

ПРОФИЛЬНАЯ ШКОЛА

В. М. Казиев

Информатика

**Информатика
в примерах
и задачах**

Москва «Просвещение» 2007

УДК 373.167.1:004
ББК 32.81я72
К14

Серия «Профильная школа» основана в 2007 г.

Казиев В. М.

К14 Информатика в примерах и задачах : кн. для учащихся 10—11 кл. / В. М. Казиев. — М. : Просвещение, 2007. — 304 с. : ил. — (Профильная школа). — ISBN 978-5-09-016550-1.

Книга предназначена для широкого круга читателей — школьников, студентов, преподавателей и родителей. Учебник может быть эффективно использован при различных формах обучения — от классической, урочной формы обучения до дистанционной, от обучения в классе или аудитории до самообучения дома или под руководством наставника (тьютора).

УДК 373.167.1:004
ББК 32.81я72

Учебное издание

Серия «Профильная школа»

Казиев Валерий Муаедович

ИНФОРМАТИКА

ИНФОРМАТИКА В ПРИМЕРАХ И ЗАДАЧАХ

Зав. редакцией *Т. А. Бурмистрова*

Редактор *А. В. Желонкин*

Младший редактор *Е. А. Андреевкова*

Дизайн обложки: *О. П. Богомолова*

Художественный редактор *О. П. Богомолова*

Компьютерная графика *О. А. Королева*

Технический редактор *Р. С. Еникеева*

Корректоры *Е. В. Барановская, Н. В. Бурдина, Л. С. Вайтман,*

Н. И. Новикова, Н. А. Смирнова

Налоговая льгота — Общероссийский классификатор продукции ОК 005-93—953000. Изд. лиц. Серия ИД № 05824 от 12.09.01. Сдано в набор 03.08.06. Подписано в печать 19.02.07. Формат 60×90^{1/16}. Бумага писчая. Гарнитура Школьная. Печать офсетная. Уч.-изд. л. 19,4. Тираж 5000 экз. Заказ № 15171 К-Л.

Открытое акционерное общество «Издательство «Просвещение». 127521, Москва, 3-й проезд Марьиной рощи, 41.

Открытое акционерное общество «Смоленский полиграфический комбинат». 214020, г. Смоленск, ул. Смольянинова, 1.

ISBN 978-5-09-016550-1

© Издательство «Просвещение», 2007
© Художественное оформление.
Издательство «Просвещение», 2007
Все права защищены

ПРЕДИСЛОВИЕ	4
1. Информатика и ее предмет	5
2. Информация и сообщение	14
3. Информационно-логические задачи и методы	25
4. Системы счисления	32
5. Алгебра высказываний и предикатов	38
6. Алгоритмы. Алгоритмы работы с числами	49
7. Алгоритмы работы с одномерными массивами	68
8. Алгоритмы работы с двумерными массивами	78
9. Алгоритмы работы с текстами	92
10. Алгоритмы работы с предикатами	100
11. Данные	106
12. Проектирование, тестирование и трассировка алгоритмов	113
13. Исполнители алгоритмов	118
14. Основы компьютера	130
15. Алгоритмические языки и методы трансляции	148
16. Вычислительная система	155
17. Формализация и аксиоматизация, модельное представление знаний	169
18. Математическое и компьютерное моделирование	177
19. Новые информационные технологии	196
19.1. Компьютерный (компьютеризированный) и виртуальный офис	—
19.2. Базы знаний и экспертные системы	229
19.3. Электронная почта и телекоммуникационный доступ	232
19.4. АРМ, САПР и интеллектуальные системы	234
19.5. Интегрированные пакеты прикладных программ	240
19.6. Средства и системы машинной графики	245
19.7. Мультимедиа, гипертекст, гипермедиа	248
19.8. Технология виртуальной реальности	251
19.9. Технологии информационного реинжиниринга	257
19.10. Объектно-ориентированные и средо-ориентированные технологии	259
19.11. CASE-технологии	262
19.12. Нейротехнологии	—
19.13. Когнитивные технологии	263
19.14. Нечеткие технологии	264
20. Компьютерные сети и системы	268
21. Информатизация общества	279
ЛИТЕРАТУРА	288

Современное понятие «информационная грамотность» включает не только то, что вкладывается обычно в понятие «грамотность пользователя, пользовательский интерфейс», но и то, что входит в понятие «грамотность постановщика, исследователя, аналитика». Без этого невозможно понимание, изучение и использование сложных междисциплинарных информационных процессов в обществе, природе, познании. Без этого невозможно обучение в профильной школе.

Парафразируя сказанное о математике Л. Д. Кудрявцевым, можно сказать: не научив основам самой науки информатики, нельзя научить ее приложениям. Научить же основам информатики можно лишь с помощью достаточно строгого формального изложения таких основополагающих категорий информатики, как информация, сообщение, формальная система, информационная система, данные, алгоритм, исполнитель, технология и др.

В данной книге читателю предлагается материал, который на достаточно строгом научном уровне и в то же время на минимальном уровне требований к предварительному уровню грамотности читателя в области информатики объясняет основы информатики и новых информационных технологий. Для этого используется специальная (апробированная многолетним опытом преподавания информатики на младших курсах вуза и в старших классах средней школы), ориентированная на понимание фундаментальных основ науки информатики и достаточно большая, полная система содержательных и формализованных определений, подробно решенных примеров, обучающих и контролирующих тестов с ответами, оригинальных задач и тем для самостоятельной работы читателя.

Книга предназначена для широкого круга читателей — школьников, студентов, их преподавателей и родителей. Учебник может быть эффективно использован при различных формах обучения — от классической, урочной формы обучения до дистанционной, от обучения в классе или аудитории до самообучения дома или под руководством тьютора, а также для организации самостоятельной работы.

И И

ИНФОРМАТИКА И ЕЕ ПРЕДМЕТ

Проследим предысторию и этапы развития информатики. Начнем с этапа добумажной информатики.

Этап иероглифической символики

Изначально носителем информации была речь. Наиболее ранние знаковые информационные системы — приметы, гадания, знаменья, живопись, графика, музыка, пластика, танец, пантомима, архитектурные сооружения, костюм, народные ремесла, обряды и др. Первые примеры информационной символики были представлены еще в каменном веке в виде рисунков на камне, в виде пиктографического письма. В бронзовом веке появились изображения повторяющихся систем понятий — идеограмм, которые с конца IV в. до н. э. превратились в иероглифическое письмо.

Этап абстрактной символики

Иероглифическое письмо сохранилось до наших дней в ряде регионов (Китай, Япония, Корея). Его сохранению способствовали удобство, наглядность и то, что народы этих стран были этнически однородны и из-за особенностей культуры, традиций, географического положения слабо мигрировали. В Средиземноморье же были различные языковые формы, развитые межнациональные торговые связи, относительно нестабильная политическая обстановка в государствах и миграции населения. Поэтому здесь за короткий исторический период завершился переход от иероглифической системы письма к абстрактной и более удобной для чтения системе клинописи на сырых глиняных табличках (III—II вв. до н. э.). Следующий исторический период создания последовательного слогового письма на глиняных табличках — вавилонский. Вавилонский язык впервые в истории начинает выполнять международные функции в дипломатии и торговле, т. е. приобретает коммуникационные и терминообразующие функции. Новым этапом явилось создание в X—IX вв. до н. э. финикийского алфавита, которое завершилось в VIII в. до н. э. появлением на основе финикийского письма греческого алфавита, впоследствии ставшего основой всех западных письменных систем. Усовершенствованием этой информационной символики стало введение во II—I вв. до н. э. в Александрии начал пунктуации. Развитие письменной символики завершается в Европе в XV в. созданием пунктуации современного вида. Появляется древнегреческая научная терминология, благодаря которой началось устранение информационной избыточности. В период Возрождения древнегреческие и латинские языки послужили основой для создания терминологических систем в различных

областях знаний. В период технической революции терминологические системы расширяются по объему и упорядочиваются за счет фундаментальных законов природы и общества, взаимопроникновения терминов различных наук. Математическая символика продолжает качественно развиваться благодаря углубленным исследованиям и фундаментальным открытиям математики (создание совершенной алгебраической символики в XIV—XVII вв., введение знаков операций в XV в., введение знаков равенства, бесконечности в XVII в., появление знаков степени, дифференциала, интеграла в XVII в. и др.).

Этап картографии, технической графики и информационной визуализации

Особая форма представления, визуализации знаний — карты, отображающие явления природы и общества в виде информативных образов и знаков. Первые карты, дошедшие до наших дней, были составлены в Вавилоне (III—I тыс. до н. э.). Карта мира была впервые составлена Птолемеем во II в. до н. э. Создание новых картографических проектов и технологий их составления происходит в конце XVI в. Возникновение технической графики относится ко времени появления ранней письменности и развивается в связи с сооружением сложных объектов (замечательные пирамиды, дворцы, шахты, водопроводные системы) в III—II тыс. до н. э. Дальнейшее развитие техническая графика получила в эпоху Возрождения в связи с конструированием сложных машин и механизмов, например, военного характера и возведением крупных городов.

Этап «каменписи», «глинописи», «древнописи», «пергаментописи»

Добумажная информационная технология характеризуется переходом к более совершенным носителям; например, запись на камне позволяет впервые добиться эффекта обезличивания процесса передачи информации. Переход к записи на глиняных табличках и деревянных дощечках позволяет перейти к информационным коммуникациям (появляется новое свойство информационной динамичности). Изобретение папируса (III тыс. до н. э.) значительно повышает емкость (актуализируется новое свойство информации — сжимаемость). Появление пергамента завершает добумажную фазу, так как появляется оптимальный носитель информации — книга (IV в. до н. э.). На развитие механизма информационного взаимодействия людей в добумажную эпоху оказывают влияние социальные, политические, региональные и другие факторы. В каменном веке пиктограмма представляла собой общедоступное информационное сообщение, что соответствовало низкому уровню развития труда и социальной иерархии. На этапе создания первых государств глиняные и деревянные таблички хранились в закрытом помещении, а пользоваться ими могла только аристократия, поэтому появилась потребность в обучении. Появляются централи-

званные хранилища информации. В этот период библиотеки становятся доступными для свободных граждан центрами сосредоточения информационных носителей. Качественно новый, динамичный и открытый характер приобретает информационные коммуникации, когда в крупных государствах (Греция, Персия, Египет) возникла хорошо налаженная почтовая связь. Впервые появляется инструмент массовой информационной коммуникации.

Рассмотрим теперь этап бумажной информатики. Бумажный этап развития информатики можно отсчитывать, видимо, с X в., когда бумага стала производиться в странах Европы. С расширением торговли появились городские почты: с XV в. — частная почта, с XVI в. — королевская почта. Благодаря этим коммуникациям информационная деятельность начинает расширяться, появляются первые университеты (Италия, Франция), которые начинают играть роль центров хранения и передачи информации, культуры и знания. Классическое университетское образование базируется на фундаментальности, универсальности, гармонизации образования, методов и средств актуализации информации.

Этап книгопечатания

Книгопечатание было изобретено в Германии в XV в. как массовая деятельность и стало началом нового научного этапа в естествознании (станок Гутенберга, 1440—1450 гг.). Главным качественным достижением того времени стало возникновение систем научно-технической терминологии в основных отраслях знаний, появились журналы, газеты, энциклопедии, географические карты.

Этап технической (индустриальной) революции XIX в.

Книгопечатание развивало науки, способствовало систематизации и формализации знаний по отраслям. Эти знания можно было теперь быстро тиражировать (появляется еще одно важное свойство информации). Знания стали доступны многим, в том числе и территориально удаленным друг от друга, а также удаленным по времени участникам трудового процесса (усиливаются пространственно-временные свойства информации). Появляются признаки параллелизма в передаче и актуализации информации, знаний. Начала раскручиваться спираль технической цивилизации: текущее знание — текущее общественное производство — новое знание — новое общественное производство. Печатный станок резко повысил пропускную способность социального канала обмена знаниями. Новый этап в развитии информатики, связанный с технической революцией XIX в., ассоциируется с началом создания регулярной почтовой связи как формы стабильных международных коммуникаций. Возникли фотография (1839), телеграф (1832), Всеобщий почтовый союз (1874), Всемирная почтовая конференция (1878), телефон (1876), радио (1895), кинематограф (1905), беспроводная передача изображения (1911), промышленное телевидение (1920), цифровые фотография и телевидение, сотовая связь (конец XX в.).

Этап математизации и формализации знаний

С развитием промышленной революции становится все более острой потребность в создании системы описания и использования профессиональных знаний, введения фундаментальных и профессиональных понятий, формирования основных элементов технологии, формализации профессиональных знаний. Первые признаки этого процесса восходят к временам, когда жрецы отказались от всеобщего контроля и перешли к индивидуальной специализации (появились первые специалисты — звездочеты, лекари и др.). Наиболее успешно развивается в этот период процесс формализации астрономических знаний. Появляются книги с астрономическими формулами, таблицами, а на их базе разрабатываются навигационные инструменты, что позволяло передавать профессиональные знания и умения, например за несколько лет обучать профессии мореплавателя. Отчуждаемость профессиональных знаний от их носителей до самого последнего времени определялась возможностью формализации профессиональных знаний математическими методами и аппаратом. Области профессиональных знаний, которые оказались более формализуемыми, получили название точных или естественных наук — математика, физика, биология, химия и др. Остальные науки стали называть гуманитарными. Процесс формализации знаний, как правило, сводился к попыткам выделения из всего многообразия сведений в некоторой области человеческой деятельности небольшой части, логически определяющей достаточно многое (система аксиом и правила вывода). Отправитель и получатель информации (знаний) пользовались некоторым общим набором правил для их представления и восприятия — формализмом представления знаний. Мысль, которую нельзя выразить формализмом (языком), не может быть включена в информационный обмен, в обмен знаниями. В отраслях науки формируются специфические языковые системы, среди которых особенно важен язык математики как информационная основа знаний в точных, естественных науках, как язык науки. Свои языки имеют химия (язык структурных химических формул, например), физика (язык описания атомных связей), биология (язык генетических связей и кодов) и т. д. Нынешний этап развития информатики характерен созданием и становлением языка информатики.

Этап информатизации, информационно-логического представления знаний

С появлением ЭВМ впервые стал возможен способ записи и долговременного хранения профессиональных знаний, ранее формализованных математическими методами (алгоритмов, программ, баз данных, эвристик и т. д.). Эти знания, а также опыт, навыки, интуиция уже могли широко использоваться и без промежуточного воздействия на человека влиять на режим работы производственного оборудования (на автоматизацию процессов). Процесс записи ранее формализованных профессиональных знаний в форме,

готовой для воздействия на механизмы (автоматы), получил изначально название «программирование». В начале 70-х гг. начал наблюдаться информационный кризис: резко возрос объем научнотехнической публикации; специалистам различных областей стало трудно общаться; возрос объем используемой неопубликованной информации; возникли сложности в восприятии, переработке информации, выделении нужной информации из общего потока и др. Если машины и системы автоматизации в сфере материального производства постоянно совершенствовались и производительность труда росла, то в сферу обработки информации средства автоматизации проникали с большим трудом. Численность людей в информационной сфере к началу 80-х гг. в большинстве развитых стран составляла около 60% от общего числа занятых в производстве и продолжала расти, т. е. ЭВМ применялись там, где существовала формальная постановка задач (алгоритм). Кроме этого, ЭВМ использовались для хранения и обработки по стандартным процедурам больших наборов данных.

Этап автоформализации знаний

Этот этап тесно связан с развитием персональных компьютеров и вычислений, благодаря которым стало возможным формальное описание (актуализация, передача, хранение, сжатие) исследователями накопленных знаний, опыта, профессиональных умений и навыков. Развиваются когнитивные методы и средства, позволяющие строить решения проблем по ходу их рассмотрения, на лету, особенно эффективно в тех случаях, когда исследователю неизвестен путь решения. Развиваются методы виртуализации и визуализации. Этот этап очень важен для информатики, ибо он стал позволять решать межпредметные задачи, как правило, плохо структурируемые и формализуемые, а также позволил использовать типовые инструментальные системы. Используется когнитивная графика, порождающая новые решения, а также виртуальный мир — искусственное трехмерное пространство (одну из осей координат можно условно считать пространственной, другую — временной, третью — информационной) и визуальные среды.

Рассмотрим современный период безбумажной информатики.

Этап развитой безбумажной информатики и глобальных систем связи (Интернет), оформления информатики в фундаментальную научную дисциплину

Переход к безбумажной информатике, электронным информационным технологиям и использованию сетей Интернет, информационному производству товаров и услуг характерен для всех стран, вступивших в стадию построения информационного общества.

Двадцать первый век можно считать веком информационного сообщества, единого и доступного мирового информационного про-

странства (поля), которое будет улучшать как производительные силы и производственные отношения, так и личность, общество.

Информатика — молодая наука. Часто и обоснованно информатика трактуется как фундаментальная научная и образовательная область, как теоретическая (математическая) информатика, которая не может ограничиться рамками инженерных, технократических применений, рамками процедурного программирования. Информатика имеет мощные формальные средства для глубокого изучения сути явлений и систем, а также их практической интерпретации, для усиления междисциплинарных связей, изучения сложных процессов и систем. Информатика завершает этап несколько хаотичного развития и уже накопила достаточный опыт и знания для ее систематизации, осмысления, структурирования, теоретизации, превращения в фундаментальную науку.

Информатика, рассматриваемая с точки зрения хранения и преобразования информации, как правило, сводится к компьютерам, служащим определенным практическим целям. Это ресурсный, или технократический, подход к информатике. Если же информатика рассматривается с коммуникационных позиций, например с позиции передачи знаний, то она неотъемлемый фрагмент общей культуры общества, основа информационной культуры. Оба подхода должны быть взаимосвязаны. Абсолютизация первого подхода приводит к заблуждению, что технические возможности предопределяют цели развития общества (возникают технократические утопии). Абсолютизация второго может привести к забвению технических возможностей информатики, недооценке технических нововведений, к излишнему формализму и отрыву от практики.

Информатика — наука об информации, информационных потоках и процессах, о моделях и моделировании, об алгоритмах и алгоритмизации, о программах и программировании для различных классов исполнителей алгоритмов, в частности компьютеров, об их использовании в обществе, познании.

Появление информатики вызвано возникновением и развитием индустрии, технологии сбора, обработки, передачи, актуализации информации, особенно с помощью компьютеров и автоматизированных систем и устройств. Информатика появилась как итог развития математики, физики, управления, техники и других наук.

ПРИМЕР

Задача построения математической модели эпидемии гриппа — задача из области теоретической информатики. Задача построения алгоритма исследования этой модели для прогноза эпидемии — задача из теоретической информатики. Задача разработки компьютерной программы для прогноза эпидемии является, например, задачей практической информатики. Задача разработки технических устройств обеспечения прогноза, например разработки системы передачи, обработки данных о больных, — задача технической информатики.

Предмет информатики точно невозможно определить в силу его сложности, многосторонности, динамической изменчивости. Тем не менее можно отметить следующие три основные ветви информатики, определяемые ее познавательной (теоретической) и прагматической (практической) функциями, ее внутренней и внешней сущностью. При этом заметим, что деление информатики как науки и человеческой деятельности на те или иные части зависит от целей, задач, ресурсов и часто оно условное.

Теоретическая (математическая) информатика (brainware) изучает теоретические проблемы информатики (большей частью связанные с формальными системами, моделями, алгоритмами и теорией программирования, кодирования и организации систем).

Практическая (прикладная) информатика (software) изучает практические, конкретные проблемы информатики (большей частью связанные с программированием и использованием моделей, программными и компьютерными технологиями и системами).

Техническая (инженерно-физическая) информатика (hardware) изучает инженерно-физические, технические проблемы информатики (большей частью связанные с разработкой и использованием технических средств обработки информации, ЭВМ и систем ЭВМ, сетей).

Информатика базируется на следующих основных и важных понятиях:

- информация и сообщение, в частности получение, переработка, сжатие, актуализация информации сообщениями;
- алгоритм и алгоритмизация, в частности программа и программный комплекс, проектирование программ и программирование;
- система и структура, отношение и связь, порядок, выбор, в частности информационные системы и отношения в них;
- модель и моделирование, в частности описание и исследование систем с помощью моделей и моделирования;
- исполнитель и его операционная среда, в частности ЭВМ и система ЭВМ;
- языки и грамматики, в частности алгоритмические языки, языки программирования, языки общения с различными системами и средами;
- проектирование систем и технология, в частности информационная, компьютерная технология.

Предметная область информатики — информационные процессы и системы, модели, языки их описания, технологии их актуализации, направленные как на получение знаний (**внутренняя сущность информатики**), так и на применение знаний, принятие на их основе решений (**внешняя сущность информатики**).

Эти информационные процессы могут происходить в живых существах (организмах), автоматах (технических устройствах), обществе, в индивидуальном и общественном сознании.

Информатика, как и математика, является наукой для описания и исследования проблем других наук. Она предоставляет свои общие и/или частные методы исследования другим наукам, помогает прокладывать и усиливать междисциплинарные связи, исследовать проблемы различных наук, цементирует их своими идеями, методами, технологиями и результатами.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Чем в основном вызвано появление информатики как науки?
2. Перечислите три основные ветви информатики. Укажите их основные различия и сходства. Приведите примеры задач.
3. На каких основных понятиях базируется информатика?
4. Чем отличается внутренняя и внешняя сущность информатики?
5. Какова роль информатики, ее методов для развития общества?
6. Какова роль информатики в среде других наук?
7. Какая из наук близка к информатике по целям и почему?

ЗАДАЧИ

1. Заполните правильно места, помеченные знаком вопроса:
 - а) Предметная область науки информатики — это?.
 - б) Термин «информатика» был впервые введен?.
 - в) Информатика возникла, потому что?.
2. Напишите эссе на тему «История появления информатики».
3. Напишите эссе на тему «Снижение эффективности информационного обмена в период, предшествующий информатике».
4. Напишите эссе на тему «Междисциплинарная связь информатики с математикой (физикой, экономикой, биологией и др.)».
5. Напишите эссе на тему «Применение информатики в общественной жизни (науке, быту, познании)».
6. Напишите эссе на тему «Личность, внесшая важный вклад в информатику».
7. Напишите эссе на тему «Пути развития информатики».

ТЕСТ

1. Информатика — наука, изучающая...
 - а) ЭВМ, алгоритмы, программы;
 - б) структуру, свойства информации, системы ее актуализации;
 - в) компьютерные сети и системы;
 - г) программирование, компьютерные системы, Интернет.

2. Наиболее полно предмет информатики отражен в пункте:
 - а) информационные ресурсы, методы и технологии;
 - б) ресурсы общества;
 - в) информационные и компьютерные системы;
 - г) знания и информация.
3. Информатика — это:
 - а) междисциплинарная наука;
 - б) раздел математики;
 - в) техническая наука;
 - г) раздел теории алгоритмов и их исполнителей.
4. Информатика имеет три основные ветви:
 - а) brainware, hardware, software;
 - б) hardware, software, techware;
 - в) hardware, software, netware;
 - г) hardware, software, compware.
5. Информационные процессы изучались и использовались:
 - а) с древнейших времен;
 - б) с начала XVIII в.;
 - в) с начала XIX в.;
 - г) с момента появления первых ЭВМ.
6. Термин «информатика» образован соединением двух слов:
 - а) информация и статика;
 - б) информация и автоматика;
 - в) информация и математика;
 - г) информация и энергетика.
7. В основные функции информатики как науки входит:
 - а) разработка методов изучения информационных процессов;
 - б) разработка моделей информационных процессов;
 - в) построение технологий применения (актуализации) знаний;
 - г) все перечисленное в пунктах а, б, в.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
б	а	а	а	а	б	г

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Теория информационного взрыва — причины и последствия.
2. История информатики как науки о знаниях.
3. История информатики как науки о технологиях.
4. Информатика как междисциплинарная наука.
5. Методологическая роль информатики.
6. Информационное поле человечества.
7. Открытия в науке и технике и появление информатики.

И 2 ИНФОРМАЦИЯ И СООБЩЕНИЕ

Понятие «информация» (informatio — разъяснение, осведомление, изложение) является одним из основных, ключевых понятий не только в информатике, но и в математике, физике и др. Понятие «информация» — плохо формализуемое и структурируемое объемное и всеобщее понятие. Как правило, это понятие в курсе информатики не определяется, принимается как исходное базовое понятие, неопределяемый термин. Ниже мы часто будем использовать термин *система*, который упрощенно будет означать выделяемый из окружающей среды комплекс элементов (объектов) с общей целью, общими ресурсами, связями и аспектами рассмотрения.

Информация трактуется по-разному, например как:

- сущность, которая вызывает изменения в некоторой информационно-логической (инфологической — состоящей из данных, знаний, абстракций и т. д.) модели системы (математика, системный анализ);
- сообщения, полученные системой от внешнего мира в процессе адаптивного управления, приспособления (теория управления, кибернетика);
- отрицание энтропии, отражение меры хаоса в системе (термодинамика);
- связи, устраняющие неопределенность в системе (теория информации);
- вероятность выбора в системе (теория вероятностей);
- отражение разнообразия в системе (физиология);
- отражение материи, атрибут сознания, интеллекта системы (философия).

В соответствии с парадигмой информатики как фундаментальной образовательной и научной дисциплины необходимо введение в это фундаментальное понятие информатики не только на научном, строго понятийном, но и на доступном, содержательном уровне.

Алфавит — конечное множество знаков для представления сообщений, т. е. цепочек из этих знаков (слов), образуемых по некоторым правилам.

ПРИМЕР

Примерами алфавитов являются: алфавит арифметики — множество из десяти цифр, знаков арифметических операций и десятичной точки (запятой); алфавит из знаков русского языка, который ошибочно часто называется алфавитом русского языка; алфавит русского языка — множество знаков русского языка, знаков препинания и пробела; алфавит из точки и тире в азбуке Морзе и др.

Информация — это некоторая последовательность сведений, знаний, сообщений, выражаемых с помощью некоторого алфавита символов, жестов, звуков. Информация или регистрируется, или преобразовывается, или передается, или используется (актуализируется) с помощью некоторых сообщений. Формы облачения информации в сообщения различны.

ПРИМЕР

Для живых существ форма информации — сигналы, жесты, для различных технических устройств — сигналы.

Информация по отношению к окружающей среде (используемой ее среде) бывает трех типов: входная, выходная и внутренняя.

Входная информация (по отношению к окружающей среде) — информация, которую система воспринимает из окружающей среды.

Выходная информация (по отношению к окружающей среде) — информация, которую система выдает в окружающую среду.

Внутренняя (внутрисистемная) информация (по отношению к системе) — информация, которая хранится, перерабатывается, используется только внутри системы, актуализируется лишь подсистемами системы. Это идеализированное (с точки зрения физики открытых систем) понятие.

ПРИМЕР

Человек воспринимает, обрабатывает входную информацию, например данные о погоде на улице, и формирует выходную реакцию — форму одежды. При этом используется и внутренняя информация, например генетически заложенная (приобретенная) физиологическая информация, в частности о «морозостойкости» человека.

- Информация по отношению к конечному результату делится на:
- **исходную** (на начало актуализации этой информации);
 - **промежуточную** (от начала до завершения актуализации);
 - **результатирующую** (после завершения ее актуализации).

ПРИМЕР

При решении системы линейных алгебраических уравнений информация о методах решения, среде реализации, входных данных (источники, точность и т. д.), размерности системы и т. д. является исходной информацией; информация о совместности системы уравнений, численных значениях корня и т. д. — результирующей; информация о текущих состояниях коэффициентов при решении системы уравнений — промежуточной.

- Информация по изменчивости при ее актуализации бывает:
- **постоянная** (не изменяемая никогда при ее актуализации);
 - **переменная** (изменяемая при актуализации);
 - **смешанная** — условно-постоянная (или условно-переменная).

ПРИМЕР

В задаче определения дальности полета снаряда артиллерийского орудия информация об угле наклона орудия может быть переменной, информация о начальной скорости вылета снаряда — постоянной, а информация о координатах цели — условно-постоянной.

Возможна классификация информации и по другим признакам:

- по стадии использования в системе (первичная, вторичная);
- по полноте для системы (избыточная, достаточная, недостаточная);
- по отношению к цели системы (синтаксическая, семантическая, прагматическая);
- по отношению к элементам системы (статическая, динамическая);
- по отношению к структуре системы (структурная, относительная);
- по отношению к управлению системой (управляющая, советующая, преобразующая, смешанная);
- по отношению к территории (федеральная, региональная, местная, относящаяся к юридическому лицу, относящаяся к физическому лицу, смешанная);
- по доступу (открытая или общедоступная, закрытая или конфиденциальная, смешанная);
- по предметной области, по характеру использования (статистическая, коммерческая, справочная, научная, учебная и т. д.).

Основные свойства информации (и сообщений):

- полнота (минимально необходимые сообщения для понимания);
- актуальность (своевременность, необходимость);
- ясность (выразительность сообщений на языке интерпретатора);
- адекватность, точность, корректность (актуализации знаний);
- интерпретируемость и понятность (интерпретатору);
- достоверность (отображения сообщениями);
- массовость (применимость ко всем проявлениям);
- кодируемость и экономичность (актуализации сообщений);
- сжимаемость и компактность (сообщений);
- защищенность и помехоустойчивость (актуализации сообщений);
- устойчивость (к изменениям входных данных);
- доступность (интерпретатору сообщений, для приема-передачи);
- ценность (значимость на уровне подготовки потребителя к восприятию).

Информация — это содержание сообщения, сообщение — форма проявления информации. Это приращение, развитие, актуализация знаний, возникающие в процессе интеллектуальной деятельности человека. Никакая информация не появляется сразу — этому предшествует этап накопления, осмысления, систематизации опытных данных, взглядов. Знание — продукт такого процесса. Мышление — необходимый атрибут такого процесса.

ПРИМЕР

Рекламный щит — красочный кусок железа, но информация, заложенная в сообщениях на щите, должна обладать всеми вышеперечисленными свойствами, и только тогда щит будет ассоциироваться у интерпретатора (человека) с рекламируемым товаром и актуализировать информацию о потребительских свойствах товара. При этом вся форма представления рекламы должна строиться с учетом понятности интерпретатору, быть информативной. Пока символы не организованы определенным образом, не используются для некоторой определенной цели, они не отражают информацию.

Сообщения измеряются в битах, байтах, килобайтах (К или Кбайт), мегабайтах (М), гигабайтах (Г), терабайтах (Т), петабайтах (П) и эксабайтах (Э). Любые сообщения кодируются в ЭВМ в цифровом виде с помощью алфавита из нулей и единиц, записываются и реализуются в ЭВМ в битах.

Соотношения между единицами измерения сообщений:

1 бит (от слова binary digit — двоичная единица) — это 0 или 1,

1 байт = 8 бит, например байт вида 10001110,

1 килобайт (1 К) = 2^{10} байт, или 2^{13} бит,

1 мегабайт (1 М) = 2^{10} К, или 2^{20} байт, или 2^{23} бит,

1 гигабайт (1 Г) = 2^{10} М = 2^{20} К = 2^{30} байт = 2^{33} бит,

1 терабайт (1 Т) = 2^{10} Г = 2^{20} М = 2^{30} К = 2^{40} байт = 2^{43} бит,

1 петабайт (1 П) = 2^{10} Т = 2^{20} Г = 2^{30} М = 2^{40} К = 2^{50} байт = 2^{53} бит,

1 эксабайт (1 Э) = 2^{10} П = 2^{20} Т = 2^{30} Г = 2^{40} М = 2^{50} К = 2^{60} байт = 2^{63} бит.

Последние три единицы (есть еще и три следующие за ними) редко используются. Обычно достаточно остальных.

ПРИМЕР

Найти неизвестные x и y , если верны соотношения

$$16^y \text{ (М)} = 8^x \text{ (бит)}; 32^x \text{ (К)} = 2^y \text{ (М)}.$$

Выравниваем единицы измерения информации:

$$16^y \text{ (М)} = 16^y \cdot 2^{10} \cdot 2^3 \text{ (бит)}; 2^y \text{ (М)} = 2^y \cdot 2^{10} \text{ (К)}.$$

Отбрасывая теперь размерности информации, получаем

$$2^{4y+23} = 2^{3x}; 2^{5x} = 2^{y+10}.$$

Отсюда получаем систему уравнений $\begin{cases} 4y + 23 = 3x, \\ 5x = y + 10, \end{cases}$ откуда $x = 1, y = -5$.

ПРИМЕР

Матричный принтер имеет скорость печати 180 байт/с. Определим время, необходимое для распечатки 10 листов, если каждый лист вмещает 60 строк по 30 символов. Каждый лист содержит $60 \cdot 30 = 1800$ символов, а так как каждый символ

кодируется байтом, то всего на одном листе будет 1800 байт информации. Для печати одного листа потребуется $1800 : 180 = 10$ (с). Для печати 10 листов необходимо 100 с (около 17 мин).

Кодом называется правило, описывающее соответствие данного набора знаков одного множества X знакам другого множества Y , который может и совпадать с X . Если для каждого символа из Y существует код для X , то это правило называется декодированием. Любая информация может быть воспринята и обработана приемником этой информации только в виде некоторых закодированных сообщений на языке сигналов, жестов, символов и т. д.

Кодирование — процесс преобразования букв (слов) одного алфавита в буквы (слова) другого алфавита.

ПРИМЕР

Код (одношаговый) Цезаря: каждый символ алфавита, на котором записано сообщение, заменяется другим, определенным символом этого же алфавита, например отстоящим от него вторым символом алфавита. Двухшаговый код Цезаря кодирует каждую букву алфавита буквой, следующей за ней через одну букву алфавита. Закодируем текст «Яснаяпогода» одношаговым кодом Цезаря, устанавливая соответствия вида «а» — «б», «б» — «в», «в» — «г» и т. д. Закодированный текст будет иметь вид «Атобаридпнеб». Раскодируйте текст «Обкщйл», закодированный этим кодом.

При внутреннем представлении сообщений в памяти ЭВМ все символы кодируются байтами. Количество всех различных символов, которые можно при этом представить в ЭВМ, равно $2^8 = 256$, что, например, больше количества разных символов на стандартной клавиатуре ЭВМ (не специального назначения, например, на клавиатуре настольной издательской системы).

ПРИМЕР

Словом, закодированным последовательностью битов вида 11110001110100011111000110011111, если коды букв М, И, О условно равны 11110001, 11010001, 10011111, будет слово МИМО. Запишем, используя предыдущие коды букв М, И, О, битовую информацию, соответствующую слову ОМИММО. Это слово запишется в виде:

100111111111000111010001111100011111000110011111.

ПРИМЕР

Человек способен различать примерно 130 градаций яркости. Определим, сколько битов необходимо, чтобы их закодировать, т. е. какой минимальной разрядности комбинации битов нужны для этого. Для кодировки 1—2 цветов нужна однобитовая комбинация (например, белый — бит 0, черный — бит 1), кодируются 2^1 цвета; для кодировки 1—4 цветов необходима двухбитовая комбинация (например, белый — 00, жел-

тый — 01, синий — 10, черный — 11), кодируются 2^2 цвета; для кодировки 1—8 цветов необходима трехбитовая комбинация (например, оранжевый — 000, белый — 001, черный — 010, синий — 011, красный — 100, голубой — 101, зеленый — 110, бежевый — 111), кодируются 2^3 цвета. Отсюда видно, что для кодировки 130 цветов необходимо выбрать число разрядов n так, чтобы выполнялось неравенство $2^n \geq 130$. Так как $2^8 = 256 \geq 130$, а $2^7 = 128 < 130$, то необходимо брать для кодировки 130 градаций яркости восьмиразрядные битовые комбинации (байты).

Правила кодирования (шифрования) не могут быть произвольными. Они должны быть такими, чтобы зашифрованное сообщение можно было прочесть (расшифровать). Однотипные правила (типа правила Цезаря) можно объединять в классы. Если внутри класса правил выбран параметр (числовой, табличный и т. д.), изменением значений которого можно перебрать все правила, то такой параметр называется **шифровальным ключом**.

Имеются две группы шифров: шифры перестановки и шифры замены. **Шифр перестановки** изменяет только порядок следования символов исходного сообщения. **Шифр замены** меняет каждый символ кодируемого сообщения на другие символы, не меняя порядка их следования.

ПРИМЕР

Шифр Цезаря — шифр типа перестановки. Шифр, состоящий в замене каждого символа алфавита русского языка его двухзначным порядковым номером, — шифр типа замены.

Восстановление некоторого зашифрованного сообщения по ключу — **расшифрование**, а восстановление (прочтение) его без известного ключа — **вскрытие (взлом) шифра**.

Кодирование и шифрование — разные процедуры. При кодировании нет секретного ключа (достаточно знать соответствие символов), а при шифровании необходимо наличие ключа (нужно знать и шифр, и ключ к шифру). Отличны и их цели. Кодирование не ставит основной целью недоступность для чтения, а ставит целью более сжатое, компактное и быстрое представление и преобразование текста. Шифрование же ставит целью сделать сообщение недоступным для чтения без обладания ключом к шифру.

ПРИМЕР

С некоторой планеты поступают импульсы. Отождествив один импульс с полубайтом (4 бита), получаем последовательность импульсов (кодированных сообщений):

$X[1] = 000100001001010101100001,$

$X[2] = 000100010000100101010101100001,$

$X[3] = 00010001000100001001010101010101100001$ и т. д.

Выведена гипотеза: если сообщения $X[i]$, $i = 1, 2, 3, \dots$, подчиняются квадратичному закону (т. е. каждое $X[i]$ представимо в виде квадрата некоторого числа в какой-то системе счисления), то на планете живут разумные существа. Определим, разумные ли существа живут на планете. Переведем полубайты в десятичную систему: $X[1] = 109561$, $X[2] = 11095561$, $X[3] = 1110955561$, ... Можно записать равенство $X[n] = 111...1109555...55561$. Многоточие соответствует n единицам или пятеркам. Запишем последнее равенство в виде

$$\begin{aligned} X[n] &= 11...1 \cdot 10^{n+4} + 9 \cdot 10^{n+2} + 55...5 \cdot 10^2 + 61 = \\ &= (10^n - 1)/9 \cdot 10^{n+4} + 9 \cdot 10^{n+2} + 5(10^n - 1) \cdot 10^2/9 + 61 = \\ &= (10^{2n+4} - 14 \cdot 10^{n+2} + 49)/9 = [(10^{n+2} - 7)/3]^2. \end{aligned}$$

Итак, последовательность сообщений подчиняется закону:

$$X[1] = 331^2, X[2] = 3331^2, X[3] = 33331^2, \dots$$

Вывод: на планете живут разумные существа.

Для измерения информации используются другие, более сложные принципы и методы, например, часто используется мера информации по К. Шеннону.

Количество информации — числовая величина, адекватно характеризующая информацию по разнообразию, структурированности, определенности, выбору состояний системы.

Если рассматривается система, которая может принимать одно из n возможных состояний, то актуальна задача оценки такого выбора, исхода. Такой оценкой может стать мера информации (события). Мера информации — критерий оценки количества информации.

1. Мера Р. Хартли. Пусть имеется N состояний системы S или N опытов с различными, равновероятными, последовательными состояниями системы. Если каждое состояние системы закодировать, например, двоичными кодами определенной длины d , то эту длину необходимо выбрать так, чтобы число всех различных комбинаций было не меньше чем N . Наименьшее число, при котором это возможно, или мера разнообразия множества состояний системы, задается формулой Р. Хартли:

$$H = \log_2 N \text{ (бит).}$$

ПРИМЕР

Чтобы определить состояние системы из двух возможных состояний, т. е. получить некоторую информацию о системе, необходимо задать один вопрос. Узнав состояние, мы увеличиваем суммарную информацию о системе на 1 бит ($I = \log_2 2$). Для 3—4 возможных состояний информация равна 2 битам ($I = \log_2 4$). Если система имеет n различных состояний, то максимальное количество информации равно $I = \log_2 n$.

ПРИМЕР

Выясним, сколько бит информации несет каждое двузначное число aa со всеми значащими цифрами (отвлекаясь при этом от его конкретного числового значения). Так как таких чисел может быть всего 90 ($10 - 99$), то количество информации будет $I = \log_2 90$ или приблизительно $I = 6.5$. Так как в таких числах значащая первая цифра имеет 9 значений ($1 - 9$), а вторая — 10 значений ($0 - 9$), то $I = \log_2 90 = \log_2 9 + \log_2 10$. Приблизительное значение $\log_2 10$ равно 3.32. Итак, сообщение в одну десятичную единицу несет в себе в 3.32 больше информации, чем в одну двоичную единицу (чем $\log_2 2 = 1$), а вторая цифра в числе aa несет в себе больше информации, чем первая, если цифра a обоих разрядов неизвестна; если же эта цифра a известна, то выбора нет — информация равна нулю.

Утверждение Хартли: если во множестве $X = \{x_1, x_2, \dots, x_n\}$ искать произвольный элемент, то для его нахождения необходимо иметь не менее $\log_2 n$ (единиц) информации.

ПРИМЕР

Имеются 192 монеты, из которых одна фальшивая. Определим, сколько взвешиваний нужно произвести, чтобы вывить ее. Если положить на весы равное количество монет, то получим три независимые возможности: а) левая чашка ниже; б) правая чашка ниже; в) чашки уравновешены. Таким образом, каждое взвешивание дает количество информации $I = \log_2 3$, следовательно, для определения фальшивой монеты нужно сделать не менее k взвешиваний, где наименьшее k удовлетворяет условию $\log_2 3^k \geq \log_2 192$. Отсюда $k \geq 7$ или $k = 7$ ($k = 8$ — если считать за взвешивание последнее, очевидное для определения монеты и поэтому лишнее). Итак, необходимо сделать не менее 7 взвешиваний (7 достаточно).

Уменьшение (увеличение) H говорит об уменьшении (увеличении) разнообразия состояний N системы. Обратное, как это следует из формулы Хартли, также верно.

2. Мера К. Шеннона. Мера Шеннона оценивает информацию отвлеченно от ее смысла:

$$I = - \sum_{i=1}^n p_i \log_2 p_i,$$

где n — число состояний системы; p_i — вероятность (относительная частота) перехода системы в i -е состояние, а сумма всех p_i равна 1.

Если все состояния равновероятны, $p_i = 1/n$, то $I = \log_2 n$. Если выбор i -го варианта предопределен заранее (выбора, собственно, нет, $p_i = 1$), то $I = 0$.

Сообщение о наступлении события с меньшей вероятностью несет в себе больше информации, чем сообщение о наступлении события с большей вероятностью. Сообщение о наступлении достоверно наступающего события несет в себе нулевую информацию (событие все равно произойдет когда-либо).

ПРИМЕР

Если положение точки в системе клеток известно, например, она находится в k -й клетке, т. е. все $p_i = 0$, кроме $p_k = 1$, то тогда $I = \log_2 1 = 0$ и мы новой информации не получаем.

Если в формуле Шеннона обозначить $f_i = -n \log_2 p_i$, то получим, что I можно понимать как среднеарифметическое величин f_i . Отсюда f_i можно интерпретировать как информационное содержание символа алфавита с индексом i и величиной p_i вероятности появления этого символа в сообщении, передающем информацию.

Если k — коэффициент Больцмана, $k = 1.38 \cdot 10^{-16}$ эрг/град, то выражение

$$S = -k \sum_{i=1}^n p_i \ln p_i$$

в термодинамике известно как энтропия, или мера хаоса в системе. Сравнение выражений I и S дает, что I можно понимать как энтропию из-за нехватки информации.

Нулевой энтропии соответствует максимальная информация. Основное соотношение между энтропией и информацией:

$$I + S(\log_2 e)/k = \text{const},$$
$$dI/dt = -((\log_2 e)/k) dS/dt.$$

Увеличение (уменьшение) меры Шеннона — свидетельство уменьшения (увеличения) энтропии (организованности) системы. При этом энтропия может являться мерой дезорганизации систем от полного хаоса ($S = S^{\max}$) и полной информационной неопределенности ($I = I^{\min}$) до полного порядка ($S = S^{\min}$) и полной информационной определенности ($I = I^{\max}$) в системе.

ПРИМЕР

Чем ближе к нам движущийся объект, тем полнее информация, обрабатываемая нашими органами чувств, тем более информативен и структурирован (упорядочен) объект. Чем больше информации мы имеем о компьютерной технике, тем меньше психологический барьер перед компьютером (согласно основному соотношению между энтропией и информацией).

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое информация? Приведите пример информации.
2. Как классифицируется информация? Приведите примеры.
3. Чем отличается информация от сообщения? Приведите примеры.
4. Что такое кодирование, декодирование? Приведите примеры.
5. В чем смысл количества информации? Какова связь количества информации с энтропией, хаосом?
6. Что означает термин «актуализация информации»?
7. Каковы основные свойства информации?

ЗАДАЧИ

1. Принтер печатает 3,2 Кбайт в минуту а каждый символ печатаемого документа представляется байтом. За сколько минут напечатает принтер документ из 16 страниц, если на одной странице 64 строки по 64 символа?
2. Один том из 50 томов БСЭ (Большой советской энциклопедии) состоит примерно из 50 печатных листов. Один печатный лист — это 40 000 символов. Сколько мегабайт необходимо для записи всей БСЭ? Можно ли поместить всю БСЭ на CD (емкостью 700 Мбайт)?
3. Найдите x и y из системы:
а) 32^x (байт) = 128^y (К), 256^{x-1} (М) = 2^{y+2} (Г);
б) 2^x (К) = 256^y (бит), 1024^y (Г) = 2^{x-2} (М).
4. Сколько вопросов, как минимум, нужно задать, чтобы отгадать закодированное натуральное число: а) от 1 до 31; б) от 1 до 100; в) от 1 до N ? (Обоснуйте ответ.)
5. Для точности передачи сообщений и ликвидации шумов в сообщениях используют трехкратную последовательную передачу каждого символа. В результате сбоя при передаче информации приемником было принято сообщение: «*PPRRRAOOCSS*HT*OO*». Какое осмысленное сообщение передавалось, если в каждой передаче символа сделана только одна ошибка и есть только один неопознанный символ (помечен знаком «*»)?
6. Некоторая система может находиться в четырех состояниях с вероятностями: в первом (худшем) — 0.1, во втором и третьем (среднем) — 0.25, в четвертом (лучшем) — 0.4. Чему равно количество информации в системе?
7. Некоторая система может принимать 128 различных равновероятных состояний. Каково количество информации в системе, если состояние системы неизвестно? Чему равно количество информации, если известно, что система находится в состоянии номер 8?

ТЕСТ

1. Любой алфавит обладает всеми перечисленными свойствами:
а) компактность, конечность, сжимаемость, строгость;
б) компактность, полнота, ясность, сжимаемость;
в) полнота, конечность, уникальность, информативность;
г) полнота, информативность, уникальность, универсальность.
2. Код, заданный на множестве A , — это соответствие:
а) слова алфавита A другому слову A ;
б) одного знака алфавита A двум и более знакам A ;
в) двух и более знаков A одному знаку A ;
г) знака алфавита A другому знаку A .

3. Формула Хартли имеет вид $H = \log_2 N$, где N — количество:
 - а) равновозможных (равновероятных) состояний системы;
 - б) разноразличных (разновероятных) состояний системы;
 - в) наиболее возможных (вероятных) состояний системы;
 - г) наименее возможных (вероятных) состояний системы.
4. Формула Шеннона (p_i — относительная частота i -го состояния системы):
 - а) $I = -\sum_{i=1}^n P_i \log_2 P_i$; б) $I = \sum_{i=1}^n P_i \log_2 P_i$; в) $I = -\sum_{i=1}^n P_i \ln P_i$; г) $I = \sum_{i=1}^n P_i \ln P_i$.
5. Независимо от предметной области информация бывает:
 - а) входная, выходная, внутренняя (внутрисистемная);
 - б) входная, выходная, сетевая, научная;
 - в) научно-техническая, бытовая, результирующая;
 - г) входная, выходная, результирующая.
6. По отношению к результату информация бывает:
 - а) исходная, постоянная, результирующая;
 - б) исходная, переменная, результирующая;
 - в) исходная, внутренняя, выходная;
 - г) исходная, промежуточная, результирующая.
7. Наибольшее число свойств информации дано в списке:
 - а) полнота, семантическая, открытость, массовость, связность;
 - б) полнота, массовость, достоверность, устойчивость, ценность;
 - в) полнота, ценность, модифицируемость, закрытость, адекватность;
 - г) полнота, массовость, адекватность, ценность, закрытость.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
в	г	а	а	а	г	б

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Информация как знание.
2. Информация как абстракция.
3. Информация как мера порядка и организации в системе.
4. Информация как мера структурированности системы.
5. Информация как уменьшение неопределенности в системе.
6. Способы и методы получения информации.
7. Информация, ее кодирование и защита.

И 3

ИНФОРМАЦИОННО-ЛОГИЧЕСКИЕ ЗАДАЧИ И МЕТОДЫ

Основная проблема информатики — построение, описание, исследование, хранение, актуализация объектов или процессов реального или идеального происхождения (знаний), установление соответствия или связей между ними.

Информационно-логическая задача — задача, в которой необходимо установить некоторые информационные или логические связи, определяющие поведение, эволюцию объекта, и сделать необходимые причинно-следственные логические выводы. Эти задачи возникают в различных областях и часто являются плохо формализованными и структурированными. Их нужно хорошо формализовать и структурировать. Насколько хорошо возможно будет это сделать, настолько хорошо и полно будет решена проблема. Часто это зависит от владения исследователем методами информатики, математики, технологий.

ПРИМЕР

Пусть имеются только 2 пустых ведра емкостью 3 и 7 л. Нужно налить в ведро 5 л воды из реки за наименьшее число операций — переливаний из одного ведра в другое. Можно опорожнять ведро. Решение задачи — последовательность операций:

- 1) наполняем ведро емкостью 7 л полностью;
- 2) переливаем из большего ведра в меньшее;
- 3) опорожняем меньшее ведро для дальнейших операций;
- 4) переливаем из большего ведра в меньшее;
- 5) выливаем (опорожняем) меньшее ведро;
- 6) переливаем из большего ведра в меньшее;
- 7) наливаем полное большее ведро;
- 8) доливаем меньшее ведро из большего.

ПРИМЕР

В городах Нальчик, Москва, Серпухов, Пятигорск живут четыре супружеские пары, причем в каждом городе живет только одна супружеская пара. Имена этих супругов: Антон, Борис, Давид, Григорий, Ольга, Светлана, Мария, Екатерина. Антон живет в городе Нальчике, Борис и Ольга — супруги, Григорий и Светлана не живут в одном городе, Мария живет в Москве, Светлана — жительница Серпухова. Определим, кто на ком женат и кто где проживает. Составим, исходя из условия задачи, таблицу возможностей, отмечая знаком «+» возможные, а знаком «-» невозможные ситуации. Согласно условию задачи получаем следующую таблицу:

Город	Антон	Борис	Давид	Григорий	Ольга	Мария	Светлана	Екатерина
Нальчик	+	—	—	—	—	—	—	+
Москва	—	—	+	+	—	+	—	+
Серпухов	—	—	+	—	—	—	+	+
Пятигорск	—	+	+	+	+	—	—	+

Из анализа этой таблицы видно, что:

- 1) Антон и Екатерина — супруги и живут в Нальчике (единственно возможный вариант в Нальчике);
- 2) Борис и Ольга живут в Пятигорске;
- 3) Григорий и Мария — супруги и живут в Москве (единственно возможный вариант из разрешенных вариантов в Москве);
- 4) Светлана и Давид живут в городе Серпухове.

ПРИМЕР

В следующей записи сложения одна буква обозначает одну цифру (каждой букве соответствует одна цифра): $3BC + CBA = 603$. Найдем все эти цифры за наименьшее число рассуждений (одно рассуждение — одно утвердительное высказывание относительно одного разряда):

- 1) анализируя вторые разряды, замечаем, что $B = 5$; так как $B+B$ оканчивается на ноль, то либо $B = 0$ (это невозможно, так как A, B, C не равны нулю), либо $B = 5$ (из младшего разряда не было переноса);
- 2) так как $B = 5$, то из второго разряда в старший разряд переносится единица, т. е. $C = 2$;
- 3) из младшего разряда заключаем, что $A = 1$.

Решение найдено за три шага.

ПРИМЕР

Петя, Вася, Дима, Коля и Гена хотят (каждый отдельно) пойти в кинотеатры, где они не были. При этом Петя не был в «Победе», «Востоке», «Авроре», «Мире»; Вася был в «Авроре», «Мире»; Дима не был лишь в «Авроре»; Коля не был в «Победе», «Мире»; Гена был в «Востоке», «Мире», «Юности». Найдем, кто и куда должен пойти (в перечисленные кинотеатры). За одно рассуждение принять одно повествовательное предложение относительно кинотеатра или юноши. Эту задачу решим для разнообразия без составления таблицы. Первое рассуждение: Дима не был только в «Авроре», поэтому он идет в «Аврору». Второе рассуждение: так как Гена не был только в «Авроре» и «Победе», то он идет в «Победу». Третье рассуждение: так как Коля не может уже идти в «Победу», то он идет в «Мир». Четвертое рассуждение: так как Петя не может пойти уже в «Победу», «Аврору» или «Мир», то он идет в «Восток». Пятое рассуждение: Васе однозначно остается кинотеатр «Юность».

Для решения инфологических задач необходимо хорошо владеть методами получения и использования информации. Эти методы можно разделить на три группы, часто перекрываемые друг другом, особенно во время решения практических проблем.

1. **Эмпирические методы**, или методы получения эмпирической информации (эмпирических, опытных данных).
2. **Теоретические методы**, или методы получения теоретической информации (построение теорий, получение теоретических знаний).
3. **Эмпирико-теоретические методы** (смешанные, полуэмпирические), или методы получения эмпирико-теоретической информации.

Охарактеризуем кратко основные эмпирические методы:

1. **Наблюдение** — сбор первичной информации о системе.
2. **Сравнение** — установление общего и различного в системе.
3. **Измерение** — нахождение эмпирических законов, фактов для системы.
4. **Эксперимент (простой)** — целенаправленное преобразование системы.

Кроме этих классических форм их реализации, в последнее время используются и такие формы, как опрос, интервью, тестирование, классификация и др.

Охарактеризуем основные эмпирико-теоретические методы:

1. **Абстрагирование** — установление общих свойств системы, процесса, замещение их моделью.
2. **Анализ** — разъединение системы, процесса на связанные части (подсистемы) с целью выявления их взаимосвязей.
3. **Синтез** — соединение подсистем в систему, процесс с целью выявления их взаимосвязей.
4. **Индукция** — получение знания о системе, процессе на основе знаний о подсистемах.
5. **Дедукция** — получение знания о подсистемах на основе знаний о системе, процессе.
6. **Аналогия** — получение знания о системе, процессе с помощью перехода от сходства одних сторон, свойств, связей к другим.
7. **Моделирование, использование приборов** — получение знания о системе, процессе с помощью модели или приборов.

В последнее время часто используются и такие формы, как мониторинг (система наблюдений и анализа состояний системы), деловые игры и ситуации, экспертные оценки, совещания и конференции, имитации (имитационная процедура) и др.

Охарактеризуем кратко основные теоретические методы:

1. **Идеализация** — получение знаний о системе, процессе путем мысленного конструирования, представления в мышлении систем или подсистем, не существующих в действительности.
2. **Формализация** — получение знаний о системе, процессе с помощью знаков или формул, языков искусственного происхож-

дения, например языка математики (или математическое, формальное описание, представление).

3. **Аксиоматизация** — получение знаний о системе, процессе с помощью некоторых, специально для этого сформулированных аксиом и правил вывода из этой системы аксиом.
4. **Виртуализация** — получение знаний о системе, процессе созданием особой среды, обстановки, ситуации, которую реально без этой среды невозможно реализовать и без которой нельзя получить соответствующие знания.
5. **Ассоциация** — получение знаний о системе, процессе путем связывания некоторых процедур формирования идей.

Методы обычно применяются многоуровневым образом, и можно предложить схему, представленную на рисунке 1.

ПРИМЕР

В приведенной ниже последовательности чисел 2, 8, 48, 384, ... каждый элемент получен по строгому правилу. Найдем и опишем это правило, используя методы наблюдения, сравнения, анализа и синтеза, абстрагирования, индукции и дедукции, формализации. Нетрудно логически определить, что каждое новое число a_n — произведение всех первых четных чисел до $2n$ -го, где n — номер рассматриваемого числа: $2 = 2$, $8 = 2 \cdot 4$, $48 = 2 \cdot 4 \cdot 6$, $384 = 2 \cdot 4 \cdot 6 \cdot 8$. В общем случае имеет место формула вида $a_n = 2 \cdot 4 \cdot 6 \cdot \dots \cdot (2n - 2) \cdot (2n)$. Это число называется факториалом четного числа и обозначается $(2n)!!$.

ПРИМЕР

Друзья X, Y, Z, U, V должны поехать в разные города $A, B, B, Г, Д, Е$. При этом X может ехать только в $A, Б, Д$; Y может ехать только в $A, Б$ и B ; Z может ехать только в B ; U не может ехать никуда, куда может ехать Y ; V не может ехать только в $Д$ и $Е$. Необходимо определить, в каком городе мог

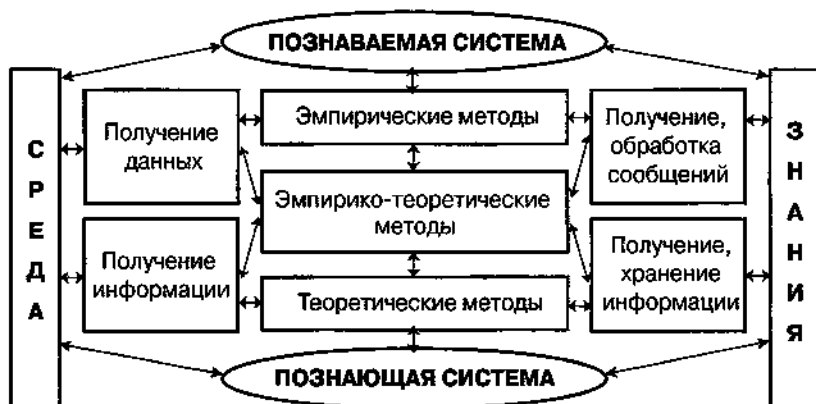


Рис. 1. Структура познания системы

быть каждый из них, если вдвоем они не были ни в одном городе. Решение необходимо найти за минимальное число рассуждений. Одно рассуждение — утверждение об одном городе или о человеке. Определим также, каковы оценки возможности поездки в города каждого, если под возможностью понимать вероятность поездки в этот город или относительную частоту поездок в город. Составим таблицу возможностей каждого, пометив города, куда может ехать каждый, знаком «+», а в противном случае — знаком «-». Получим

	X	Y	Z	U	V
A	+	+	-	-	+
B	+	+	-	-	+
B	-	+	+	-	+
Г	-	-	-	+	+
Д	+	-	-	+	-
E	-	-	-	+	-

Рассмотрим два случая: а) в E кто-то был и б) в E никого не было. Для случая а из анализа этой таблицы (т. е. условия задачи) следует: 1) U должен ехать в E (иначе в E никто не поедет); 2) Z может ехать только в B; 3) Y должен ехать только в Г (иначе в Г никто не едет); 4) X должен ехать только в Д (так как U в Д уже не может ехать); 5) V может ехать только в A или в B (т. е. задача имеет два решения). Аналогично рассматривается случай б. Вероятности

	X	Y	Z	U	V
A	1/3	1/3	0	0	1/4
B	1/3	1/3	0	0	1/4
B	0	1/3	1	0	1/4
Г	0	0	0	1/3	1/4
Д	1/3	0	0	1/3	0
E	0	0	0	1/3	0

поездок в города вначале можно изобразить таблицей вида

Определите самостоятельно основные методы получения информации, использованные при решении этой задачи.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Какая задача называется информационно-логической?
2. Как может быть решена информационно-логическая задача?
3. Каковы основные типы методов получения информации?
4. Каковы эмпирические методы получения информации?
5. Каковы теоретические методы получения информации?
6. Каковы эмпирико-теоретические методы получения информации?
7. Охарактеризуйте по одному методу актуализации информации каждого типа.

ЗАДАЧИ

1. У Петрова, Иванова, Семенова и Николаева в фамилиях содержится часть (корневая) их имен, причем только у Николаева они совпадают, т. е. Николаева зовут Николай. Семенова зовут не Петр. Укажите фамилии и имена каждого за минимум утвердительных высказываний.
2. Задан квадрат $ABCD$ со стороной 10 см, около которого описана окружность и в который окружность вписана. Вписанная окружность разделена на 4 равные части. Как раскрасить тремя цветами все полученные части описанной окружности, чтобы никакие две соседние части не были окрашены в одинаковый цвет? Достаточно ли двух цветов? Как раскрасить части, если во вписанную окружность вписать другой квадрат?
3. Определите неизвестные десятичные цифры, помеченные знаком «*», если в нижеприведенной записи каждое слагаемое — осмысленное слово русского языка (начиная со второго): $\Pi + \Pi* + \Pi** + \Pi*** + \Pi**** = 60355$.
4. Имеются только вагоны по 7 и 11 т. Составьте два поезда из минимального числа вагонов для перевозки 241 т, чтобы число вагонов в каждом поезде было минимальным, причем количество вагонов каждого вида в новых поездах не должно превышать количества вагонов в исходном поезде. Ответ обоснуйте за минимальное число кратких и утвердительных рассуждений.
5. Некоторые вкладчики хотят сделать вклады, при этом Артем не согласен быть вкладчиком банков «МММ», «ННН», «ККК»; Николай согласен вкладывать только в банки «МММ», «ТТТ», «РРР»; Алексей согласен быть вкладчиком любого банка; Михаил — быть вкладчиком только банков «РРР» и «МММ»; Владимир — быть вкладчиком любого банка, кроме «ККК». Вкладчиками каких банков могут быть Николай, Владимир и Алексей? Определите понятие рассуждения и ответ обоснуйте за минимальное число рассуждений.
6. Имеются трубы длиной 3 и 5 м. Докажите, что можно построить водопровод в виде непересекающейся ломаной $ABCDEF$, где $AB = 19$ м, $BC = 22$ м, $CD = 26$ м, $DE = 31$ м, $EF = 84$ м. Можно ли уменьшить число стыков труб, если добавить трубы длиной 7 м? Укажите используемые при решении этой задачи методы получения информации.
7. Составьте и решите одну информационно-логическую задачу.

ТЕСТ

1. К эмпирическим методам получения информации относятся:
 - а) наблюдение, сравнение, индукция, дедукция;
 - б) наблюдение, эксперимент, индукция, дедукция;
 - в) наблюдение, сравнение, измерение, эксперимент;
 - г) наблюдение, сравнение, измерение, аналогия.

2. К эмпирико-теоретическим методам получения информации относятся:
 - а) сравнение, анализ, синтез, индукция, эксперименты;
 - б) анализ, синтез, индукция, дедукция, идеализация;
 - в) анализ, синтез, эксперимент, исторический, логический;
 - г) анализ, синтез, аналогия, дедукция, индукция.
3. К теоретическим методам получения информации относятся:
 - а) анализ, синтез, идеализация, формализация, аналогия;
 - б) анализ, аналогия, формализация, аксиоматизация, идеализация;
 - в) идеализация, формализация, аксиоматизация, виртуализация, ассоциация;
 - г) анализ, синтез, идеализация, формализация, ассоциация.
4. Анализ — это метод поиска информации о системе с помощью:
 - а) использования аналитических записей;
 - б) разделения системы на подсистемы и выявления их связей;
 - в) соединения подсистем в систему и выявления их связей;
 - г) методов математического анализа.
5. Синтез — метод поиска информации о системе с помощью:
 - а) разделения системы на подсистемы и выявления их связей;
 - б) синтетических материалов;
 - в) соединения подсистем в систему и выяснения их взаимосвязей;
 - г) наблюдений и опытных данных.
6. Индукция — метод поиска информации о системе на основе:
 - а) знаний о подсистемах;
 - б) знаний о системе;
 - в) рекуррентных зависимостей;
 - г) частных заключений.
7. Дедукция — метод поиска информации о системе с помощью:
 - а) разделения системы на подсистемы и поиска их связей;
 - б) деструктивных методов анализа;
 - в) общих заключений;
 - г) знаний о системе.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
в	г	в	б	в	а	г

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Информационно-логические задачи и методы их исследования.
2. Классические эмпирические методы получения информации.
3. Классические теоретические методы получения информации.
4. Классические эмпирико-теоретические методы получения информации.
5. Неклассические эмпирические методы получения информации.
6. Неклассические теоретические методы получения информации.
7. Неклассические эмпирико-теоретические методы получения информации.

СИСТЕМЫ СЧИСЛЕНИЯ

Алфавит X из p символов и правила записи (изображения) и обработки чисел с помощью символов этого алфавита называются **системой счисления** с основанием p . Число x в системе с основанием p обозначается как $(x)_p$ или x_p .

Все системы счисления строятся по общему принципу: выбирается p — основание системы, а любое число x записывается в виде комбинации степеней веса p от 0 до n -й степени, т. е. $(x)_{10} = x_n p^n + x_{n-1} p^{n-1} + \dots + x_1 p^1 + x_0 p^0$.

ПРИМЕР

$$1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 1 = 13_{10},$$

$$157_8 = 1 \cdot 8^2 + 5 \cdot 8^1 + 7 \cdot 8^0 = 64 + 40 + 7 = 111_{10},$$

$$A6F_{16} = 10 \cdot 256 + 6 \cdot 16 + 15 \cdot 1 = 2671_{10}.$$

В большинстве систем вес цифры, символа алфавита зависит от их места в записи числа или слова. Такая система называется **позиционной**, в противном случае она называется **непозиционной**.

ПРИМЕР

Непозиционная система — римская система записи чисел с алфавитом вида $X = \{I (1), V (5), X (10), L (50), C (100), D (500), M (1000)\}$, где в скобках указаны веса символов (не зависящие от позиции символа). Примеры римских чисел: III (3), IV (4), V (5), VI (6), IX (9), XI (11), DCL (650).

Для изображения десятичных дробей используется подобная формула разложения по степеням основания.

ПРИМЕР

$$110,001_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 6,125_{10}; \quad 510$$

$$A, B_{16} = A \cdot 16^0 + B \cdot 16^{-1} = 10 \cdot 1 + 11 \cdot 0,0625 = 10,6875_{10}.$$

Процедура перевода десятичных чисел в p -ю систему счисления:

- 1) перевести целую часть числа x , для чего последовательно делить сперва целую часть $[x]_{10}$, а затем все получаемые частные на p до тех пор, пока не получим в очередном частном число, меньшее p ; изображение $[x]_p$ получается последовательным приписыванием к последнему частному остатков от деления — от последнего до первого;
- 2) перевести мантиссу числа, т. е. $\{x\}_{10}$, для чего последовательно умножать сначала исходную мантиссу, а затем мантиссы получаемых чисел на p до тех пор, пока не получим мантиссу, равную нулю, или нужное количество цифр; изображение $\{x\}_p$ получается приписыванием к целой части первого произ-

ведения второй такой же цифры и т. д. — до последней цифры целой части;

3) общий результат будет иметь вид $(x)_p = [x]_p, \{x\}_p$.

ПРИМЕР

Найти: $12,8_{10} = ?_2$. Решение найдем по частям:

1) Переводим вначале целую часть числа $12_{10} : 12_{10} = 1100_2$.

2) Переводим теперь мантиссу числа 0,8 (снизу подчеркнуты цифры, идущие в изображение мантиссы в двоичной системе): $0,8 \cdot 2 = 1,6$; $0,6 \cdot 2 = 1,2$; $0,2 \cdot 2 = 0,4$; $0,4 \cdot 2 = 0,8$; $0,8_{10} = 0,1100110\dots_2$, т. е. перевод точно невозможно осуществить, результат — бесконечное периодическое число.

3) Записываем результат: $12,8_{10} = 1100,1100110011\dots_2 = 1100,(1100)_2$.

ПРИМЕР

Найдем $29,25_{10} = ?_8$: 1) $29_{10} = 35_8$; 2) $0,25_{10} = 0,2_8$; 3) $29,25_{10} = 35,2_8$.

Найдем $79,26_{10} = ?_{16}$: 1) $79_{10} = 4F_{16}$; 2) $0,26_{10} = 0,40_{16}$; 3) $79,26_{10} = 4F,4_{16}$.

При переводе дробной части мы ограничились пахождением двух значащих цифр после запятой, ибо точно перевод сделать невозможно.

Для перевода из 2-й системы в 8-ю и наоборот, из 2-й в 16-ю и наоборот, из 8-й в 16-ю и наоборот используется таблица вида

ОСНОВАНИЕ СИСТЕМЫ							
10	2	8	16	10	2	8	16
0	0	000	0000	8	—	—	1000
1	1	001	0001	9	—	—	1001
2	—	010	0010	10	—	—	1010
3	—	011	0011	11	—	—	1011
4	—	100	0100	12	—	—	1100
5	—	101	0101	13	—	—	1101
6	—	110	0110	14	—	—	1110
7	—	111	0111	15	—	—	1111

При переводе в 8-ю систему или из нее необходимо группировать в тройки биты, а при переводе в 16-ю или из нее необходимо группировать их в четверки битов. Можно добавлять, если нужно, незначащие нули (слева от целой части и справа от мантиссы) или отбрасывать их.

ПРИМЕР

Рассмотрим переводы в смешанных системах:

а) из 2-й системы в 8-ю (двоично-восьмеричное изображение):

$$101,10111_2 = \overbrace{101,101110}^{5_8 \quad 5_8 \quad 6_8} = 5,56_8;$$

б) из 8-й системы в 2-ю (восьмерично-двоичное изображение):
 $6, 24_8 = \underbrace{6}_2, \underbrace{2}_2, \underbrace{4}_2 = 110, 0101_2$;
 $110_2 \ 010_2 \ 100_2$

в) из 2-й системы в 16-ю (двоично-шестнадцатеричное изображение): $101, 10111_2 = \underbrace{0101}_{5_{16}}, \underbrace{10111000}_{11(B)_{16}} = 5, B8_{16}$;
 $5_{16} \ 11(B)_{16} \ 8_{16}$

г) из 16-й системы в 2-ю (шестнадцатерично-двоичное изображение): $1A, F3_{16} = \underbrace{1}_{0001_2}, \underbrace{A}_{1010_2}, \underbrace{F}_{1111_2}, \underbrace{3}_{0011_2} = 11010, 11110011_2$;

д) из 8-й системы в 16-ю (восьмерично-шестнадцатеричное изображение): $15, 7_8 = \underbrace{1}_{001_2}, \underbrace{5}_{101_2}, \underbrace{7}_{111_2} = \underbrace{D}_{1101_2}, \underbrace{E}_{1110_2} = D, E_{16}$;

е) из 16-й системы в 8-ю (шестнадцатерично-восьмеричное изображение): $1A, B_{16} = \underbrace{1}_{0001_2}, \underbrace{A}_{1010_2}, \underbrace{B}_{1011_2}, \underbrace{0}_{011_2}, \underbrace{1}_{101_2}, \underbrace{0}_{100_2} = 11010, 1011011_2$;

Сложение в двоичной системе счисления осуществляется по правилам: $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, $1 + 1 = 2_{10} = 10_2$.

В последнем случае цифра 1 учитывается в старшем разряде.

Каждый раз, когда при сложении получается сумма, большая основания системы счисления, необходимо сделать перенос в следующий разряд.

Таблица вычитания в двоичной системе счисления имеет вид

$$0 - 0 = 0, \ 1 - 0 = 1, \ 1 - 1 = 0, \ 0 - 1 = 10 - 1 = 1.$$

В этом случае цифру 1 ($10_2 = 2_{10}$) заняли у старшего разряда.

Таблица умножения в двоичной системе счисления имеет вид

$$0 \cdot 0 = 0, \ 0 \cdot 1 = 0, \ 1 \cdot 0 = 0, \ 1 \cdot 1 = 1.$$

Таблица деления в двоичной системе счисления имеет вид

$$0 : 0 = 1, \ 1 : 0 = 1, \ 0 : 1 = 0, \ 1 : 1 = 1,$$

где знак «!» означает невыполнимость операции деления.

Обратным кодом числа в системе с основанием p называется число в этой системе, получаемое заменой цифры или символа в каждом разряде числа на его дополнение до максимальной цифры в системе (т. е. до количества $p - 1$).

Дополнительный код = обратный код + единица в младшем разряде.

ПРИМЕР

а) $10011 \Rightarrow$ двоичное число, $01101 \Rightarrow$ дополнительный код;

б) $457 \Rightarrow$ восьмеричное число, $321 \Rightarrow$ дополнительный код;

в) $A9 \Rightarrow$ шестнадцатеричное число, $57 \Rightarrow$ дополнительный код.

Вычитание с помощью дополнительного кода: найти дополнительный код вычитаемого (разрядности, как у уменьшаемого) и сложить этот код с уменьшаемым; результат — сумма без старшего разряда (отбрасывается).

ПРИМЕР

Выполним вычитание напрямую и через дополнительный код:

$$\begin{array}{r} - 110110_2 \\ \underline{10101_2} \\ 100001_2 \end{array} \quad \begin{array}{r} + 110110_2 \\ \underline{101011_2} \\ 1100001_2 \end{array}$$

ПРИМЕР

Примеры арифметических действий в системах счисления:

А. Сложение.

$$\begin{array}{r} + 3209_{10} \\ \underline{8051_{10}} \\ 11260_{10} \end{array} \quad \begin{array}{r} + 10011_2 \\ \underline{1010_2} \\ 11101_2 \end{array} \quad \begin{array}{r} + 165_8 \\ \underline{273_8} \\ 460_8 \end{array} \quad \begin{array}{r} + A56_{16} \\ \underline{BB7_{16}} \\ 160D_{16} \end{array}$$

В. Вычитание (обычное вычитание и через дополнительный код).

$$\begin{array}{r} - 527_{10} \\ \underline{136_{10}} \\ 391_{10} \end{array} \quad \begin{array}{r} + 527_{10} \\ \underline{864_{10}} \\ 1391_{10} \end{array} \quad \begin{array}{r} - 1101_2 \\ \underline{1100_2} \\ 1_2 \end{array} \quad \begin{array}{r} + 1101_2 \\ \underline{0100_2} \\ 10001_2 \end{array}$$

$$\begin{array}{r} - 176_8 \\ \underline{67_8} \\ 107_8 \end{array} \quad \begin{array}{r} + 176_8 \\ \underline{711_8} \\ 1107_8 \end{array} \quad \begin{array}{r} - AB5_{16} \\ \underline{77A_{16}} \\ 33B_{16} \end{array} \quad \begin{array}{r} + AB5_{16} \\ \underline{886_{16}} \\ 133B_{16} \end{array}$$

С. Умножение.

$$\begin{array}{r} \times 23_{10} \\ \underline{45_{10}} \\ + 115 \\ \underline{92} \\ 1035_{10} \end{array} \quad \begin{array}{r} \times 1011_2 \\ \underline{11_2} \\ + 1011 \\ \underline{1011} \\ 100001_2 \end{array} \quad \begin{array}{r} \times 37_8 \\ \underline{25_8} \\ + 133 \\ \underline{76} \\ 1113_8 \end{array} \quad \begin{array}{r} \times A5_{16} \\ \underline{2C_{16}} \\ + 7BC \\ \underline{28A} \\ 305C_{16} \end{array}$$

Д. Деление.

$$\begin{array}{r} - 11101_2 \mid 101_2 \\ \underline{101} \\ 1001 \\ \underline{101} \\ 1000 \\ \underline{101} \\ 110 \\ \underline{101} \\ \vdots \end{array}$$

Деление дает бесконечную дробь, поэтому, ограничиваясь, например, четырьмя разрядами в частном, получаем результат $101,1100_2$. Это число в десятичной системе равно 5,75. Следовательно, мы получили результат с точностью до четвертого разряда: $29:5 = 5,75$. При делении указывают либо точность результата, либо количество разрядов в мантиссе результата. Наибольшее целое n -разрядное число, которое можно записать в системе счисления с основанием p , равно:

$$x_p^{\max} = \sum_{i=0}^{n-1} (p-1)p^i = \sum_{i=0}^{n-1} p^{i+1} - \sum_{i=0}^{n-1} p^i = p^n - 1.$$

Наименьшее целое n -разрядное число в этой системе равно:

$$x_{\min} = -x_p^{\max} = 1 - p^n.$$

ПРИМЕР

Вычислим наибольшее и наименьшее 5-разрядное целое число в системе счисления с основанием 4. В системе счисления с осно-

ванием 4 и числом разрядов 5 представим диапазон: $-1023 = 1 - 4^5 \leq (x)_4 \leq 4^5 - 1 = 1023$. Например, для двоичной системы $x^{\max} = 111...111 = 2^n - 1$, $x_{\min} = 1 - 2^n$, а для восьмеричной системы счисления $x^{\max} = 777...777 = (7^n - 1)$, $x_{\min} = (1 - 7^n)$. Многоточие означает n разрядов.

ПРИМЕР

В саду 100 фруктовых деревьев — 14 яблонь и 42 груши. Определим, в какой системе счисления посчитаны деревья. Из условия задачи и переводом указанных чисел в систему счисления с пока неизвестным основанием p можно записать равенство вида $1 \cdot p^2 + 0 \cdot p^1 + 0 = 1 \cdot p^1 + 4 + 4 \cdot p^1 + 2$ или $p^2 - 5p - 6 = 0$. Из корней этого уравнения подходит в качестве основания системы только один: $-p = 6$. Деревья посчитаны в шестеричной системе.

ПРИМЕР

Число $x = 4840$ записано в системе счисления с основанием p и $\text{mod}(x, y) = 0$, где y — десятичное число 11. Найдем основание системы p , если $p < 31$. Так как число 4840 в системе счисления с основанием p можно записать в виде $4840 = 4p^3 + 8p^2 + 4p^1 = 4p(p^2 + 2p + 1) = 4p(p + 1)^2$ и по условию оно делится на простое число 11, то либо p , либо $p + 1$ делится на 11, т. е. $p = 11a$ или $p = 11a - 1$, где $a = 1, 2, \dots$. Например, при $p < 11$ единственным возможным решением будет $p = 10$ (при $a = 1$); при $p < 31$ — возможные решения $p = 10, 11, 21, 22$. Подумайте над решением задачи при $p > 31$.

История развития систем счисления достаточно интересна. Приведем лишь некоторые факты. Счет вначале велся с помощью пальцев рук (пятерками и затем десятками). В некоторых странах сохранился счет с основанием 12 (например, в Великобритании), 20 (например, во Франции — «quatre-vingts» или «четыре-двадцать», т. е. 80; у древних адыгов счет велся аналогично: «тощџиц», т. е. «двадцать-три» — 60) и др.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Как перевести числа из десятичной системы счисления в p -ю?
2. Какая система позиционная (непозиционная)? Приведите примеры.
3. Объясните переводы чисел из одной системы счисления в другую по схемам: $2 \Leftrightarrow 8$, $8 \Leftrightarrow 2$, $2 \Leftrightarrow 16$, $16 \Leftrightarrow 2$, $8 \Leftrightarrow 16$, $16 \Leftrightarrow 8$. Приведите примеры.
4. Как выполняют арифметические операции в p -й системе?
5. Что такое обратный код? Приведите примеры в различных системах.
6. Что такое дополнительный код? Приведите примеры.
7. Чему равно максимальное и минимальное целое число в системе счисления с основанием p ?

ЗАДАЧИ

1. Переведите число $95,61_{10}$ в двоично-восьмеричную, а также двоично-шестнадцатеричную системы. Ответ проверьте обратным переводом.
2. Переведите в восьмеричную систему число $2F, A5C_{16}$. Чему равно это число в десятичной системе? Найдите его дополнительный код.
3. Переведите в шестнадцатеричную систему число $25,31_8$. Чему равно это число в десятичной системе? Найдите его дополнительный код и переведите его в десятичную систему.
4. Вычислите выражение:
 $110,01_2 - 234,4_8 + FA, C8_{16} \cdot 22_{10} : 1,1_2 + 12,88_{10} : 11,010101_2 - F, A_{16}$.
5. В саду 1000 деревьев — 140 яблонь и 420 груш. В какой системе счисления посчитаны деревья? Найдите общее число деревьев в двоичном виде.
6. Имеются ящики: 4 черных, 3 красных, 2 желтых и 1 зеленый (посчитаны в десятичной системе). В каждом черном ящике находятся $(21)_p$ шара (шаров), в красном и желтом — $(23)_p$ шара, в зеленом — $(111)_p$ шара. Определите p , если всего было $(244)_p$ шаров.
7. Число $x = (176)_p$ (рассматриваемое в системе счисления с основанием p , $1 < p < 20$) делится нацело на $(7)_{10}$. Найдите (не перебирая) p и число $(x)_8$.

ТЕСТ

1. Разность восьмеричных чисел $152,71$ и $66,66$ равна восьмеричному числу: а) $64,13$; б) $63,03$; в) $64,03$; г) $63,13$.
2. Разность $A5,6A_{16} - F, C1_{16}$ равна шестнадцатеричному числу: а) $95,99$; б) $95,B9$; в) $95,A9$; г) $94,A9$.
3. Значение выражения $111,01_2 + 6,4_8 + B,9_{16}$ равно: а) $56,8_{10}$; б) $11001,0101_2$; в) $30,24_8$; г) $16,D1_{16}$.
4. Дополнительный код к двоичному числу 11011 равен двоичному числу: а) 11010 ; б) 00100 ; в) 00101 ; г) 11100 .
5. Сумма восьмеричных чисел $153,45$ и $44,55$ равна восьмеричному числу: а) $222,22$; б) $220,22$; в) $202,22$; г) $222,02$.
6. Сумма шестнадцатеричных чисел $A5,B$ и $CF,8$ равна шестнадцатеричному числу: а) $174,3$; б) $175,4$; в) $175,3$; г) $165,3$.
7. Разность двоичных чисел $100,101$ и $11,011$ равна двоичному числу: а) $11,01$; б) $1,01$; в) $10,01$; г) $1,11$.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
в	в	б	в	б	в	б

АЛГЕБРА ВЫСКАЗЫВАНИЙ И ПРЕДИКАТОВ

Информатика изучает информацию, сообщения или знаковые системы, последовательности. Алгебра — адекватный математический аппарат описания действий со знаковыми системами. Поэтому алгебраический аппарат наилучшим образом подходит для описания и формализации различных информационных систем. Информационные процессы лучше структурируются и формализуются с помощью алгебр, алгебраических структур.

Алгеброй A называется некоторая совокупность определенных элементов X с заданными над ними определенными операциями f (часто определяемые по сходству с операциями сложения и умножения чисел), которые удовлетворяют определенным свойствам — аксиомам алгебры.

Операция f называется n -местной, если она связывает n операндов (объектов, участников этой операции).

Утверждение (высказывательная форма) — основная единица, неделимая с точки зрения информации, семантического смысла знаний.

Высказывание — повествовательное утверждение, про которое можно однозначно сказать, что оно истинно или оно ложно. Эти два значения всевозможных высказываний обозначаются «истина» и «ложь», «true» и «false» или «1» и «0». Переменная, значениями которой могут быть лишь значения «1» или «0», называется логической переменной или булевой переменной.

ПРИМЕР

Рассмотрим словосочетания:

- 1) Москва — столица Российской Федерации.
- 2) Житель города Нальчика.
- 3) $5 - 9 + 8$
- 4) $5 - 9 + 8 = 4$
- 5) В пятую неделю зимы выпал снег.
- 6) На юге Африки живут пингвины.

Высказывание должно быть однозначно истинным или ложным, поэтому высказываниями являются только утверждения 1, 4, 6.

ПРИМЕР

Не является высказыванием и парадокс лжеца (Эвбулид, учитель Демосфена, около 350 лет до н. э.): «То, что я сейчас утверждаю, ложно», ибо если оно истинно, то оно ложно, а если допустить, что оно ложно, то оно истинно. Это неопределенная высказывательная форма. Аналогичный при-

мер принадлежит Геделю (1931): «То, что утверждается в этом предложении, не может быть доказано». Если его можно доказать, то его нельзя доказать, а если его можно опровергнуть, то его можно доказать. Предложения такого рода не могут быть доказаны или опровергнуты в пределах языка (теории, алгебры), с помощью которого они сформулированы.

Рассматривая высказывания, мы не обращаем внимания на их внутреннюю структуру и можем как разлагать их на структурные части, так и объединять их.

Предикат — выражение с логическими переменными, имеющее смысл при любых допустимых значениях этих переменных. Количество этих переменных в записи предиката называется его местностью.

Простые высказывания или предикаты не зависят от других высказываний или предикатов, а сложные зависят хотя бы от двух простых.

ПРИМЕР

Выражение « $x > y$ » — предикат, « $x > 5$ » — предикат, а « $7 > 5$ » — высказывание. Утверждение «Ясная погода» не является высказыванием.

Логической функцией $f(x)$ называется функция, у которой аргумент x — логическая переменная, а значения функции из множества $R(f) = \{1, 0\}$.

ПРИМЕР

Заданы предикаты вида p = «число x делится нацело на 5» и q = « y — день недели». Найдем множество истинности предикатов p и q , если $x \in \{1, 4, 6, 16, 20, 26, 30, 35\}$, $y \in \{\text{первый, вторник, среда, 1999, зима, выходной, праздник, воскресенье}\}$. Получаем, что $D(p) = \{20, 30, 35\}$, $D(q) = \{\text{вторник, среда, воскресенье}\}$.

Множество логических переменных $x, y \in X$ с определенными над ним операциями: $\bar{x}$ — отрицания или инверсии, $x \vee y$ — логического сложения или дизъюнкции, $x \wedge y$ — логического умножения или конъюнкции — называется алгеброй предикатов (и высказываний), если эти операции удовлетворяют аксиомам вида:

1. Аксиома инволюции: $\bar{\bar{x}} = x$.
2. Аксиомы коммутативности: $x \wedge y = y \wedge x$, $x \vee y = y \vee x$.
3. Аксиомы ассоциативности: $(x \wedge y) \wedge z = x \wedge (y \wedge z)$, $(x \vee y) \vee z = x \vee (y \vee z)$.
4. Аксиомы идемпотентности: $x \wedge x = x$, $x \vee x = x$.
5. Аксиомы поглощения: $x \wedge (x \vee y) = x$, $x \vee (x \wedge y) = x$.
6. Аксиомы дистрибутивности: $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.
7. Аксиомы де Моргана: $\overline{x \wedge y} = \bar{x} \vee \bar{y}$, $\overline{x \vee y} = \bar{x} \wedge \bar{y}$.
8. Аксиомы нейтральности: $x \wedge (y \vee \bar{y}) = x$, $x \vee (y \wedge \bar{y}) = x$.
9. Аксиома существования: двух особых элементов (называемых «единица: 1» и «нуль: 0»), причем $\bar{0} = 1$, $\bar{1} = 0$ и $\bar{x} \vee x = 1$, $\bar{x} \wedge x = 0$.

Из этих аксиом следует ряд других соотношений, например: $x \wedge 1 = x$, $x \vee 0 = x$, $x \vee 1 = 1$, $x \wedge 0 = 0$, $x \vee \bar{x} = 1$, $x \wedge \bar{x} = 0$.

Три базовые операции алгебры предикатов определяются таблицей их значений (в алгебре предикатов из-за дискретности значений логических функций часто используется табличная форма задания функции; булеву функцию n переменных можно полностью определить таблицей из 2^n строк).

x	y	$\bar{x}$	$\bar{y}$	$x \wedge y$	$x \vee y$
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

ПРИМЕР

Пусть на R заданы одноместный предикат $x = «a - 3 > 0»$ и двухместный предикат $y = «b + 4 < a»$. Запишем предикаты $\bar{x}$, $\bar{y}$, $x \wedge y$, $\bar{x} \vee \bar{y}$. Эти предикаты имеют вид

$\bar{x} = «a - 3 \leq 0»$, $\bar{y} = «b + 4 \geq a»$, $x \wedge y = «\begin{cases} a > 3 \\ a > b + 4 \end{cases}»$, $\bar{x} \vee \bar{y} = «a \leq 3$
или $a \leq b + 4»$.

Таблица всех значений логической функции называется таблицей истинности функции.

ПРИМЕР

Составим таблицу истинности функции $z = \overline{x \wedge (\bar{x} \vee y)}$. Она имеет вид

x	y	$\bar{x}$	$\bar{x} \vee y$	$z = x \wedge (\bar{x} \vee y)$	$z = \overline{x \wedge (\bar{x} \vee y)}$
0	0	1	1	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	1	0	1	1	0

ПРИМЕР

Заполним таблицу истинности трехместной логической функции $w = x \vee y \wedge z$. Эта таблица имеет вид

x	y	z	$y \wedge z$	w
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	1
0	1	1	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

ПРИМЕР

Изобразим множество истинности двухместного предиката вида $p(x, y) = \text{«модуль числа } x \text{ равен модулю числа } y\text{»}$, если задана область изменения аргументов: $x, y \in [0; 1]$. Имеем $E(p) = \{(x, y) : |x| = |y|\} = \{(x, y) : (x = y) \vee (x = -y)\} = \{(x, y) : x = y\} \cup \{(x, y) : x = -y\} = E(p_1) \cup E(p_2)$. Смысл предикатов $p_1(x, y)$ и $p_2(x, y)$ очевиден. Множество изображается графиком $y = |x|$ (рис. 2).

При выполнении логических операций в компьютере они сводятся к поразрядному сравнению операндов, которые кодируются (представляются) битами. Сравнения могут быть точными как с целыми числами, так и с точностью до определенного числа разрядов, причем все биты при этом считаются значащими (даже те нулевые биты, которые приписаны слева, ибо мы здесь имеем дело не с кодированием числовых величин, а с кодированием логических величин). Эти операции достаточно быстро выполняются, так как сводятся к выяснению совпадения (несовпадения) битов. Логические операции выполняются ЭВМ быстро («почти аппаратно»), поэтому можно не «бояться» использования в программах сложных логических условий. «Бояться» можно логически и вычислительно сложных процедур.

В логических формулах определено старшинство операций, например скобки, отрицание, конъюнкция, дизъюнкция (остальные, не базовые операции пока не учитываем).

Всегда истинные формулы называют тавтологиями.

Логические функции эквивалентны, если совпадают их таблицы истинности (т. е. совпадают области определения и значения, а также сами значения функции при одних и тех же допустимых значениях переменных). Если это совпадение происходит на части множества допустимых значений, то формулы называются эквивалентными на этой части.

Задача упрощения логического выражения состоит в преобразовании его к более простому (по числу переменных, операций или операндов) эквивалентному выражению.

ПРИМЕР

Упростим: (аксиома дистрибутивности) $= ((x \vee \bar{x}) \wedge y) \vee (x \wedge \bar{y}) =$
(аксиома нейтральности) $= y \vee (x \wedge \bar{y}) =$ (аксиома дистрибутивности) $= (y \vee x) \wedge (y \vee \bar{y}) =$
 $= (y \vee x) \wedge 1 = y \vee x = x \vee y.$

Задача доказательства равенства двух логических выражений (функций) состоит в установлении эквивалентности этих функций на некотором множестве значений всех переменных, входящих в данную функцию.

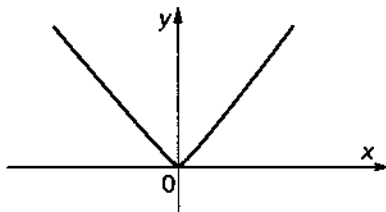


Рис. 2. График функции $y = |x|$

ПРИМЕР

Докажем равенство логических выражений $\overline{x \vee y} \vee \overline{x} \wedge y \vee \overline{x} = \overline{x}$. Используя аксиомы алгебры предикатов, получаем равенства $\overline{x \vee y} \vee \overline{x} \wedge y \vee \overline{x} = \overline{x} \wedge \overline{y} \vee \overline{x} \wedge y \vee \overline{x} = \overline{x} \wedge (\overline{y} \vee y) \vee \overline{x} = \overline{x} \wedge 1 \vee \overline{x} = \overline{x} \vee \overline{x} = \overline{x}$. Левая часть равенства приведена к правой части, т. е. равенство доказано.

Такие задачи решаются с помощью аксиом алгебры предикатов одним из способов:

- правая часть равенства приводится к левой части;
- левая часть равенства приводится к правой части;
- обе части равенства приводятся к третьему выражению.

Логические функции позволяют решать инфологические задачи.

ПРИМЕР

Брауну, Джонсу и Смиту предъявлено обвинение в соучастии в ограблении банка. В ходе следствия Браун сказал, что преступники были на синем «бьюике», Джонс сказал, что это был черный «крайслер», Смит утверждал, что это был «форд», но не синий. Они договорились запутать следствие и поэтому указывали неправильно либо марку, либо цвет автомобиля. Определим цвет и марку автомобиля. Рассмотрим простые высказывания вида x = «машина — синяя», y = «машина — бьюик», z = «машина — черная», u = «машина — крайслер», v = «машина — фورد». Высказывание Брауна можно записать в виде сложного логического выражения вида $\overline{x} \wedge y$, высказывание Джонса — $z \wedge u$, высказывание Смита — $\overline{x} \wedge v$. Так как в каждом из этих выражений одна из переменных принимает значение 1 («истина»), то истинны и дизъюнкции вида $x \vee y = 1$, $z \vee u = 1$, $\overline{x} \vee v = 1$. По определению конъюнкции $(x \vee y) \wedge (z \vee u) \wedge (\overline{x} \vee v) = 1$. Упростим выражение:

$$\begin{aligned} 1 &= (x \vee y) \wedge (z \vee u) \wedge (\overline{x} \vee v) = (x \wedge z \vee y \wedge z \vee x \wedge u \vee y \wedge u) \wedge (\overline{x} \vee v) = \\ &= x \wedge z \wedge \overline{x} \vee y \wedge z \wedge \overline{x} \vee x \wedge u \wedge \overline{x} \vee y \wedge u \wedge \overline{x} \vee x \wedge z \wedge v \vee y \wedge z \wedge \\ &\quad \wedge v \vee x \wedge u \wedge v \vee y \wedge u \wedge v = \\ &= 0 \vee y \wedge z \wedge \overline{x} \vee 0 \vee 0 \vee 0 \vee 0 \vee 0 = y \wedge z \wedge \overline{x}. \end{aligned}$$

Здесь было использовано то, что одновременно не могут быть истинными два высказывания относительно цвета или два высказывания относительно марки машины. Так как конъюнкция принимает значение «истина» только тогда, когда $y = 1$, $z = 1$, $\overline{x} = 1$, то заключаем, что автомобиль был черным «бьюиком».

ЭВМ, ее электронные блоки состоят из сотен тысяч вентилях (логических устройств, базовых схем), объединяемых по правилам и законам (аксиомам) алгебры вентилях в схемы, модули. Вентиль — атом, из которого состоит ЭВМ. Логические схемы предназначены для реализации различных функций алгебры логики и реализуются с помощью трех базовых логических элементов (вентилей, логических схем или релейно-контактных схем). Они вос-

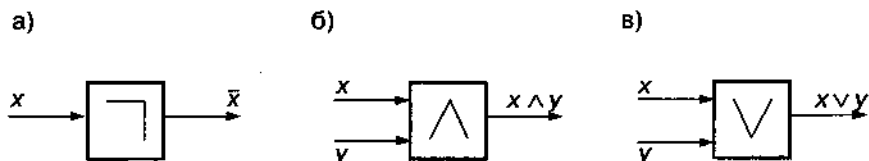


Рис. 3. Схемы базовых вентилях

производят функции полупроводниковых схем. Работу вентиляхных схем мы, как и принято, будем рассматривать в двоичной системе.

Функции отрицания, дизъюнкции и конъюнкции реализуют соответственно логические схемы, называемые инвертором, дизъюнктом и конъюнктом. Условные изображения схем инвертора (а), дизъюнктора (б) и конъюнктора (в) приведены на рисунке 3.

ПРИМЕР

Транзисторные схемы, соответствующие логическим схемам: — (инвертор), $\vee$ (дизъюнктор), $\wedge$ (конъюнктор), показаны на рисунке 4 (с. 44).

Из базовых логических элементов собирают, конструируют сложные логические схемы ЭВМ: сумматоры, шифраторы, дешифраторы и др. Большие (БИС) и сверхбольшие (СБИС) интегральные схемы содержат (на кристалле кремния площадью в несколько квадратных сантиметров) десятки тысяч вентилях. Это возможно еще и потому, что базовый набор схем является функционально полным, а представление логических констант в них одинаково (одинаковы электрические сигналы, представляющие 1 и 0), т. е. входы и выходы схем можно соединять, присоединяя одну схему к другой и «вкладывая» схемы друг в друга. Так образуются узлы ЭВМ — ячейки памяти, регистры, шифраторы, дешифраторы и т. д.

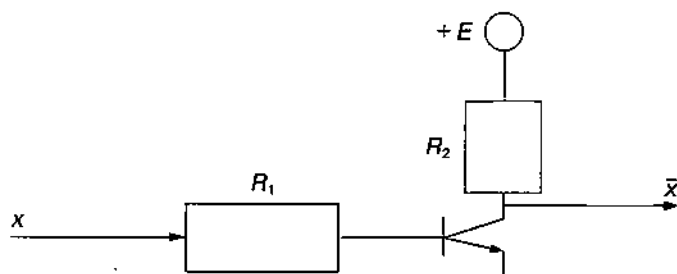
ПРИМЕР

В двоичной системе таблицу суммирования цифры x и цифры y и получения цифры z с учетом переноса p в некотором разряде чисел x и y можно изобразить так:

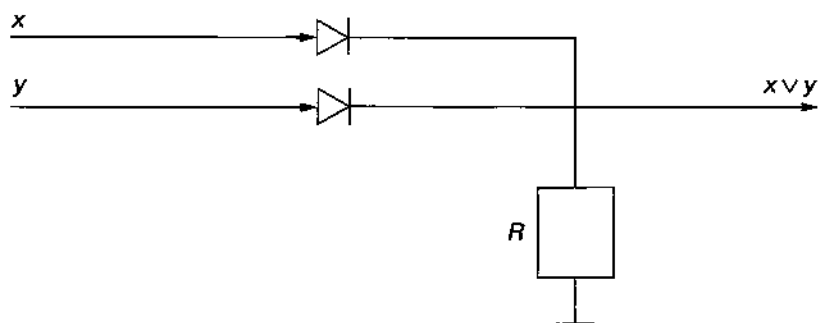
x	y	z	p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Эту таблицу можно интерпретировать как совместно изображаемую таблицу логических функций (предикатов) вида $z = \bar{x} \wedge$

а) инвертор



б) дизъюнктор



в) конъюнктор

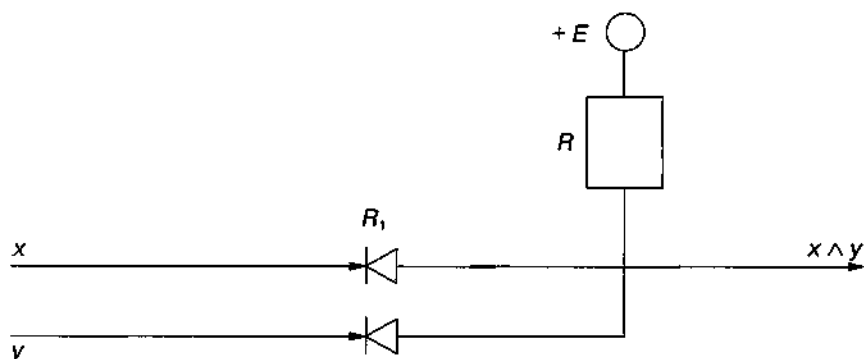


Рис. 4. Базовые транзисторные схемы

$\wedge y \vee x \wedge \bar{y}$, $p = x \wedge y$. Логический элемент, соответствующий этим функциям, называется **одноразрядным сумматором** и имеет следующую схему (обозначим ее как Σ или Σ_i , если мы хотим акцентировать i -й разряд; рис. 5).

Последовательный сумматор — комбинация одноразрядных сумматоров (рис. 6).

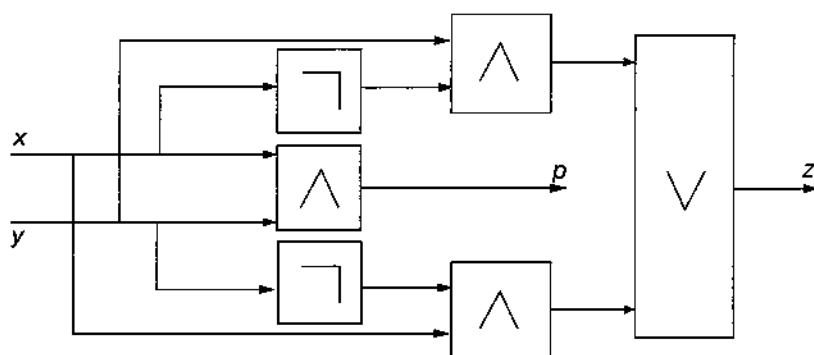


Рис. 5. Схема одноразрядного сумматора

ПРИМЕР

«Черным ящиком» называется устройство (схема и т. д.), содержимое которого неизвестно и может быть определено только по проявлениям входа-выхода ящика. В «черном ящике» находится логическая схема, которая в ответ на некоторую последовательность входных логических констант выдает последовательность логических констант, получаемых после выполнения логической схемы внутри «черного ящика». Определим логическую функцию внутри «черного ящика» (рис. 7), если операции выполняются с логическими константами для вход-

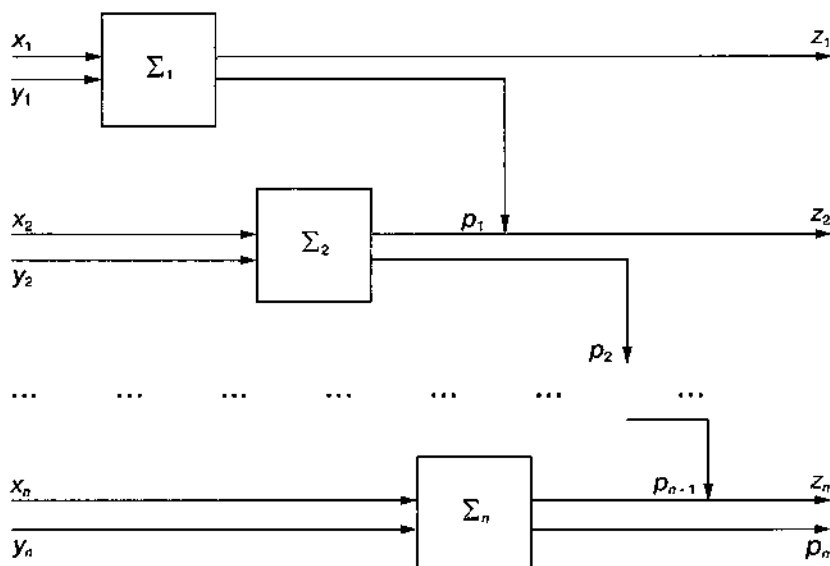


Рис. 6. Условная схема n -разрядного последовательного сумматора

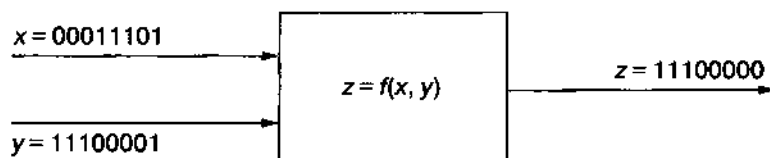


Рис. 7. Схема «черного ящика»

ных последовательностей (поразрядно). Из анализа входных значений x , y и поразрядного сравнения логических констант в этих сообщениях с константами в z — результате выполнения функции в «черном ящике» — видно, что подходит, например, функция вида $z = (x \vee y) \wedge \bar{x}$. Действительно, $x \vee y = 00011101 \vee 11100001 = 11111101$, $z = (x \vee y) \wedge \bar{x} = 11111101 \wedge 11100010 = 11100000$.

Важной задачей является минимизация числа вентилей для реализации той или иной схемы (устройства), что необходимо для более рациональной, эффективной реализации этих схем, для большей производительности и меньшей стоимости ЭВМ.

ПРИМЕР

Построим схему для функции $u = \overline{x \wedge y \wedge z \vee t}$. Схема показана на рисунке 8.

ПРИМЕР

Определим логическую функцию $u = f(x, y, z, t)$, реализуемую логической схемой (рис. 9).

Искомая логическая функция имеет вид $u = \overline{x \vee y} \vee (\bar{z} \wedge t)$.

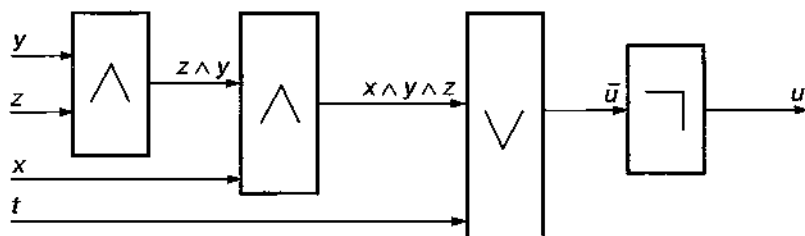


Рис. 8. Логическая схема для функции $u = \overline{x \wedge y \wedge z \vee t}$

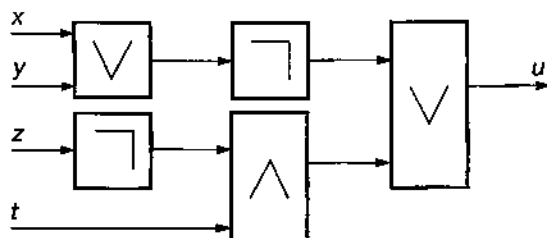


Рис. 9. Логическая схема для функции $u = \overline{x \vee y} \vee (\bar{z} \wedge t)$

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое алгебра? Приведите примеры алгебр.
2. Что такое высказывание, предикат, логическая функция? Приведите примеры. Чем они отличаются?
3. Каковы аксиомы алгебры предикатов? Какие из них имеют аналоги в алгебре чисел (арифметике), а какие нет?
4. Какие логические функции эквивалентны? Приведите примеры.
5. В чем суть задачи упрощения логического выражения?
6. Что такое логическая схема, вентиль и для чего они применяются? Приведите примеры.
7. Как комбинируются базовые логические элементы в более сложные логические схемы?

ЗАДАЧИ

1. Упростите и составьте таблицу истинности функции $z = (\bar{x} \vee \bar{y} \vee x \wedge y) \wedge \bar{x}$.
2. Обсуждая свои возможности по поступлению в вуз, абитуриенты A, B, B высказали предположения: A — «Я не смогу поступить, а B поступит»; B — « B не поступит, а A поступит»; B — «Или я не поступлю, или B не поступит». После экзаменов выяснилось, что каждый высказал одно верное и одно ложное простое утверждение. Кто поступил в вуз, если поступил лишь один?
3. Подберите три функции, эквивалентные данной функции, но более простого вида (с меньшим числом операций и операндов): $z = x \vee y \wedge \bar{y} \wedge x \vee x \wedge x$.
4. Из указанных ниже функций отметьте (с обоснованием своих рассуждений) эквивалентные между собой функции:
 - а) $z = x \vee x \vee y \vee y \vee x \wedge y \vee x \vee y \vee y \vee x \wedge y$;
 - б) $u = x \vee x \vee y \vee y \vee x \wedge y$;
 - в) $w = x \vee x \vee y \vee y \vee x \wedge y$;
 - г) $s = x \vee y \vee x \wedge y \vee y \vee x \wedge y$;
 - д) $v = x \vee x \vee y \vee y \vee x \wedge y$.
5. Определите логическое выражение внутри «черного ящика» (рис. 10).

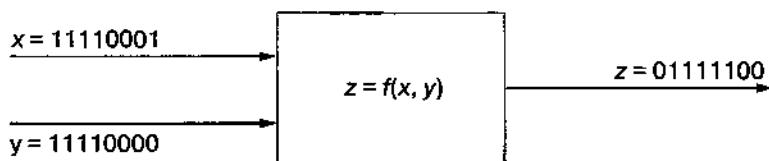


Рис. 10. Логическая схема для функции $u = \bar{x} \wedge y \wedge (\bar{z} \vee v)$

6. Постройте логические схемы, соответствующие функциям:
 а) $z = \overline{x \wedge y} \vee t \wedge \overline{x} \wedge y \vee t$; б) $z = x \vee y \wedge \overline{x} \vee t \vee x \wedge y$.
 Постройте эквивалентные логические схемы с минимумом инверторов, конъюнкторов, дизъюнкторов.
7. Определите логические функции, реализуемые схемами.

ТЕСТ

- Высказыванием не является выражение:
 а) «ясная погода»;
 б) «Нальчик — столица США»;
 в) «23 февраля 2002 г. было холоднее, чем 25 февраля 2002 г.»;
 г) « $2 + 2 = 4$ ».
- Одноразрядный сумматор $Z = X + Y$ с учетом переноса P можно определить логическими функциями:
 а) $Z = \overline{X} \wedge Y$, $P = X \wedge Y$; в) $Z = \overline{X} \wedge Y \vee X \wedge \overline{Y}$, $P = \overline{X} \wedge \overline{Y}$;
 б) $Z = \overline{X} \wedge Y \vee X \wedge Y$, $P = X \wedge \overline{Y}$; г) $Z = \overline{X} \wedge Y \vee X \wedge \overline{Y}$, $P = X \wedge Y$.
- Укажите множество истинности $D(p)$ предиката $p = «x$ не простое число», если $x \in \{1, 6, 10, 7, 17, 4, 9, 100\}$:
 а) $D(p) = \{1, 9, 7\}$; в) $D(p) = \{6, 10, 4, 9, 100\}$;
 б) $D(p) = \{100, 6, 10\}$; г) $D(p) = \{1, 6, 10, 4, 9, 100\}$.
- Область, нестрого (включая границы) заключенная между окружностями $x^2 + y^2 = 4$ и $x^2 + y^2 = 9$, описывается предикатом:
 а) $(x^2 + y^2 = 4)$ или $(x^2 + y^2 = 9)$; в) $(x^2 + y^2 \geq 4)$ и $(x^2 + y^2 \leq 9)$;
 б) $(x^2 + y^2 = 4) \cdot (x^2 + y^2 = 9)$; г) $(x^2 + y^2 > 4)$ или $(x^2 + y^2 < 9)$.
- Выражение $\overline{x} \wedge y \vee z \vee \overline{y} \wedge z \vee y$ эквивалентно выражению:
 а) $\overline{x} \wedge y \vee z \vee y$; б) $z \vee y \vee \overline{x} \wedge y$; в) $x \vee x \wedge y$; г) 1.
- Предикат $x \vee y \vee \overline{x} \wedge y \vee x$ равен:
 а) 0; б) 1; в) x ; г) y .
- Предикат $\overline{x \vee y} \vee (\overline{x} \wedge (y \vee x \wedge \overline{y}) \vee x)$ равен:
 а) 1; б) 0; в) x ; г) y .

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
а	г	в	в	г	б	а

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

- Джордж Буль и его необыкновенная алгебра.
- Алгебра контактно-релейных схем (переключательных схем).
- Алгебра логики как теории доказательств и рассуждений.
- Развитие логических систем (учений) от Аристотеля.
- Тавтологии, силлогизмы и парадоксы.
- Логические элементы ЭВМ.
- Логика и ее использование при доказательстве прямых, обратных и противоположных утверждений.

А

Б

АЛГОРИТМЫ. АЛГОРИТМЫ РАБОТЫ С ЧИСЛАМИ

Алгоритмы и алгоритмизация (часто называемые алгоритмикой) — основной и наиболее важный раздел информатики, обучающий постановке и алгоритмическому решению задач.

Алгоритм — система точных и полных предписаний, задающих действия (описывающих как их содержание, так и порядок), которые должен выполнить исполнитель алгоритма (человек, автомат) для нахождения решения любой задачи из рассматриваемого класса задач.

Указанный класс задач должен иметь общий метод решения. В этом смысле алгоритм есть описание реализации некоторого метода решения задач из определенного класса задач. Двум различным методам решения одного и того же класса задач могут соответствовать различные алгоритмы.

ПРИМЕР

Квадратное уравнение можно решить с помощью дискриминанта (один алгоритм) или же с помощью теоремы Виета — сведением к системе уравнений (совсем другой алгоритм).

Алгоритм удовлетворяет основным свойствам алгоритмов:

1. **Дискретность** — предписания алгоритма для любого набора допустимых входных данных конечны и упорядочены.
2. **Понятность исполнителю** — предписания алгоритма формулируются на входном языке исполнителя и используют лишь операции, реализуемые данным исполнителем (входящие в операционную среду исполнителя).
3. **Результативность** — выполнение команд алгоритма для любого набора допустимых входных данных должно приводить к результату, который может оказаться и отрицательным, показывающим, например, что задача в данном случае неразрешима.
4. **Массовость** — применимость алгоритма к любой задаче из достаточно широкого класса задач (нет ни одной корректной задачи из этого класса, решение которой нельзя получить данным алгоритмом).
5. **Детерминированность (определенность)** — все предписания алгоритма однозначно определены и приводят к однозначному результату для любого набора допустимых входных данных (не требуется уточнение предписаний по ходу исполнения алгоритма).
6. **Формализованность** — предписания алгоритма должны быть записаны на некотором формальном языке (математическом, алгоритмическом, программирования, блок-схем и др.).

7. **Полнота (команд)** — алгоритм должен содержать и в своем описании, и в области своего определения все необходимое для решения любой задачи из класса задач, решаемых данным алгоритмом.
8. **Непротиворечивость** — любые два предписания алгоритма не должны никогда (т. е. для любого допустимого набора входных данных) приходиться в противоречие.
9. **Конструктивность** — предписания алгоритма должны содержать всю необходимую информацию о последовательности и содержании действий для конструирования решения задачи при любых допустимых наборах входных данных, т. е. по алгоритму можно всегда получать результаты, интерпретируемые в определенной операционной среде (исполнителя) при любом допустимом наборе входных данных.

Если для некоторого класса задач нельзя построить (в операционной среде данного типа исполнителей) конструктивный алгоритм решения всех задач этого класса, то этот класс задач называется **алгоритмически неразрешимым**.

ПРИМЕР

Алгоритмически неразрешимы, например, проблема решения диофантовых уравнений $P(x) = 0$, где $P(x)$ — многочлен с целочисленными коэффициентами, а также проблема трисекции угла с помощью циркуля и линейки (деление угла на три равные части); если изменить операционную среду исполнителя, задачи могут быть разрешимы, например задача трисекции будет разрешима (с заданной точностью), если расширить операционную среду введением в нее какого-то угломерного инструмента (требуемой точности).

Алгоритмическая неразрешимость не приговор, не основание для того, чтобы не браться за решение данного класса задач. Для каждой такой задачи или подкласса (выбираемого при каких-то определенных условиях, гарантирующих разрешимость) данного класса задач обычно строят свою последовательность шагов, обязательно удовлетворяющих всем свойствам алгоритмов для решения в данном случае, или так называемую процедуру решения. Такие процедуры могут быть и эффективными.

ПРИМЕР

Хотя формул для получения всех корней уравнений четвертой степени и нет (в школе не изучают), тем не менее многие школьники могут составить конструктивный алгоритм решения уравнений вида $x^4 + (a + b + c + d)x^3 + (ab + cd)x^2 + (ac + bc + ad + bd + abc + abd - acd - bcd)x + abcd = 0$, если они догадываются сгруппировать, как надо (как?), члены этого уравнения и записать его в виде $(x - a)(x - b)(x - c)(x - d) = 0$.

Алгоритм, записанный на некотором языке, состоит из заголовка и тела алгоритма.

Заголовок алгоритма (описательная часть алгоритма) состоит из имени алгоритма, входных и выходных параметров, описания

назначения, цели алгоритма, типов и области определения используемых входных, выходных и вспомогательных данных.

Тело алгоритма (функциональная часть алгоритма) состоит из последовательности предписаний, преобразующих, актуализирующих данные.

Алгоритм может существовать в двух основных состояниях:

- **статическое состояние** (не актуальное, пассивное, «теоретическое» состояние), или состояние алгоритма, при котором ему еще не передано управление, например программа на диске ЭВМ; в этом состоянии входные параметры алгоритма еще не получили конкретные числовые (текстовые, логические) значения и не распределены в структуре памяти исполнителя алгоритма (такие параметры называются формальными);
- **динамическое состояние** (актуальное, активное, «практическое» состояние, точнее, состояние для каждого входного набора данных — свое состояние), или состояние алгоритма, при котором ему передано управление (данные, необходимые для его актуализации, или так называемые фактические параметры) и конкретные значения всех входных параметров алгоритма.

ПРИМЕР

Алгоритм (программа), записанный на языке программирования и сохраненный на флоппи-диске, — статический алгоритм. Алгоритм, скопированный с этого диска на жесткий диск, откомпилированный и загруженный в оперативную память вместе с необходимыми входными данными и получивший команду на выполнение, — динамический алгоритм.

Выполнение алгоритма состоит из двух основных составных частей: **управления**, или изменения номера выполняемой команды в соответствии с внутренней структурой, упорядоченностью команд алгоритма, и **преобразования** — изменения информации по пути к достижению целей алгоритма.

На школьном алгоритмическом языке (ШАЯ) любой алгоритм имеет следующую стандартную структуру:

```
алг<имя алгоритма>(<типы и имена входных и выходных параметров>)  
дан0 <условия задачи и применимости алгоритма>  
надо | <цель составления и выполнения алгоритма>  
нач  
  <описания типов и имен используемых в алгоритме рабочих  
  объектов>  
  <команды ввода входных данных>  
  <тело алгоритма (последовательность команд алгоритма)>  
  <команды вывода результатов (выходных данных)>  
кон.
```

Приведем таблицу используемых в языке функций и процедур.

Обычная запись	Учебный алгоритмический язык
Корень квадратный из x	<code>sqrt(x)</code>
Модуль $ x $	<code>abs(x)</code>
Знак числа ($-1, 0$ или 1)	<code>sign(x)</code>
$\sin x$	<code>sin(x)</code>
$\cos x$	<code>cos(x)</code>
$\operatorname{tg} x$	<code>tg(x)</code>
$\operatorname{ctg} x$	<code>ctg(x)</code>
$\arcsin x$	<code>arcsin(x)</code>
$\arccos x$	<code>arccos(x)</code>
$\operatorname{arctg} x$	<code>arctg(x)</code>
$\operatorname{arcctg} x$	<code>arcctg(x)</code>
Натуральный логарифм $\ln x$	<code>ln(x)</code>
Десятичный логарифм $\lg x$	<code>lg(x)</code>
Степень числа $e = 2, 7 \dots$ или e^x	<code>exp(x)</code>
Остаток от деления целого x на целое y	<code>mod(x, y)</code>
Частное от деления целого x на целое y	<code>div(x, y)</code>
Целая часть числа x (вещественного)	<code>int(x)</code>
Случайное число от 0 до x	<code>rnd(x)</code>
Максимум из чисел x, y	<code>max(x, y)</code>
Минимум из чисел x, y	<code>min(x, y)</code>
Длина текста x в символах	<code>длина(x)</code>

ПРИМЕР

Результаты применения этих функций: $\operatorname{sqrt}(9) = 3$, $\operatorname{abs}(-5) = 5$, $\operatorname{sign}(-6.8) = -1$, $\sin(0) = 0$, $\cos(0) = 1$, $\operatorname{tg}(\pi/4) = 1$, $\operatorname{ctg}(\pi/4) = 1$, $\arcsin(1) = \pi/2$, $\arccos(0) = 0$, $\operatorname{arctg}(1) = \pi/4$, $\operatorname{arcctg}(1) = \pi/4$, $\ln(1) = 0$, $\lg(10) = 1$, $\exp(1) = e$, $\operatorname{mod}(23, 5) = 3$, $\operatorname{mod}(3, 5) = 3$, $\operatorname{mod}(20, 5) = 0$, $\operatorname{div}(23, 5) = 4$, $\operatorname{div}(20, 5) = 4$, $\operatorname{div}(3, 5) = 0$, $\operatorname{int}(2.7) = 2$, $\operatorname{int}(2) = 2$, $\operatorname{int}(-2.7) = -3$, $\operatorname{rnd}(0) = \text{случайное число, например } 0.231356$, $\operatorname{max}(4, 7) = 7$, $\operatorname{min}(4, 7) = 4$, длина ("информ") = 6.

Порядок выполнения операций (старшинство операций — по убыванию) в языке таков:

- 1) вычисление выражений в скобках;
- 2) вычисление стандартных функций;
- 3) возведение в степень (обозначается «**», например, $x^5 \rightarrow x^{**}5$);
- 4) умножение и деление (обозначаются «*» и «/»);
- 5) сложение и вычитание (обозначаются «+» и «-»).

ПРИМЕР

Выражение $a + b * c - d / t * (v / n / m)$ вычисляется в следующем порядке (слева направо): $b * c$, $a + c * c$, v / n , $(v / n) / m$, d / t ,

$(d/t)*(v/n/m)$, $a + b*c - d/t*(v/n/m)$. Это выражение эквивалентно математической дроби $a + bc - \frac{dv}{tnm}$.

Рассмотрим базовые простые команды.

1. Команда описания (заголовка) алгоритма имеет вид:

алг <имя алгоритма> (арг <аргументы>, рез <результаты>),
где <имя алгоритма> — имя, которым снабжается алгоритм при разработке (имя должно кратко, полно, грамотно отражать суть разрабатываемого алгоритма).

Указываются и типы этих переменных (целые, вещественные, символьные, литерные, логические).

ПРИМЕР

Команда алг Максимум из трех (арг вещ a, b, c , рез вещ m).

2. Ввод — команда ввода в рассмотрение (в тело алгоритма) тех или иных входных параметров:

ввод (<вводимые параметры>).

ПРИМЕР

Команда ввод (a, b) — ввод значений параметров a и b .

3. Вывод — команда вывода тех или иных выходных параметров алгоритма:

вывод (<выводимые параметры>).

ПРИМЕР

Команда вывод ($x, "y = ", y, z$) — вывода значений x, y, z , причем значение параметра y сопровождается выдачей пояснительного текста " $y =$ " перед выдачей его числового значения.

4. Присваивание (или операция явного присваивания) — это команда вида:

<идентификатор> := <выражение>,

где <идентификатор> соответствует имени некоторого объекта, <выражение> — некоторое допустимое в рамках языка данного исполнителя выражение. Знак «:=» указывает исполнителю, что, во-первых, необходимо определение текущего значения (состояния) <выражения>, во-вторых, текущее значение переменной, имя которой указано с помощью <идентификатора>, необходимо заменить на новое значение, равное найденному значению <выражения>.

ПРИМЕР

Команда $y := x - 2*y$. Берем текущие значения x, y , затем вычисляем $z = x - 2*y$ и заменяем текущее значение y на значение z .

5. Команда начала алгоритма — команда нач. Не имеет параметра.
6. Команда завершения алгоритма — команда кон. Не имеет параметра.

7. Команда вставки комментариев в текст алгоритма имеет вид:

<комментируемый объект> | <текст комментария>,

где знак «|» отделяет их друг от друга. Обычно комментарий записывают за комментируемым объектом.

ПРИМЕР

$a := a - 2$	диаметр вала уменьшаем на 2
$b := 5 + b$	диаметр маховика увеличиваем на 5
$c := a + b$	новое значение суммы диаметров

Различают три базовые алгоритмические структуры: следование, или последовательная структура, ветвление (разветвляющаяся, или условная, структура), повторение (циклическая структура, или цикл).

1. Структура типа следование (последовательность) состоит из двух простых команд, для которых указано, какая из них выполняется первой, а какая — второй. Запись этой структуры имеет вид:

<команда-предшественник>

<команда-преемник>.

ПРИМЕР

$R := 2.45$	определяем значение радиуса пластины R
$S := \pi * R^2$	вычисляем площадь круга

2. Структура типа ветвления в полной форме состоит из некоторого предиката, проверяемого на истинность при обращении к структуре, и двух команд, одна из которых выполняется при условии истинности предиката, а другая — при условии ложности предиката. Запись этой структуры имеет вид:

если <предикат>

то <команда 1>

иначе <команда 2>

все.

ПРИМЕР

если $x > 2$ | если текущее значение $x > 2$

то $y := x$ | то значение y полагаем равным значению x

иначе $y := -x$ | иначе (т. е. при $x \leq 2$) значение y равно $-x$

все.

Структура типа ветвления в неполной форме является частным случаем структуры ветвления в полной форме, в которой при условии ложности предиката (невыполнения условия) никаких действий, кроме передачи управления следующей команде, не осуществляется. Запись этой структуры имеет вид:

если <предикат>

то <команда>

все.

Структуры типа повторения или циклы служат для компактной записи нескольких однотипных команд, повторяемых с изменяющимися параметрами.

Структура типа повторения «с предусловием» (цикл типа «пока») записывается в виде:

нц пока <предикат>

<тело цикла>

кц.

Телом цикла называется набор повторяемых команд, т. е. цикл типа «пока» имеет вид:

<u>нц</u> пока <предикат>	заголовок цикла
<команда 1>	первая команда тела цикла
<команда 2>	вторая команда тела цикла
	...
<команда n>	последняя команда тела цикла
<u>кц</u>	конец цикла.

Часть команды цикла нц пока <предикат> называется заголовком цикла.

Структура типа «пока» выполняется по следующему алгоритму:

- 1) проверяется истинность предиката, т. е. вычисляется его значение (0 или 1);
- 2) если предикат имеет значение 0, то управление передается команде 4;
- 3) если предикат имеет значение 1, то выполняется тело цикла и затем после выполнения последней команды тела цикла управление передается команде 1;
- 4) конец алгоритма, т. е. завершение структуры цикла типа «пока».

ПРИМЕР

Пусть необходимо находить сумму всех четных элементов ряда нечетных чисел до тех пор, пока эта сумма не превысит значение n . Мы сейчас забываем о лучшем, но чисто математическом решении этой задачи через определение суммы арифметической прогрессии по соответствующей формуле. Структура цикла типа «пока» позволяет не только перебирать четные числа натурального ряда чисел, но и проверять указанное в условии примера условие окончания добавления к предыдущему значению суммы очередного нечетного числа:

$s := 1$	начальное значение суммы до входа в цикл — один
$i := 1$	количество просуммированных чисел в начале — одно
<u>нц пока</u> $i < n$	заголовок цикла — проверяет условие выхода из цикла
$i := i + 2$	переходим к следующему суммируемому числу ряда
$s := s + i$	добавили к старому значению суммы новое число

кц.

Другая, менее универсальная форма организации повторений группы команд — цикл типа «с постусловием» или цикл типа «до». Структура типа «цикл с постусловием» записывается в виде:

нц для <переменная> от <начало> до <конец>
 <тело цикла>

кц.

Эта структура также называется **явной структурой цикла с пересчетом**.

Здесь <переменная> — это некоторый идентификатор изменяемой (пересчитываемой) переменной, <начало> — заданное начальное значение этой переменной, <конец> — заданное конечное значение переменной.

Этот цикл выполняется по алгоритму:

- 1) выполняется тело цикла при <переменная> = <начало> (<начало> ≤ <конец>) в соответствии с его логикой и порядком команд;
- 2) вычисляется значение предиката <переменная> > <конец> после завершения последней команды тела;
- 3) если значение предиката равно 0, то управление передается пункту 1, а если значение предиката равно 1, то управление передается пункту 4;
- 4) конец алгоритма, т. е. завершение всей структуры или цикла типа «до».

ПРИМЕР

Вычислить (табулировать) значения остатка на складе, если каждый день отпускаются товары на сумму в h р., в первый день было отпущено товара на сумму в a р., отпуск товара осуществляется в течение n дней. Структура цикла типа перебора:

нц для t от 1 до n | заголовок цикла вычисления остатка a на складе

$a := a - h$ | вычисляется остаток (в рублях) после отпуска
кц.

Приведем примеры алгоритмов (их фрагментов), не удовлетворяющих основным свойствам алгоритмов.

ПРИМЕР

Фрагмент алгоритма, для которого не выполнено свойство результативности:

а) ввод (a, b, c)

б) $D := b^2 - 4*a*c$

в) если $D > 0$

то вывод ("корни есть")

$x1 := -b + \text{sqrt}(D)$

$x2 := -b - \text{sqrt}(D)$

все.

Этот фрагмент нельзя считать телом алгоритма решения квадратного уравнения, ибо он не дает никакого результата для a, b, c , при которых дискриминант D будет отрицательным (а любой такой набор является допустимым для задачи).

ПРИМЕР

Пример фрагмента, не удовлетворяющего свойству дискретности, можно представить себе (вообразить, используя понятие «бесконечность») в виде алгоритма из бесконечного числа ко-

манд. Пример фрагмента алгоритма (процедуры), не удовлетворяющего этому свойству в другом смысле:

- 1) установить метеостанции на каждом квадратном километре;
- 2) снимать данные с метеостанции каждую минуту;
- 3) осуществлять прогноз погоды с использованием этих данных и математических моделей потоков тепла, воздуха, влажности, давления и т. д.

ПРИМЕР

Пример фрагмента процедуры, не удовлетворяющего свойству детерминированности:

- 1) взять линейку;
- 2) взять стержень;
- 3) совместить один конец стержня с началом линейки, совместить стержень с линейкой;
- 4) выдать полученную длину стержня по линейке.

Этот фрагмент не может служить процедурой измерения стержня, ибо необходимо уточнение команды 1 с учетом единиц измерения стержня; например, измерить стержень длины 9,5 м метровой линейкой (линейкой с ценой деления 1 м) нельзя.

ПРИМЕР

Пример фрагмента, не удовлетворяющего свойству массовости:

- 1) ВВОД (p, q)
 - 2) $D := p^2 - 4 \cdot q$
 - 3) если $D > 0$
 то вывод ("есть действительные корни")
 $x1 := (-p + \text{sqrt}(D))/2$
 $x2 := (-p - \text{sqrt}(D))/2$
 иначе вывод ("нет действительных корней")
- все.

Этот фрагмент не может служить телом алгоритма решения квадратного уравнения, но может служить телом алгоритма нахождения действительных корней приведенного квадратного уравнения.

ПРИМЕР

Фрагмент, приведенный выше (по свойству детерминированности), не удовлетворяет также и свойству формализованности (поэтому по отношению к нему не употреблялось понятие «алгоритм», а употреблялось понятие «процедура»).

ПРИМЕР

Если ЭВМ не имеет транслятора с Бейсика, то свойство понятности для этого исполнителя и программы на Бейсике не выполнено.

ПРИМЕР

Фрагмент алгоритма с командами вида (если допустимый набор a, b — действительные числа) и нарушением свойства

непротиворечивости:

- 1) ввод (a, b)
- 2) $x := \text{int}(a)$
- 3) $y := \text{mod}(a, b)$

Он содержит две конфликтующие команды 1 и 3, например, при допустимых параметрах $a = 5, b = 4.8$. Команды 1 и 2 не конфликтуют при этом. Для того чтобы застраховаться от появления таких ситуаций, необходимо осуществлять, например, контроль входных данных.

ПРИМЕР

Фрагмент алгоритма, приведенный по свойству массовости, не удовлетворяет и свойству конструктивности, так как не содержит команд, обрабатывающих ситуацию $D \leq 0$.

Рассмотрим в этой и следующих нескольких главах примеры составления алгоритмов. Алгоритмы, как правило, приводятся в статическом виде — без ввода исходных данных и вывода результатов (которые считаются передаваемыми через заголовок алгоритма). Для записи алгоритмов здесь и ниже будем использовать школьный алгоритмический язык — он вполне достаточен для обучения составлению алгоритмов. Есть и другие способы (языки, псевдокоды) для записи алгоритмов, например язык блок-схем, язык структурограмм и др.

Кроме того, алгоритмы будут в полной мере комментированы, так как обучение комментированию алгоритмов также одна из важных дидактических задач образовательной информатики.

ПРИМЕР

Составим алгоритм вычисления кусочно-заданной функции:

$$(*) \quad y = \begin{cases} x^2, & x \in]-\infty; 0[\\ 1 - x^2, & x \in [0; +\infty[\end{cases}.$$

Алгоритм имеет вид:

алг	Кусочная функция (арг цел n , вещ x , рез вещ y)
дано	функция $y=y(x)$, определенная формулой (*)
	x — абсцисса точки для вычисления значений функции
надо	вычислить функцию и сохранить вычисленные значения
нач	начало тела алгоритма
если	($x < 0$) какой частью формулы (*) надо воспользоваться?
	то $y := x^{**}2$ воспользуемся верхней частью (*)
	иначе $y := 1 - x^{**}2$ воспользуемся нижней частью (*)
все	конец условной структуры
кон	конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения наибольшего общего делителя двух заданных натуральных чисел:

алг НОД (арг цел m , n , рез цел k)
дано | два натуральных числа m и n
надо | найти k — наибольший общий делитель этих чисел
нач | начало тела алгоритма
нц пока ($m \neq n$) | заголовок цикла «пока» для нахождения НОД
 если ($m > n$) | проверяем, какое из чисел m и n больше
 то $m := m - n$ | если $m > n$, меняем m на $m - n$
 иначе $n := n - m$ | если $m \leq n$, n заменяем на $n - m$
 все | конец условной структуры
кц | конец цикла
 $k := m$ | присваиваем значение НОД переменной k
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм вычисления чисел Фибоначчи без запоминания вычисляемых членов последовательности Фибоначчи:

алг Числа Фибоначчи (арг цел n)
дано | номер n искомого числа Фибоначчи и закон образования этих чисел: $x[n] = x[n - 1] + x[n - 2]$, где $x[1] = 1$, $x[2] = 1$
надо | найти n -е число Фибоначчи без запоминания
нач | начало тела алгоритма
 цел i , | порядковый номер i -го определяемого числа $x[i]$
 b , | переменная для запоминания числа $x[i - 1]$
 a , | переменная для запоминания числа $x[i - 2]$
 r | переменная для запоминания вычисляемого члена $x[i]$
 $a := 1$ | задаем явно первое число Фибоначчи
 $b := 1$ | задаем явно второе число Фибоначчи
вывод (a , b) | выдача пользователю первых двух чисел Фибоначчи
нц для i от 3 до n | заголовок цикла для вычисления i -го числа
 $r := a + b$ | вычисление следующего числа Фибоначчи
 $a := b$ | замена предыдущего числа на последующее (для r)
 $b := r$ | замена значения $(i - 2)$ -го на $(i - 1)$ -е
 вывод (r) | к получению и выводу нового i -го числа r
кц | конец цикла вычисления чисел Фибоначчи
кон | конец тела алгоритма.

ПРИМЕР

Составим алгоритм выделения всех цифр любого заданного натурального n -разрядного числа x , выдавая при этом цифры пользователю с младшего разряда заданного числа:

алг Выделение цифр (арг цел n , x)
дано | n -разрядное число x с цифрами $x[1]$, $x[2]$, ..., $x[n]$
надо | вывести по мере выделения все цифры этого числа
нач | начало тела алгоритма
 цел i | переменная цикла (номер выделяемой цифры)

x , | заданное число
 y | выделенная очередная цифра
нц для i от 1 до n | заголовок цикла для выделения цифр
 $y := \text{mod}(x, 10)$ | выделяем последнюю цифру текущего значения x
вывод (y) | выдача очередной цифры числа
 $x := \text{div}(x, 10)$ | из x отбрасываем последнюю цифру
кц | конец тела цикла
кон | конец тела алгоритма.

ПРИМЕР

Составим эффективный алгоритм решения квадратного уравнения (с наименьшим числом операций умножения, возведения в степень, деления, сложения, вычитания, операндов, переменных и констант). Обычный, классический алгоритм можно встретить часто в учебниках, но с точки зрения информатики нас это, как будет показано ниже, не вполне удовлетворяет. Математика помогает информатике, но и информатика помогает математике.

Алгоритм РКУ — Решение Квадратного Уравнения имеет вид:

алг РКУ (арг вещ a, b, c , рез лит s , вещ x_1, x_2)
дано | квадратное уравнение с коэффициентами a, b, c
надо | решить оптимальным способом это уравнение
нач | начало тела алгоритма
если ($a = 0$) | является ли уравнение квадратным на самом деле?
 то если ($b = 0$) | является ли уравнением на самом деле?
 то | если нет, то — блок исключительных случаев
 если ($c = 0$) | является ли соотношение тождеством?
 то $s :=$ "Уравнение выродилось в тождество $0 = 0$ "
 иначе $s :=$ "Такого уравнения не существует"
 вывод (s) | вывод результата
 все | конец блока исключительных случаев
 иначе | тело для «иначе» условной команды
 $s :=$ "Уравнение — уравнение первой степени"
 $x_1 := -c/b$ | решаем полученное уравнение
 вывод (x_1, s) | вывод результата
 все | конец проверки исключительных параметров уравнения
 иначе | блок, идентифицирующий два различных корня
 $x_1 := -b/(2*a)$ | вспомогательное выражение
 $x_2 := x_1*x_1 - c/a$ | выражение для дискриминанта
 если ($x_2 > 0$) | существуют ли два различных корня?
 то | блок двух различных корней
 $s :=$ "Уравнение имеет два различных корня" | да
 $x_2 := \text{sqrt}(x_2)$ | оптимизированный дискриминант
 $x_1 := x_1 - x_2$ | находим первый корень
 $x_2 := x_1 + 2*x_2$ | находим второй корень

```

        вывод( $x_1$ ,  $x_2$ ,  $s$ ) | вывод результата
        | конец блока двух различных корней
    иначе | нет (двух различных корней)
        если ( $x_2 = 0$ ) | есть ли два равных корня?
            то | начало блока двух равных корней
                 $s :=$  "Уравнение имеет два равных корня" | да
                 $x_2 := x_1$  | находим второй равный корень
                вывод ( $x_1$ ,  $x_2$ ,  $s$ ) | вывод результата
            иначе  $s :=$  "Уравнение не имеет решений" | нет
        все | конец блока двух равных корней
    все | конец блока идентификации двух различных корней
все | конец проверки всех параметров заданного уравнения
кон | конец всего алгоритма.

```

Легко видеть, что использовано меньше операций и операндов, чем в классическом алгоритме. В таблице ниже приведен сравнительный анализ количества операций в классическом через вычисление дискриминанта алгоритме (он имеется практически в любом учебнике для начинающих) и в приведенном алгоритме. Для понимания оценен лишь блок, идентифицирующий два различных корня квадратного уравнения; в нашем алгоритме он выделен соответствующим комментарием, в классическом алгоритме ему соответствует фрагмент:

```

 $D := b*b - 4*a*c$ 
если ( $D > 0$ ) | существуют ли два различных корня?
    то  $s :=$  "Уравнение имеет два различных корня" | да
         $x_1 := (-b + \text{sqrt}(D))/(2*a)$  | находим первый корень
         $x_2 := (-b - \text{sqrt}(D))/(2*a)$  | находим второй корень
    все.

```

Вычитание и изменение знака считаются однопорядковыми операциями. Функция означает вычисление значения стандартной функции (без детализации количества операций в самой процедуре, микропрограмме вычисления функций). В данном случае это вычисление корня. Переменные означают количество используемых различных переменных. Предикат означает проверку какого-либо предиката на истинность.

Количество операций	Классический алгоритм	Приведенный алгоритм
Сложение	1	1
Вычитание	4	3
Умножение	5	3
Деление	2	2
Функция	2	1
Переменная	7	6
Предикат	1	1

Полезно с точки зрения программирования (примеры по алгоритмам должны акцентировать внимание и на аспекты программирования) заметить, что были «сэкономлены» наиболее длительные по времени их исполнения операции — вычитание, вычисление корня, умножение, т. е. программа по этому алгоритму будет работать намного быстрее. Интересно провести вычислительный эксперимент по решению пакета задач из 1000 различных квадратных уравнений и сравнивать затем время счета по этому и классическому алгоритмам (предоставляем возможность осуществить это читателям самостоятельно).

ПРИМЕР

Составим алгоритм округления заданного целого числа x до числа, кратного заданному целому числу y . Алгоритм имеет вид:

алг ОкруглениеДоКратного (арг цел x, y , рез цел x)
дано | два целых числа x и y
надо | округлить x до числа, кратного заданному целому y
нач | начало алгоритма и описание вспомогательных данных
цел r | остаток от деления
 $r := \text{mod}(x, y)$ | находим остаток от деления
если $(2 \cdot (y - r) \leq y)$ | если $y - r$ не превышает $y/2$,
то $x := x + y - r$ | округляем x в сторону большего кратного y ,
иначе $x := x - r$ | округляем x в сторону меньшего кратного y
все | конец выяснения округляемости
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм вычисления факториала заданного натурального числа n , т. е. произведения $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$ с использованием рекуррентной формулы $n! = (n-1)! \cdot n$. Алгоритм имеет вид:

алг ФакториалЧисла (арг цел n , рез цел f)
дано | натуральное n
надо | найти для этого числа его факториал: $n!$
нач | начало тела алгоритма
цел i | переменная цикла для вычисления факториала
 $f := 1$ | начальному значению f присваивается 1, т. е. $1! = 1$
нц для i от 1 до n | цикл умножения текущего f на текущее i
 $f := f \cdot i$ | предыдущее значение f умножаем на i
кц | конец цикла
кон | конец алгоритма.

ПРИМЕР

Заданы натуральное число и его разрядность. Вычислить среднее арифметическое ненулевых цифр и среднее геометрическое ненулевых цифр. Алгоритм имеет вид:

алг СредниеАрифмГеомНенулевыхЦифр (арг вещ x , n , рез вещ s , p)

дано | натуральное число x разрядности n

надо | вычислить s — среднее арифметическое, p — среднее геометрическое ненулевых цифр

нач | начало алгоритма

цел k | количество (счетчик) ненулевых цифр числа

$s := 0$ | начальное значение суммы цифр

$p := 1$ | начальное значение произведения цифр

$k := 0$ | начальное значение счетчика ненулевых цифр

нц для i от 1 до n | цикл перебора (выделения) цифр числа

$y := \text{mod}(x, 10)$ | выделяем очередную цифру y

если $(y \neq 0)$ | если эта цифра не нуль

$\text{то } s := s + y$ | то добавляем его в текущую сумму

$p := p * y$ | умножаем на него текущее произведение

$k := k + 1$ | и увеличиваем счетчик цифр

все | конец проверки цифры

кц | конец цикла перебора цифр числа

если $(k > 0)$ | если была найдена хоть одна ненулевая цифра

$\text{то } s := s/k$ | то вычисляем среднее арифметическое

$p := p^{1/k}$ | и среднее геометрическое этих цифр

вывод ("сумма =", s , "произведение =", p | вывод s , p

иначе вывод ("таких цифр нет") | отрицательный результат

все | конец проверки наличия ненулевой цифры

кон | конец алгоритма.

ПРИМЕР

Вычислить b — корень из числа a с точностью ϵ . Алгоритм имеет вид:

алг КореньЧислаПриближенно (арг вещ a , ϵ , рез вещ b)

дано | вещественные числа a , ϵ ($0 < \epsilon < 1$)

надо | вычислить b — корень из числа a с точностью ϵ

для вычисления b используется математическая формула:
 $b[n+1] = (b[n] + a/b[n])/2$, $b[0]$ — приближение b
 вычисления прекращаются при $\text{abs}(b[n+1] - b[n]) < \epsilon$

нач | начало алгоритма и описания переменных

цел n , | переменная, указывающая количество приближений

вещ $b0$ | начальное приближение b

если $(a > 1)$ | если число a больше единицы

$\text{то } b0 := 1$ | то нулевому значению b присваиваем единицу

иначе $b0 := 0.01$ | иначе b упрощенно присваиваем 0.01

все | конец определения начального значения b

$n := 1$ | определяем количество повторений

$b := 0.5 * (b0 + a/b0)$ | вычисление очередного значения корня

нц пока $\text{abs}(b - b0) \geq \epsilon$ | пока разность b и $b0$ больше допустимого

$b0 := b$ | меняем старое значение приближения на новое
 $b := 0.5 * (b0 + a/b0)$ | и затем вычисляем уточненное новое
 $n := n + 1$ | увеличиваем количество сделанных приближений
кц | конец цикла вычислений приближений
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм вычисления значения корня y (для любого неотрицательного x) по формуле $y = \sqrt{x + \sqrt{x + \dots + \sqrt{x}}}$. Алгоритм имеет вид:

алг ВложенныеКорни (**арг** вещ x , **цел** n , **рез** **цел** y)
дано | вещественное x , натуральное n , формула для корня
надо | вычисления значения y (для любого x) по данной формуле
нач | начало алгоритма и описания переменных
 цел i | номер вычисленных внутренних (вложенных) корней
 $y := 0$ | обнуляем начальное значение корня
нц для i **от** 1 **до** n | цикл перебора корней
 $y := y + x$ | наращиваем значение y на x
 $y := \text{sqrt}(y)$ | вычисляем корень из текущей суммы
кц | конец цикла вычисления корней
кон | конец алгоритма.

ПРИМЕР

Найдем количество простых чисел, заключенных между двумя заданными вещественными положительными числами. Алгоритм («самый понятный») имеет вид:

алг ЧислоПростыхЧисел (**арг** вещ a, b , **рез** **цел** k)
дано | вещественные числа $a, b > 0$.
надо | найти количество простых чисел между ними
нач | начало алгоритма и описания переменных
 цел y | текущее значение делителя для проверки на простоту
если $(\text{int}(a) \neq a)$ | если число a не целое
 то $a := \text{int}(a) + 1$ | то присваиваем a более подходящее
все | конец проверки a
если $(\text{int}(b) \neq b)$ | если число b не целое
 то $b := \text{int}(b)$ | то присваиваем b более подходящее
все | конец проверки b
 $k := 0$ | начальное значение счетчика целых чисел
нц для x **от** a **до** b | цикл перебора целых чисел между a и b
 $y := 2$ | начальному значению делителя y присваиваем 2
 нц пока $(y \leq \text{int}(x/2))$ **и** $(\text{mod}(x, y) > 0)$ | цикл проверки
 | делимости
 $y := y + 1$ | берем следующий делитель y
 кц | конец цикла перебора возможных делителей текущего x
 если $\text{mod}(x, y) > 0$ | если x не делится на последнее y
 то $k := k + 1$ | то x — простое и его считаем
 все | конец цикла
кц | конец цикла перебора числа
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм перевода заданного десятичного натурального числа в двоичную систему. Алгоритм имеет вид:

алг ПереводЧислаДвоичнуюСистему (цел n , рез цел a);
дано | число n в десятичной системе счисления
надо | перевести число n из десятичной системы чисел в двоичную
нач | начало тела алгоритма
 цел i , | счетчик числа делений на 2
 c , | текущее значение очередного с конца разряда,
 | т. е., если выдавались c , равные 1, 0, 0, 1, 0, 1, то
 | двоичное изображение c равно 100101
 $i := 0$ | начальное число делений на основание 2 для перевода n
нц пока ($n \geq 2$) | цикл, пока частное больше либо равно 2
 $i := i + 1$ | увеличиваем число делений на 2
 $c := \text{mod}(n, 2)$ | запоминаем остаток от деления n на 2
 $n := \text{div}(n, 2)$ | новое значение частного от деления на 2
 вывод ("новый разряд двоичного числа от конца равен", c)
кц | конец тела перевода
кон | конец тела алгоритма.

ПРИМЕР

Задано целое число n и основание системы p . Вычислим максимальное и минимальное n -разрядные числа в этой системе. Алгоритм имеет вид:

алг МаксМинЧислоСистемеСчисления (арг цел p , n , рез цел x , y)
дано | целое число n и основание системы p
надо | найти x — максимальное и y — минимальное n -разрядное
 | число в системе счисления с основанием p
нач | начало тела алгоритма
 цел i , | переменная цикла
 s | вспомогательная переменная
 x | минимальное n -разрядное число
 y | максимальное n -разрядное число
 $s := 0$ | обнуляем переменную s , где накапливаем сумму
 $y := 1$ | объединили переменную y для произведения
нц для i от 0 до $n - 1$ | заголовок цикла по элементам системы
 $s := s + (p - 1) * y$ | суммируем цифры максимального числа
 $y := y * p$ | находим новую степень основания p
кц | конец цикла
 $y := -x$ | находим искомое минимальное число
кон | конец тела алгоритма.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Дайте определение алгоритма, укажите и объясните его основные свойства. Приведите примеры алгоритмов с нарушением свойств.

2. Перечислите основные базовые алгоритмические операции, укажите и объясните их свойства. Приведите примеры использования операций.
3. Перечислите основные базовые алгоритмические структуры, укажите и объясните их свойства. Приведите примеры использования структур.
4. Охарактеризуйте основные атрибуты, назначение и функции заголовка и тела алгоритма.
5. Что такое алгоритмически неразрешимые проблемы? В чем отличие алгоритма от процедуры?
6. Запишите каждую базовую структуру в общем виде. Приведите частные случаи этих структур.
7. Какие способы представления алгоритмов вы знаете? Приведите пример.

ЗАДАЧИ

1. Составьте алгоритм определения, стоит ли в заданном числе y на заданной n -й позиции (т. е. в задаваемом разряде номер n) заданная цифра x . Отсчет разрядов ведите от младшего разряда.
2. Составьте алгоритм вычисления значения произведения максимально и минимально допустимых в системе счисления с основанием p и с максимальной разрядностью n целого числа (т. е. чисел, соответствующих понятиям положительной и отрицательной бесконечностей в этой системе).
3. Составьте алгоритм выдачи всех цифр и знака заданного целого n -разрядного числа x . Цифры выдавайте со старшего разряда.
4. Составьте алгоритм нахождения корня из числа a с заданной степенью точности ($a > 0$). Используйте известную рекуррентную формулу для вычисления квадратного корня. Начальное приближение можно задавать равным 1 при $a > 1$ и равным 0 при $a < 0$.
5. В последовательности чисел 1, 2, 6, 24, 120, ... каждый элемент (начиная со второго) получен по следующему закону: каждое новое число — произведение всех натуральных чисел от 1 до n , где n — порядковый номер этого числа в данном ряду (такое произведение — факториал числа n и обозначается как $n!$). Составьте алгоритм для вычисления любого элемента этой последовательности по задаваемому номеру n .
6. В последовательности чисел 1, 2, 3, 5, 7, 11, ... каждый элемент не имеет делителей, кроме себя и 1 (такие числа — простые). Составьте алгоритм для вычисления n -го члена последовательности (n задается).
7. Составьте алгоритм, который находит количество нулей в записи любого натурального задаваемого числа. Предусмотрите два случая: а) разрядность числа задана; б) разрядность числа не задана.

ТЕСТ

1. Алгоритм не обладает свойством:
 - а) результативности;
 - б) массовости;
 - в) детерминированности;
 - г) положительного решения задачи.
2. Больше всего свойств алгоритмов указано в варианте:
 - а) массовость, результативность, динамичность, дискретность;
 - б) дискретность, массовость, статичность, непрерывность;
 - в) массовость, результативность, дискретность, полнота;
 - г) массовость, непрерывность, линейность, тестируемость.
3. Количество «не свойств» алгоритмов в списке {программируемость, результативность, динамичность, читабельность, полнота, удобство} равно: а) 3; б) 4; в) 5; г) 6.
4. Значение s после выполнения фрагмента $s := 0; i := 5; \text{нц пока } (i > 2) i := i - 1; s := s + i; \text{кц}$ равно: а) 0; б) 5; в) 14; г) 9.
5. Во фрагменте $a := \text{да}; x := 13 \text{ нц пока } a; a := (\text{abs}(x) > 5); x := \text{div}(x, 3) + 3; \text{кц}$ тело цикла выполняется: а) 5 раз; б) 4 раза; в) 3 раза; г) 2 раза.
6. Фрагмент $s := 0; \text{нц для } i \text{ от } 1 \text{ до } n; s := s + i; \text{кц}$ находит сумму: а) натуральных последовательных чисел; б) четных последовательных натуральных чисел; в) произведений попарно соседних натуральных чисел; г) нечетных последовательных натуральных чисел.
7. Фрагмент $p := 1; \text{нц для } i \text{ от } 1 \text{ до } n; p := p * i; \text{кц}$ находит: а) произведение натуральных последовательных чисел; б) произведение четных натуральных чисел; в) произведение попарно соседних натуральных чисел; г) произведение нечетных натуральных чисел.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
г	в	б	г	в	а	а

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Узбекский математик аль-Хорезми из Хивы.
2. Основные работы Геделя, Клини, Черча, Поста, Тьюринга по теории алгоритмов.
3. Работы А. А. Маркова по теории алгорифмов.
4. Алгоритмическая разрешимость и неразрешимость.
5. Оценка сложности алгоритмов: интуитивный и научный подход.
6. P - и NP -полные алгоритмы. Асимптотическая сложность.
7. Математические и логические основы алгоритмизации.

А

ЛГОРИТМЫ РАБОТЫ С ОДНОМЕРНЫМИ МАССИВАМИ

Одномерный массив (синонимы: вектор, ряд, последовательность, линейная таблица) представляет собой совокупность значений — числовых, текстовых (символьных) или же логических, перенумерованных в каком-то порядке: $x_1, x_2, \dots, x_n$. Массив имеет общее для всех элементов массива имя. Если определен массив, то для выделения какого-то конкретного элемента массива необходимо указать его порядковый номер в этом ряду.

ПРИМЕР

Последовательность чисел 9, -6, 5, 9, 100, 0, -1.7 может образовать одномерный массив размерности 7, например, с именем x вида $x[1] = 9, x[2] = -6, x[3] = 5, x[4] = 9, x[5] = 100, x[6] = 0, x[7] = -1.7$.

Выражение, определяющее численное значение порядкового номера элемента массива — **индексное выражение**, его значение — **индекс элемента**.

ПРИМЕР

Можно ссылаться на элемент $x[7]$, элемент $x[i]$, элемент $x[3 + j - k]$ массива x . Индексные выражения равны соответственно 7, i , $3 + j - k$, а при текущих значениях переменных $i = 2, j = 0, k = 1.8$ эти индексные выражения определяют соответственно индексы 7, 2 и 1-го элементов массива.

Для обозначения (введения в рассмотрение нового типа объектов) одномерного массива в учебном алгоритмическом языке вводится служебное слово **таб**, после которого приводится имя массива и в квадратных скобках его размерность в виде $[m:n]$, где m — номер первого элемента массива (необязательно равный 1!), n — номер последнего элемента (шаг равен 1).

ПРИМЕР

Вышеуказанная последовательность из семи чисел может быть описана в алгоритмах, записываемых на ШАЯ в виде **таб** $x[1:7]$ или **вещ** **таб** $x[1:7]$.

Как видно из этого примера, одинаковые по числовому значению, но имеющие различные индексы элементы массива считаются различными.

ПРИМЕР

Составим алгоритм вычисления суммы конечного ряда чисел:

алг СуммаКонечногоРяда (**арг** **цел** n , **вещ** **таб** $x[1:n]$, **рез** **вещ** s)
нач | начало тела алгоритма
 цел i | переменная цикла суммирования

<u>s</u> := 0		начальное значение суммы
<u>нц</u> <u>для</u> <u>i</u> <u>от</u> 1 <u>до</u> <u>n</u>		заголовок цикла по числу элементов
<u>s</u> := <u>s</u> + <u>x</u> [<u>i</u>]		к сумме <i>i</i> - 1 членов добавляем <i>x</i> [<i>i</i>]
<u>кц</u>		конец цикла
<u>кон</u>		конец алгоритма.

ПРИМЕР

Составим алгоритм вычисления суммы бесконечного ряда чисел $a[1], a[2], a[3], \dots$ с точностью ϵ , полагая, что точность будет достигнута при номере члена ряда не более чем 100 000 (это число играет роль «бесконечности» в этом случае и нужно для того, чтобы объявить размерность массива *a*):

алг СуммаБесконечногоРяда (арг вещ e, вещ таб a[1 : 100 000], рез вещ s)

дано | ряд $a[1], a[2], \dots, a[n]$ вещественных чисел и точность ϵ
 | (добавление слагаемого изменяет сумму не более чем на ϵ)

надо | вычислить сумму *s* с точностью ϵ

нач | начало тела алгоритма

цел i, | переменная цикла добавления к сумме *x*[*i*]
 вещ p | переменная для значения суммы *i* - 1 членов

i := 1 | начальное значение номера текущего члена ряда

p := *a*[1] | вначале сумму членов (*i*=1) берем равной *a*[1]

s := *p* + *a*[2] | вычисляем сумму двух первых членов

нц пока abs(*s* - *p*) > e | пока суммы *s* и *p* не близки

i := *i* + 1 | переход к следующему члену ряда

p := *s* | предыдущую сумму меняем на вычисленное

s := *s* + *a*[*i*] | вычисление нового значения *s*

кц | конец тела цикла

кон | конец алгоритма.

ПРИМЕР

Составим алгоритм поиска произведения элементов конечного ряда чисел:

алг ПроизведениеКонечногоРяда (арг цел n, вещ таб a[1 : *n*], рез вещ p)

дано | последовательность из *n* элементов $a[1], a[2], \dots, a[n]$

надо | найти произведение элементов $p = a[1] * a[2] * \dots * a[n]$

нач | начало тела алгоритма

цел i, | *i* - переменная цикла домножения на новый *a*[*i*]

p := 1 | начальное значение *p* равно 1

нц для i от 1 до n | заголовок цикла для вычисления *p*

p := *p* * *a*[*i*] | домножаем произведение *i* - 1 членов на *a*[*i*] и получаем новое значение

кц | конец тела цикла

кон | конец алгоритма.

ПРИМЕР

Составим алгоритм вычисления значения полинома по схеме Горнера:

алг СхемаГорнера (арг цел n, вещ x, вещ таб a[0 : *n*], рез вещ p)

дано | полином вида $a[0]*x^n + a[1]*x^{n-1} + \dots + a[n]$,
 (дано натуральное число n , вещественные числа $a[0:n]$ и x)
надо | вычислить значение полинома в точке x по схеме Горнера:
 1) при $n = 0$, $P = a[0]$
 2) при $n = 1$, $P = a[0]*x + a[1] = (a[0])*x + a[1]$
 3) при $n = 2$, $P = a[0]*x^2 + a[1]*x + a[2] = (a[0]*x + a[1])*x + a[2]$
 ...
 $n) P = a[0]*x^n + a[1]*x^{n-1} + a[2]*x^{n-2} + \dots + a[n-1]*x + a[n] =$
 $= (a[0]*x^{n-1} + a[1]*x^{n-2} + a[2]*x^{n-3} + \dots + a[n-1])*x + a[n]$
 Таким образом, всегда верно рекуррентное соотношение Горнера:
 Текущее $P :=$ Предыдущее $P*x + a[\text{текущее}]$
нач | начало тела алгоритма
цел i | переменная цикла по индексам $i = 1, 2, \dots, n$
 $p := a[0]$ | начальное значение переменной p полагаем равным $a[0]$
нц для i **от** 1 **до** n | заголовок цикла реализации схемы Горнера
 $p := p*x + a[i]$ | по рекуррентной формуле Горнера
кц | конец тела цикла
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм вычисления чисел Фибоначчи (см. выше) с запоминанием вычисленных ранее чисел:

алг ЧислаФибоначчи-2 (**арг цел** n , **рез цел таб** $x[1:n]$)
дано | номер n последнего вычисляемого числа Фибоначчи
надо | n -е число Фибоначчи запомнить в таблице $x[1:n]$
нач | начало тела алгоритма
цел i , | порядковый номер i -го определяемого числа $x[i]$
 $x[1] := 1$ | задаем явно первое число Фибоначчи
 $x[2] := 1$ | задаем явно второе число Фибоначчи
нц для i **от** 3 **до** n | заголовок цикла для вычисления i -го числа
 $x[i] := x[i-1] + x[i-2]$ | вычисление следующего числа
кц | конец цикла по i
кон | конец тела алгоритма.

ПРИМЕР

Составим алгоритм инвертирования ряда, т. е. записи в обратном порядке:

алг ИнвертированиеРяда (**арг цел** n , **вещ таб** $x[1:n]$)
дано | натуральное число n и ряд чисел $x[1], x[2], \dots, x[n]$
надо | получить ряд $x[n], x[n-1], \dots, x[2], x[1]$
нач | начало тела алгоритма
цел i , | переменная цикла инвертирования
 m , | переменная для середины массива, т. е. $\text{int}(n/2)$
вещ y | буферная переменная для перестановки элементов
 $m := \text{int}(n/2)$ | находим середину массива x
нц для i **от** 1 **до** m | заголовок цикла перестановки двух чисел
 $y := x[i]$ | запоминаем в y значение $x[i]$

$x[i] := x[n - i]$	заменяем $x[i]$ значением симметричного относительно середины ряда элемента $x[n - i]$
$x[n - i] := y$	
	на место $x[n - i]$ ставим $x[i]$, его значение было запомнено в буфере y
<u>кц</u>	конец цикла по i
<u>кон</u>	конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения номеров максимального и минимального элементов в массиве $x[1], x[2], \dots, x[n]$:

алг НомераМаксМинРяда (арг цел n , вещ таб $x[1:n]$, рез цел max , min)

дано | массив $x[1:n]$

надо | найти номера максимального и минимального элементов массива $x[1:n]$

нач | начало алгоритма и описания вспомогательных переменных

цел i | номер текущего элемента массива

max | максимальный из просмотренных членов ряда

min | минимальный из просмотренных членов ряда

$max := x[1]$ | переменная max равна 1-му элементу

$max := 1$ | считаем номером максимального элемента 1

$min := x[1]$ | переменная min равна 1-му элементу

$min := 1$ | считаем номером минимального элемента 1

нц для i от 1 до n | цикл перебора всех элементов массива
(часто начинают со 2-го, но возможен тест $n = 1$!)

если $max < x[i]$ | если i -й элемент больше

то $max := x[i]$ | то max присваиваем i -й, еще больший

$max := i$ | и запоминаем его номер

все | конец сравнения i -го элемента с текущим значением

если $min > x[i]$ | если i -й элемент меньше

то $min := x[i]$ | то min присваиваем i -й элемент массива

$min := i$ | и запоминаем его номер

все | конец сравнения i -го элемента с текущим значением

кц | конец цикла перебора элементов массива

кон | конец алгоритма.

ПРИМЕР

Составим алгоритм вычисления значений выражений вида $x = a[1] * a[2] + \dots + a[n - 1] * a[n]$ и $y = (-1)^n * a[1] * a[2] * \dots * a[n]$.

Алгоритм имеет вид:

алг ЗначенияВыражений (арг цел n , вещ таб $a[1:n]$, рез вещ x , y)

дано | ряд чисел $a[1], a[2], \dots, a[n]$

надо | найти $x = a[1] * a[2] + \dots + a[n - 1] * a[n]$, $y = (-1)^n * a[1] * \dots * a[n]$

нач | начало алгоритма и описания вспомогательных переменных

цел i , | номер текущего элемента ряда

$x := 0$ | начальное значение суммы

$y := 1$ | начальное значение произведения

<u>нц</u> <u>для</u> i <u>от</u> 1 <u>до</u> $n - 1$	цикл вычисления x и y
$x := x + a[i] * a[i + 1]$	к предыдущей сумме добавляем $a[i] * a[i + 1]$
$y := y * (-a[i])$	произведение умножаем на $-a[i]$
<u>кц</u>	конец цикла суммирования и умножения
<u>кон</u>	конец алгоритма.

ПРИМЕР

Составим алгоритм удаления максимального элемента ряда чисел. Алгоритм имеет вид:

алг УдалениеМаксРяда (арг цел n , вещ таб $x[1:n]$, рез вещ таб $x[1:n]$)

дано | ряд чисел $x[1], x[2], \dots, x[n]$

надо | найти максимальный элемент ряда и удалить его

нач | начало алгоритма и описания переменных

цел i , | номер текущего элемента ряда

k , | вспомогательная переменная

вещ $maxx$ | текущее значение максимума

$maxx := x[1]$ | начальное значение максимального элемента

$k := 1$ | и его номер первый

нц для i от 1 до n | цикл перебора чисел ряда

если $(x[i] > maxx)$ | если элемент большего $maxx$

то $maxx := x[i]$ | то меняем текущий максимум и

$k := i$ | текущий номер максимума ряда

все | конец сравнения текущего элемента

кц | конец цикла перебора элементов ряда

нц для i от k до $n - 1$ | цикл удаления максимального элемента

$x[i] := x[i + 1]$ | сдвигаем все за максимальным вперед на одну позицию

кц | конец цикла удаления ряда

$n := n - 1$ | уменьшаем число элементов на один (удаленный)

кон | конец алгоритма.

ПРИМЕР

Составим алгоритм поиска наибольшего m и наименьшего k из последовательности целых чисел $x_1, x_2, \dots, x_n$, но среди элементов на четных местах:

алг МинМаксРядаЧетМест (арг цел n , цел таб $x[1:n]$, рез цел m, k)

дано | ряд целых чисел $x[i], i = 1, 2, \dots, n$

надо | найти минимум m и максимум k на четных местах

нач | начало алгоритма и описания переменных

цел i , | i — номер текущего четного места

$i := 2$ | начальное значение i полагаем первому четному номеру

$k := x[2]$ | начальное значение минимального элемента ряда

$m := x[2]$ | начальное значение максимального элемента ряда

нц пока $(i \leq n)$ | цикл перебора элементов на четных местах

$m := \max(m, x[i])$ | наибольшее m из текущих m и $x[i]$

$k := \min(k, x[i])$ | наименьшее k из текущих k и $x[i]$

$i := i + 2$ | переход к следующему элементу
кц | конец цикла перебора требуемых по условию элементов ряда
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения максимального расстояния от заданных точек из 2-й четверти до начала координат и номер точки с максимумом этого расстояния. Алгоритм имеет вид:

алг МаксРасстТочек (арг цел n , вещ таб $x[1:n]$, $y[i:n]$, рез вещ $maxd$, цел k)
дано | вещественные числа — массивы $x[i:n]$, $y[i:n]$
надо | найти максимум расстояний от точек 2-й четверти до $O(0; 0)$ и номер этой точки
нач | начало алгоритма и описания вспомогательных переменных
 цел i | номер текущего элемента массива
 вещ d | текущее расстояние между точками $(x[i], y[i])$, $O(0; 0)$
 $maxd := 0$ | в качестве максимального расстояния возьмем нулевое
 нц для i от 1 до n | цикл перебора всех элементов массива
 если $(x[i] < 0)$ и $(y[i] > 0)$ | если точка лежит во 2-й четверти,
 то $d := \sqrt{x[i]^2 + y[i]^2}$ | расстояние до $O(0; 0)$
 если $(maxd < d)$ | сравниваем с текущим максимумом длин
 то $maxd := d$ | и фиксируем новый максимум
 $k := i$ | номер точки с новым максимумом
 все | конец сравнения с текущим максимумом
 все | конец проверки, лежит ли точка во 2-й четверти
кц | конец цикла перебора точек
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения наибольшего m и наименьшего k из целочисленной последовательности чисел $x_1, x_2, \dots, x_n$, но только среди элементов на четных местах и только до тех пор, пока не встретится первый нечетный элемент. Алгоритм имеет вид:

алг МиниМакРядаДоНечет (арг цел n , цел таб $x[1:n]$, рез цел m, k)
дано | ряд целых чисел $x[i]$, $i = 1, 2, \dots, n$
надо | найти минимум m и максимум k нечетных элементов данного ряда, но только среди элементов на четных местах
нач | начало алгоритма и описания вспомогательных переменных
 цел i | i — номер текущего четного места
 $i := 2$ | начальное i полагаем первому четному номеру
 $k := x[2]$ | начальное значение минимального элемента ряда
 $m := x[2]$ | начальное значение максимального элемента ряда
 нц пока $(i \leq n)$ | перебор всех элементов
 если $(\text{mod}(x[i], 2) = 0)$ | если элемент четный
 то $m := \max(m, x[i])$ | выбор максимума
 $k := \min(k, x[i])$ | выбор минимума из текущего числа

		на четном месте и из максимума из предшество-
		вавших ему таких же
<u>все</u>		конец проверки
$i := i + 2$		переход к следующему на четном месте
<u>кц</u>		конец цикла перебора требуемых по условию элементов
		ряда
<u>кон</u>		конец алгоритма.

ПРИМЕР

Составим алгоритм сортировки (по возрастанию) заданной последовательности вещественных чисел $x[1], x[2], \dots, x[n]$ (сортировка обменами):

алг СортировкаОбменами (арг цел n , вещ таб $x[1:n]$)

<u>дано</u>		вещественная последовательность $x[1], x[2], \dots, x[n]$
<u>надо</u>		получить ряд из них вида $x[1] < x[2] < \dots < x[n]$
<u>нач</u>		начало тела алгоритма
<u>цел</u> i		переменная цикла нахождения места текущего $x[i]$
j		переменная цикла сравнения $x[i]$ и $x[j]$, $j = i + 1, \dots, n$
<u>вещ</u> u		переменная перестановки элементов ряда (буфер)
<u>нц для</u> i <u>от</u> 1 <u>до</u> $n - 1$		цикл поиска места i -го текущего элемента
<u>нц для</u> j <u>от</u> $i + 1$ <u>до</u> n		цикл сравнения $x[i]$ с $x[j + 1], \dots, x[n]$
<u>если</u> $x[i] > x[j]$		проверка: нужно ли переставлять?
<u>то</u>		если нужно — блок перестановки $x[i]$ с $x[j]$
$u := x[j]$		запоминание значения $x[j]$ в буфере u
$x[j] := x[i]$		на место $x[j]$ ставим $x[i]$
$x[i] := u$		на место $x[i]$ ставим $x[j]$ из буфера u
<u>все</u>		конец условной структуры
<u>кц</u>		конец цикла сравнения текущего элемента
<u>кц</u>		конец цикла поиска места каждого элемента
<u>кон</u>		конец алгоритма.

ПРИМЕР

Составим алгоритм циклического сдвига элементов ряда на k позиций вперед (влево). Алгоритм имеет вид:

алг ЦикличСдвигВперед (арг цел n , вещ таб $x[1:n]$, рез вещ таб $x[1:n]$)

<u>дано</u>		вещественный ряд чисел $x[1], x[2], \dots, x[n]$
<u>надо</u>		получить ряд вида $x[k + 1], x[k + 2], \dots, x[n], x[1], x[2], \dots, x[k]$
<u>нач</u>		начало тела алгоритма и описание вспомогательных данных
<u>цел</u> i		номер текущего элемента ряда (переменная цикла)
<u>вещ</u> u		переменная для фиксации 1-го элемента
<u>нц для</u> i <u>от</u> 1 <u>до</u> k		цикл сдвигов k раз (текущего элемента номер i)
$u := x[i]$		переменной u присваивается 1-й элемент
<u>нц для</u> j <u>от</u> 1 <u>до</u> $n - 1$		цикл сдвига $x[j]$ вперед
$x[j] := x[j + 1]$		с помощью последовательных сдвигов
<u>кц</u>		конец цикла сдвигов
$x[n] := u$		в конец ряда ставим 1-й элемент (копию u)
<u>кц</u>		конец цикла выбора текущего элемента
<u>кон</u>		конец алгоритма.

ПРИМЕР

Составим алгоритм циклического сдвига элементов ряда на k позиций назад (вправо). Алгоритм имеет вид:

алг ЦикличСдвигНазад (арг цел n , вещ таб $x[1:n]$, рез вещ таб $x[1:n]$)

дано | вещественный ряд чисел $x[1], x[2], \dots, x[n]$

надо | получить $x[n-k+1], x[n-k+2], \dots, x[n], x[1], \dots, x[n-k]$

нач | начало тела алгоритма и описания вспомогательных данных

цел i | номер текущего элемента ряда (переменная цикла)

вещ y | переменная для запоминания 1-го элемента

нц для i от 1 до k | цикл сдвигов k раз (текущего элемента i)

$y := x[n]$ | y присваиваем последний элемент

нц для j от 1 до $n-1$ | цикл сдвига $x[j]$ назад

$x[j+1] := x[j]$ | с помощью сдвигов

кц | конец цикла сдвигов

$x[1] := y$ | в начало ставим последний элемент (копию y)

кц | конец цикла выбора текущего элемента

кон | конец алгоритма.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое одномерный массив, каковы его синонимы? Приведите примеры.
2. Что такое индексное выражение, индекс элемента? Приведите примеры.
3. Зачем вводятся одномерные массивы? Укажите одну из причин.
4. Как описываются одномерные массивы на учебном языке?
5. Когда можно считать два массива полностью идентичными?
6. Какие два массива будут различными?
7. Какие операции определены для элементов массивов?

ЗАДАЧИ

1. Составьте алгоритм поиска суммы наибольшего и наименьшего рядов.
2. Составьте алгоритм нахождения числа положительных и отрицательных чисел, находящихся между наибольшим и наименьшим элементами ряда.
3. Составьте алгоритм нахождения количества чисел ряда $x_1, x_2, \dots, x_n$, меньших:
а) заданного числа; б) среднего арифметического членов ряда.
4. Составьте алгоритм нахождения суммы неотрицательных чисел ряда.

5. Составьте алгоритм удаления из ряда целых чисел всех четных чисел на нечетных местах.
6. Составьте алгоритм нахождения произведения значений полинома n -й степени $P(x)$ в точке x на значение полинома m -й степени $Q(x)$ в точке x .
7. Составьте алгоритм замены наибольшего и наименьшего элементов заданной последовательности чисел на их противоположные значения.

ТЕСТ

1. Фрагмент $s := x[1]$; нц для i от 1 до n ; если $s > x[i]$ то $s := x[i]$; все; кц; находит:
 - а) минимальный элемент массива чисел $X = \{x[i], i = 1, 2, \dots, n\}$;
 - б) сумму отрицательных элементов X ;
 - в) индекс минимального элемента X ;
 - г) максимальный элемент X .
2. Фрагмент $p := 1$; нц для i от 1 до n ; $p := p * \text{abs}(x[i])$; кц; если $(n > 1)$ то $p := p^{**}(1/n)$; все; определяет:
 - а) произведение модулей всех чисел ряда $X = \{x[i], i = 1, 2, \dots, n\}$;
 - б) корень квадратный из произведения модулей чисел X ;
 - в) среднее геометрическое модулей чисел ряда X ;
 - г) среднее арифметическое модулей всех чисел X .
3. Фрагмент $s := 0$; нц для i от 0 до $n - 1$; $s := s + i + 1$; кц; находит сумму:
 - а) натуральных последовательных чисел;
 - б) четных последовательных натуральных чисел;
 - в) произведений попарно соседних натуральных чисел;
 - г) нечетных последовательных натуральных чисел.
4. Фрагмент $p := 1$; нц для i от 1 до n ; $p := p * x[i]$; кц; находит:
 - а) произведение натуральных последовательных чисел;
 - б) произведение элементов заданного ряда чисел;
 - в) произведение попарно соседних натуральных чисел;
 - г) произведение нечетных натуральных чисел.
5. Цикл из команд нц для i от 1 до n ; $y[i] := \text{div}(i, 2)$; кц; находит:
 - а) целые части «половин» натуральных последовательных чисел;
 - б) сумму четных натуральных чисел;
 - в) остатки от «половин» натуральных последовательных чисел;
 - г) сумму нечетных натуральных чисел.
6. Значение $P_n(x) = a_0 x^n + a_1 x^{n-1} + \dots + a_n$ можно найти фрагментом алгоритма:
 - а) $P := a[1]$; нц для i от 2 до n ; $P := P * x$; кц;
 - б) $P := a[n]$; $i := n$; нц пока $i > 1$; $P := P * x + a[i]$; $i := i - 1$; кц;
 - в) $P := a[0]$; $i := 1$; нц пока $i < n$; $P := P * x + a[i]$; $i := i + 1$; кц;
 - г) $P := a[0]$; нц для i от 1 до n ; $P := P * x$; кц.

7. Результат f фрагмента $m := a; n := \text{abs}(b);$ нц пока $(m > 0)$ и $(n > 0);$
если $(m > n)$ то $m := \text{mod}(m, n);$ иначе $n := \text{mod}(n, m);$ все; кц;
 $f := m + n;$ равен 7 для:
- а) $a = 81, b = -27;$ в) $a = 84, b = -27;$
б) $a = 84, b = -35;$ г) $a = 81, b = -21.$

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
а	в	а	б	а	в	б

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Алгоритм сортировки ряда чисел обмeнaми.
2. Алгоритм сортировки ряда чисел методом «пузырька».
3. Алгоритм сортировки ряда чисел Шелла.
4. Алгоритм сортировки ряда чисел с помощью кучи.
5. Алгоритм быстрой сортировки.
6. Теоремы алгоритмики рядов.
7. Комбинаторные задачи (алгоритмы).

АЛГОРИТМЫ РАБОТЫ С ДВУХМЕРНЫМИ МАССИВАМИ

Двухмерным массивом (синонимы: прямоугольная таблица, матрица) называется совокупность векторов одинаковой размерности и одинакового типа. Векторы можно рассматривать как по вертикали (векторы-столбцы), так и по горизонтали (векторы-строки), при этом может измениться их размерность. Общий вид двухмерного массива:

(столбец 1)	(столбец 2)	...	(столбец n)	
x_{11}	x_{12}	...	x_{1n}	(строка 1)
x_{21}	x_{22}	...	x_{2n}	(строка 2)
...	...	...	...	...
x_{m1}	x_{m2}	...	x_{mn}	(строка m)

Массив, как и вектор, имеет общее имя, например массив x . Для выделения элемента массива нужно указать два его индекса (целочисленные значения двух индексных выражений) — номер строки и номер столбца, на пересечении которых стоит этот элемент.

ПРИМЕР

Элемент $x[3, 2]$ — элемент, который стоит на пересечении 3-й строки и 2-го столбца массива x .

Матрицы, как и векторы, должны быть в алгоритме описаны служебным словом **таб**, но в отличие от вектора матрица имеет описание двух индексных выражений, разделяемых запятыми. Первое индексное выражение определяет значения номеров строк, а второе — столбцов.

ПРИМЕР

Если матрица x описана в виде **вещ таб $x[1:2, 3:5]$** , то это определяет таблицу из двух строк (от 1-й до 2-й строки) и трех столбцов (от 3-го до 5-го столбца) вида

(столбец 1)	(столбец 2)	(столбец 3)	
x_{13}	x_{14}	x_{15}	(строка 1)
x_{23}	x_{24}	x_{25}	(строка 2)

Для перебора элементов двухмерного массива $x[1:n, 1:m]$ нужно организовать два цикла: один цикл (внешний) **нц для i от 1 до n** — для перебора всех строк массива, а другой (внутренний) **нц для j от 1 до m** — для перебора всех элементов данной (текущей) строки номер i массива, т. е. элементов по столбцам. Эта группа циклов позволяет перебирать элементы массива построчно от строки к строке. Можно организовать перебор элементов и по столбцам: внешний цикл **нц для j от 1 до m** , а внутренний — **нц для i от 1 до n** .

ПРИМЕР

Цикл ввода всех элементов массива $x[1:n, 1:m]$ построчно:

```
нц для  $i$  от 1 до  $n$   
  нц для  $j$  от 1 до  $m$   
    ввод ( $x[i, j]$ )  
  кц  
кц.
```

ПРИМЕР

Цикл вывода всех элементов массива $x[1:n, 1:m]$ по столбцам:

```
нц для  $j$  от 1 до  $m$   
  нц для  $i$  от 1 до  $n$   
    вывод ( $x[i, j]$ )  
  кц  
кц.
```

Если номера строки и столбца можно связать математически так, что один номер восстанавливается по другому, то нет необходимости в полном переборе всех элементов массива, а часто и в двойном цикле.

ПРИМЕР

Пусть нужно найти сумму только тех элементов матрицы $x[1:n, 1:n]$, которые расположены на двух диагональных линиях — главной диагонали, соединяющей $x[1, 1]$ и $x[n, n]$, и побочной диагонали, соединяющей $x[1, n]$ и $x[n, 1]$. Легко установить следующую связь на главной диагонали $j = i$. На побочной диагонали эта связь находится из непосредственно индуктивно устанавливаемого равенства $i + j = n + 1$, т. е. $j = n + 1 - i$. Тогда тело алгоритма суммирования имеет вид:

```
 $s := 0$   
нц для  $i$  от 1 до  $n$   
   $s := s + x[i, i] + x[i, n + 1 - i]$   
кц.
```

Иногда в качестве решения предлагается фрагмент:

```
 $s := 0$   
нц для  $i$  от 1 до  $n$   
  нц для  $j$  от 1 до  $i$   
    если ( $i = j$ ) или ( $i + j = n + 1$ )  
      то  $s := s + x[i, j]$   
  все  
кц  
кц.
```

При всей видимости корректности и несмотря на то, что этот фрагмент выдаст правильный результат, его нельзя считать приемлемым решением задачи, так как он производит просмотр всех элементов (n^2 действий), а предыдущий — только n действий.

Объекты, связи, которые не изменяются от элемента к элементу в цикле, называются **инвариантами цикла** (специально упрощено это сложное понятие).

ПРИМЕР

В предыдущем примере инварианты цикла — формулы $j = i$, $j = n + 1 - i$.

ПРИМЕР

Составим алгоритм вычисления суммы и произведения всех элементов $a[i, j]$, где $i = 1, 2, \dots, n$; $j = 1, 2, \dots, m$ заданной матрицы A :

алг СуммаПроизведение (арг цел n, m , вещ таб $a[1:n, 1:m]$, рез вещ s, p)

дано | матрица A из элементов $a[i, j]$, $i = 1, 2, \dots, n$, $j = 1, 2, \dots, m$

надо | вычислить сумму s и произведение p элементов A

нач | начало алгоритма и описания переменных

цел i | номер текущей строки матрицы

j | номер текущего столбца матрицы

$s := 0$ | обнуляем переменную s , где накапливаем сумму

$p := 1$ | переменную p для произведения равно 1

нц для i от 1 до n | цикл перебора всех строк матрицы

нц для j от 1 до m | цикл перебора элементов текущей строки

$s := s + a[i, j]$ | к предыдущему значению s

добавляем текущий элемент

$p := p * a[i, j]$ | старое значение произведения p

умножаем на текущий элемент

кц | конец цикла перебора столбцов

кц | конец цикла перебора строк

кон | конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения номера первого столбца целочисленной матрицы A из n строк и m столбцов, в котором содержится наименьшее число четных и отрицательных элементов:

алг СтолбецМинЧетОтриц (арг цел n, m , вещ таб $a[1:n, 1:m]$, рез цел q)

дано | матрица A из элементов $a[i, j]$, $i = 1, 2, \dots, n$, $j = 1, 2, \dots, m$

надо | найти столбец q с минимумом четных и отрицательных

нач | начало алгоритма и описания переменных

цел i , | номер текущей строки матрицы

j , | номер текущего столбца матрицы

l , | количество искоемых элементов в текущем столбце

k , | максимум искоемых элементов по предыдущим

$k := n$ | начальное значение k — максимально возможное

(считаем все элементы столбца подходящими)

нц для j от 1 до m | цикл перебора всех столбцов и нахождения l

$l := 0$ | обнуляем счетчик искоемых элементов столбца

```

ни для  $i$  от 1 до  $n$  | цикл перебора элементов столбца
если  $(\text{mod}(a[i, j], 2) = 0$  и | проверка условия четности и
 $a[i, j] < 0)$  | отрицательности  $a[i, j]$ ; если эти
то  $i := i + 1$  | условия выполнены, увеличиваем  $i$ 
все | конец проверки на четность и отрицательность
к | конец цикла по элементам текущего столбца матрицы
если  $(i < k)$  | если для текущего столбца  $i < k$ 
то  $k := i$  | то меняем  $k$  на  $i$  и меняем текущий номер
 $q := j$  | искомого столбца на номер текущего столбца
все | конец сравнения текущего значения  $k$  с текущим  $i$ 
к | конец цикла перебора и испытания всех столбцов
кон | конец алгоритма

```

ПРИМЕР

Составим алгоритм поиска количества и произведения всех нечетных элементов по каждой четной строке матрицы. Алгоритм имеет вид:

алг НечЭлЧетСтр (арг цел n, m , таб $x[1:n, 1:m]$, рез цел таб k , $p[1:\text{int}(n/2)]$)

дано | целочисленная таблица x из n строк и m столбцов

надо найти произведение нечетных элементов четных строк

нач	начало алгоритма
-----	------------------

цел i , переменная цикла по строкам (номер текущей строки)

j номер элемента матрицы по столбцам

номер нечетного элемента в четной строке

$i := 2$ номер текущей (первой вначале) четной строки — это 2

$l := 1$ | номер нечетного элемента в четной строке (вначале — 1)

нц пока ($i \leq n$) | цикл по четным строкам матрицы

$p[l] := 1$ произведение нечетных текущей строки

$k[I] := 0$	количество нечетных элементов данной строки
-------------	---

ны для i от 1 до m | цикл по столбцам матрицы

если $(\text{mod}(X[i], J, 2) = 1)$	если текущий элемент нечетный
то $k[J] := k[J] + 1$	то число элементов увеличиваем на 1

$\rho[l] := \rho[l] * x[i, l]$	умножаем произведение нечетных
--------------------------------	--------------------------------

все | конец проверки элемента текущей строки

кц | конец цикла по столбцам

$j := j + 2$ | переход к следующей четной строке

кц | конец цикла по строкам

кон | конец алгоритма.

ПРИМЕР

Составим алгоритм выделения всех элементов матрицы a , не совпадающих ни с одним элементом матрицы b , в отдельный массив c . Все матрицы целочисленные. Алгоритм имеет вид:

алг Отделение (арг цел n, m, s, k , таб $a[1:n, 1:m], b[1:s, 1:k]$,
рез цел таб $c[1:s*k + n*m]$)

ПРИМЕР

Составим алгоритм нахождения строки матрицы с минимумом нулей:

алг МинНулейСтрока (арг цел n, m , вещ таб $a[1:n, 1:m]$, рез цел k, r)

дано | вещественный массив x из n строк и m столбцов

надо | найти номер k строки массива x с минимумом r нулей

нач | описание используемых вспомогательных переменных

цел i , | переменная для перебора строк матрицы x

j , | переменная для перебора элементов строки номер i

s | переменная для подсчета количества нулей в строке i

$r := m$ | начальное минимальное число нулевых элементов

| полагаем равным максимально возможному, считая,

$k := 1$ | что наименьшее число нулей находится в первой строке
| (изменяем, если есть строка, в которой меньше нулей)

нц для i от 1 до n | заголовок цикла для перебора всех n строк

$s := 0$ | начальное число нулевых элементов в строке i

нц для j от 1 до m | цикл перебора элементов строки i

если $(x[i, j] = 0)$ | если текущий элемент равен 0, то

то $s := s + 1$ | увеличим счетчик нулей в строке

все | конец проверки на нуль

кц | конец цикла перебора элементов текущей строки

если $(r > s)$ | если количество нулей текущей строки

| меньше минимума нулей в предыдущих

то | то строка содержит минимальное число нулей

$r := s$ | и запоминаем их количество

$k := i$ | запоминаем номер этой строки

все | конец условия

кц | конец цикла по строкам

кон | конец алгоритма.

ПРИМЕР

Составим алгоритм, который для каждой строки заданной целочисленной матрицы $x[1:n, 1:m]$ определяет количество элементов, кратных заданному целому числу y . Алгоритм имеет следующий вид:

алг Кратность (арг цел n, m, y , таб $x[1:n, 1:m]$, рез цел таб $k[1:n]$)

дано | целочисленная матрица $x[i, j]$, $i = 1, \dots, n$;
 $j = 1, \dots, m$

надо | определить вектор из $k[j]$ — числа кратных y в строке i

нач | начало алгоритма и описания переменных

цел i , | номер текущей строки матрицы x

j | номер текущего столбца матрицы x

нц для i от 1 до n | цикл по строкам

$k[j] := 0$ | начальное количество кратных элементов в строке

нц для j от 1 до m | цикл по элементам текущей i -й строки
 если $(\text{mod}(x[i, j], y) = 0)$ | если элемент $x[i, j]$ кратен y
 то $k[j] := k[j] + 1$ | то увеличиваем количество кратных
 все | конец цикла проверки элементов на кратность
кц | конец цикла по элементам текущей строки
кц | конец цикла по строкам (выбора новой строки)
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм удаления заданной строки матрицы x , состоящей из m строк и n столбцов. Алгоритм имеет вид:

алг УдСтроки (арг цел n, m, k , вещ таб $x[1:m, 1:n]$, рез вещ таб $x[1:m, 1:n]$)
дано | матрица вещественных чисел x из m строк и n столбцов
надо | удалить строку матрицы с заданным номером k
нач | начало алгоритма и описание переменных
 цел i , | номер текущей строки матрицы
 цел j | номер текущего столбца матрицы
нц для i от k до $m - 1$ | цикл перебора строк, начиная с k -й
 нц для j от 1 до n | цикл перебора элементов текущей строки
 $x[i, j] := x[j + 1, j]$ | меняем элемент текущей строки на
 | элемент следующей строки для каждого столбца
 кц | конец цикла по передвижению элементов строки
кц | конец цикла выбора текущей строки (переход к новой)
 $m := m - 1$ | число строк уменьшаем на 1 (удаляем последнюю,
 | передвинутую на место строки $m - 1$)
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения суммы и произведения всех элементов матрицы, расположенных на главной диагонали. На главной диагонали все элементы $x[i, j]$ обладают следующим свойством: индекс строки элемента равен индексу столбца или $i = j$. С учетом этого алгоритм имеет вид:

алг СуммаПроизвГлавДиаг (арг цел n , вещ таб $x[1:n, 1:n]$, рез
вещ s, p)
дано | квадратная матрица x размерности n строк и n столбцов
надо | найти сумму и произведение на главной диагонали x
нач | начало алгоритма и описание переменных
 цел i , | переменная цикла по строкам матрицы
 цел j | переменная цикла по столбцам матрицы
 $s := 0$ | начальное значение суммы
 $p := 1$ | начальное значение произведения
нц для i от 1 до n | цикл перебора всех строк матрицы
 $s := s + x[i, i]$ | к предыдущему значению суммы добавляем
 | следующий элемент главной диагонали

$p := p * x[i, j]$ | домножаем предыдущее значение произведения

кц | конец цикла перебора всех строк матрицы

кон | конец алгоритма.

ПРИМЕР

Составим алгоритм вставки новой строки номер k (т. е. новая строка ставится на место старой k -й строки, которая становится $(k+1)$ -й строкой в новой матрице) и нового столбца номер l (т. е. новый столбец становится на место старого l -го столбца, который становится $(l+1)$ -м столбцом в новой матрице) в заданную матрицу A из n строк и m столбцов. Для вставки строки номер k в матрицу A необходимо последовательно переставить все элементы строки номер n на место соответствующих (расположенных в том же столбце) элементов новой строки $n+1$, затем элементы строки номер $n-1$ на место элементов строки номер n и т. д. до строки номер k матрицы, сдвигаемой на место строки номер $k+1$; затем на место старой k -й строки вставляются элементы новой k -й строки, например, из массива $b[1:m]$. Одну такую перестановку, т. е. передвижение всех элементов одной строки на одну позицию по столбцам назад, можно осуществить с помощью цикла от 1 до m . Для перестановки всех строк от последней строки матрицы до вставляемой строки необходим еще один (внешний) цикл по строкам от n -й до $(k+1)$ -й. Для вставки элементов столбца номер l , формируемого из элементов одномерного массива $c[1:n]$, необходимо организовать два аналогичных цикла. Алгоритм имеет следующий вид:

алг СтрСтб (арг цел n, m, k, l , вещ таб $a[1:n, 1:m], b[1:m], c[1:n]$, рез вещ таб $a[1:n+1, 1:m+1]$)

дано | матрица A из элементов $a[i, j], i = 1, 2, \dots, n; j = 1, 2, \dots, m$

надо | вставить новую строку номер k и новый столбец номер l

нач | начало алгоритма и описания вспомогательных переменных

цел i, j | i — номер передвигаемой назад строки
| j — номер передвигаемого назад столбца

нц для i от k до n | цикл передвижения строк (от строки номер n)

нц для j от 1 до m | сдвиг элементов $(n-i+k)$ -й строки

$a[n-i+k+1, j] := a[n-i+k, j]$ | элемент столбца следующей строки меняем на элемент в предыдущей

кц | конец цикла передвижения элементов текущей строки

кц | конец цикла передвижения всех строк

нц для i от 1 до m | цикл вставки новой k -й строки

$a[k, i] := b[i]$ | новая строка из элементов массива b

кц | конец цикла формирования элементов новой строки A

$n := n + 1$ | число строк увеличиваем на 1 (на вставленную строку)

нц для j от 1 до m | цикл передвижения столбцов с номера l

нц для i от 1 до n | цикл сдвига элементов $(j+1)$ -го столбца

$a[i, m - j + l + 1] := a[i, m - j + l]$ | элемент строки
 | меняем на элемент в предыдущем столбце
кц | конец цикла передвижения элементов текущего столбца
кц | конец цикла передвижения всех столбцов
нц для i от 1 до n | цикл вставки нового столбца на место l -го
 $a[k, l] := b[l]$ | новый столбец из элементов массива c
кц | конец цикла формирования элементов нового столбца A
 $m := m + 1$ | число столбцов увеличиваем на 1
кон | конец алгоритма.

Это простой алгоритм. Число явных операций присваивания (т. е. не учитывая неявных, используемых для увеличения параметров цикла на 1 в заголовках циклов), как нетрудно вычислить, равно $2nm - km - ln + 2m + 2n + 2$.

ПРИМЕР

Дана квадратная матрица $x[1:n, 1:n]$. Составим алгоритм нахождения значения такого элемента и его индексов, который является наибольшим среди элементов главной (ведущей из левого верхнего угла матрицы в правый нижний угол) и побочной (ведущей из правого верхнего угла в нижний левый угол) диагоналей. Как сказано выше, зная номер строки любого элемента этих двух диагоналей, можно автоматически определять и номер столбца, на пересечении которых стоит элемент. Поэтому при работе с такими элементами нет необходимости организовывать второй цикл — по столбцам матрицы, достаточен цикл по строкам (в котором вычисляем индекс столбца j). С учетом этого алгоритм имеет вид:

алг МаксЭлемДиагон (арг цел n , вещ таб $x[1:n, 1:n]$, рез цел k, l , вещ m)

дано | квадратная матрица x размерностью n строк и n столбцов

надо | найти максимальный элемент m среди элементов главной и побочной диагоналей матрицы, а также его индексы k и l

нач | начало алгоритма и описание вспомогательных переменных

цел i , | i — переменная цикла по строкам матрицы

$m := x[1, 1]$ | начальное значение максимума

$k := 1$ | номер строки исходного начального максимума

$l := 1$ | номер столбца начального максимума

нц для i от 1 до n | цикл перебора всех строк матрицы

если ($m < x[i, l]$) | если m меньше элемента диагонали

то $m := x[i, l]$ | то меняем его на текущий элемент

$k := i$ | и затем фиксируем номер его строки и

$l := i$ | номер его столбца

все | конец проверки элемента на главной диагонали

если ($m < x[l, n - i + 1]$) | m меньше элемента побочной?

то $m := x[l, n - i + 1]$ | меняем его значение

$k := l$ | фиксируем номер его строки и

$l := n - i + 1$ | номер его столбца

все | конец сравнения текущих элементов с m
кц | конец цикла перебора строк матрицы
кон | конец алгоритма.

ПРИМЕР

Дана квадратная матрица $x[1:n, 1:n]$. Составим алгоритм нахождения среднего геометрического положительных элементов, находящихся под главной диагональю (не включая ее элементы). Нетрудно заметить, что под главной диагональю находятся элементы $x[i, j]$, обладающие следующим свойством: индекс строки элемента больше, чем индекс столбца. Если индекс строки таких элементов может меняться от $i = 2$ до $i = n$, то индекс столбца элемента строки i в указанном множестве элементов может изменяться соответственно от $j = 1$ до $j = i - 1$ (начало идет от $i = 2$ из-за того, что элементы диагонали не включаются, т. е. из первой строки в рассматриваемое множество не попадает ни один элемент). Таким образом, нет необходимости перебирать все $n \cdot n$ элементов матрицы — достаточно перебрать $n \cdot (n - 1) / 2$ элементов (объясните почему). Для больших n это может значительно сократить число операций. С учетом этого алгоритм примет вид:

алг СреднГеомПоложитЭл (арг цел n , вещ таб $x[1:n, 1:n]$, рез вещ p)

дано | квадратная матрица x размерности n строк и n столбцов
надо | найти среднее геометрическое положительных элементов под главной диагональю

нач | начало алгоритма и описание переменных

цел i , | i — переменная цикла по строкам матрицы

j , | j — переменная цикла по столбцам матрицы

k | k — счетчик числа положительных элементов

$p := 1$ | начальное значение произведения элементов по условию

$k := 0$ | начальное значение счетчика числа элементов по условию

нц для i от 2 до n | цикл перебора всех строк матрицы

нц для j от 1 до $i - 1$ | цикл перебора элементов строки
 | до предшествующего элемента на главной
 | диагонали

если $x[i, j] > 0$ | если текущий элемент положителен,

то $p := p \cdot x[i, j]$ | то домножаем на него p

$k := k + 1$ | и увеличиваем счетчик k

все | конец проверки на положительность элемента

кц | конец цикла перебора всех строк матрицы

кц | конец цикла перебора элементов текущей строки

если $k > 0$ | если найден хоть один элемент по условию

то $p := p^{**}(1/k)$ | то находим среднее геометрическое

иначе $p := 0$ | иначе среднее геометрическое равно 0

все | конец нахождения среднего геометрического элементов

кон | конец алгоритма.

ПРИМЕР

Дана квадратная целочисленная матрица $x[1:n, 1:n]$. Составим алгоритм нахождения среднего арифметического нечетных элементов, расположенных над главной диагональю, включая и элементы диагонали. Нетрудно заметить, что на главной диагонали и над ней все элементы $x[i, j]$ обладают следующим свойством: индекс столбца элемента больше или равен индексу строки, т. е. если индекс i строки таких элементов может меняться от $i = 1$ до $i = n$, то индекс j столбца любого элемента строки i в указанном множестве элементов может изменяться соответственно от $j = i$ до $j = n$ (от $i = 1$ идет начало из-за того, что элементы самой диагонали включаются в рассматриваемое множество). Таким образом, нет никакой необходимости перебирать все $n \cdot n$ элементы матрицы — достаточно перебрать $n \cdot (n + 1) / 2$ элементов (почему?). При больших значениях n это значительно может сократить число операций. Итак, алгоритм примет вид:

алг СреднАрифмНечет (арг **цел** n , **цел** таб $x[1:n, 1:n]$, **рез** **вещ** s)
дано | квадратная матрица x размерности n строк и n столбцов
надо | найти s — среднее арифметическое нечетных элементов над главной диагональю матрицы (включая ее)
нач | начало алгоритма и описание вспомогательных переменных
 цел i , | i — переменная цикла по строкам матрицы
 цел j , | j — переменная цикла по столбцам матрицы
 цел k | k — счетчик положительных элементов по условию
 $s := 0$ | начальное значение суммы элементов по условию
 $k := 0$ | начальное значение количества элементов по условию
нц для i **от** 2 **до** n | цикл перебора всех строк матрицы
 нц для j **от** i **до** n | цикл рассмотрения элементов строки начиная от элемента на главной диагонали
 если $\text{mod}(x[i, j], 2) = 1$ | если элемент нечетен
 то $s := s + x[i, j]$ | то изменяем s
 $k := k + 1$ | и увеличиваем значение k
 все | конец проверки на нечетность текущего элемента
 кц | конец цикла перебора всех строк матрицы
кц | конец цикла перебора элементов текущей строки
если $k > 0$ | если был найден хоть один элемент по условию за-
 | дачи, то
 то $s := s/k$ | находим среднее арифметическое
все | конец нахождения среднего арифметического
кон | конец алгоритма.

ПРИМЕР

Дана квадратная матрица $x[1:n, 1:n]$. Составим алгоритм занесения в отдельный одномерный массив y всех элементов матрицы x , лежащих под побочной диагональю, включая элементы и самой диагонали. Под побочной диагональю (включая ее элементы) все элементы $x[i, j]$ имеют следующее свойство: если индекс строки i таких элементов может меняться от $i = 1$ до $i = n$, то индекс столбца j любого элемента строки i в указанном множестве элементов может изменяться соответственно от $j = n - i + 1$ до $j = n$ (если элементы самой диагонали не будут включены, то i изменяется от 2 до n , а j изменялось бы от $n - i + 2$ до $j = n$). Таким образом, нет необходимости перебирать все n^2 элементов матрицы — достаточно перебрать $n*(n+1)/2$ элементов (или же $n*(n-1)/2$ элементов, если элементы диагонали не включаются) (объясните почему), что при больших значениях n может значительно сократить число операций. С учетом сказанного «понятный» алгоритм примет вид:

алг ЭлПодПобОтд (арг цел n , вещ таб $x[1:n, 1:n]$, рез вещ $y[1:n*(n+1)/2]$)

дано | квадратная матрица x размерности n строк и n столбцов

надо | занести в y элементы x на и под побочной диагональю x

нач | начало алгоритма и описание переменных

цел i , | переменная цикла по строкам матрицы

j , | переменная цикла по столбцам матрицы

k | текущий номер заполняемого места массива y

$p := 1$ | начальное значение произведения элементов по условию

$k := 0$ | число элементов матрицы над диагональю

нц для i от 1 до n | цикл перебора всех строк матрицы x

нц для j от $n - i + 1$ до n | цикл для элементов строки
 от $x[i, n - i + 1]$ на побочной диагонали

$k := k + 1$ | вычисление номера текущего y

$y[k] := x[i, j]$ | заполнение нового элемента y

кц | конец цикла перебора всех строк матрицы

кц | конец цикла перебора элементов текущей строки

кон | конец алгоритма.

Можно предложить эффективный алгоритм, непосредственно вычисляя номера элементов по номерам i и j соответствующего элемента массива x .

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое двухмерный массив, каковы его синонимы? Приведите примеры.
2. Что такое индекс элемента двухмерного массива?
3. Как описываются матрицы на учебном алгоритмическом языке? Приведите примеры описания массивов.

4. Как описывается матрица на языке программирования? Приведите пример.
5. Когда можно считать две матрицы полностью идентичными?
6. Какие операции определены относительно матриц (как операндов операции)? Приведите примеры выполнения этих операций.
7. Какие операции определены относительно элементов матриц (как операндов операции)? Приведите примеры.

ЗАДАЧИ

1. Составьте алгоритм нахождения минимального значения из всех максимумов по строкам заданной матрицы.
2. Составьте алгоритм нахождения суммы положительных элементов по каждой четной и нечетной строкам заданной матрицы отдельно.
3. Составьте алгоритм упорядочивания по возрастанию максимальных во всех строках матрицы элементов.
4. Составьте алгоритм нахождения суммы по четным и произведению по нечетным столбцам заданной произвольной матрицы.
5. Составьте алгоритм нахождения произведения всех элементов матрицы, не превосходящих среднего арифметического всех ее элементов.
6. Составьте алгоритм удаления из матрицы данной строки (столбца).
7. Составьте алгоритм вставки в матрицу строки (столбца) в заданное место.

ТЕСТ

1. Фрагмент нц для i от 1 до $n + 1$; $c := a[i, l]$; $a[i, l] := a[i, n - i + 1]$; $a[i, n - i + 1] := c$; кц; кц:
 а) изменит порядок строк в таблице $A = \{a[i, j], i, j = 1, 2, \dots, n\}$;
 б) изменит порядок столбцов A ;
 в) поменяет местами элементы диагоналей;
 г) аварийно прерывается.
2. Фрагмент нц для i от 1 до n ; $i := i + 1$; $c := a[i, l]$; $a[i, l] := a[k, l]$; $a[k, l] := c$; кц:
 а) меняет порядок строк в таблице $A = \{a[i, j], i, j = 1, 2, \dots, n\}$;
 б) меняет элементы главной диагонали и k -й строки в A ;
 в) меняет местами элементы k -го столбца и побочной диагонали A ;
 г) аварийно прерывается.
3. Фрагмент нц для i от 1 до n ; $c := a[i, n - i + 1]$; $a[i, n - i + 1] := a[i, l]$; $a[i, l] := c$; кц; меняет:
 а) порядок строк в таблице $A = \{a[i, j], i, j = 1, 2, \dots, n\}$;
 б) порядок столбцов A ;
 в) местами элементы главной диагонали и побочной диагонали A ;
 г) местами элементы i -й строки и j -го столбца A .

4. Фрагмент $s := 0$ нц для i от 1 до n ; нц для j от 1 до $i-1$; $s := s + a[i, j]$; кц; кц определяет сумму s элементов матрицы $A = \{a[i, j], i, j = 1, 2, \dots, n\}$:
- на главной диагонали;
 - ниже главной диагонали;
 - выше главной диагонали;
 - на побочной диагонали.
5. Фрагмент $p := 1$ нц для i от 1 до n ; нц для j от $n-i+2$ до n ; если $a[i, j] \neq 0$ то $p := p * a[i, j]$; кц; кц находит для матрицы $A = \{a[i, j], i, j = 1, 2, \dots, n\}$ произведение p элементов:
- ненулевых, расположенных строго ниже побочной диагонали;
 - всех ненулевых;
 - всех положительных;
 - всех на побочной диагонали.
6. Фрагмент $s := 0$ нц для i от 1 до n ; если $a[i, j] = 0$ или $a[i, n-i+1] = 0$ то $s := s + a[i, j] + a[i, n-i+1]$; кц находит для матрицы $A = \{a[i, j], i, j = 1, 2, \dots, n\}$ сумму s ненулевых элементов на:
- побочной диагонали;
 - главной диагонали;
 - главной или побочной диагонали;
 - главной и побочной диагоналях.
7. Фрагмент нц для i от 1 до n ; нц для j от 1 до n ; $k := a[i, j]$; $a[j, j] := a[i, j]$; $a[i, j] := k$; кц; кц:
- транспонирует (строки делает столбцами и наоборот) матрицу $a(n \times n)$;
 - не приводит к изменению матрицы a ;
 - осуществляет циклический сдвиг строк матрицы на k позиций вниз;
 - меняет строки с номером k на столбец с тем же номером.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
г	г	в	б	а	г	б

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

- Атрибуты и операции для матриц.
- Алгоритмы обхода элементов матрицы по строкам и столбцам.
- Симметричные матрицы, их хранение и обработка.
- Трехдиагональные матрицы, их хранение и обработка.
- Треугольные матрицы, их хранение и обработка.
- Теоремы алгоритмики матриц.
- Классификация задач на использование матриц.

А

АЛГОРИТМЫ РАБОТЫ С ТЕКСТАМИ

Символ (синонимы: **буква, литера**) — любой знак, допускаемый для записи сообщений на данном языке кодирования этих сообщений, для описания переменных, значения которых — один символ. Используется ключевое слово **сим**.

Текст — некоторая последовательность символов (литер), допустимых в рассматриваемом алфавите, множестве символов.

ПРИМЕР

Если рассматривается компьютерный ввод текста, то допустим любой набор знаков, встречающихся на клавишах (с небольшим исключением). Вводимая последовательность знаков может иметь смысл, например "Утро", или же не иметь, например "ее + 0".

Для описания переменных, значениями которых могут быть лишь тексты, используется ключевое слово **лит**. Текстовые (символьные) константы обычно заключаются в апострофы.

ПРИМЕР

В алфавите арифметики символами будут, например, знаки 0, -, +, 9,), =. Числа 17, -9, 1.9 не являются символами данного алфавита, так как представляют собой комбинации из двух или трех символов этого алфавита, т. е. являются текстами в этом алфавите. Примерами текстов в этом алфавите могут быть: 001, -9.98 + 89.34, $2 + 2 = 4$, (45-5.7) : 67.7.

Текстовые константы вида "АРБА", "2 + 2", "-8", "8 - 9 = -1".

Если переменная имеет описание **лит**, то она может принимать значение и пустого слова, и только одного символа. Если же переменная имеет описание **сим**, то принимать значение слова более чем из одного символа она уже не может.

ПРИМЕР

Корректны описания
и присваивания:

сим a, b, c, **лит** x, y

a: = "a"

b: = "c"

c: = " + "

x: = "x + y = 4"

y: = x + " x = 4".

Некорректны описания
и присваивания (почему?):

сим a, b, c, **лит** x, y

a: = ""

b: = "ac"

c: = " + / - "

x: = x + y = 4

y: = x" + "y.

Тексты представляются в ЭВМ, как и числа, битовыми комбинациями.

Для работы с текстами используют следующие основные операции и процедуры: **конкатенации** (сцепления) текстов, **выявления** длины текста, **выделения** части текста (например, символа).

ПРИМЕР

Конкатенация двух текстовых констант $a = \text{"Паро"}$ и $b = \text{"ход"}$ дает текстовую константу $c = a + b = \text{"Пароход"}$, а применение функции (процедуры) «длина» к c дает значение 7: длина (c) = 7. Пробел также считается символом. Операция выделения 4-го символа c дает результат $c[4] = \text{"о"}$; операция выделения подтекста, начинающегося с 4-го и заканчивающегося 6-м символом, дает результат $c[4:6] = \text{"охо"}$. Этот результат совпадает с результатом конкатенации вида $c[4] + c[5] + c[6] = \text{"охо"}$.

ПРИМЕР

Составим алгоритм подсчета числа несовпадающих символов (на одинаковых позициях) в двух заданных текстах a и b :

алг ПодсчетНесовпадений (арг лит a, b , рез цел k)
дано | тексты a и b
надо | подсчитать k — число несовпадающих символов в a и b
нач | начало алгоритма
цел i, j | переменная цикла сравнения символов в текстах a и b
 j | минимальное значение из длин (в символах) a и b
 $k := \text{abs}(\text{длина}(a) - \text{длина}(b))$ | начальное значение счетчика
 $j := \min(\text{длина}(a), \text{длина}(b))$ | выбираем минимум из длин a и b
нц для i от 1 до j | цикл сравнения символов a и b на i -х позициях
если ($a[i] \neq b[i]$) | если эти символы не равны
то $k := k + 1$ | то увеличиваем счетчик k на 1
все | конец условной структуры
кц | конец цикла
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм, который выясняет, сколько раз заданная буква a входит в заданный текст t . Алгоритм имеет вид:

алг ЧислоВхожденийБуквы (арг лит t , сим a , рез цел k)
дано | текст t , буква a
надо | выяснить, сколько раз a входит в t
нач | начало алгоритма и описания переменных
цел i, n | i — номер текущего символа
 n | n — длина данного текста
 $k := 0$ | счетчик числа символов в тексте вначале обнуляем
 $n := \text{длина}(t)$ | нахождение длины текста t
нц для i от 1 до n | цикл перебора текста
если $t[i] = a$ | если текущая буква совпадает с заданной

то $k := k + 1$ | то увеличиваем число совпадений
 все | конец проверки
 кц | конец цикла перебора символов в тексте
 кон | конец алгоритма.

ПРИМЕР

Составим алгоритм удаления из заданного текста a символов, входящих в заданный текст b . Алгоритм имеет вид:

алг УдалениеВстречающихсяСимволов (арг лит a , b , рез лит a)
 дано | текст a и текст b
 надо | из текста a удалить все символы, входящие в текст b
 нач | начало алгоритма
 цел i , | количество символов текста a
 j , | количество символов текста b
 m | длина текста a
 $m := \text{длина}(a)$ | переменной присваивается длина текста a
 нц для i от 1 до m | перебор элементов текста a
 нц для j от 1 до $\text{длина}(b)$ | перебор символов текста b
 если $(a[i] = b[j])$ | если символ a совпадает с символом b
 то | организуем цикл удаления i -го символа a
 нц для k от i до $m - 1$ | цикл сдвигает все
 $a[k] := a[k + 1]$ | символы a от k -го вперед
 кц | конец цикла удаления символа из a
 все | конец проверки совпадения символов
 $m := m - 1$ | уменьшение длины из-за удаленного символа
 кц | конец цикла по элементам текста b
 кц | конец цикла по элементам текста a
 кон | конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения длины самого длинного слова в тексте a , если слова разделены лишь одним пробелом. Алгоритм имеет вид:

алг МаксимДлинноеСлово1 (арг лит a , рез цел n)
 дано | текст a , слова в котором отделены одним пробелом
 надо | найти длину самого длинного слова в тексте a
 нач | начало алгоритма и описания переменных
 цел i , | номер текущего символа в тексте
 $k := 0$ | обнуляем счетчик символов слова
 $n := 0$ | обнуляем счетчик всего текста
 нц для i от 1 до $\text{длина}(a)$ | цикл перебора символов в a
 если $(a[i] \neq " ")$ | если символ не равен пробелу
 то $k := k + 1$ | то считаем количество символов
 иначе $n := \max(n, k)$ | выбираем максимальное
 $k := 0$ | обнуляем счетчик длины нового слова
 все | конец подсчета символов
 кц | конец цикла перебора
 кон | конец алгоритма.

ПРИМЕР

Составим алгоритм замены всех окончаний «ый» на «ая» в заданном тексте. Алгоритм имеет вид:

алг ЗаменаОкончаний (арг лит a , рез лит a)
дано | текст a , слова в котором отделены одним пробелом
надо | заменить все окончания «ый» на «ая» в тексте a
нач | начало алгоритма и описания переменных
 цел i | номер текущего символа в тексте
 $a := a + " "$ | для проверки слова в конце добавляем пробел
 нц для i от 1 до $\text{длина}(a) - 3$ | цикл перебора всех символов a
 если $a[i:i+2] = \text{"ый"}$ | если 3 символа — «ый» и пробел
 то $a := a[i:i-1] + \text{"ая"} + a[i+2:\text{длина}(a)]$ | то меняем
 все | конец замены символов
 кц | конец цикла перебора
 кон | конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения в заданном тексте числа всех подтекстов заданной длины:

алг ЧислоПодтекстов (арг цел d , лит t , рез цел n)
дано | задан текст t из подтекстов, разделенных пробелами
надо | найти число подтекстов заданной длины d в тексте t
нач | начало алгоритма
 цел i , | переменная цикла для перебора всех символов текста t
 m , | переменная подсчета символов выделяемого подтекста
 сим x | вспомогательная переменная, текущий символ t
 $m := 0$ | начальное значение счетчика символов в подтексте
 нц для i от 1 до $\text{длина}(t)$ | заголовок цикла выбора символа
 $x := t[i]$ | присваиваем x значение символа на i -м месте в t
 если $(x \neq " ")$ | если символ x не пробел
 то $m := m + 1$ | то число символов подтекста увеличиваем
 иначе | иначе (x — пробел) сравниваем длину с d
 если $m = d$ | если длина подтекста равна d , то
 то $n := n + 1$ | увеличиваем число подтекстов
 все | конец внутренней условной структуры
 $m := 0$ | обнуляем длину подтекста снова
 все | конец проверки, является ли символ пробелом
 кц | конец цикла
 кон | конец алгоритма.

ПРИМЕР

Составим алгоритм вставки в заданный текст t заданного текста p с заданной позиции n :

алг ВставкаТекстаВПозицию(арг цел n , лит t , p)
дано | текст t , вставляемый текст (подтекст) p , позиция вставки n
надо | вставить в текст t с позиции номер n заданный текст p
нач | начало алгоритма

$t := t[1:n-1] + p + t[n:\text{длина}(t)]$ | вырезаем из t подтекст
 | от 1-го до $(n-1)$ -го символа, к нему «цепляем» p и затем «цепляем»
 | подтекст $t[n:m]$ от n -го до последнего символа текста
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм удаления из заданного текста t его подтекста от заданной позиции n до заданной позиции m :

алг УдалениеПодтекста (арг цел n, m , лит t)
дано | текст t , позиции n начала и m конца удаляемой части
надо | удалить из t подтекст от позиции n до позиции m
нач | начало алгоритма
цел k | k — переменная, запоминающая число символов в t
 $k := \text{длина}(t)$ | находим число символов в t , включая пробел
 $t := t[1:n-1] + t[m+1:k]$ | вырезаем от 1-го до $(n-1)$ -го символа и «цепляем» часть от $(m+1)$ -го символа до последнего
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм, который в заданном тексте t вместо всех слов на букву «а» вставляет заданное слово. Алгоритм имеет вид:

алг ВставкаСловаВместоБуквы (арг лит s, t , рез лит t)
дано | текст t , слово s
надо | в текст t вместо буквы «а» вставлять слово s
нач n | длина текста t
 $n := \text{длина}(t)$ | длина текста t
 $i := 1$ | начальное значение символа данного слова
нц пока $(i \leq n)$ | цикл перебора текста
 если $(t[i] = \text{«а»})$ | если текущая буква есть «а»
 то $t := t[1:i-1] + s + t[i+1:n]$ | меняем букву «а» на s .
 все | конец проверки на букву «а»
кц | конец цикла перебора символов в тексте
кон | конец алгоритма.

ПРИМЕР

Составим алгоритм нахождения текста, записанного в обратном порядке:

алг ИнвертированиеТекста (арг лит a)
дано | произвольный текст a
надо | записать a в обратном порядке
нач | начало алгоритма
цел i , | переменная цикла
 n , | вычисляемая длина текста
сим b , | буфер перестановки симметричных элементов
 $n := \text{длина}(a)$ | вычисляем длину текста

<u>если</u> $n > 1$		если «есть что переставлять», то
<u>то</u> <u>нц</u> <u>для</u> i <u>от</u> 1 <u>до</u> $\text{int}(n/2)$		цикл перестановки
$b := a[i]$		запоминаем текущий символ
$a[i] := a[n - i + 1]$		переставляем симметричный
$a[n - i + 1] := b$		передвигаем запомненный
<u>кц</u>		конец цикла перестановки
<u>все</u>		конец условной структуры
<u>кон</u>		конец алгоритма.

ПРИМЕР

Составим алгоритм подсчета числа вопросительных, восклицательных и утвердительных предложений в тексте:

алг ЧислоПредложений (арг цел n , лит таб $b[1:n]$, рез цел k, m)

дано | текст, состоящий из n предложений $b[i]$, $i = 1, 2, \dots, n$

надо | найти число вопросительных, восклицательных предложений

нач | начало алгоритма

цел i , | i — переменная цикла перебора предложений

k , | k — переменная для числа вопросительных

m , | m — переменная для числа восклицательных

d , | d — переменная для вычисления длины предложения

лит w , | w — рабочая переменная

$k := 0$ | обнуление числа вопросительных предложений

$m := 0$ | обнуление числа восклицательных предложений

нц для i от 1 до n | заголовок цикла просмотра предложений $b[i]$

$w := b[i]$ | выбор предложения $b[i]$ заданного текста

$d := \text{длина}(w)$ | находим длину текущего предложения $b[i]$

$w := w[d:d]$ | вырезаем из текущего w последний знак

если $w = "?"$ | если последний символ — это «?»

то $k := k + 1$ | то увеличиваем счетчик вопросительных

все | конец условия

если $w = "!"$ | аналогично поступаем с восклицательными

то $m := m + 1$

все

кц | конец цикла

кон | конец алгоритма.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое текст, каковы его синонимы? Приведите примеры.
2. Чем отличается текстовая переменная от символьной?
3. Чем отличается текстовая переменная от числовой?
4. Чем отличается текстовая переменная от логической?
5. Что такое конкатенация текстов? Коммутативна ли она?
6. Каковы операции, позволяющие работать с текстами?
7. Как представляются в ЭВМ тексты? Приведите примеры.

ЗАДАЧИ

1. Составьте алгоритм подсчета количества вхождений в заданный текст заданного символа.
2. Составьте алгоритм подсчета количества вхождений в заданный текст заданного слова.
3. Составьте алгоритм, который удаляет в заданном тексте все символы, совпадающие с одним из заданных символов. Заданные символы перечислены в символьном одномерном массиве.
4. Составьте алгоритм поиска в заданном тексте слова наибольшей длины, если слова разделены одним пробелом.
5. Составьте алгоритм, который в заданном тексте все окончания «ый» слов текста (слова разделены несколькими пробелами) меняет на окончание «ая».
6. Составьте алгоритм вычисления количества знаков препинания в тексте (литерной строке длиной не более чем 255 символов).
7. Составьте алгоритм лексикографического (алфавитного) упорядочивания заданных слов по первым символам алфавита.

ТЕСТ

1. С текстовыми данными нельзя выполнять операцию (функцию):
а) конкатенации; в) вычисления длины;
б) сложения; г) проверки на совпадение с буквой.
2. С текстовыми данными можно выполнять операцию:
а) умножения; в) увеличения на единицу;
б) уменьшения на единицу; г) уменьшения длины.
3. Результат конкатенации $C = A + B$ текстов $A = \text{"кот"}$, $B = \text{"кто"}$ равен:
а) $C = \text{"кто кот"}$; в) $C = \text{"кот кто?"}$;
б) $C = \text{"кот кто"}$; г) $C = \text{"коткто"}$.
4. Фрагмент $x := \text{длина}(y)$; $i := 1$; нц пока ($i \leq x$); $a := y[i]$; $y[i] := y[n - i + 1]$; $y[n - i + 1] := a$; $i := i + 1$; кц; вывод(z); выдаст результат z при входном параметре $y = \text{"informatics"}$:
а) $z = \text{"informatics"}$; в) $z = \text{"scitamatics"}$;
б) $z = \text{"inform"}$; г) $z = \text{"mrofni"}$.
5. Фрагмент $x := \text{длина}(y)$; $i := 1$; нц пока ($i \leq x/2$); $a := y[i]$; $y[i] := y[n - i + 1]$; $y[n - i + 1] := a$; $i := i + 1$; кц; вывод(z); выдаст результат z при входном параметре $y = \text{"informatics"}$:
а) $z = \text{"informatics"}$; в) $z = \text{"scitamrofni"}$;
б) $z = \text{"inform"}$; г) $z = \text{"mrofni"}$.
6. Фрагмент $a := \text{" " + } a$; $k := 0$; нц, для i от 1 до $\text{длина}(a) - 1$; если ($a[i] = \text{" "}$) и ($a[i + 1] = B$) то $k := k + 1$; все; кц; вывод($"k = "$); для текста a из слов, разделенных одним пробелом, выводит количество k всех:
а) слов текста A , начинающихся на букву B ;
б) слов текста A , заканчивающихся на букву B ;

- в) слов текста A , начинающихся на букву B ;
 г) слов текста A , заканчивающихся на букву B .
7. Фрагмент $B := " " + B$; $m := \text{длина}(A)$; $n := \text{длина}(B)$; $k := 0$; нц
 для i от 1 до $n - m + 1$; если ($B[i] = " "$) и ($B[i + 1 : i + m - 1] = B$)
 и ($B[i + m] = " "$) то $k := k + 1$; кц; вывод(" $k = "$ ") выводит число k
 всех вхождений:
 а) текста A в текст B ; в) слов текста A в текст B ;
 б) текста B в текст A ; г) слов текста B в текст A .

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
б	г	г	а	в	а	б

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Текстовые данные, их атрибуты и особенности.
2. Конкатенация и селекция.
3. Вставка и удаление.
4. Стандарты представления алфавитно-цифровой информации.
5. Упаковка и распаковка как универсальные процедуры.
6. Стековое представление текстовых выражений.
7. Тип лит (string) и его особенности.

10

АЛГОРИТМЫ РАБОТЫ С ПРЕДИКАТАМИ

Для выполнения различных операций, связанных с обработкой логических данных, используются соответствующие алгоритмы, называемые алгоритмами работы с предикатами или логическими данными. Этот класс алгоритмов очень важен при решении задач поиска, упорядочивания и т. д.

Предикаты в алгоритмах используются не только в условных структурах, «флагах» и циклах типа «пока», но и при решении различных логических задач.

ПРИМЕР

Дана целочисленная матрица $x[1:n, 1:m]$. Логическая переменная k принимает значение «истина» лишь только тогда, когда существует хотя бы один элемент массива x , кратный заданному числу y . Алгоритм решения задачи имеет вид:

алг ЕстьКратныйИзМассива?(арг цел n, m , таб $x[1:n, 1:m]$, рез лог k)

дано | целочисленная таблица x из n строк и m столбцов

надо | найти значение логической переменной k

нач | начало алгоритма

цел i , | номер текущей строки матрицы

j , | номер текущего столбца матрицы

$i := 1$ | номер первой рассматриваемой строки

$j := 1$ | номер первого элемента первой строки

нц пока $(i \leq n) \text{ и } (\text{mod}(x[j, i], y) \neq 0)$ | пока не дошли до конца или не
| встретили первый кратный y элемент матрицы x

$j := j + 1$ | номер элемента текущего столбца

нц пока $(j \leq m) \text{ и } (\text{mod}(x[j, i], y) \neq 0)$ | поиск кратного y эле-
| мента в текущей строке

$j := j + 1$ | переход к элементу следующего столбца

кц | конец цикла проверки элементов текущей строки

$i := i + 1$ | переход к следующей строке

кц | конец цикла перебора строк

если $(i > n) \text{ или } (j > m)$ | если найден хотя бы один элемент

то $k := \text{«истина»}$ | то $k = \text{«истина»}$

иначе $k := \text{«ложь»}$ | иначе $k = \text{«ложь»}$

все | конец проверки, найден ли был элемент, кратный y

кон | конец алгоритма.

ПРИМЕР

Задано число a и уравнения вида $kx + b = 0$, $cx^2 + dx + f = 0$. Определить, является ли $x = a$ корнем одного из этих уравнений, т. е. найти $z1, z2$, принимающие значения текстов «корень», «не корень». Составим алгоритм решения этой задачи:

алг КореньУравнений? (**арг** вещ a, k, b, c, d, f , **рез** лог $z1, z2$)
дано | число a и уравнения вида $kx + b = 0$, $cx^2 + dx + f = 0$
надо | является $x = a$ корнем одного из этих уравнений, т. е.
| найти $z1, z2$ со значениями «истина» (корень) или «ложь»
нач | начало тела алгоритма
если ($a*k + b = 0$) | проверяем, корень ли первого уравнения
 то $z1 := \text{«истина»}$ | a — корень уравнения $kx + b = 0$
 иначе $z1 := \text{«ложь»}$ | a — не корень уравнения $kx + b = 0$
все | конец проверки для первого уравнения
если ($c*a**2 + d*a + f = 0$) | проверка для второго уравнения
 то $z2 := \text{«истина»}$ | a — корень уравнения $cx^2 + dx + f = 0$
 иначе $z2 := \text{«ложь»}$ | a — не корень уравнения $cx^2 + dx + f = 0$
все | конец проверки для второго уравнения
кон | конец алгоритма.

ПРИМЕР

На плоскости заданы три произвольные точки $A(x1; y1)$, $B(x2; y2)$, $C(x3; y3)$. Существует ли треугольник ABC ? Можно предложить несколько различных алгоритмов проверки существования треугольника, но мы выберем наиболее простой (но не наиболее грамотный и четкий с точки зрения геометрии!) алгоритм, основанный на том факте, что если треугольник существует, то его площадь не равна нулю. Площадь будем находить по формуле Герона. Алгоритм решения этой задачи будет иметь вид:

алг СуществуетТреуг?(**арг** вещ $x1, y1, x2, y2, x3, y3$, **рез** лог q , вещь p, s)
дано | три произвольные точки $A(x1; y1)$, $B(x2; y2)$, $C(x3; y3)$
надо | если треугольник ABC существует ($q = \text{«истина»}$)
| то вычислим его периметр p и площадь s
нач | начало тела алгоритма
 вещь a , | длина отрезка AB
 b , | длина отрезка BC
 c , | длина отрезка AC
 p , | периметр треугольника ABC
 s | площадь треугольника ABC
 $a := \text{sqrt}((x1 - x2)**2 + (y1 - y2)**2)$ | находим сторону AB
 $b := \text{sqrt}((x2 - x3)**2 + (y2 - y3)**2)$ | находим сторону BC
 $c := \text{sqrt}((x1 - x3)**2 + (y1 - y3)**2)$ | находим сторону AC
 $p := (a + b + c)/2$ | находим полупериметр треугольника
 $s := \text{sqrt}(p*(p - a)*(p - b)*(p - c))$ | формула Герона

если $(s = 0)$ | если площадь нуль, то треугольник не существует
 то $q := \text{"ложь"}$
 иначе $q := \text{"истина"}$
 все | конец проверки существования треугольника
 кон | конец алгоритма.

ПРИМЕР

Дан ряд чисел $x[1], x[2], \dots, x[n]$. Составим алгоритм проверки, убывает ли эта последовательность. Алгоритм имеет вид:

алг УбываетЛиРяд? (арг цел n , вещ таб $x[1:n]$, рез лит a)
 дано | ряд чисел $x[1], x[2], \dots, x[n]$
 надо | проверить, убывает ли эта последовательность
 нач | начало тела алгоритма
 цел i | номер числа в последовательности
 $i := 1$ | номер первого элемента в последовательности
 нц пока $(i < n)$ и $(x[i] > x[i + 1])$ | цикл проверки убывания ряда
 $i := i + 1$ | пока убывание есть, перейти к следующему
кц | конец цикла нарушения условий убывания
если $(i = n)$ | если цикл был прерван при достижении конца ряда
 | (второе условие в заголовке не нарушилось)
 то $a := \text{"ряд убывает"}$ | то a присваивается нужный текст
 иначе $a := \text{"ряд не убывает"}$ | a получает другое значение
 все | конец проверки условия убывания ряда
 кон | конец алгоритма.

ПРИМЕР

Дана последовательность чисел $x[1], x[2], \dots, x[n]$. Составим алгоритм определения, образует ли эта последовательность арифметическую или геометрическую прогрессию. Характеристическое свойство арифметической прогрессии:

$$x_i = \frac{x_{i+1} + x_{i-1}}{2}, \quad i = 2, 3, n-1.$$

Характеристическое свойство геометрической прогрессии:

$$x_i = \sqrt{x_{i+1} \cdot x_{i-1}}, \quad i = 2, 3, n-1.$$

Алгоритм имеет вид:

алг Прогрессия? (арг цел n , вещ таб $x[1:n]$, рез лог s, p)
 дано | n вещественных чисел $x[1], x[2], \dots, x[n]$
 надо | образует ли она какую-то прогрессию?
 нач | начало тела алгоритма
 цел i | номер числа в последовательности
 $i := 2$ | начальный элемент
 нц пока $(i < n)$ и $(x[i]^2 = x[i + 1] + x[i - 1])$ | цикл испытания эле-
 | мента ряда на «характеристическое свойство»
 $i := i + 1$ | переход к следующему числу ряда
кц | конец цикла испытания текущего числа ряда

если $(i = n)$ | если цикл был прерван по достижении конца ряда
 | (второе условие в заголовке не нарушилось)
 то $s := \text{"истина"}$ | то прогрессия
 иначе $s := \text{"ложь"}$ | иначе не прогрессия
 все | конец проверки условия убывания ряда
 $i := 2$ | начальный элемент
 нц пока $(i < n)$ и $(x[i]**2 = x[i+1]*x[i-1])$ | цикл испытания
 | элемента ряда на «характеристическое свойство»
 $i := i + 1$ | переход к следующему числу ряда
 кц | конец цикла испытания текущего числа ряда
 если $(i = n)$ | если цикл был прерван по достижении конца ряда
 | (второе условие в заголовке не нарушилось)
 то $p := \text{"истина"}$ | то прогрессия
 иначе $p := \text{"ложь"}$ | иначе не прогрессия
 все | конец проверки условия убывания ряда
 кон | конец алгоритма.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое высказывание, предикат, логическое условие, в чем их алгоритмическое назначение? Приведите примеры.
2. Как используются логические условия в алгоритмах? Классифицируйте их использование по типам. Приведите примеры.
3. Что такое флаг? Каково его назначение? Когда желательно использовать флаг в алгоритмах? Приведите примеры.
4. Что такое логическая структура алгоритма? В чем состоит ее отладка?
5. Как грамотно составлять условие завершения цикла типа «пока»? Как можно использовать аксиомы де Моргана при этом? Приведите примеры.
6. Есть ли чисто логические и чисто арифметические задачи?
7. Что такое двоичный (бинарный) поиск? Приведите примеры.

ЗАДАЧИ

1. Логическая переменная k принимает значение «истина» только тогда, когда в заданной матрице нет отрицательных или нулевых элементов. Составьте алгоритм нахождения значения переменной k для любой заданной матрицы.
2. Логическая переменная k принимает значение «истина» только тогда, когда заданный текст не содержит ни одной десятичной цифры. Составьте алгоритм нахождения значения переменной k для заданного текста.
3. Заданы числа a, k, b, d, m, n и система уравнений вида $kx + b = 0, ax + d = 0$. Определите, является ли пара $x = m, y = n$ корнем системы, т. е. найдите $z1, z2$, принимающие значения «истина», «ложь». Составьте алгоритм.

4. На плоскости заданы три произвольные точки $A(x_1; y_1)$, $B(x_2; y_2)$, $C(x_3; y_3)$. Проверьте, является ли треугольник ABC (если он существует) прямоугольным. Составьте алгоритм.
5. На плоскости заданы три произвольные точки $A(x_1; y_1)$, $B(x_2; y_2)$, $C(x_3; y_3)$. Проверьте, является ли этот треугольник ABC (если он существует) равнобедренным. Составьте алгоритм.
6. На плоскости заданы четыре произвольные точки $A(x_1; y_1)$, $B(x_2; y_2)$, $C(x_3; y_3)$, $D(x_4; y_4)$. Проверьте, можно ли провести через эти точки прямоугольник $ABCD$. Составьте алгоритм.
7. Составьте алгоритм, который проверяет, является ли заданный ряд чисел невозрастающей или неубывающей последовательностью, и выдает соответствующее логическое сообщение.

ТЕСТ

1. Алгоритм — логический (для работы с логическими данными), если в нем:
 - а) есть хотя бы одно логическое условие;
 - б) в основном происходит обработка логических данных;
 - в) есть цикл типа «пока» с некоторым логическим условием;
 - г) вычисляется значение некоторого предиката.
2. Если цикл «пока» должен выполняться до тех пор, пока не выполнится условие A , то предикат заголовка цикла имеет вид:
 - а) ни пока (A) < тело цикла > кц;
 - б) ни пока (A) или не (A) < тело цикла > кц;
 - в) ни пока не (A) < тело цикла > кц;
 - г) ни пока (A) и не (A) < тело цикла > кц.
3. Во фрагменте ни пока ($i < n$) и ($x[i] > x[i+1]$); $i := i + 1$; если ($x[i] < 0$) то $x[i+1] := \text{abs}(x[i])$; все; кц окончание цикла определяется предикатом:
 - а) ($i < n$);
 - б) ($x[i] > x[i+1]$);
 - в) ($x[i] < 0$);
 - г) а и б.
4. Флаг в алгоритмах — это:
 - а) символ при выходе из цикла;
 - б) предикат для завершения цикла;
 - в) управляющая логическая переменная;
 - г) управляющая логическая, числовая или текстовая переменная.
5. Флаг для окончания цикла может быть заменен подходящим, корректным:
 - а) предикатом заголовка цикла типа «пока»;
 - б) циклом типа «до»;
 - в) условным оператором в теле цикла;
 - г) оператором аварийного прерывания цикла.
6. Область, нестрого (с границами) заключенная между окружностями $x^2 + y^2 = 4$ и $x^2 + y^2 = 9$, описывается предикатом:
 - а) ($x^2 + y^2 = 4$) или ($x^2 + y^2 = 9$);
 - б) ($x^2 + y^2 = 4$) $\cdot$ ($x^2 + y^2 = 9$);
 - в) ($x^2 + y^2 \geq 4$) и ($x^2 + y^2 \leq 9$);
 - г) ($x^2 + y^2 = 4$) и не ($x^2 + y^2 = 9$).

7. Правильным утверждением является:
- а) тело цикла должно изменять значение предиката заголовка;
 - б) заголовок цикла должен содержать истинный предикат;
 - в) заголовок цикла при выходе из цикла всегда не определен;
 - г) переменная любого цикла модифицируема в теле цикла.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
б	в	г	г	а	в	а

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Методы и алгоритмы обработки логических данных.
2. Универсальность цикла типа «пока». Циклы с предусловием и циклы с постусловием.
3. Флаги и их использование в алгоритмах.
4. Алгоритмы поиска по образцу.
5. Алгоритмы бинарного поиска.
6. Алгоритмы обработки предикатов.
7. Теоремы об алгоритмах обработки логических данных.

ДАННЫЕ

Данные — сообщения, структуры, относящиеся к одному процессу, объекту. Текущее динамическое состояние данных называют значением данных.

Данные, используемые при решении задач некоторого типа, класса, нельзя организовывать или структурировать без определения и понимания алгоритма решения этого класса задач. В свою очередь, сами методы, а следовательно, реализующие их алгоритмы существенным образом зависят от структуры обрабатываемых ими данных.

Тип данных характеризует множество значений, объектов, к которому принадлежат данные. Типы данных либо задаются перечислением значений этого типа, например **простые типы данных**, либо могут быть образованы объединением ранее определенных каких-то типов в разрешенную корректную структуру — **структурированные типы данных**.

ПРИМЕР

Опишем простые типы данных «специальность» и «пол» и структурированный тип «сотрудник»: специальность = (учитель, врач, рабочий); пол = (мужской, женский); сотрудник = (специальность, пол).

Для всех типов данных определена базовая операция присваивания. Для обозначения конкретных, текущих значений данных используются константы. Любая переменная простого типа может «хранить» в текущий момент времени только одно значение, например скалярное. Часто образуются данные, которые строятся иерархически из простых типов данных.

В языках описания алгоритмов есть так называемые **стандартные типы**, для которых их множество значений не определяют специально, например типы вида цел, вещ, лит, сим, лог и др.

Опишем стандартные простые типы: цел, вещ, сим, лит, лог.

1. Тип «целые числа».

- 1) Имя типа данных: цел.
- 2) Область определения типа: целые числа $z \in \mathbb{Z}$.
- 3) Константы типа: одна десятичная цифра со знаком «-» (или без него) или последовательность десятичных цифр со знаком «-» (или без него).
- 4) Ограничения: ограничения на разрядность определяются только физически (аппаратно).
- 5) Разрешенные для типа операции: $:=$, $+$, $-$, $*$, $**$, $/$, $=$, $\neq$, $<$, $>$, $\leq$, $\geq$.

6) Функции от аргумента типа: `int`, `mod`, `abs`, `sqrt`, `sign`, `sin`, `cos`, `tg`, `ctg`, `arcsin`, `arccos`, `arctg`, `arctgtg`, `ln`, `lg`, `exp`, `div`, `rnd`.

2. Тип «вещественные числа».

1) Имя типа данных: вещ.

2) Область определения типа: вещественные числа $r \in R$.

3) Константы типа: число типа цел, десятичная точка, число типа цел без знака [буква «Е», число типа цел].

4) Ограничения: величина и точность физического представления, аппаратные.

5) Разрешенные для типа операции: `:=`, `+`, `-`, `*`, `**`, `/`, `=`, `≠`, `<`, `>`, `≤`, `≥`.

6) Функции от аргумента типа: как в предыдущем случае, кроме `mod`, `div`.

Тип данных вещ при реализации на уровне представления и физическом уровне может реализоваться в двух формах — с фиксированной точкой и с плавающей точкой.

3. Тип «символьная константа».

1) Имя типа данных: сим.

2) Область определения типа: литеры (символы) языка.

3) Константы типа: литера в кавычках типа (") или (').

4) Ограничения: символ «пробел» не отображается визуально.

5) Разрешенные для типа операции: `:=`, `=`, `≠`, `<`, `>`, `≤`, `≥`.

6) Функции от аргумента типа: длина, [] (вырезка, сечение).

4. Тип «литерная константа».

1) Имя типа данных: лит.

2) Область определения типа: последовательность литер (символов).

3) Константы типа: литерная последовательность в кавычках (") или (').

4) Ограничения: «пробел» не отображается, длина константы ограничена физически (аппаратно).

5) Разрешенные для типа операции: `:=`, `+`, `=`, `≠`, `<`, `>`, `≤`, `≥`.

6) Функции от аргумента типа: длина, [] (вырезка, сечение).

5. Тип «логическая константа».

1) Имя типа данных: лог.

2) Область определения типа: 1 («истина»), 0 («ложь»).

3) Константы типа: 1, 0.

4) Ограничения: неопределенное состояние невозможно.

5) Разрешенные для типа операции: и, или, не, `=`, `≠`, `<`, `>`, `≤`, `≥`.

6) Функции от аргумента типа: отсутствуют.

Массивы (как регулярные структуры данных) можно описать, как и выше.

1) Имя типа данных: цел мас (или вещ мас, сим мас, лог мас, лит мас).

2) Область определения типа: целые (вещественные, символьные, логические, литерные) значения, область определения которых находим, как и для стандартных простых типов.

3) Константы типа: константами могут быть элементы массива.

4) Ограничения: определяются только типами индексных выражений, границами их допустимых изменений (границы типа: цел), однородностью элементов массива.

5) Разрешенные для данного типа операции:

— для работы с объектами-массивами (не во всех языках) — присваивание;

— для работы с элементами массива — операция селекции, ссылки на элемент массива (`[ ]`).

6) Функции от аргумента типа: отсутствуют.

Для представления (единой структурой) совокупности неоднородных данных, связанных между собой общим видом записи и целью, используются записи. Например, информация обо всех учениках какой-то школы представляется записями вида (назовем такую запись — Ученик):

<Ф.И.О.><Класс><Школа><Домашний адрес><Пол>.

Каждая компонента записи называется элементом записи, а его место в записи — полем записи. Каждое поле имеет имя поля и тип поля. Для ссылки на поле записи используется форма:

<имя переменной типа запись>.<имя поля>.

Для идентификации записи используется служебное слово, например в языке Паскаль — `record` (запись).

Например, в записи Ученик ссылка на элемент «Школа» имеет вид: `Ученик.Школа`.

Сослаться на конкретного, *i*-го ученика можно так: `Ученик[i].Школа`.

Структура данных типа «запись (простая)» описывается следующим образом:

1) Имя типа данных: Запись (типы данных).

2) Объявление типа записи:

Имя записи и служебное слово для идентификации записи

поле записи 1: тип элемента поля записи 1;

...

поле записи *n*: тип элемента поля записи *n*.

3) Константы типа: отсутствуют.

4) Ограничения: определяются типами элементов поля записи, границами изменения типа.

5) Операции: операция селекции (`.`) по полю записи; операция конкатенации селекторов (или последовательного добавления селекторов друг к другу).

6) Функции от аргумента типа: отсутствуют.

ПРИМЕР

Опишем простые (неречисляемые) данные вида:

Мес = (январь, февраль, март, апрель, май, июнь, июль).

Год = (1995, 1996, 1997, 1998, 1999, 2000, 2001).

Ден = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28).

ПРИМЕР

На их основе можно определить записи:

Дата = Запись

ДенДат: Ден;
МесДат: Мес;
ГодДат: Год;

Юноша = Запись

Фамилия: лит;
Имя: лит;
ДенРожд: Дата;
Статус = (студент,
школьник);

Можно переменной *Z* типа «Юноша» присвоить значения таким образом:

Z = Юноша("ИВАНОВ", "АЛЕКСЕЙ", Дат(04.04.1995), студент);

Селекция переменной *Z* по селекторам

«Имя», «ДенРожд.ГодДат»

будет иметь вид:

Z.Имя = "АЛЕКСЕЙ",
Z.ДенРожд.ГодДат = 1995.

Для ссылки на запись и селекции ее по полю записи нужно указать имя поля и записи.

Структура данных определяет алгоритм решения задачи.

ПРИМЕР

Задана некоторая фирма, у которой *n* сотрудников. Составим алгоритм определения оклада заданного сотрудника (по Ф.И.О.). Выберем следующую структуру данных: Ф.И.О.; отдел фирмы; телефон отдела и оклад сотрудника — и примем следующие типы и форматы этих данных (это называется спецификацией структуры): Ф.И.О. — строка, содержащая не более 30 символов, например Николаев Петр Иванович; отдел — целое двухразрядное число, например отдел 06; телефон — литерная строка, содержащая не более 10 символов, например 8-88-90; оклад (в р.) — целое, не более чем 5-разрядное число, например 500 р. Выберем в качестве базовой используемой структуры данных вектор и решим задачу с его помощью. Статический алгоритм:

алг Оклад(арг цел *n*, лит *f*)

дано | описанная выше структура данных из *n* сотрудников

надо | определить оклад заданного сотрудника (по Ф.И.О.)

нач | начало тела алгоритма и объявление типов объектов

лит *x*, | переменная «Ф.И.О.»

f, | задаваемый сотрудник (его Ф.И.О.)

цел *i*, | счетчик цикла поиска

<u>сим</u> <u>мас</u> Сотрудник [1:n]		Сотрудник — массив Ф.И.О.
<u>цел</u> <u>мас</u> Оклад [1:n]		Оклад — массив окладов
$i := 1$ начальное значение номера записи		
<u>нц</u> <u>пока</u> ($i \leq n$) <u>и</u> (Сотрудник [i] $\neq f$)		условие цикла
$i = i + 1$ если сотрудник не найден — к следующему		
<u>кц</u>		конец цикла — переход к новому сотруднику
<u>если</u> $i < n$		если, не дойдя до конца, мы вышли из цикла
<u>то вывод</u> ("Оклад"; f ; "равен: "; Оклад [i])		то мы нашли
<u>иначе вывод</u> ("такого сотрудника нет")		иначе не нашли
<u>все</u>		конец проверки успешности поиска
<u>кон</u>		конец алгоритма.

Решим эту же задачу для записи Сотрудник с указанными выше полями. Алгоритм на учебном алгоритмическом языке:

алг Оклад(арг цел n , лит f)

дано | описанная выше структура данных из n сотрудников

надо | определить оклад заданного сотрудника (по его Ф.И.О.)

нач | начало тела алгоритма и объявление структур и типов

Запись Сотрудник | структура типа «запись» с полями

ФИО: лит; | поле 1

Отдел: цел; | поле 2

Телефон: лит; | поле 3

Оклад: цел; | поле 4

f | задаваемый сотрудник (его Ф.И.О.)

цел i | переменная цикла поиска

$i := 1$ | начальное значение номера записи

нц пока ($i \leq n$) и (Сотрудник [i]. Ф.И.О. $\neq f$) | условие цикла

$i := i + 1$ | если сотрудник не найден — к следующему

кц | переход к новой записи структуры (новому сотруднику)

если $i \leq n$ | если все записи не были просмотрены

то вывод ("Оклад"; f ; "равен: "; Сотрудник [i]. Оклад) | то нашли

иначе вывод ("такого сотрудника не нашли") | не нашли

все | конец проверки успешности поиска

кон | конец алгоритма.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое структура данных? Почему она так называется?
2. Какими основными элементами (атрибутами) определяются стандартные простые типы данных?
3. Чем отличается простой тип данных от структурированного?
4. Охарактеризуйте базовую структуру данных «массив». Каковы его атрибуты и операции над ним? Приведите примеры.
5. Охарактеризуйте базовую структуру данных «запись». Каковы ее атрибуты и операции над ней? Приведите примеры.

6. В чем состоит отличие объекта, переменной для обозначения этого объекта, типа переменной и описания типа переменной?
7. Как соотносятся алгоритм и структура для решения задачи?

ЗАДАЧИ

1. Опишите структуру данных «выпускаемая продукция», в которой отражаются данные: год выпуска, код продукции и др.
2. Опишите структуру данных «абитуриент с атрибутами»: факультет, группа, домашний адрес, телефон, год рождения, баллы — и укажите операции для этой структуры.
3. Задана группа из n студентов и их оценки. Составьте алгоритм нахождения всех студентов, имеющих оценки ниже средней по группе, и всех отличников по задаваемой дисциплине.
4. Задана группа студентов и их оценки по предметам. Составьте алгоритм выдачи студентов: а) занимающихся только на «5»; б) имеющих хотя бы одну оценку «2».
5. Заданы текстовые массивы A (эталон диктанта), B (диктант ученика) не более чем из n предложений, каждое из предложений не более чем из m знаков. Составьте алгоритм и структуру данных проверки диктанта для каждого из k учеников.
6. Задана группа учеников и их оценки по предметам. Составьте алгоритм выдачи учеников с успеваемостью выше средней по группе.
7. Составить алгоритм поиска вуза с нужной специальностью, определив адекватную структуру данных.

ТЕСТ

1. К стандартным типам данных относятся все перечисленные:
 - а) цел (integer), вещь (real), сим (char), лит (string), лог (boolean);
 - б) цел, вещ, сим, лит, лог, хэш, очередь;
 - в) цел, вещ, сим, лит, лог, запись, стек;
 - г) цел, вещ, сим, лит, лог, запись, очередь, стек, хэш.
2. Для структуры типа «запись» разрешена операция:
 - а) сложения селекторов;
 - б) умножения селекторов;
 - в) последовательной селекции;
 - г) перестановки полей записи.
3. Ошибки описания данных могут сказываться только на:
 - а) этапе проектирования структуры данных;
 - б) этапе программирования;
 - в) этапе физической реализации;
 - г) на всех перечисленных этапах.
4. В списке {«стол», «функция», «вектор», «матрица», 100, «функция $y = 2x + 3$ », «переменная», «множество N »} абстрактных типов данных всего:
 - а) 7; б) 6; в) 5; г) 4.

5. При решении задачи правильно:
 - а) данные и алгоритмы разрабатывать независимо друг от друга;
 - б) данные разрабатывать после разработки алгоритма;
 - в) алгоритм разрабатывать после разработки данных;
 - г) периодически менять структуры данных и алгоритма.
6. При решении задачи правильно упрощать:
 - а) алгоритм за счет усложнения данных;
 - б) данные за счет усложнения алгоритма;
 - в) и данные, и алгоритм за счет памяти;
 - г) алгоритм за счет более адекватной структуры данных.
7. Типы данных в развитых языках программирования:
 - а) можно описывать по желанию пользователя;
 - б) можно никогда не описывать;
 - в) можно всегда описывать;
 - г) можно описывать, если они сложные.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
а	в	г	б	в	г	в

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Данные и их типы. Простые типы данных.
2. Абстрактные структуры данных и операции с ними, их свойства.
3. Представление структур данных в структурах памяти (компьютера).
4. Структура данных «множество».
5. Структура данных «дерево».
6. Структура данных «куча».
7. Структура данных «стек».

12

ПРОЕКТИРОВАНИЕ, ТЕСТИРОВАНИЕ И ТРАССИРОВКА АЛГОРИТМОВ

Нисходящим проектированием алгоритмов, проектированием алгоритмов сверху вниз или методом последовательной (пошаговой) нисходящей разработки алгоритмов называется метод составления алгоритмов, который исходную задачу (алгоритм) разбивает на ряд вспомогательных подзадач (подалгоритмов), формулируемых и решаемых в терминах более простых и элементарных операций (процедур). Они вновь разбиваются на более простые и элементарные до тех пор, пока не дойдем до команд исполнителя. В этих командах можно представить и выполнить полученные на последнем шаге разбиений подалгоритмы (команды системы команд исполнителя).

Восходящий метод, наоборот, опираясь на заранее определяемый корректный набор подалгоритмов, строит завершенные подзадачи более общего назначения, от них к еще более общим до тех пор, пока не дойдем до уровня, на котором можно записать решение поставленной задачи. Этот метод известен как проектирование снизу вверх.

Структурные принципы алгоритмизации (структурные методы алгоритмизации) — это принципы формирования алгоритмов из базовых структур (следование, ветвление, повторение), используя их соединение или вложение друг в друга с соблюдением правил, гарантирующих читабельность и исполняемость алгоритма сверху вниз.

ПРИМЕР

Спроектируем нисходящим методом алгоритм решения уравнения $a\sin^2x + b\sin x \cos x + c\cos^2x = d$ (считаем a, b, c, d отличными от нуля). Сделаем первый шаг — разобьем задачу на следующие подзадачи:

1. Замена $d = d\sin^2x + d\cos^2x$ и приведение подобных членов.
2. Решение однородного тригонометрического уравнения $(a - d)\sin^2x + b\sin x \cos x + (c - d)\cos^2x = 0$.

На следующем этапе задача 2 разбивается на новые подзадачи:

- 2.1. Доказательство того факта, что $\cos x \neq 0$ (не детализируем далее эту стандартную математическую задачу).
- 2.2. Деление на $\cos^2x$ обеих частей уравнения.
- 2.3. Решение уравнения $(a - d)\tg^2x + b\tg x + c - d = 0$.

На следующем этапе разработки алгоритма последняя задача может быть разбита на новые подзадачи:

- 2.3.1. Замена $y = \tg x$.

- 2.3.2. Решение уравнения $(a - d)y^2 + by + c - d = 0$.

На следующем этапе разбиваем последнюю задачу:

2.3.2.1. Вычисление дискриминанта $D = b^2 - 4(a-d)(c-d)$.

2.3.2.2. Определение знака D .

2.3.2.3. Нахождение корней y_1, y_2 при $D > 0$.

2.3.2.4. Решение уравнений $\operatorname{tg} x = y_1, \operatorname{tg} x = y_2$.

Таким образом, можно «собрать» алгоритм из подалгоритмов:

1) Замена $d = d \sin^2 x + d \cos^2 x$ и приведение подобных членов.

2) Деление на $\cos^2 x \neq 0$ обеих частей уравнения.

3) Замена $y = \operatorname{tg} x$.

4) Вычисление дискриминанта $D = b^2 - 4(a-d)(c-d)$.

5) Если $D < 0$, то записать ответ — решений нет; иначе найти:

$y_1 = \frac{-b + \sqrt{D}}{2(a-d)}, y_2 = \frac{-b - \sqrt{D}}{2(a-d)}, x_{1k} = \operatorname{arctg} y_1 + pk, x_{2k} = \operatorname{arctg} y_2 + pk,$
 $k \in Z$ и записать ответ — объединение множеств $\{x_{1k}\} \cup \{x_{2k}\}$.

ПРИМЕР

Составим восходящим методом алгоритм поиска $S = \sum_{i=1}^n \frac{\sin x}{x^i}$.

Построим подзадачи 1-го уровня:

1.1. Ввести значения n, x .

1.2. Определение начальных значений $i = 1, S = 0, P = 1$.

1.3. Вычисление степени умножением текущего P на x .

1.4. Вычисление $\sin x$.

Построим теперь подзадачи 2-го уровня:

2.1. Вычисление текущего значения дроби.

2.2. Увеличение текущего значения i на 1 ($i = i + 1$).

Построим, наконец, подзадачи 3-го уровня:

3.1. Добавление вычисленного значения дроби к текущему S .

3.2. Если $i \leq n$, то переход к выполнению подзадач 2-го уровня.

Объединив эти две подзадачи, получим тело искомого алгоритма.

Тестирование алгоритма — это проверка правильности или неправильности работы алгоритма на специально заданных тестах или тестовых примерах — задачах с известными входными данными и результатами (иногда достаточно их приближения).

Тестирование алгоритма не дает 100%-ной гарантии правильности алгоритма для всех входных данных и сложных алгоритмов.

Существуют сложные математические, формальные методы доказательства правильности (верификации) алгоритмов. Для простых алгоритмов грамотное тестирование может дать полную картину работоспособности алгоритма.

Трассировка — процедура визуализации текущего состояния алгоритма после выполнения каждой команды. Осуществляется с помощью так называемых трассировочных таблиц, в которых каждая строка соответствует определенному состоянию алгоритма (фрагмента алгоритма), а столбцы — состоянию параметров (входных, выходных и промежуточных).

Процесс поиска и исправления (явных или неявных) ошибок в алгоритме называется отладкой алгоритма.

Некоторые ошибки в сложных программных комплексах могут выявиться только в процессе их эксплуатации, поэтому есть еще последний этап (последняя возможность) поиска и исправления ошибок — этап сопровождения. На этом этапе, кроме поиска и исправления скрытых ошибок (не обнаруженных на предыдущих этапах), выполняются также необходимые работы по уточнению и улучшению документации, обучению персонала и др.

В каждой среде программирования имеются средства трассировки (пошагового выполнения программы). Процедуры тестирования и трассировки используются не только при проверке правильности алгоритма, но и при обосновании его неправильности.

ПРИМЕР

Определим функцию фрагмента алгоритма и значения элементов массива a при заданных начальных условиях: $n = 4$, $a[1] = 2$, $a[2] = -6.5$, $a[3] = 2.8$, $a[4] = 0$:

```

нц для  $i$  от 1 до  $n$ 
  если  $a[i] > 0$ 
    то  $a[i] := 1$ 
  иначе если  $(a[i] < 0)$ 
    то  $a[i] := -1$ 
  иначе  $a[i] := 0$ 
всё
кц.

```

Алгоритм заменяет каждое число ряда $a[1]$, $a[2]$, ..., $a[n]$ на его знак, т. е. выполняет те же функции, что и функция $\text{sign}(x)$ по отношению к $x = a[i]$, $i = 1, 2, \dots, n$. Выполнив алгоритм, получим: $a[1] = 1$, $a[2] = -1$, $a[3] = 1$, $a[4] = 0$.

ПРИМЕР

Выполним трассировку алгоритма поиска индекса наибольшего числа с нечетным индексом в целых числах $x_1, x_2, \dots, x_n$:

```

алг Поиск (арг цел  $n$ , цел таб  $x[1:n]$ , рез цел  $k$  лит  $s$ )
дано | последовательность целых чисел  $x[1], x[2], \dots, x[n]$ 
надо | найти индекс  $k$  максимального числа  $x[k]$ ,  $k = 1, 2, \dots, n$ 
нач | описание типов вспомогательных переменных
  цел  $i$ , | вспомогательная переменная (цикла)
  вещ  $y$  | вспомогательная переменная текущего максимума
   $k := 1$  | начальное состояние  $k$  (определено)
   $y := x[1]$  | начальное состояние  $y$  (определено)
   $i := 1$  | начальное состояние  $i$  (определено)
  нц пока  $(i \leq n)$  | заголовок цикла поиска  $k$ 
    если  $(y \leq x[i])$  | если  $x[i]$  превышает текущее  $y$ 
      то  $y := x[i]$  | то фиксируем большее из них в  $y$ 
       $k := i$  | и его индекс  $i$  в  $k$ 

```

все | конец условной структуры
 $i := i + 2$ | переходим к следующему элементу
 кц | конец цикла по i
 если ($k > 0$) | проверка положительности решения
 то $s :=$ "задача имеет решение"
 иначе $s :=$ "задача не имеет решения"
 все | конец условной структуры
 кон | конец алгоритма.

Произведем трассировку при входных данных $n = 5$, $x[1] = 7$,
 $x[2] = 2$, $x[3] = 1$, $x[4] = 10$, $x[5] = 18$ с помощью таблицы;
 $s =$ "нет" ("да") соответствует «задача решена (не решена)»:

i	s	k	y	$y \leq x[i]$	$i \leq n$
1	Нет	1	7	Да	Да
3	Нет	1	7	Нет	Да
5	Нет	5	18	Да	Да
7	Да	5	18	Да	Нет

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. В чем суть метода нисходящего проектирования алгоритмов? Укажите его достоинства и недостатки (если они есть).
2. В чем суть метода восходящего проектирования алгоритмов? Укажите его достоинства и недостатки (если они есть).
3. Объясните принципы структурного проектирования.
4. Что такое тестирование алгоритма, для чего оно применяется?
5. Что такое верификация алгоритма, для чего она применяется?
6. Что такое трассировка алгоритма, для чего она применяется?
7. Что такое отладка алгоритма (программы)?

ЗАДАЧИ

1. Составьте восходящим методом алгоритм нахождения наименьшего из сумм строк, столбцов, диагоналей матрицы $A(n \times n)$.
2. Составьте нисходящим методом алгоритм сортировки ряда.
3. Составьте восходящим методом алгоритм нахождения количества чисел ряда, меньших среднего арифметического ряда.
4. Приведите один из разработанных вами первых алгоритмов в соответствии с принципами структурного программирования.
5. Протестируйте алгоритм решения квадратного уравнения на минимально необходимом наборе тестов.
6. Составьте к задаче сортировки ряда по убыванию минимально необходимый набор тестов. Протестируйте алгоритм.
7. Определите функцию фрагмента, запишите его кратко, выполните его трассировку при указанных начальных данных.

$x := a[0]$

$y := b[n]$

нц для i от 1 до n

$x := x*s + a[i]$

$y := y*s + b[n-i]$

кц

Начальные данные

$n = 4, a[i] = i, b[i] = 1 - i, s = 2, i = 0-4.$

ТЕСТ

- Ошибки в программе устраняются в процессе:
а) отладки; в) тестирования;
б) сопровождения; г) всего перечисленного в а, б, в.
- Метод разбиения задачи на более мелкие, логически связанные подзадачи называется:
а) нисходящим; в) разбивающим;
б) восходящим; г) измельчающим.
- Метод «сбора» задачи из более мелких, связываемых логически подзадач называется:
а) нисходящим; в) собирающим;
б) восходящим; г) связывающим.
- Метод конструирования алгоритма из базовых структур — это:
а) последовательный; в) конструктивный;
б) композиционный; г) структурный.
- Проверка правильности работы алгоритма на специально заданных примерах называется:
а) трассировкой; в) верификацией;
б) тестированием; г) сопровождением.
- Формальное доказательство правильности алгоритма — это:
а) трассировка; в) верификация;
б) тестирование; г) сопровождение.
- Исправление скрытых ошибок в программе в процессе ее эксплуатации происходит на этапе:
а) трассировки; в) проектирования;
б) тестирования; г) сопровождения.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
г	а	б	г	б	в	г

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

- Метод нисходящего проектирования алгоритмов (программ).
- Метод восходящего проектирования алгоритмов (программ).
- Метод структурного проектирования алгоритмов (программ).
- Тестирование алгоритмов (программ).
- Верификация алгоритмов.
- Трассировка и пошаговое выполнение.
- Отладка программы.

И 13

ИСПОЛНИТЕЛИ АЛГОРИТМОВ

Исполнитель иногда понимается как «автомат, способный выполнять автоматически некоторые программные действия, в частности ЭВМ». Такая трактовка понятия «исполнитель» может привести как к непониманию роли и места основных исполнителей — человека и автомата в процессе целенаправленной обработки информации (в лучшем случае), так и к различным технократическим утопиям (в худшем случае).

Исполнитель — это некоторая абстрактная, биологическая, техническая, организационная или смешанного типа система, способная преобразовывать информацию или сообщения из некоторого фиксированного множества, называемого **операционным пространством**. Работа исполнителя осуществляется в командном или программном режиме. Любой исполнитель состоит из устройства управления, «рабочего инструмента» исполнителя, имеет класс исполнимых алгоритмов.

Человек — специфический исполнитель. Сообщения, которые он обрабатывает, хранятся и передаются, как правило, в языковой форме (разговор, письмо, жесты, мимика и др.). В качестве устройства управления исполнителя — человека или высшего животного выступает мозг и нервная система, передающая и воспринимающая сигналы от окружающего мира с помощью органов чувств.

ПРИМЕР

Передающий орган человека (эффектор) — голосовой аппарат (орган речи) передает сообщения с помощью физических носителей — звуковых волн (15—20 000 Гц). Воспринимающий орган (рецептор) — слуховые мембраны воспринимают эти сигналы. Способ восприятия — слуховой. Время реакции на сообщение составляет порядка 200 мс до ответа или реакции, так как раздражение от сигналов проводится по нервным путям к мозгу, где они обрабатываются, и затем ответ проводится к эффектору. Обработка сенсорных сигналов и выработка ответа на них происходит на четырех различных уровнях нервной системы: на уровне коры головного мозга, на уровне таламуса, на уровне продолговатого мозга, на уровне спинного мозга. Структура организации и функции головного мозга человека приведены на рисунке 10. Примечания: 1 — кора головного мозга (управление высшей нервной деятельностью); 2 — борозды коры головного мозга (преобразование входной информации в импульсы, передаваемые двигательной системе); 3 — таламус (управление вводом-выводом данных в кору головного мозга); 4 — средний мозг (ввод-вывод данных от ор-

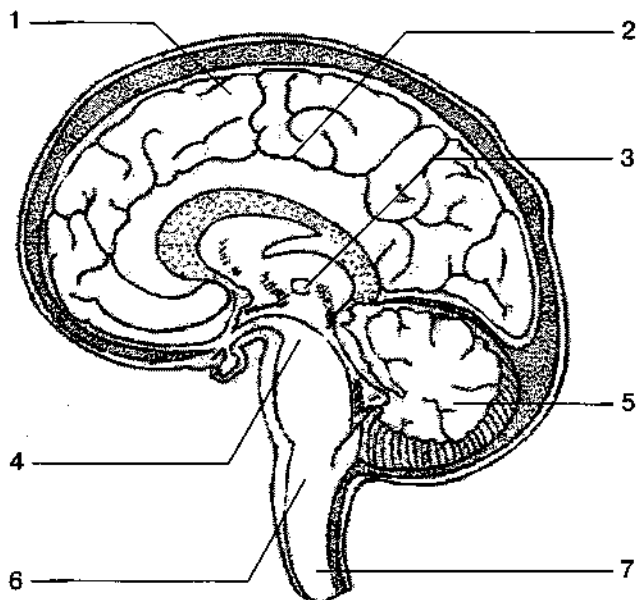


Рис. 10. Структура мозга человека

ганов чувств и импульсы управления мышцами); 5 — мозжечок (центр выработки выходных импульсов для двигательной системы); 6 — продолговатый мозг и мозговой мост (центр управления вводом-выводом данных в каналы связи спинного ствола); 7 — спинной ствол (основной канал связи для передачи данных, поступивших от органов чувств и выходных импульсов управления мышцами).

Центром деятельности организма является **нервная система**, выполняющая две основные функции:

- **коммуникативную** — передачу информации, получаемой от рецепторов кожи, глаз, ушей и других органов, к нервным центрам и, наоборот, от нервных центров к эффекторам, например мышцам;
- **интеграцию и переработку** получаемой информации, программирование наиболее адекватной реакции — от простых рефлексов до сложных мыслительных процессов.

Первая функция обеспечивается периферической нервной системой, состоящей из соматической системы (обеспечивающей связь с внешним миром) и вегетативной системы (обеспечивающей связь с внутренними органами). Вторая функция обеспечивается центральной нервной системой. Рассмотрим ее структуру (рис. 11).

Структурная единица нервной системы — **нейрон**, нервная клетка. Кора переднего мозга содержит около 10^3 – 10^4 нейронов, каждый из которых соединяется с другими нейронами синапсами,

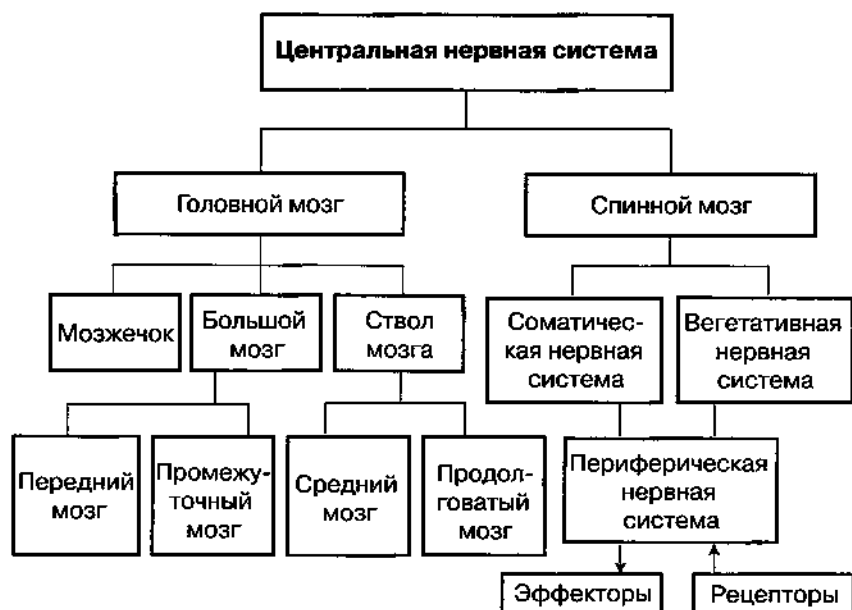


Рис. 11. Структура центральной нервной системы

образуя кластеры нейронов, обменивающихся сообщениями типа «да» — «нет», «возбужден» — «не возбужден».

Нейроны — микроскопические массы протоплазмы, заключенные в мембраны, выполняющие функции проводников сигнала от мозга и к мозгу. На конце каждого пучка нервных волокон есть рецепторы (клетки ввода) и эффекторы (клетки вывода). Рецепторы посылают сигналы по нейронным цепям в головной мозг, где с помощью нервной системы ведется поиск релевантной ситуации информации, ее распознавание и обработка.

ПРИМЕР

При прикосновении, например, рукой к горячему рецепторы пальцев передают возбуждение через нервные окончания и цепочки нейронов в мозг, где определяется релевантное в ситуации действие — сигнал мышечной системе рук о необходимости сделать судорожное движение — отдернуть руку.

ПРИМЕР

При произнесении слова, как показывают опыты, необходимо управление примерно 50 мышцами (лица, гортани), участвующими в процессе произношения слова, и для этого требуется около 250 обращений к образу этого слова в памяти человека.

Особенностью мозга является высокое качество и скорость обработки информации. Нейроны выполняют обработку со скоростью

всего около 100 инструкций в секунду (сравнить с ЭВМ, выполняющей миллионы инструкций в секунду), но они быстрее и эффективнее решают наиболее сложные (в частности, для ЭВМ) задачи распознавания и классификации, принятия решений и другие плохо формализуемые и структурируемые проблемы.

Человеческий мозг — система параллельно работающих структур для выработки, принятия логических (алгоритмических, рациональных) решений. Там, где невозможно принять такое решение (не удастся найти связи структур), принимается эвристическое решение. Каждая проблема имеет свой порог формализуемости.

ПРИМЕР

Такая «простая» для человека задача, как нахождение решения квадратного уравнения, имеет свой порог формализации. Если ищем вещественные корни, то порог определяется известной формулой для корней. За порогом этой формализации, если необходимо находить «очень точные» рациональные корни, эта задача неразрешима для человеческого мозга, как и бесконечен итерационный процесс вычисления корня.

В каждом нейроне коры головного мозга одновременно обрабатываются возбуждения — мотивации, целеполагания, внешние возбуждения — отражения текущего состояния управляемого объекта, возбуждения памяти (опыта). Их согласованная обработка дает картину объекта и позволяет принимать решения. Так, мозг, непрерывно перебирая результаты всех прошлых действий в аналогичных ситуациях и сравнивая их с текущей ситуацией, выбирает вариант, наиболее подходящий, целесообразный и эффективный в данной конкретной ситуации. Если не найдется такая ситуация, то выбирается такое состояние, результат которого будет наиболее адекватен; этот результат и запоминается далее.

Отметим, что процесс обработки информации в мозге до сих пор недостаточно раскрыт и изучен и представляет собой одну из сложнейших и интереснейших проблем (не только нейрофизиологии, нейропсихологии, но и нейроматематики и нейроинформатики в связи с актуальностью разработки нейрокомпьютеров и нейроподобных вычислительных сред и систем).

Нейроны бывают различного типа: сенсорные (от рецептора кожи к спинному мозгу); нейроны сетчатки (от рецепторов сетчатки глаза к зрительному нерву); двигательные (от рецепторов мышц к двигательной коре). Они образуют своеобразные регистры (зрительные, слуховые, тактильные и др.). Нейрон служит для передачи информации за счет нервных импульсов. Расшифровка нервных импульсов (информации) происходит в соответствующих областях коры головного мозга. Нейроны коры мозга функционируют параллельно. Это их важное свойство и достоинство.

ПРИМЕР

В коре переднего мозга около 50 млрд нейронов. Они организованы примерно в 600 млн функционирующих параллельно систем. Производительность такого типа «процессора» (нейроподобной системы) очень впечатляет (оцените!).

Память человека бывает: **сенсорная** — на уровне рецепторов; **кратковременная** — с момента передачи сигналов от рецепторов до момента их интерпретации; **долговременная** — от кратковременной задержки информации в памяти до ее сжатия, передачи на долговременное хранение, т. е. ее структурирования и архивации.

В непосредственную память (сенсорную память) человека поступает информация от органов чувств, далее она переводится в оперативную память (память сознания), а затем она пересылается в долговременную память с помощью подсознания. В долговременной памяти, в частности, хранятся: формы поведения (генетически заложенные или приобретенные); объекты и образы, приобретенные в процессе обучения правилам и процедуры обнаружения и идентификации объектов; правила выборки и организации информации; сведения (сообщения) из жизненного опыта; бытовые и профессиональные навыки и умения.

Человеческая память обладает способностью хранить, запоминать, актуализировать информацию, приобретенную ранее мозгом и управляющую поведением человека (рис. 12).

Стимулы сенсорных регистров, например рецепторов, передают поступающую извне информацию в кратковременную память, из нее — в долговременную, а из нее информация может пересылаться в кратковременную память.

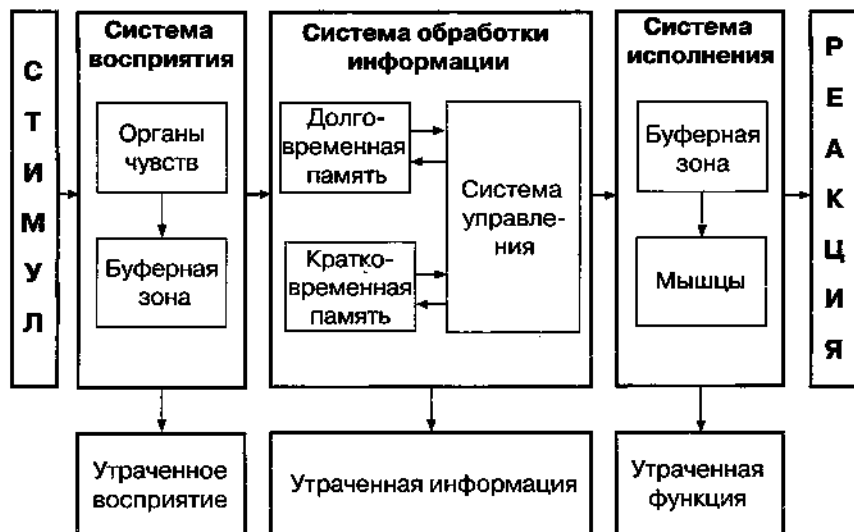


Рис. 12. Структура обработки информации человеком

ПРИМЕР

При взгляде на (конкретный) стол у человека может из долговременной памяти ассоциироваться понятие «стол» (абстрактное), а также информация, связанная (ассоциативно-интуитивными связями) с этим понятием, например сведения о потребительских качествах стола.

Информация представляется в виде информационных кодов — упорядоченных признаков, которые определяют объект (связь, понятие). Коды связываются (ассоциируются) между собой, образуя структуры памяти. Новая информация удерживается в памяти образованием связей между копиями кодов объектов (событий), т. е. образованием структур памяти вначале в кратковременной, а затем и в долговременной памяти. Так образуются хранилища знаний. Поиск нужного знания (хранилища, где оно находится) осуществляется направленно. Для нахождения соответствующей структуры памяти используются признаки кодов, определяющих нужную структуру. Каждая структура имеет свой адрес, по которому она извлекается из памяти в нужной ситуации.

ПРИМЕР

На таком принципе основывается задание номеров телефонов некоторых служб: 03 — «третья по важности служба», 911 — «номер, о котором часто рассказывали родители на случай чрезвычайной ситуации».

В живом организме в отличие от автомата, машины передача, хранение или обработка информации в конечном счете происходит с помощью биохимических реакций. Сообщениями, передающими информацию, служат при этом молекулярные системы.

ПРИМЕР

Белок — фермент, являющийся катализатором определенной биохимической реакции, есть преобразователь сигнала, превращающий одни сигнальные молекулы в другие. Любой организм живет за счет химических балансовых реакций, разностей концентрации химических веществ. Основа живого — белок, он синтезируется из аминокислот, молекул определенного вида (рис. 13). Здесь R — различные радикалы. Белковая цепь — некоторое слово (состояние) в алфавите, состоящем из 20 аминокислот (букв). Память или мозг человека может запоминать за счет обмена веществ, метаболизма (анаболизма).

У живых организмов потенциалы действий (электрические сигналы) проводят нервные волокна, а у растений роль нервных волокон выполняют жилки — проводящие пучки, пронизывающие все ткани и органы. По этим жилкам проходит вода, минеральное питание и электрические сигналы. В живых организмах (в отличие от растений) имеется «диспетчерская», откуда управляющие сигналы от внешних раздражителей поступают к различ-

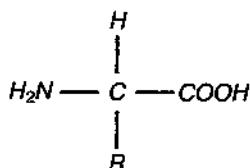


Рис. 13. Молекула белка

ным органам. «Диспетчерская» — это центральная нервная система, связанная с помощью аксонов с нейронами (кластерами, группами нейронов) различного типа: зрительными, обонятельными, слуховыми и др.

Физико-химические реакции (клеточного метаболизма) осуществляются под управлением ферментов, структура которых записана в ДНК гена. Генетическая запись (информация) передается от ДНК к РНК, от РНК к белку, регулируя, помогая самоорганизовываться белковой структуре. Информация, записанная в виде последовательности нуклеотидов в ДНК гена, перезаписывается на молекулу РНК, а затем генерируется последовательность аминокислотных остатков, осуществляется синтез белка, создается первичная структура будущей молекулы белка и, наконец, определяется белковая структура, ее функции. Синтез белка, клетки, клеточных структур — центральное событие в жизни организма. Считывание сообщений информационной РНК и их трансляция приводят к появлению в клетке молекул белка, информационные РНК способны кодировать белки.

Перейдем ко второму классу исполнителей — автоматам.

Автомат — устройство, которое в автоматическом режиме (без участия человека) выполняет прием, передачу, хранение и преобразование ресурсов (вещества, энергии, информации). Это происходит в соответствии с заложенным в него алгоритмом, представленным на языке автомата.

Автомат — конечный, если его функционирование (вход, выход, состояния) можно описать конечными последовательностями объектов, символов, алфавитов.

Алгоритм представляет собой единство информационной и управляющей частей алгоритма, а любой конечный автомат реализует алгоритм (алгоритмы), конечный автомат можно представить как совокупность автоматов: **управляющего автомата**, который определяет порядок выполнения шагов комплекса, и **операционного автомата**, реализующего шаги, выполняемые комплексом. Их взаимодействие можно изобразить схематично (рис. 14). Взаимодействие осуществляется следующим образом: на вход операционного автомата по каналу 1 поступает преобразуемая информация; с выхода операционного автомата по каналу 2 снимаются результаты преоб-

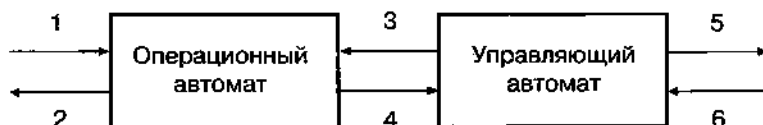


Рис. 14. Схема взаимодействия управляющего и операционного автоматов

разований, получаемые по алгоритму, выполняемому автоматом; по каналу 3 поступают некоторые управляющие воздействия, которые могут изменять в соответствии с внутренней логикой, структурой алгоритма последовательность выполнения команд алгоритма; по каналу 4 на вход этого управляющего автомата поступают признаки, характеризующие преобразуемую алгоритмом информацию; по каналу 5 поступает от управляющего автомата сигнал окончания операции; по каналу 6 — сигналы, определяющие выполняемую команду и ее начало.

ПРИМЕР

Примерами исполнителей-автоматов могут быть следующие устройства: справочный автомат на вокзале, светофор на перекрестке, устройство управления станком, микрокалькулятор, автомат для продажи газированной воды, телефон-автомат, электрорубильник, ракета.

ПРИМЕР

Исполнитель «Чертежник» может выполнять простые команды: налево(a) — поворот налево на угол в a радиан; направо(a) — направо на a радиан; вперед(a) — движение вперед на a см; назад(a) — назад на a см; поднять — поднять (от бумаги) карандаш; опустить — опустить конец карандаша для рисования. Последовательность команд «Чертежника», которая позволяет сделать рисунок (эскиз) квадрата со стороной 5 см на плоскости (начальное положение исполнителя — начало координат, а его начальный «взгляд» — по оси x , $x > 0$), будет иметь вид: опустить; вперед(5); направо($\pi/2$); вперед(5); направо($\pi/2$); вперед(5); направо($\pi/2$); вперед(5); поднять.

Эту программу для исполнителя можно записать компактнее:

```
i := 1
опустить
нц пока (i < 5)
    вперед(5)
    направо( $\pi/2$ )
    i := i + 1
кц
поднять.
```

Машина Тьюринга — это конечный автомат, предложенный английским ученым А. М. Тьюрингом в 1936 г. Она состоит из бесконечной, или полубесконечной, или конечной ленты, разбитой на ячейки, головки машины (ГМ), которая в любой дискретный момент времени $t = 0, 1, 2, \dots, T$ может обозреть только одну ячейку управляющего (головкой) устройства (УУ). В каждую ячейку можно записать только один символ, а все различные символы образуют конечный алфавит машины $X = \{x_0, x_1, \dots, x_n\}$. Символ x_0 считается пустым символом, т. е. он в ячейке никак не изображается или изображается пробелом. Головка машины всегда располо-

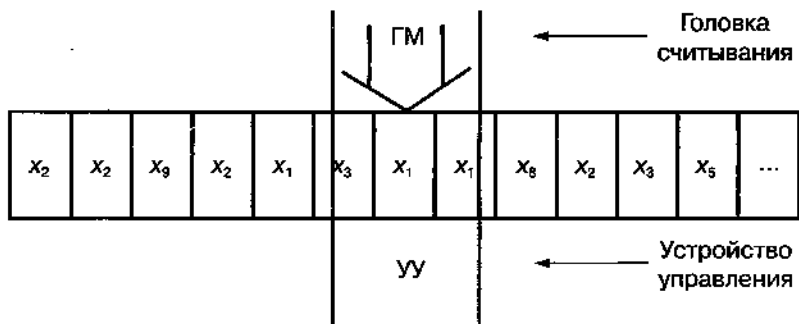


Рис. 15. Схема машины Тьюринга

жена над текущей ячейкой ленты и может выполнять команды типа: «Читать символ в текущей обозреваемой ячейке», «Писать символ в текущую обозреваемую ячейку», причем при записи пустого символа x_0 в ячейку происходит стирание символа, ранее находившегося в ячейке. Управляющее устройство машины, перемещая головку на позицию влево или вправо, может выдавать команды головке машины. Все различные состояния машины (или состояния головки машины в каждый момент времени работы автомата, за такт ее работы) образуют некоторое конечное множество внутренних состояний автомата $S = \{s_0, s_2, \dots, s_n\}$, среди которых есть начальное состояние s_0 , а также конечное состояние s_n .

На рисунке 15 приведена схема машины Тьюринга.

В каждый такт работы машины Тьюринга ГМ считывает символ из обозреваемой ячейки, затем изменяет свое текущее состояние на новое состояние (определяемое прочтенным символом и текущим состоянием) и реализует очередную команду УУ, в которой указано, какой нужно символ записать и в какую ячейку ГМ нужно переместиться (или же остаться на месте). Наборы этих правил, команд образуют алгоритм работы машины Тьюринга.

До начала работы машины некоторые ячейки ленты нужно заполнить символами из X (кроме x_0), и машина выводится из состояния s_0 . С помощью даже такой примитивной (по числу и типу выполняемых операций) машины Тьюринга можно вычислять различные функции — как числовые, так и логические, символьные (в зависимости от X). При этом машина Тьюринга допускает недетерминированность алгоритмического процесса, т. е. его незавершенность за конечное число шагов.

Справедлив следующий важный в информатике тезис.

Тезис Тьюринга — Черча. Для любой вычислимой функции можно построить машину Тьюринга, которая вычисляет эту функцию (для любого алгоритма конструируема машина Тьюринга, реализующая этот алгоритм или эквивалентная этому алгоритму по результатам выполнения).

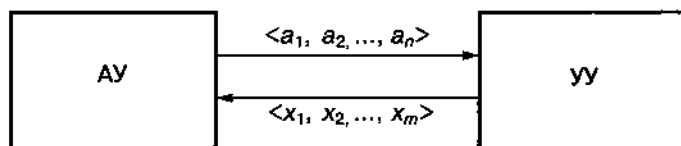


Рис. 16. Схема взаимодействия АУ и УУ

Разновидность исполнителя автомата — процессор ЭВМ. Совокупность АУ и УУ, объединенных информационными и управляющими шинами, называется **процессором ЭВМ**. Процессор выполняет функции, аналогичные мозгу человека. Взаимодействие АУ и УУ определяется составом и структурой команд программы ЭВМ. Команды программы выполняются в соответствии с линейным порядком их перечисления в программе до тех пор, пока не встретится команда ветвления, передачи управления в зависимости от каких-то предикатов или безусловно.

Взаимодействие операционного (АУ) и управляющего (УУ) автоматов происходит, как показано на рисунке 16.

Автомат АУ формирует и подает на вход автомата УУ некий набор предикатов вида $\langle a_1, a_2, \dots, a_n \rangle$, а получает набор $\langle x_1, x_2, \dots, x_m \rangle$ кодов микроопераций — элементарных операций, выполняемых за один такт работы ЭВМ и преобразующих текущее состояние процессора.

Проверяя на каждом шаге (такте) работы процессора требуемые логические условия, автомат АУ выдает соответствующий выходной сигнал, зависящий только от состояния автомата АУ, но не от сигнала на его входе.

Автомат УУ по сигналу кода операции очередной выбранной из ОЗУ и расшифрованной команды устанавливается в начальное состояние и затем начинает выполнение некоторой микропрограммы, или последовательности микроопераций, реализующих некоторую процедуру. Микропрограмма $\langle x_1, x_2, \dots, x_m \rangle$ вызывает последовательное изменение состояния автомата АУ. Выполнение микропрограммы заканчивается, если УУ переходит в заключительное состояние. В заключительном состоянии УУ находит и расшифровывает очередную команду, затем УУ устанавливается в новое начальное состояние и выполняет новую микропрограмму.

Аппаратно реализуются в ЭВМ элементарные базовые операции и базовые микрооперации.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое исполнитель алгоритмов?
2. Укажите основные классы исполнителей алгоритмов.
3. Каковы атрибуты исполнителя алгоритмов?
4. Что такое операционная среда исполнителя?
5. Что такое конечный автомат? Укажите его основные элементы.

6. Что такое машина Тьюринга? Укажите ее основные элементы.
7. Каковы основные команды машины Тьюринга?

ЗАДАЧИ

1. Ресурсы человеческого мозга рассчитаны на переработку информации в 16 бит/с. Какое количество информации перерабатывает человек за свою жизнь (средняя продолжительность жизни 70 лет)?
2. Для выживания человека необходимо условно 10^{30} — 10^{40} бит информации. Рост наследственной информации условно оценим в 10^{10} — 10^{20} бит. Как и откуда может человек черпать недостающую информацию?
3. Мозг человека содержит около 10^{30} синапсов. Чему примерно равно количество различных состояний мозга? Сравните с числом всех атомов Вселенной, оцениваемым как 10^{80} .
4. Исполнитель «Робот» может выполнять команды: вперед(a) — вперед на a см; назад(a) — назад на a см (a — вещественное); поворот(a, b) — поворот на угол в a радиан направо ($b = 1$) и налево ($b = 0$); поднять — поднять груз; опустить — опустить груз. Запишите наименьшую последовательность команд «Робота» для того, чтобы переместить грузы, расположенные в вершинах равностороннего треугольника со стороной x см, в центр треугольника. Каждый груз нужно поднимать за один раз. Модифицируйте команды «Робота». Решите новую задачу.
5. Исполнитель «Вычислитель» может выполнять следующие четыре команды: сложить(a, b, c) — найти сумму $a + b$ и результат присвоить переменной c ; вычесть(a, b, c) — выполнить $c := (a - b)$; умножить(a, b, c) — выполнить $c := a * b$; разделить(a, b, c) — выполнить $c := a / b$. Запишите наименьшую последовательность команд «Вычислителя» для того, чтобы вычислить значение функции $y = [(x + 2)^2(x - 2) - (x^3 - 6x^2 + 12x - 8)] \times \times (x^2 + 5x + 6) / (x - 2) + 5 / (x - 4)$ в точках $x = 1$ и $x = 5$. Составьте другую задачу для этого исполнителя. Модифицируйте операционную среду исполнителя.
6. Исполнитель «Железнодорожник» может выполнить команды: расцепить(a, b, n, m, k) — расцепить поезд номер n в поезд номер m из a вагонов и в поезд номер k из b вагонов; удалить(n) — удалить поезд номер n , причем это возможно только тогда, когда в нем всего один вагон; определить(n, a) — определить число a вагонов поезда номер n . Запишите минимум команд «Железнодорожника» для удаления со станции поезда номер 23 из 11 вагонов.
7. Придумайте и опишите операционную среду, систему команд одного исполнителя, составьте задачу для него и решите ее (постройте алгоритм решения задачи в операционной среде построенного исполнителя). Как модифицировать систему команд исполнителя для расширения его операционной среды?

ТЕСТ

1. Исполнитель алгоритмов не имеет систему:
а) допустимых команд; в) аналога устройства управления;
б) аварийной остановки; г) аналога устройства исполнения.
2. Исполнитель алгоритмов — это:
а) ЭВМ; в) автомат;
б) человек; г) все вместе: а, б, в.
3. Машина Тьюринга — это:
а) механическая машина; в) арифмометр;
б) конечный автомат; г) машина алгоритмизации.
4. В составе команд любой машины Тьюринга нет команды:
а) «читать символ из текущей ячейки»;
б) «писать символ в текущую ячейку»;
в) «переместить на одну ячейку»;
г) «если выполнено условие, то остановиться».
5. Универсальная машина Тьюринга (имитирует любую машину Тьюринга):
а) теоретически может быть построена;
б) теоретически не может быть построена;
в) может быть построена практически для некоторых случаев;
г) теоретически не определена.
6. Исполнителем алгоритмов не может быть:
а) человек; в) процессор ЭВМ;
б) конечный автомат; г) арифмометр.
7. Исполнитель алгоритма имеет в составе все перечисленное:
а) управляющий, исполнительный, аварийный автоматы;
б) управляющий, информационный, вычислительный автоматы;
в) управляющий и операционный автоматы, операционную среду;
г) управляющий, рабочий, запасной автоматы.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
б	г	б	г	а	г	в

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Человек как исполнитель алгоритмов.
2. Автомат как исполнитель алгоритмов.
3. Природа как исполнитель алгоритмов.
4. Машина Поста.
5. Машина Тьюринга.
6. Мозг человека как компьютерная система.
7. Нейросистемы и нейрокомпьютинг.

14

ОСНОВЫ КОМПЬЮТЕРА

Минимум знаний, необходимых человеку для решения профессиональных задач и приобщения к знаниям, накопленным с помощью компьютера, а также решения различных бытовых задач с помощью компьютера, называют **компьютерной грамотностью**.

- Основные составляющие понятия **компьютерная грамотность**:
- **мировоззренческая составляющая**: ознакомление с ролью и местом компьютера, информационных технологий в современном мире, с их ролью в развитии общества, познании природы;
 - **общеобразовательная составляющая**: ликвидация психологического барьера перед компьютером, знакомство с основными принципами устройства и работы компьютеров, их применением, выработка умения и навыков грамотного использования компьютера всегда, когда в этом возникает необходимость, приобщение к общеобразовательным знаниям, накопленным с помощью компьютера, и др.;
 - **профориентационная составляющая**: подготовка специалистов по компьютерным системам и сетям, их применению, например программистов, мастеров, операторов, администраторов, приобщение к профессиональным знаниям на компьютере и др.;
 - **воспитательная составляющая**: воспитание терпения, аккуратности, порядка в работе, предусмотрительности, умения организовывать и вести диалог с собеседником (компьютером или человеком), привитие навыков исследовательского, творческого подхода к делу, умения вникать в сущность явлений, особенно не лежащих на поверхности изучаемых проблем, а находящихся на стыке различных наук и областей;
 - **коммуникативная составляющая**: широкое внедрение и использование компьютерных систем связи, передачи, обработки информации, Интернет.

Классификацию компьютеров можно проводить по различным критериям: по объему памяти, быстродействию, качеству и объему программного обеспечения, технологии использования и др. Дадим классификацию в обобщенном виде на основе быстродействия, оперативной памяти и разрядности компьютера. Эти характеристики стабильные (хотя оперативную память можно несколько наращивать). Классы часто перекрываются.

1. **Суперкомпьютеры** (быстродействие — от нескольких тысяч до сотен тысяч MIPS или миллионов операций в секунду над числами с фиксированной запятой, оперативная память — от нескольких тысяч до сотен тысяч Мбайт, разрядность — 64—256).

2. **Компьютеры общего назначения** (быстродействие — от нескольких тысяч до сотен тысяч MIPS, оперативная память —

от нескольких сотен до десятков тысяч Мбайт, разрядность — 64—128); часто эти компьютеры называют универсальными, большими, но эти определения, на наш взгляд, менее удачны, так как большинство компьютеров по своему назначению и использованию универсальны.

3. **Мини-компьютеры** (быстродействие — от нескольких десятков до сотен MIPS, оперативная память — несколько десятков или сотен Мбайт, разрядность — 32—64); среди этих компьютеров иногда выделяют более мощный подкласс — супермини-компьютеры.

4. **Микрокомпьютеры**, в частности персональные компьютеры (быстродействие порядка до нескольких сотен MIPS, оперативная память — несколько десятков или сотен Мбайт, разрядность — 32—64); этот класс подробнее рассмотрим ниже.

Классификация основных форм использования компьютеров:

- **мощный супервычислительный центр** — вычислительный центр, предназначенный для решения очень больших и сложных задач или реализации централизованного хранилища информации в системах обработки информации большого объема, в базах данных;
- **проблемно-ориентированные комплексы, станции и серверы** для решения достаточно широких классов задач: АСУ (автоматизированные системы управления), ИПС (информационно-поисковые системы), СУБД (системы управления базами данных), АСНИ (автоматизированные системы научных исследований), САПР (системы автоматизации проектных работ), узловой компьютер сети и др.;
- **персональные вычислительные системы** предназначены для решения профессиональных задач и автоматизации профессиональных знаний и опыта в режиме персональных вычислений, т. е. в режиме, когда все ресурсы компьютера предоставлены одному (персонально) пользователю в каждый момент работы;
- **робототехнические и микропроцессорные системы** с программным управлением, используемые для автоматизации и контроля процессов, встраиваемые в те или иные приборы, производства и технологии.

ПРИМЕР

Супервычислительный центр (оснащенный одним или несколькими суперкомпьютерами) может быть предназначен для решения государственных проблем, например обороны, космоса, прогноза погоды, макроэкономического прогнозирования и др. Решение таких проблем часто находится за пределами возможностей современных вычислительных систем. В частности, для более точного (с точностью до одного градуса) прогноза погоды необходимо совместно решать многие сотни различных математических задач, уравнений.

Под архитектурой компьютера понимается не внешний вид компьютера, его габариты и расположение, а структура и реализа-

ция информационных и управляющих связей между основными узлами компьютера: устройствами ввода-вывода, процессором, памятью.

Память компьютера можно представить как последовательность ячеек памяти. Количество ячеек памяти определяет объем памяти компьютера. Ячейка памяти — это физическое устройство, в которое можно записать или с которого можно считать последовательность битов. Каждый бит числа в ячейке памяти хранится в нужном разряде.

ПРИМЕР

Запишем числа 13_{10} , 17_{10} , -31_{10} в формате целых чисел в 8-разрядную ячейку памяти, изобразив ее схематически. Старший разряд ячейки памяти (нулевой) будет содержать бит знака числа. Например, договоримся, что содержит 1 — если число отрицательное, и 0 — если число положительное. Учитывая, что $13_{10} = 1101_2$, $17_{10} = 10001_2$, $-13_{10} = -11111_2$, получаем следующие примеры записи чисел в 8-разрядную ячейку:

0	0	0	0	1	1	0	1	0	0	0	1	0	0	0	1
1	0	0	1	1	1	1	1								

ПРИМЕР

Запишем в формате с фиксированной (например, между шестым и седьмым разрядами) запятой числа -13.75_{10} и 16.34_{10} в 13-разрядную ячейку памяти и определим погрешность такого представления чисел. Записывая знак числа в старший нулевой разряд ячейки памяти для $-13.75_{10} = 1101.11_2$, $16.34_{10} = 10000.0101011100001..._2$ (перевод неточен!), получаем

	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	1	0	1	1	1	0	0	0	0	0
	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	0	0	0	1	0	1	0	1	1

Относительную погрешность такого представления (абсолютная погрешность зависит от величины операнда) можно оценить весом разряда, не вмещаемого в разрядную сетку, т. е. $2^{-7} \approx 0.0078125$.

ПРИМЕР

Запишем в формате с «плавающей» запятой числа 90_{10} , 6.25_{10} , -0.0023_{10} , если в 13-разрядной ячейке памяти знак числа и знак порядка записываются соответственно в нулевой и первый старшие разряды ячейки, мантисса записывается в разря-

ды 2—8, порядок — в разряды 9—12. Указанные числа в нормализованной форме с «плавающей» запятой имеют вид:

а) $90 = 0.9 \cdot 10^2$; б) $6.2_{10} = 0.625 \cdot 10^1$; в) $-0.0023 = -0.23 \cdot 10^{-2}$,

а мантиссы чисел в двоичной системе:

а) $0.9_{10} = 0.(11100)_2$; б) $0.625_{10} = 0.101_2$;

в) $0.23_{10} = 0.001110101110..._2$.

Итак, имеем следующие представления чисел по условию задачи:

а)

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	1	0	0	1	1	0	0	1	0

Знак Мантисса числа Порядок
числа и

б)

0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	1	0	0	0	0	0	0	0	1

в)

1	1	0	0	1	1	1	0	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---

Что касается размещения выполняемых команд в памяти компьютера, то они (в битовом изображении) размещаются в специальных устройствах — регистрах или в определенных местах (разрядах) ячеек памяти компьютера.

ПРИМЕР

Формат команды может быть таким:

0	1	2	...	m	m+1	m+2	...	n
Код выполняемой операции					Адреса операндов			

Каждой команде процессор ставит в соответствие операцию, код которой указан в кодовой части команды или на специальном регистре. Далее производится расшифровка этой операции, затем из адресной части команды извлекаются адреса ячеек памяти, которые содержат операнды — числа, с которыми необходимо выполнить операцию, а далее выполняется операция с этими числами, после чего результат помещается в ячейку памяти, адрес которой указан в адресной части команды. Этот принцип называется **адресным принципом** построения команд компьютера, а сам принцип хранения команд программы в памяти компьютера аналогично хранению данных, обрабатываемых программой, — **программным принципом** работы компьютера.

ПРИМЕР

Имеется трехадресная ЭВМ с кодом операции сложения, равным 02; рассмотреть выполнение команды 02 150 102 010 в ЭВМ, которая хранится в ячейке памяти номер 105 так, как это показано ниже:

Адрес	Операция	Операнды операции		
105	02	150	102	010
Поле адреса ячейки	Поле кодовой части	Поле первого адреса	Поле второго адреса	Поле третьего адреса

После считывания из 105-й ячейки памяти информации вида 02150102010 (представленной в двоичном виде, по полям!) процессор определяет, что нужно сложить (код операции — 02) число из 150-й ячейки памяти с содержимым 102-й ячейки и полученную сумму записать в ячейку с номером 010.

ПРИМЕР

Фрагмент программы в машинных кодах в трехадресном компьютере с операциями 01 (вычитание), 02 (сложение), 03 (умножение), 04 (деление) может иметь вид:

Адрес ячейки	Код операции	Адрес 1	Адрес 2	Адрес 3	Пояснения
100	01	0500	1201	1000	$A3 := A1 - A2$
101	02	1000	1510	1510	$A3 := A1 + A2$
102	04	0039	1000	0030	$A3 := A1 / A2$
103	03	0030	1510	1510	$A3 := A2 * A1$
105	03	1510	1510	1510	$A3 := A1 * A2$

В компьютере имеется также память на регистрах (сверхоперативная память, СОЗУ).

Регистр — электронное устройство, способное, как и ячейка памяти, запомнить и хранить (временно) последовательность битов определенной длины. Обычно в компьютере имеются регистры различного назначения: регистры адреса команды, команды, особой ситуации, первого аргумента, второго аргумента, суммы (результата), или сумматор, контроля узлов компьютера и др.

Эти устройства реализуются более дорогими и чувствительными физическими устройствами, и поэтому по сравнению с основной памятью (адресуемой памятью) регистровая память невелика.

ПРИМЕР

Регистр адреса команды $R1$ служит для хранения информации об адресе очередной выполняемой команды. Компьютер воспринимает содержимое $R1$ как адрес команды, которая должна быть выполнена в текущий момент. Регистр команды $R2$ служит для выбора из памяти очередной команды, ее расшифровки и выполнения. Адрес команды находится в регистре $R1$. Регистр особой ситуации $R3$ можно воспринимать как одноразрядный регистр, в который записывается: бит 1 — если при выполнении текущей очередной команды произошла особая или аварийная ситуация, например деление на 0; бит 0 — если команда не привела к аварийной ситуации. Регистры первого $R4$ и второго $R5$ операндов служат для приемов из памяти и хранения операндов двухместных операций на время выполнения операций. Регистр результата, сумматор $R6$, служит для выполнения операций находящейся в регистре команды, т. е. для накапливания результата. Пусть регистры $R4$ и $R5$ содержат числа $6_{10} = 110_2$, $2_{10} = 10_2$ соответственно, а регистры $R1$ и $R2$ содержат числа $10_{10} = 1010$, $2_{10} = 10_2$ соответственно. Это означает: из 10-й ячейки выбрать команду номер 2 (например, сложение), из $R4$ выбрать число 6 (первое слагаемое), из $R5$ выбрать число 2 (второе слагаемое) и сложить их на сумматоре $R6$. Если при выполнении операции возникает «переполнение» т. е. длина числа на сумматоре превосходит разрядность регистра $R5$, то в регистр $R6$ записываются правые (младшие) биты результата, а остальные будут потеряны. Тогда в регистр $R3$ помещается один бит — бит возникновения особой или аварийной ситуации, и программа может аварийно завершиться. Если бы в нашем примере разрядность $R6$ была равна 3, так и произошло бы.

Кроме оперативной памяти, компьютер имеет **внешнюю память (ВЗУ)** с большей емкостью, но с меньшим быстродействием.

Главное отличие внешней памяти от оперативной в том, что в компьютере имеется непосредственно или аппаратно реализованный доступ только к ячейкам оперативной памяти. Для использования информации, хранящейся во внешней памяти, нужно ее копию переслать в оперативную память.

Внешняя память реализуется с помощью **внешних носителей информации** — магнитных дисков (гибких, жестких).

Структура памяти компьютера приведена на рисунке 17, где стрелками с жирными концами указано направление передачи информации, остальными стрелками — направление передачи управляющих сигналов.

Степень интеграции элементов микропроцессора можно повышать до определенных пределов как уменьшением топологических размеров, так и увеличением кристаллов, их «слойности».

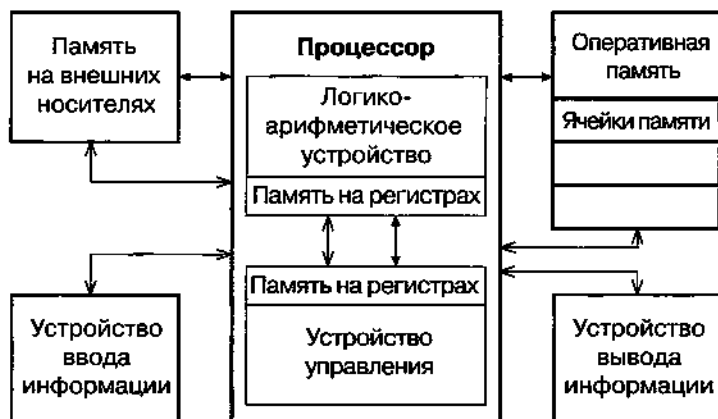


Рис. 17. Схема взаимодействия памяти и устройств ЭВМ

Не менее важной характеристикой кристаллов является время обращения к единице кода в запоминающем элементе — время выборки бита без изменения характеристик запоминания.

ПРИМЕР

Примеры достижений традиционных носителей, технологий — десятки мегабит (десятки миллионов бит) в кристалле, время выборки бита — единицы наносекунды. При этом минимальный топологический размер элементов не может быть ниже 1 мкм. Использование нетрадиционных носителей, технологий, новых принципов обработки информации позволяет существенно изменить эти параметры. В частности, использование сегнетоэлектрических пленок (пленок, структура которых позволяет при определенных условиях температуры и давления запоминать и измерять биты под действием внешнего электрического поля, за счет электрической поляризации) обеспечивает энергонезависимую память около 10 лет, емкость памяти с динамически произвольным выбором (DRAM) около 64 мегабит. Другое нетрадиционное направление исследований — биотехнология, использование биочипов, биосенсоров, устройств, у которых чувствительной структурой является биологический материал, слой: ферменты, ткани, бактерии, ДНК и т. д. Они непосредственно реагируют на приток (отток) компонента и могут быть в том или ином состоянии в зависимости от концентрации этого компонента, т. е. могут хранить (генерировать) сигнал. Они функционируют как биорецепторы живого организма, преобразуя в электрические, информационные и управляющие сигналы изменения в окружающей среде.

В 1946 г. Джон фон Нейман предложил ряд идей организации компьютера, которые легли в основу **фоннеймановской**, или **клас-**

сической, архитектуры компьютера:

- поле памяти состоит из однородных ячеек с адресами;
- программа (последовательность команд) упорядочена линейно;
- хранение программы и обрабатываемых данных одинаковое;
- команды выполняются в соответствии с их порядком;
- данные обрабатываются в соответствии с командами программы;
- процессор и доступ к памяти имеют централизованное управление.

Структура ЭВМ фоннеймановской архитектуры приведена на рисунке 18 (предметная область не входит).

Запоминающее устройство (ЗУ) — устройство запоминания и хранения перерабатываемой в процессе решения задачи или актуализируемой компьютером информации. ЗУ, элементы ЗУ реализуются технически с помощью устройств, которые обладают магнитными, электрическими, оптическими, биологическими свойствами и называются соответственно магнитными чипами, электрическими чипами, оптическими чипами, биологическими чипами или триггерами.

Арифметическое устройство (АУ) выполняет арифметические, логические операции. Команды арифметического устройства состоят из базовых арифметических и логических операций: «сравнить два числа», «переслать число», «взять дизъюнкцию» и др.

Устройство управления (УУ) организует работу ЭВМ и выполняет следующие функции:

- извлекает очередную команду (т. е. машинное слово) из ячейки (ячеек) ЗУ, расшифровывает команду и приводит в состояние готовности АУ;
- выбирает из ЗУ операнды к дешифрованной команде и передает их АУ для операций;

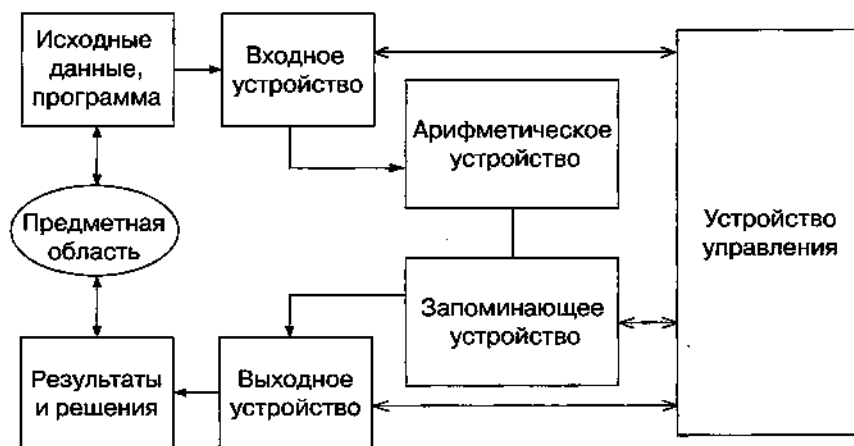


Рис. 18. Структура ЭВМ Джона фон Неймана

- пересылает результат выполнения операции из сумматора в ЗУ;
- реагирует на ход выполнения операции.

Совокупность АУ и УУ, объединенных информационными и управляющими шинами, называется **процессором компьютера**.

Взаимодействие АУ и УУ зависит от состава и структуры команд программы, которые выполняются в соответствии с порядком их перечисления в программе до тех пор, пока не встретится команда передачи управления, которая изменит этот порядок.

Обмен информацией пользователя с ЭВМ выполняется с помощью устройства ввода и устройства вывода после указания:

- типа устройства (диск, принтер, клавиатура и др.);
- атрибутов передаваемой информации (имя, размер, его местонахождение, размеры и тип файла с информацией и др.);
- режима работы устройства (режима обмена);
- программы обслуживания устройства ввода-вывода (драйверов).

В качестве примера рассмотрим **персональный компьютер**, или микрокомпьютер, работающий в режиме персональных вычислений, когда все ресурсы компьютера (адресуемая память, устройства ввода-вывода, процессорное время, программы и др.) в любой момент времени доступны только одному (персонально) пользователю компьютера. Микрокомпьютер — это компьютер, который состоит из блоков, реализованных на базе больших интегральных схем (БИС) или сверхбольших интегральных схем (СБИС), соединенных шинами данных, адресов, управления. Ядром таких компьютеров является микропроцессор, т. е. СБИС, реализующая основные функции процессора. Внешнее устройство подключается через порт — интерфейсную линию между микропроцессором и этим устройством, например порт ввода.

Персональный компьютер (ПК) удовлетворяет требованиям:

- малая стоимость, малые габариты, малое энергопотребление;
- надежность, высокий уровень интеграции компонентов, оперативность обслуживания, автономная эксплуатация без специальных условий;
- дружелюбность программного обеспечения пользователю;
- гибкость и открытость архитектуры, адаптируемость к различным применениям, совместимость многих ПК на аппаратном, программном, информационном, технологическом уровне, возможность коллективного использования периферийного оборудования.

Формы исполнения ПК: настольные; портативные, мобильные — ноутбук, записная книжка; рабочая станция, сервер.

Формы использования ПК: терминалы других компьютеров; локальные сети (общая шина данных от ЗУ одного компьютера к ЗУ другого); автономное; основное назначение ПК — стать конечной станцией доступа к глобальным сетям или станциям в сетях.

Надежность работы ПК: не менее сотен тысяч часов наработки на очередной отказ.

Память ПК: 256—1024 Мбайт ОЗУ и 40—500 Гбайт жесткого диска (HDD).

Частота тактового генератора (характеристика быстродействия): 1000—4000 МГц (1 МГц соответствует приблизительно 1 млн периодов в секунду). **Тактовый генератор** генерирует с некоторой частотой электрические импульсы, передачу битов.

На системной (материнской) плате размещаются:

- микропроцессор для управления работой всех остальных устройств компьютера и выполнения арифметических и логических операций над информацией, включающий УУ, АЛУ, микропроцессорную память;
- интерфейсная система микропроцессора для связи с другими устройствами;
- генератор тактовых импульсов;
- контроллеры устройств, интегрированных в материнскую плату;
- микросхемы ОЗУ и ПЗУ;
- интерфейс для связи всех устройств компьютера между собой;
- адаптеры клавиатуры, дисководов, портов ввода-вывода и т. д. для подключения, управления и взаимодействия внешних устройств компьютера.

Другими важными устройствами ПК являются:

- **дисковод гибких магнитных дисков** — устройство для записи данных на магнитные диски с целью долговременного хранения информации и переноса от компьютера к компьютеру (обычно формата 3,5" дюйма и емкостью 1,44 Мбайт);
- **дисковод жестких магнитных дисков, винчестер** (первые диски имели память по 20 Мбайт на каждую сторону поверхности диска и совпадали с калибром винтовки-винчестера) — устройство для долговременного хранения информации (от нескольких единиц до сотен Гбайт и высокой скоростью доступа);
- **CD-ROM (Compact Disk — Read Only Memory)** — устройство только для чтения компакт-дисков и **CD-RW** — устройство для чтения и записи (перезаписи);
- **монитор (дисплей)** — устройство отображения визуальной информации с помощью высвечивания (отклонением пучка электронной трубки) точек люминофорного покрытия трубки; такая точка называется пикселем (picture element);
- **видеокарта (видеоадаптер)** — устройство управления монитором и выводом информации на экран; располагается внутри системного блока в специальном разъеме (слоте расширения) и обеспечивает связь компьютера и монитора;
- **клавиатура** — устройство ввода информации; обычно клавиатура имеет около 100 клавиш;
- **принтер** — устройство для вывода информации на бумагу, пленку (жесткая копия документа); принтеры бывают матричные (изображение формируется ударами тончайших иглонок по красящей ленте), струйные (изображение формируется отклонением специально обработанной тончайшей струи чернил в элект-

ромагнитном поле), лазерные (изображение формируется лазерным лучом);

- **сканер** — устройство ввода в ПК графической и текстовой информации с готовых читабельных форм; сканеры бывают ручные, планшетные, рулонные;
- **плоттер (графопостроитель)** — устройство для получения на бумаге изображений большого формата;
- **дигитайзер** — кодирующий планшет для ввода графической (по координатам) информации; применяется в паре со специальным программным обеспечением и позволяет профессионально рисовать, чертить на компьютере;
- **цифровой аппарат и цифровая фотокамера** — устройства для получения, хранения и передачи в компьютер фотоизображения;
- **манипулятор-мышь** — устройство быстрого позиционирования курсора на экране и ввода данных и команд;
- **манипулятор-трекбол** — перевернутая вверх ногами мышь;
- **манипулятор-джойстик** — устройство для управления в компьютерных играх;
- **звуковая карта (адаптер)** — устройство для воспроизведения и записи с внешнего источника звуковой информации;
- **звуковые колонки** — для стереопрослушивания звуковой информации;
- **модем** — устройство для шифровки и дешифровки сигналов при передаче-приеме информации с помощью телефонной линии; позволяет согласовать электромагнитно-цифровой выход компьютера с электромагнитно-импульсным входом телефонной линии и передать ее с помощью телефонного канала на вход удаленного компьютера и наоборот.

Возможности архитектуры фон Неймана неограничены для наращивания мощности.

ПРИМЕР

Невозможна ЭВМ фон Неймана с быстродействием 10^{20} операций в секунду, разрабатываемая на принципе переноса электронов (на электронных принципах переноса информации). Действительно, самая большая скорость переноса электронов — скорость света в вакууме: $v = 3 \cdot 10^{10}$ см/с. При выполнении одной операции электрон должен пройти расстояние, не меньшее диаметра атома водорода: $s = 10^{-8}$ см. Время пути равно $t = s/v = 10^{-8}$ см $(3 \cdot 10^{10}$ см/с) $= 10^{-18}/3$ с. Итак, если удастся это технически реализовать, быстродействие компьютеров фон Неймана на принципах переноса электронов не может превышать $3 \cdot 10^{18}$ операций в секунду.

Быстродействие ЭВМ поэтому может быть увеличено на нетрадиционных технологиях передачи информации — молекулярных, нейроструктурных и др. Кроме этого, активно используются группы компьютеров и новые эффективные классы алгоритмов решения задач, повышающие производительность.

Одним из наиболее перспективных теоретических подходов увеличения производительности и мощности вычислительных систем является распараллеливание вычислительного процесса и алгоритмов решения задач. Современное состояние микропроцессорной техники позволяет соединять множества процессоров в крупные параллельные структуры, увеличивающие производительность компьютерных систем до некоторого предела пропорционально числу этих процессоров. В широком смысле параллельные вычислительные системы — вычислительные системы, состоящие из процессорных, компьютерных комплексов, которые позволяют за счет своей организации, структуры добиваться одновременного во времени (параллельного) выполнения обработки информации.

Основные виды параллельных компьютеров:

- векторно-конвейерные, параллельно выполняющие одну и ту же операцию с различными наборами, структурами данных;
- параллельно выполняющие различные операции над различными наборами, структурами данных.

При этом существен (для архитектуры параллельной системы) и факт использования общей памяти процессорами системы.

Однако отставание персональных компьютеров от суперкомпьютеров (по многим количественным и качественным архитектурным показателям) достаточно велико.

ПРИМЕР

Материнские платы, обеспечивающие одновременную работу с общей памятью двух — четырех процессоров Pentium, только недавно стали использоваться в персональных компьютерах, хотя для суперкомпьютеров архитектуры с общедоступной адресуемой памятью они уже признаны неперспективными.

Увеличение производительности в параллельных системах обеспечивается увеличением числа процессоров, но это приводит к ограничениям скорости доступа к памяти. На основе общедоступной разделяемой памяти не создают систем с большим числом процессоров. Этого недостатка лишены системы с распределенной памятью, в которых у каждого процессора, а часто и у кластера процессоров имеется своя собственная оперативная память, доступ к которой процессорами кластера осуществляется без конкуренции, хотя они и не могут воспользоваться памятью другого кластера. Число процессоров в системах с распределенной памятью теоретически не ограничено, и производительность таких систем растет линейно с увеличением числа процессоров.

ПРИМЕР

В массивно-параллельных системах (MPS) применяются обычно сотни, а иногда и тысячи процессоров (кластеров).

Для обеспечения полной эффективности вычислительного параллельного процесса необходимо загрузить вычислениями все про-

цессоры системы, т. е. необходимы распараллеливаемые алгоритмы и программы.

Вреден ли компьютер? По данным ряда исследований, лица, по роду своей работы имеющие постоянный контакт с компьютерной техникой, чаще других заболевают. Нельзя думать, что защитные экраны полностью защищают от излучения и мерцания. Факторы, оказывающие наиболее вредное влияние на здоровье, — это излучения (инфракрасное, рентгеновское, электромагнитное), вибрация и шум, электростатические поля, ультразвук строчной частоты монитора и др.

ПРИМЕР

Обследование профессиональных пользователей ЭВМ (6—8 ч в сутки), показало, что почти у всех нарушено функционирование селезенки, поджелудочной железы, они более подвержены стрессам и т. п.

Над решением проблемы безопасности при работе с компьютером сегодня трудятся ученые всего мира. Разрабатываются и различные нейтрализаторы излучения. До внедрения таких эффективных средств наилучшим средством защиты является соблюдение простых санитарно-гигиенических и эргономических правил работы с компьютером дома и в компьютерном классе для студентов и школьников. Приведем некоторые из них:

- работа с компьютером не более четырех часов подряд с 10—15 минутным перерывом после каждого часа интенсивной работы или после двух часов менее интенсивной работы;
- расстояние от глаз до поверхности экрана не менее 0,6—0,7 м;
- перемещаемость клавиатуры относительно экрана в пределах 0,5—1,0 м;
- преимущественно желтый, зеленый, серый, светло-голубой фон монитора;
- температура воздуха в помещении 15—25 °С;
- относительная влажность воздуха в помещении 45—75%;
- наличие свободной площади рабочего стола размером не менее 0,25 × 1,2 м.

Важным элементом компьютера является монитор. От плохого монитора могут появиться головная боль, ухудшение зрения, снижение производительности труда. Необходимые минимальные требования при длительной работе на ЭВМ:

- размер экрана по диагонали 17 дюймов (от него зависит рассеяние зрения);
- разрешение экрана 800 × 600 и выше (от него зависит напряжение зрения);
- частота обновления кадров 70—75 Гц (от нее зависит утомляемость зрения);
- размер зерна экрана (расстояние между точками на экране) 0,26—0,25 (от него зависит качество изображения, утомляемость зрения);

- частота кадров (мерцание экрана) 75—80 Гц (от нее зависит утомляемость зрения, появление головных болей);
- стандарт безопасности MPR-II, ISO-9241-3, Low Radiation (TCO-92), ГОСТ 27954-88 (стандартно обеспечивается изготовителями).

Эволюция технического обеспечения, которое включает в себя аппаратные средства, средства коммуникации, программное обеспечение, проходит неравномерно, скачкообразно.

ПРИМЕР

Компьютерная техника пока развивается в геометрической прогрессии. Каждые четыре года происходит удвоение производительности компьютеров. Фирмы Apple Computer, IBM, Netscape Communications, Oracle и Sun Microsystems представили эталонный перечень спецификаций и рекомендаций по устройству сетевого, независимого от используемой платформы компьютера NC (Network Computer) — компьютера, находящегося в постоянном соединении с сетью, что позволяет более гибко и полное использовать ресурсы сети. В NC будет встроено программное обеспечение просмотра www-страниц (браузеры), способное исполнять загружаемые по сети приложения, поэтому необходимость в жестком диске отпадет.

ПРИМЕР

Устройство класса Handheld PC («наручный компьютер», или «компьютер, с которым можно работать на ладони») функционирует под управлением специальной версии Windows CE, созданной только для широкого ряда устройств типа персональных цифровых помощников, коммуникационных устройств и портативных компьютеров. Операционная система, упрощенные версии Word, Excel, Internet Explorer и клиента электронной почты хранятся в ПЗУ. Предусмотрен обмен данными с настольным ПК и его приложениями и доступ к Интернету.

ПРИМЕР

Процессор Pentium-IV/1,4 ГГц — первый представитель новой серии процессоров фирмы Intel. Появился Northwood, улучшенный вариант Pentium-IV, который будет работать на частоте 2 ГГц. Из недорогих процессоров линии Celeron появился Timna, который содержит встроенную графику и контроллер памяти. Этот процессор предназначен и для ноутбуков. Для персональных станций разработаны и используются мощные процессоры на частоте 4 ГГц и выше.

В заключение приведем историю механизации (автоматизации) «умственной деятельности», которая началась с механизации (автоматизации) цифровых вычислений.

Основные этапы развития техники цифровых вычислений:

- 1) правила механизации счета, описанные на глиняных табличках в Древнем Вавилоне (1800—1600 гг. до н. э.);
- 2) счеты: римский абак (V в. до н. э.), китайский суан — пан, русские счеты на десятичной основе, японский соробан;

- 3) Леонардо да Винчи разработал проект счетной машины для выполнения действий над 12-разрядными числами (XV в.);
- 4) механические устройства для арифметических операций Д. Непера (1617);
- 5) суммирующая машина Жаккарда (Шиккарда) для сложения и вычитания (1683 г., эскизы были разработаны раньше);
- 6) машина Блеза Паскаля для всех арифметических действий (1642—1645);
- 7) машина для четырех арифметических действий Лейбница (1694);
- 8) логарифмические линейки (Англия, XVII в.);
- 9) перфокарты Жаккара для управления ткацким станком (1801—1804);
- 10) Карл Томас изобрел арифмометр (1820);
- 11) арифмометр В. Однера (1874);
- 12) «разностная» машина Чарльза Бэббиджа (1840 г., около одной операции сложения или вычитания в секунду, около 50 тыс. деталей), который организовал первую вычислительную лабораторию, где работала дочь Джорджа Байрона леди Ада Лавлейс, писавшая Бэббиджу, что «составила список операций для вычисления каждого коэффициента для каждой переменной», т. е. первую программу для вычислительной машины;
- 13) Барроуз разработал машину, печатающую исходные данные и результат (1885);
- 14) Холлерит изобрел (1886) табулятор на перфокартах (примерно в это время формируется основа для фирмы IBM);
- 15) период Джона фон Неймана (1928 г. — теория ЭВМ, 1940 г. — первый компьютер MANIAC, 1945 г. — принципы фоннеймановской архитектуры);
- 16) проект цифровой машины на базе двоичной арифметики Конрада Цузе (1936) — немецкого ученого, разработавшего в то время наиболее универсальные принципы проектирования машин, хотя о его результатах мир долго не знал, ибо все его разработки были потеряны во время войны, в том числе электрорелейная вычислительная машина (1941);
- 17) разработка в 1944 г. в США механико-электрорелейной машины «Марк-1» (Г. Айкен), затем машины «Эниак» (Эккерт и Моучли, 1946 г.), а также работы американца болгарского происхождения Джона Атанасова в этом направлении;
- 18) появление ЭВМ первого поколения (1946—1957) на базе электронных ламп (как основного элемента устройства ЭВМ);
- 19) ЭВМ второго поколения (1958—1963), база — транзисторы;
- 20) ЭВМ третьего поколения (1963—1970), база — интегральные схемы (ИС); уже в конце 60-х гг. были разработаны интегральные схемы, содержащие на кристалле кремния площадью несколько квадратных миллиметров несколько сотен вентилях, при этом сложность логических схем возросла на порядок, а процесс производства стал более технологичным;

- 21) ЭВМ четвертого поколения (1970—1985), база — Большие ИС (БИС), Сверхбольшие ИС (СВИС); разработка БИС и СВИС привела к настоящей компьютерной революции вкупе с разработкой и внедрением в различные области микропроцессоров;
- 22) ЭВМ пятого поколения (с 1985 г.), база — СВИС, опτικο-волоконные средства передачи информации, пейрокомпьютеры (компьютеры, моделирующие работу головного мозга человека), интеллектуализация ЭВМ; главная задача ЭВМ пятого поколения — обеспечение естественного интерфейса, общение пользователя за счет решения двух основных интеллектуальных функций человека: умения распознавать сообщения и ситуации (текстовые, речевые, графические, звуковые и т. д.) и умение принимать решения по этим результатам распознавания и по ранее накопленным знаниям в данной (смежной) области.
- 23) ЭВМ шестого поколения (теоретические разработки), база — средства интеллектуализации ЭВМ, например, ЭВМ способны добывать, обрабатывать информацию с помощью различных датчиков и устройств, экспертных систем и интеллектуальных систем различного назначения (говорят об интеллектуализированных ЭВМ — ЭВМ, доступных конечному пользователю в соответствии со средствами новых информационных технологий и ориентированных на способы получения знаний).

В СССР первыми ЭВМ были МЭСМ (малая электронно-счетная машина) и БЭСМ (большая электронно-счетная машина), разработанная под руководством академика Сергея Алексеевича Лебедева в 1951 г. (это год ее первого запуска, а точнее, это произошло 25 декабря 1951 г.).

Изобретение полупроводника, или транзистора, можно отнести к 1948 г. (Bell Labs, Джон Бардин, Вальтер Бреттейн и Вильям Шокли). В 1959 г. инженеры фирмы Texas Instruments разместили несколько транзисторов на одной подложке и соединили эти транзисторы без проволоки. Так родилась интегральная схема. Первая интегральная схема содержала только шесть транзисторов, а современные процессоры имеют миллионы транзисторов.

Термин «микропроцессор» впервые был употреблен в 1971 г. (фирма Intel выпустила микропроцессор серии 4004 — однокристалльный центральный 4-разрядный параллельный сумматор с шестнадцатью 4-разрядными регистрами, реализованный на 2300 транзисторах, который выполнял 45 команд). Его преемником был процессор 8008 — восьмибитовый микропроцессор, появившийся в 1972 г. В конце 1973 г. фирма Intel выпустила процессор 8080, который был в 10 раз быстрее, чем процессор 8008, и мог адресовать внушительные тогда 64 Кбайт памяти. Это был один из стимулов к созданию персональных компьютеров.

Первым ПК был компьютер «Альтаир-8800» (США, 1974—1975). Он включал процессор 8080, источник питания, переднюю панель с огромным количеством лампочек и огромной памятью в 256 байт, открытую архитектуру (разъемы), которые позволяли

подключать внешние устройства и дополнительные ресурсы, не входящие в стандартный набор. Новый процессор вдохновил другие компании писать программы, включая операционную систему CP/M (Control Program for Microprocessor — Управляющая Программа для Микропроцессора) и первую версию языка Microsoft BASIC. Фирма IBM выпустила свой первый персональный компьютер в 1975 г. (16 Кбайт памяти, встроенный интерпретатор BASIC и встроенный кассетный лентопротяжный механизм, цена около \$ 9000). Дебют IBM PS (Intel 80286) состоялся в 1981—1982 гг.

Следует отметить также мини-ЭВМ PDP-8 (фирма DEC) — яркий пример эффективного использования во многих различных областях ЭВМ с упрощенной архитектурой (всего 7 основных операций с 8-разрядными данными).

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Классифицируйте ЭВМ и дайте характеристику представителю каждого класса (память, быстродействие).
2. Какие классы задач решаются на ЭВМ? Приведите примеры.
3. Какова роль ЭВМ в процессе познания?
4. Что понимается под архитектурой ЭВМ?
5. Изобразите структурную схему ЭВМ.
6. Что такое АУ, УУ, процессор? Перечислите их функции.
7. Что такое персональный компьютер? Чем он отличается от других типов ЭВМ?

ЗАДАЧИ

1. Запишите числа 0, 42, 1000, 1, -25, -100 в формате целых чисел в шестнадцатеричную ячейку памяти, изобразив ее схематически.
2. Запишите числа 99, 12.5, 0.025, -4.18, -0.01 в формате вещественных чисел с фиксированной запятой (ячейка памяти 16-разрядная, разряды 1—8 отводятся под целую часть, разряды 9—15 — мантисса, 0 — знак). Оцените погрешность такого представления чисел.
3. Какое максимально положительное и минимально отрицательное число можно записать в указанную для задачи 2 ячейку памяти?
4. Запишите числа 99, 9999, 12.7, 0.005, -64.5, -0.002, 0 в формате чисел с «плавающей» запятой в 16-разрядную ячейку памяти (нулевой разряд — знак числа, первый разряд — знак порядка, разряды 2—12 — мантисса, разряды 13—15 — порядок). Оцените погрешность такого представления.
5. Какое максимально положительное и минимально отрицательное число можно записать в ячейку памяти, указанную в задаче 4?

6. Определите максимальную память (ОЗУ) ЭВМ, если ее разрядность равна 8, число выполняемых команд не более 100, адресность всех типов команд равна 2. Приведите расчеты.
7. Возможен ли теоретически процессор ЭВМ фоннеймановского типа с быстродействием 10^{16} операций в секунду, вмещающийся в квадратную микросхему размером в 1 см^2 ? Ответ поясните необходимыми вычислениями.

ТЕСТ

1. Устройствами ввода-вывода данных в ЭВМ являются:
 - а) АЛУ, сканер, дискета, клавиатура, принтер;
 - б) УУ, дисковод, модем, плоттер, сканер;
 - в) сканер, плоттер, мышь, дисплей, принтер;
 - г) ОЗУ, плоттер, клавиатура, принтер, мышь, дисплей.
2. Быстродействие компьютера характеризуется:
 - а) количеством выполняемых одновременно программ;
 - б) количеством машинных операций в секунду;
 - в) временем организации связи между устройствами ЭВМ;
 - г) характеристиками считывания из ОЗУ или записи в ОЗУ.
3. Компьютер не может работать без:
 - а) принтера; б) процессора; в) дискеты; г) звуковых колонок.
4. В качестве устройства памяти ЭВМ выступает:
 - а) ОЗУ; б) УУ; в) АЛУ; г) ППП.
5. Компьютер может работать без:
 - а) винчестера; б) сканера; в) процессора; г) системной шины.
6. Устройством конечного (непосредственного) ввода-вывода информации в ЭВМ не является:
 - а) принтер; б) сканер; в) процессор; г) плоттер.
7. Первым микропроцессором был микропроцессор Intel серии:
 - а) 4004; б) 8008; в) 8080; г) 80286.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
в	б	б	а	б	в	а

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. История развития технологии автоматизации вычислений.
2. История развития ЭВМ.
3. Машинная и обычная арифметика (сравнительный анализ).
4. Санитарно-гигиенические нормы безопасности работы на ЭВМ.
5. Джон фон Нейман.
6. Отечественные и зарубежные супер-ЭВМ (анализ и рейтинг).
7. Современные архитектуры ЭВМ.

15

АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ И МЕТОДЫ ТРАНСЛЯЦИИ

Математический язык как форма выражения научного знания использовался еще в древности при решении практических задач.

Развитие информатики, особенно таких ее разделов, как теория формальных языков, грамматик, структур, алгоритмических языков и методов трансляции, баз данных и знаний приводит к развитию объектов лингвистики. В частности, появилась возможность анализа и построения различного рода текстов и словарей, в том числе их автоматического построения.

Языком, определенным над алфавитом $X = \{x_1, x_2, \dots, x_n\}$, называется устная, звуковая, письменная или иная форма выражения, изображения всех слов над X , включая синтаксис (грамматику) — правила их конкатенации (соединения), образования слов и словосочетаний, структур слов и словосочетаний, семантику — правила проверки правильности, смысловой однозначности и совместности синтаксических конструкций языка, состоящих из лексем. Для устных (этнических) языков важна и фонетика — правила произношения значимых единиц речи, т. е. фонем.

ПРИМЕР

В частности, семантика изучает связи вида «знак, структура знаков $\Leftrightarrow$ значение $\Leftrightarrow$ объект»; синтаксис — связи вида «знак, структура знаков $\Leftrightarrow$ объект»; прагматика — связи вида «знак, структура знаков $\Leftrightarrow$ использование».

Функции языка — отображение, передача, сжатие, хранение, актуализация информации с помощью сообщений на этом языке. Язык — способ передачи (актуализации) информации (сообщений).

ПРИМЕР

Запишем на учебном алгоритмическом языке минимальное число условий, обеспечивающих непопадание точки $M(x; y)$ внутрь кольца, образованного окружностями с общим центром в точке $A(a; b)$ и радиусами R и r соответственно. Пусть $R > r$. Вспомним уравнения этих окружностей:

$$(x - a)^2 + (y - b)^2 = R^2, \quad (x - a)^2 + (y - b)^2 = r^2.$$

Множество точек, попавших внутрь окружности R , удовлетворяет неравенству вида $(x - a)^2 + (y - b)^2 < R^2$, а множество точек, не попавших внутрь окружности радиуса r , удовлетворяет неравенству $(x - a)^2 + (y - b)^2 > r^2$. Следовательно, чтобы точка $M(x; y)$ попадала в кольцо, образованное этими окружностями, необходимо выполнение приведенных выше двух неравенств вместе: $r^2 < (x - a)^2 + (y - b)^2 < R^2$. Условие непопадания

в кольцо точки $M(x; y)$ (по правилу де Моргана) будет на алгоритмическом языке иметь вид:

если $(r^{**2} \geq (x-a)^{**2} + (y-b)^{**2})$ и $((x-a)^{**2} + (y-b)^{**2} \geq R^{**2})$
то вывод ("Точка M не лежит в кольце")
иначе вывод ("Точка M лежит в кольце");
все.

Языки бывают естественного происхождения, например языки общения людей, используемые в некотором социуме, и искусственного происхождения, или **формальные языки**, разрабатываемые для общения человека с автоматом (ЭВМ) или для профессиональных приложений, для описания знаний, для моделирования и т. д. Во многих естественных языках, в отличие от формальных, грамматика языка определена лишь условно, часто не полностью.

С появлением же ЭВМ значительно повысились требования к точности, понятности и структурированности алгоритма. При реализации алгоритма на ЭВМ нужно быть уверенным в точности его описания, в правильности структуры алгоритма. Для этого нужна некоторая система точных правил записи, чтения, исполнения алгоритмов. В информатике такая система получила название **алгоритмического языка**. С одной стороны, он близок к естественному языку общения людей, так как состоит из слов, предложений и знаков препинания. С другой стороны, этот язык **формализованный** (формальный), так как он включает в себя и математическую символику, например числа, знаки каких-то алгебраических операций, знаки логических операций и т. д.

Основное значение алгоритмического языка — быть средством записи алгоритмов для исполнения, для изучения логики алгоритма. Основное отличие алгоритмического языка от языка программирования (хотя их часто отождествляют) состоит в том, что последний предназначен для записи и исполнения алгоритмов в виде, понятном и исполнимом ЭВМ.

Любые математические выражения записываются на алгоритмических языках по правилам не только математики, но и этого языка.

ПРИМЕР

На учебном алгоритмическом языке выражение

$$y = 2^{n^*} + \frac{x - 2\sin(x + \pi)}{\frac{e^x}{x}}$$

будет записано в виде $y = 2^{**}(n^{**2}) + (x - 2^{*}\sin(x + \pi)) / (\exp(x) / x)$. Обратите внимание на выражение в скобках. Если убрать скобки, то мы получим совсем другие выражения в соответствии со старшинством операций, выполняемых слева направо.

ПРИМЕР

Выражению на учебном языке $w = \sqrt{b*b + c*c + 2/a*x/(c+g)}$ соответствует математическое выражение вида

$$w = \sqrt{b^2 + c^2 + \frac{2x}{a(c+g)}}.$$

Вычисление значений выражений на алгоритмическом языке также происходит по законам этого языка и в соответствии с принятым в языке старшинством операций.

ПРИМЕР

Найдем $b = \ln(\exp(3)) + \max(\min(3, 2.5), 3.6) + \text{mod}(13, 4) * \text{int}(-1.5)$.
Результат: $b = 3 + \max(2.5, 3.6) + 1 * (-2) = 3 + 3.6 - 2 = 4.6$. В выражении вида $A = 3 * \ln(x) / y^{**2} * (x - 1) + \exp(y)$ вычисление подвыражений выражения происходит в следующем порядке: $x - 1$, $\ln(x)$, $\exp(y)$, y^{**2} , $3 * \ln(x)$, $3 * \ln(x) / y^{**2}$, $3 * \ln(x) / y^{**2} * (x - 1)$, A .

Первые ЭВМ поставлялись заводами без какого-то программного обеспечения. Программисту приходилось описывать в своей программе все необходимые для работы программы, вплоть до перевода чисел в другую систему. Разработка же специальных алгоритмических языков значительно упростила процесс программирования, сделала его независимым от конкретной ЭВМ и тем самым увеличила контингент программистов, которые стали называться пользователями, так как большинству из них уже, собственно говоря, не нужно было программировать в обычном (процедурном) понимании. Дальнейшее развитие программирования вначале пошло по двум направлениям: **прикладное** и **системное программирование**. В настоящее время на стыке этих направлений появилось и третье — **инструментальное программирование** как системного, так и прикладного направления. Впрочем, границы эти условны и размыты. Прикладной программист разрабатывает программы решения конкретных естественно-научных задач. Это, как правило, профессионал в некоторой области знания, обладающий еще и навыками программирования и знакомый с использованием ЭВМ в своей профессиональной области. Системный программист пишет программы автоматизации процесса написания и отладки этих прикладных программ, распределения ресурсов между прикладными программами, управления процессом прохождения таких прикладных программ на ЭВМ, например разрабатывает ОС.

Язык считается более высокого уровня, чем он ближе к языку естественному, и более низкого уровня, чем он ближе к языкам, реализуемым аппаратно, т. е. машинным.

Охарактеризуем эти уровни алгоритмических языков:

- **языки запросов**, или **непроцедурные языки**, — языки, предназначенные для формулирования условия решаемой задачи (того, «что делать», а не того, «как делать»; на таком языке описываются спецификации решаемой задачи), например язык, на котором можно осуществлять диалог с некоторым пакетом при-

кладных программ, языки имитационного моделирования, в частности язык SLAM и др.;

- языки высокого уровня, или проблемно-ориентированные языки, — языки для решения определенного, но достаточно широкого класса задач, например языки FORTRAN, BASIC, LISP и др.;
- ассемблеры — семейство языков, предназначенных для укрупнения и символической (мнемонической) записи машинных команд;
- языки микроопераций — языки разработки микропрограмм (собственно, это и есть языки машинных операций).

ПРИМЕР

Отметим следующие языки, которые оказали наибольшее влияние и преемственность в развитии языков программирования:

- ФОРТРАН — язык научно-технических расчетов с простым синтаксисом, который до сих пор развивается и любим (например, физиками, создавшими на нем большое программное обеспечение для решения своих задач), разработан в 1955 г. IBM;
- АЛГОЛ, или АЛГОЛ-60, — язык вычислительного характера с развитыми синтаксическими и семантическими средствами, разработан в 1960 г. Международным комитетом ученых;
- БЕЙСИК — простой язык для начинающих, имеющий большое количество диалектов, разработан в 1965 г. в Дортмунде;
- ПАСКАЛЬ — разработан в качестве языка учебного и исследовательского характера в 1969 г. в Цюрихе Н. Виртом, имеет строгие синтаксические и семантические правила, структуру и логику, хорош в качестве первого изучаемого языка;
- СИ — разработан лабораторией BELL (США, 1972 г.) в качестве языка поддержки и программирования операционной системы UNIX, предоставляет большие возможности для работы системных программистов;
- ПРОЛОГ — язык для логического программирования, для преобразований деревьев (язык ЭВМ пятого поколения), разработан Колмероз в 1971 — 1973 гг.;
- JAVA — язык, ориентированный на Интернет и серверы WWW;
- HTML — язык гипертекстовой разметки (Hyper Text Markup Language), был предложен Тимом Бернерсом-Ли в 1989 г. в качестве поддержки www-документов.

ЭВМ воспринимает и может исполнять только программы, написанные на ее машинном языке, в машинных кодах. Поэтому для перевода с других языков и составляется специальная программа, называемая транслятором. Построенный алгоритм может быть представлен в виде программы на языке программирования. Эта программа затем может быть преобразована в корректную последовательность команд и данных, загружаемых в память ЭВМ и выполняемых в ней аппаратно. При этом, чем выше уровень используемого при составлении программы исходного языка, тем больше

работ по трансляции программы, тем больше команд машинного языка обычно сопоставляется программе. Транслятор переводит программу с исходного языка программирования на язык загрузки ее в память и исполнения — язык ЭВМ. Во время перевода, трансляции программы, транслятор по тексту программы на исходном языке генерирует соответствующую ей последовательность на языке ЭВМ (машинных команд).

Существует два режима трансляции: компиляция и интерпретация. При интерпретации перевод на язык машины и выполнение каждой команды исходной программы осуществляется поочередно, а полученная машинная программа пригодна только для одноразового решения задачи; данные вводятся при этом по мере трансляции. При компиляции до выполнения программы осуществляется ее полный перевод на машинный язык ЭВМ, полученная программа редактируется и загружается со всеми необходимыми для выполнения транслированной программы программами ОС в память ЭВМ и получается так называемый загрузочный модуль, пригодный для многократного использования без трансляции. Компилятор преобразует программу на языке высокого уровня в эквивалентную по входу-выходу рабочую программу в машинных командах. Интерпретатор осуществляет независимую трансляцию каждой команды программы, сразу же ее выполняет, поэтому интерпретатор должен все время находиться в памяти ЭВМ вместе с самой выполняемой программой.

ПРИМЕР

Рассмотрим, как будет исполняться программа в разных средах программирования на базе стандарта одного и того же языка Бейсик — в среде Turbo-Basic (транслятор-интерпретатор) и среде QuickBasic (транслятор-компилятор). Программа для T-Basic исполняется (интерпретируется) команда за командой по циклу: «ввод команды — перевод команды на внутренний машинный язык (если там нет ошибок) — соединение (ввод) необходимых данных для выполнения данной команды — исполнение». Программа для QBasic исполняется (компилируется) по циклу: «перевод всех команд программы на внутренний язык — исправление ошибок в программе (отладка) — соединение с необходимыми для всей программы входными данными — исполнение всей программы».

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое алгоритмический язык и для чего он предназначен?
2. Что такое алфавит языка, лексический анализ, синтаксический анализ и семантический анализ?
3. Что такое язык программирования? Каково его назначение?
4. Что такое прикладное и системное программирование, каковы их задачи?
5. Проведите сравнительный анализ разных типов языков.

6. Что такое транслятор? Какие режимы трансляции бывают?
7. Каково старшинство операций в учебном языке? Запишите выражение на алгоритмическом языке.

ЗАДАЧИ

1. Запишите пять математических выражений на алгоритмическом языке.
2. Запишите пять записей алгоритмического языка на математическом языке.
3. Определите указанные функции через другие операции и функции оптимально: $\max(x, y)$, $\min(x, y)$, $\text{abs}(x)$, $\text{sign}(x)$, $\text{int}(x)$, $\text{div}(x, y)$, $\text{mod}(x, y)$, $(x**y)$.
4. Даны описания переменных: цел x, y , вещ a, b, c , сим u , лит s , лог r . Укажите неправильные места, причины появления ошибок в выражении:
 - а) $y := \sin(x)*s - \text{длина}(s) + \text{длина}(u)$;
 - б) $x := \text{int}(x/y) + \text{mod}(x, a) - \text{div}(\text{sign}(x), \text{int}(b))$;
 - в) $a := \exp(u) + s - \text{sqrt}(x + y) : (\text{длина}(u) - 1)$.
5. Вычислите значение выражения:
 - а) $y := \sin(\arcsin(1)) * \text{длина}(\text{"sort"}) + \text{длина}(\text{" "})$;
 - б) $x := \text{int}(2.6/5.9) + \text{mod}(123, 23) - \text{div}(56, \text{int}(5.8))$;
 - в) $x := \ln(\exp(3)) + \max(\min(6, 3.5), 3) + \text{mod}(3, 4) * \text{int}(-4.8) - \text{div}(5.99, 3)$.
Укажите порядок вычисления всех подвыражений.
6. Запишите условие попадания точки $A(x; y)$ в область, ограниченную окружностью с центром в точке $B(a; b)$ и радиусом r , и полосой, ограниченной параллельными прямыми, проходящими через точки $C(0; b)$, $D(a; 2b)$ и $M(a; 0)$, $N(2a; b)$.
7. Запишите условие попадания точки $A(x; y)$ внутрь треугольника, заданного координатами вершин.

ТЕСТ

1. Если переменные $x; y$ описаны как цел, а z как вещ, то количество различных ошибок в записи выражения $y := \text{mod}(y, z) + \max(\text{int}(y), y/x)$ равно:
 - а) 4; б) 3; в) 2; г) 1.
2. Для переменной x типа цел один и тот же результат при любом входном $x > 0$ дают все функции списка:
 - а) $\text{mod}(x, 1)$, $\text{div}(x, 1)$; в) $\text{int}(x)$, $\text{sign}(x)$;
 - б) $\text{int}(x)$, $\text{div}(x, 1)$; г) $\text{sign}(x)$, $\text{mod}(x, x)$.
3. Выражения $\text{abs}(-5) + \text{int}(2.6) * \text{mod}(6, 6)$; $\max(\text{mod}(12, 5), \text{div}(3, 3)) * \text{int}(7)$; $\exp(\ln(2)) + \text{mod}(13, 30)$; $\text{sign}(-8) * \text{sign}(8)$; $\text{int}(-6.7) + \text{div}(45, 56)$ в этой же последовательности равны:
 - а) 5; 14; 15; -7; -1; в) 5; 14; 15; -1; -6;
 - б) 5; 15; 14; -7; -1; г) 5; 14; 15; -1; -7.

4. Ранжирован по старшинству операций в алгоритмическом языке список:
- а) степень, умножение и деление, сложение и вычитание;
 - б) умножение, степень, деление, сложение, вычитание;
 - в) степень, сложение, умножение, вычитание, деление;
 - г) деление, умножение и степень, сложение и вычитание.
5. Число различных ошибок в записи выражения $y := \text{mod}(y, z) + \text{div}(\text{int}(y), y/x)$, где x, y описаны как цел, z как вещ, равно:
- а) 4; б) 3; в) 2; г) 1.
6. Последовательное выполнение команд $a := \text{abs}(-5) + \text{trunc}(2.6) * (6 \bmod 6)$ и $a := \max(a \bmod 5, a \text{ div } 3) * \text{trunc}(a)$ дает значение:
- а) $a = 10$; б) $a = 6$; в) $a = 5$; г) $a = 3$.
7. В списке $\sin(x)$, $\text{int}(x)$, $\text{mod}(x, y)$, $\exp(x)$, $\text{sign}(x)$, $\ln(x)$, $\text{abs}(x)$ число функций, всегда имеющих при целых x и y целочисленные значения, равно:
- а) 1; б) 2; в) 3; г) 4.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
г	б	г	а	в	в	г

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Языки естественного и искусственного происхождения.
2. Метаязыки. Язык Бэкуса—Наура.
3. Непроцедурные языки.
4. Процедурные языки.
5. Языки ассемблирования (ассемблеры).
6. Машинные коды.
7. Языки для www-приложений. Языки HTML, JAVA.

16

ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

Программное обеспечение (ПО, или software — гибкое обеспечение) — совокупность программ ЭВМ, языков и трансляторов общения с ЭВМ. Предлагаемая на рисунке 19 схема отражает основные структуры ПО и часто является условной — блоки структуры могут иногда перекрываться.

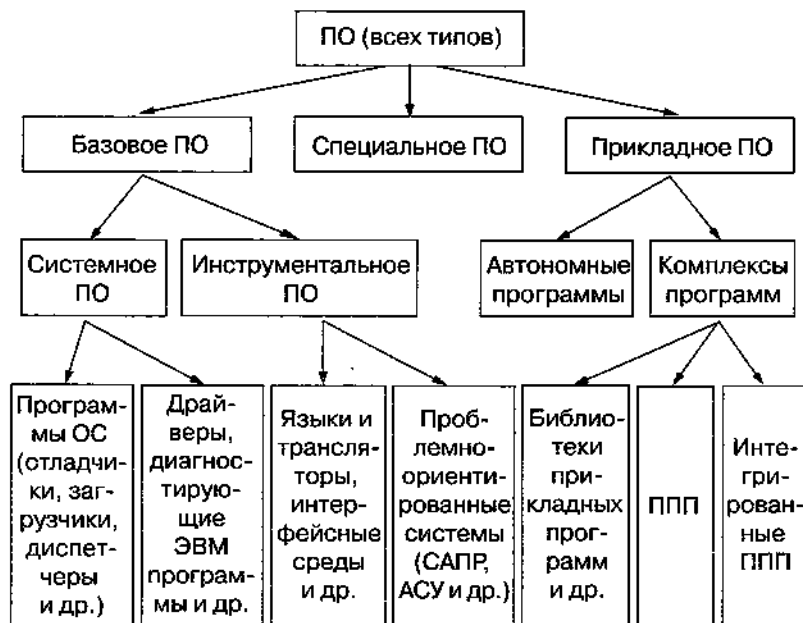


Рис. 19. Структура ПО

Классификация ПО проводится и по другим критериям, например по удобству, комфортности использования (дружественности ПО пользователю), по предоставляемому набору возможностей, по типу использования и т. д.

Техническое обеспечение (ТО, или hardware — железное, жесткое обеспечение) — это совокупность технических средств для решения задачи с помощью ЭВМ (рис. 20).

Вычислительной системой (ВС) называется совокупность ПО и ТО.

Операционная система (ОС) — совокупность программ, обеспечивающих взаимодействие всех программных и аппаратных

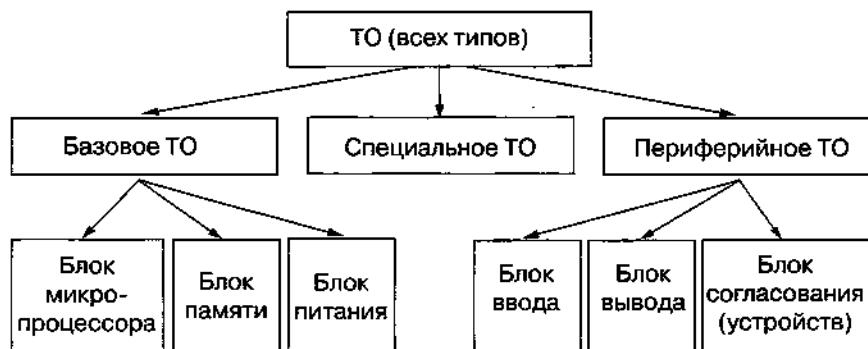


Рис. 20. Структура ТО

средств ЭВМ, т. е. работу всей ВС. Пользователь имеет дело не с самой ЭВМ, а с ВС, и это его взаимодействие с ЭВМ обеспечивает программы ОС. Процессор ЭВМ обращается за информацией только к оперативной памяти. Информация с периферийных устройств передается в ОЗУ машинными командами, операциями обмена по каналам связи процессорам специального назначения. Процессор (или центральный процессор многопроцессорной системы) и каналы работают независимо, обмениваясь специальными видами сигналов. Их совместная работа, а также ликвидация простоев процессора во время обменов невозможны без специальных систем, реализованных аппаратно, так называемых **систем прерываний**. При поступлении прерываний выполнение текущей программы прерывается и управление УУ передает ОС, которая и решает задачу «как и что делать дальше?». Возможны следующие причины прерываний: поступление сигнала о начале (завершении) обмена; сбой аппаратуры; вмешательство пользователя программы; незапланированная ситуация (в конфигурации, в программе, в режиме работы ЭВМ) и др. В момент прерывания процессор в специальных регистрах сохраняет текущее состояние данной (прерванной) программы в необходимом для ее дальнейшего продолжения виде и объеме (без потери устойчивой работы и информации). Прерывания могут быть: **аппаратные**, вызванные сигналом от какого-либо устройства; **логические**, вызванные возникновением нестандартной ситуации в работе центрального процессора; **программные**, вызванные из программы с помощью специального кода.

ПРИМЕР

Аппаратное прерывание может поступить от сетевого принтера, в случае необходимости «перекачки» данных в буфер. Логическое прерывание может возникнуть при делении двух чисел, когда делитель равен нулю (или очень мал). Программное прерывание может вызвать вирус, который таким образом перехватывает управление в системе и внедряется в систему.

Любая ОС значительно упрощает работу программиста, преобразуя физическую или техническую ЭВМ в другую, виртуальную (воображаемую) ЭВМ, которую легче и удобнее программировать, с которой легче логически работать. Кроме того, ОС обеспечивает пользователю доступ к программам ОС, сегментирование программ, использование стандартных процедур и т. д.

ПРИМЕР

При составлении прикладной программы программист совсем не заботится о распределении памяти под данные к программе, ему в принципе неважно, в какие участки памяти попадут его данные и откуда будут извлекаться эти данные при работе программы.

Основные функции операционной системы:

- 1) планирование работ при решении задач;
- 2) управление памятью;
- 3) управление заданиями (наборами связанных программ);
- 4) управление данными;
- 5) управление устройствами;
- 6) восстановление работоспособности в случае сбоев;
- 7) управление процессором.

Управление памятью состоит в эффективном размещении в ОЗУ в каждый такт времени выполнения задания (независимой единицы работы ОС) всех необходимых фрагментов задания; программист освобождается от изнурительной необходимости указания в самой программе областей памяти, где будут размещены фрагменты данной программы. Физическое размещение программы в памяти или в адресном пространстве задания выполняет ОС по указанным в программе логическим (математическим) адресам, объектам. Кроме того, в функции ОС по управлению ПО входит и распределение памяти между заданиями или пакетами заданий.

Управление процессором (процессорами). Функции управления процессором: создание и завершение текущего задания; предоставление текущей программе процессорного времени; фиксирование и реакция на состояние текущей программы; квантование программы — выделение текущей программе кванта времени, например 100 мс. При выполнении программы информация о состояниях регистров процессора (до и после выполнения задания) запоминается в специальной таблице, которая содержит необходимую информацию и определяется состоянием текущей программы в ОС. К этой таблице имеют доступ только программы ОС, которые по информации из этой таблицы изменяют текущее состояние программы, продолжают процесс выполнения задания.

Управление периферийными устройствами. Периферийным устройством называется устройство, подключаемое к ЭВМ некоторыми шинами к портам. Основные функции управления устройствами ЭВМ: организация и контроль обмена информацией между ОП и внешними устройствами по запросам заданий; обслуживание

определенной конфигурации устройств с помощью соответствующих каналов и драйверов (программ, обслуживающих устройства); отслеживание состояния устройства и реакции на его работу.

В современных ЭВМ роль каналов обмена с внешними устройствами выполняют специализированные процессоры (канальные процессоры). При получении очередного запроса на обмен по каналу ОС проверяет его занятость и в случае необходимости ставит запрос в очередь на его обслуживание; при этом информация из области обмена блокируется на обслуживание только данным каналом — защищается область обмена запроса.

Управление заданиями и прохождением заданий. В задании указываются работы, которые должна выполнить ОС: трансляция программ, выделение и распределение памяти и т. д. Особая программа ОС, называемая **планировщиком заданий**, собирает (опрашивая устройства, регистры состояния устройств) информацию о текущем состоянии всех заданий, назначает им приоритеты выполнения, следит за выделением и освобождением ресурсов для этих устройств и т. д. Все задания на выполнение образуют пакет заданий, а выбор задания на выполнение из пакета зависит от многих факторов: от эффективности по времени, по использованию аппаратуры, по загрузке процессора, частоте обращений и т. д. Вся такая статистика собирается по заданиям и по пакету и учитывается при выборе нужной стратегии выполнения заданий планировщиком.

Управление данными. Используемые заданием данные и структуры данных с помощью специальных программ отображаются на реальные физические структуры, носители данных. Широко используется принцип виртуализации, который может реализовываться **файловой системой** — специальной системой обмена данными между программами пользователя и операционной системой.

Файл может отображаться на различные физические носители (копирование файлов), включая ячейки ОЗУ, а следовательно, быть доступным нескольким программам. Файловая система должна обеспечивать основные операции над файлами: создание, модификация (в том числе расширение и сжатие), уничтожение, чтение (запись), перемещение файла.

Работу с файлом пользователь осуществляет на абстрактном уровне (при проектировании структуры данных) или на логическом уровне (при программировании), а ОС — на физическом уровне (при реализации).

Файловая система ведет справочник файлов, в котором регистрируются файлы, используемые в данном задании в данный момент, — открытые файлы.

Операционные системы бывают различных классов:

- **однозадачные**, решающие в каждый момент времени только одну задачу;
- **многозадачные мультипрограммной обработки**, загружающие в ОЗУ последовательность (пакет) независимых задач, организующие очередность обслуживания их процессором и устройствами,

- а затем решающие их по очереди, выделяя для каждой ресурсы ЭВМ на некоторый квант времени, например на 0.1 с;
- **реального времени**, обрабатывающие информацию по мере поступления от источника, например от какого-то метеорологического спутника и др.

ПРИМЕР

Рассмотрим основные элементы ОС Windows.

Работа с мышью. После загрузки Windows на экране появляется курсор мыши. Перемещение мыши по столу вызывает перемещение курсора по экрану, а команды передаются нажатием (щелчком) кнопок.

Курсором мыши можно выполнять различные действия:

- 1) после появления вертикальной черты (|) можно установить курсор ввода, если щелкнуть левой кнопкой мыши на позиции указателя мыши;
- 2) после появления стрелки, направленной налево вверх (↖), можно выбирать из «покрываемой» этой стрелкой линейки меню, команд меню, кнопок;
- 3) после появления стрелки, направленной направо вверх (↗), можно выделять строки текста, ячейки таблицы;
- 4) после появления крестообразного курсора (✚) можно перемещать окно или изменять его размер, зацепив рамку окна этим курсором.

Чтобы активизировать какой-либо объект, необходимо указать его курсором мыши и нажать не левую или правую кнопку мыши и сразу отпустить ее — это действие назовем «щелчок по клавише мыши» или «щелкнуть мышкой».

Левая кнопка мыши используется для запуска приложений и открытия папки (двойной щелчок — выполнение двух последовательных щелчков); для выбора пунктов меню и выделения отдельных объектов, таких, как файл, ярлык, окно (одинарный щелчок); для выделения некоторых областей текста или рисунка; для перемещения объекта по экрану (для этого указать курсором мыши объект, нажать левую кнопку мыши и, не отпуская левой кнопки мыши, перетащить курсор в нужное место, затем отпустить левую кнопку — происходит фиксация нового положения объекта). Правая кнопка мыши используется в основном для вызова меню из пунктов, которые подходят для данного объекта.

Папка, приложение и документ. Папка — группа файлов, объединенных по какому-либо критерию. Это развитие понятия каталога (директории). Приложение — это обычно прикладная или инструментальная программа, используемая в среде Windows. Документ — это любой объект (текстовый, графический, аудио- или видеофайл), обрабатываемый некоторым приложением. Разрешены любые имена файлов до 255 символов, в качестве которых можно использовать любые символы, кроме «?», «\», «*», «"», «<», «>», «|», в том числе и русского алфавита (лучше латинские имена).

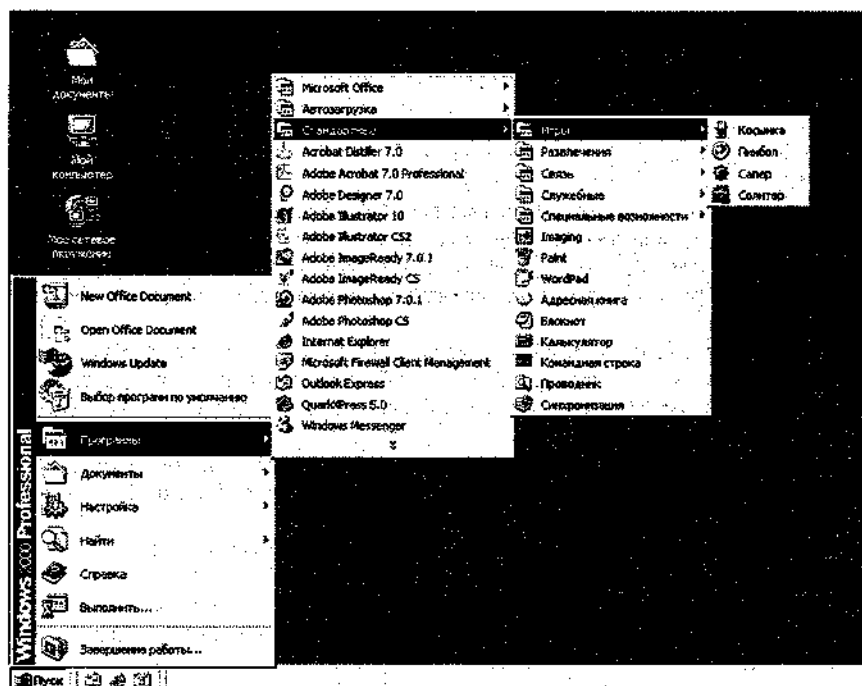


Рис. 21. Рабочий стол и Панель задач

Рабочий стол и Панель задач. Рабочий стол и Панель задач появляются на экране при включении компьютера и по окончании загрузки (рис. 21).

На Рабочем столе располагаются пиктограммы (иконки, значки). Пиктограмма — это небольшой рисунок (слева на рисунке 21) с комментариями (справа), отражающий назначение того объекта (приложения, документа), которому он соответствует. **Ярлык** — разновидность иконки в Windows. Он представляет собой указатель на объект и помогает найти его на диске. В левом нижнем углу ярлыка находится стрелка. Результат щелчка по ярлыку тот же, что и по самому объекту. Двойной щелчок по ярлыку открывает объект, с которым он связан.

На стандартном Рабочем столе располагаются следующие пиктограммы.

Мой компьютер. Двойной щелчок по пиктограмме **Мой компьютер** открывает папку с одноименным названием; с помощью этой папки пользователь имеет доступ ко всем папкам и файлам на компьютере, к дискам, различным устройствам и их настройке.

Корзина. Служит для удаления папок и файлов, их временного хранения на диске (до полного избавления от них в режиме **Очистить корзину** или до необходимости достать из корзины — ре-

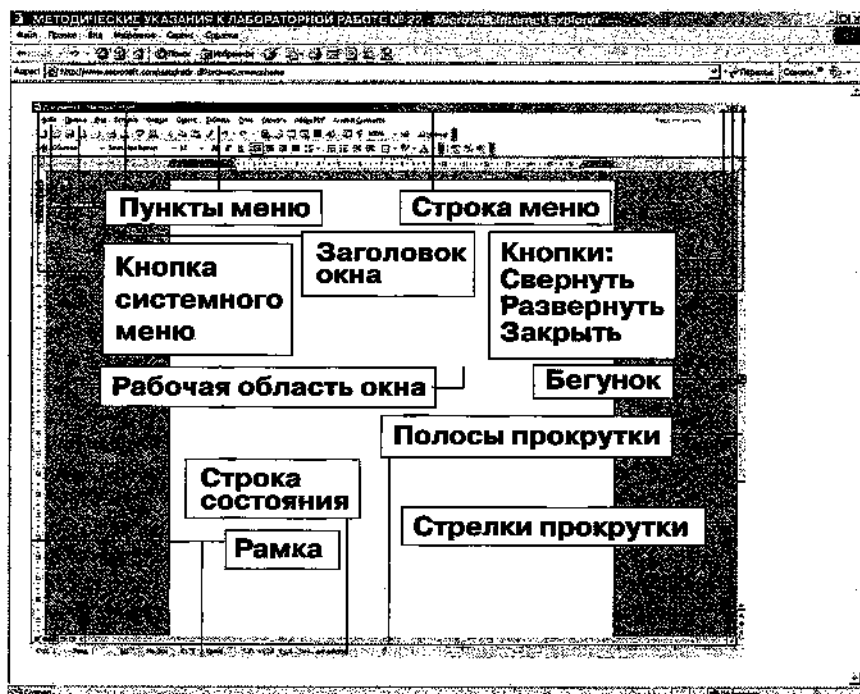


Рис. 22. Окна рабочего стола

жим **Восстановить файл**, если до этого **Корзина** не была очищена); содержимое **Корзины** можно просмотреть двойным щелчком по пиктограмме **Корзины**.

Пользователю постоянно доступна **Панель задач**. **Панель задач** может занимать различное положение: находиться на **Рабочем столе** или появляться только тогда, когда курсор мыши приближается к границе экрана (всплывать). **Панель задач** содержит кнопку **Пуск** с логотипом **Windows** и кнопки с именами открытых приложений и папок. При установке указателя мыши на кнопке появляется подсказка с названием приложения и документа. Кнопки можно использовать для переключения между окнами.

Справа на панели задач расположены пиктограммы **Системные часы**, **Звук**, **Индикатор текущей раскладки клавиатуры** (Ru соответствует кириллице, En — латинице). Щелчок **Индикатора раскладки клавиатуры** вызывает окно переключения языка. После установки указателя мыши на **Системных часах** появляется текущая дата. Двойной щелчок **Системных часов** вызывает на **Рабочий стол** часы с календарем, и можно корректировать текущее время и дату. Щелчок значка **Звук** вызывает появление значка с движком, управляющим громкостью звука, и можно включать/выключать звук (**Вкл/Выкл**).

Окна. **Окно** — обрамленная область экрана, в которой может отображаться приложение, документ, сообщение. На **Рабочем столе** можно увидеть три типа окон: окна приложений, документов и вспомогательные (рис. 22).

В верхней части каждого окна приложений находится строка заголовка, в которой отображается имя открытого приложения. В левой части строки заголовка находится кнопка вызова системного меню, активизируемого щелчком левой клавиши мыши. Чтобы убрать его, достаточно щелкнуть той же клавишей в любом месте Рабочего стола. В правой части строки заголовка расположены три кнопки, причем на двух из них пиктограммы всегда постоянны, а на средней кнопке пиктограммы изменяются в зависимости от состояния окна (распахнуто на весь экран или нет). Левая кнопка (со значком подчеркивания) сворачивает окно до иконки на Панели задач. Средняя кнопка (с двумя прямоугольниками) разворачивает окно на весь экран или восстанавливает его первоначальные размеры. Правая кнопка (с крестиком) закрывает окно и прекращает выполнение приложения.

Под строкой заголовка окна приложений находится строка меню. Команды меню активизируются двумя способами: меню открывается при щелчке левой клавиши мыши по названию меню или с помощью клавиатуры. В каждом пункте меню подчеркнута первая буква. Одновременным нажатием клавиш **Alt + Буква**, подчеркнутая в имени меню, можно открыть меню. Команды меню можно выбрать клавишами, соответствующими подчеркнутым буквам.

Ряд команд можно вызвать на выполнение с помощью клавиш или комбинаций клавиш, не открывая меню. Для них справа от их имени указывается соответствующая комбинация. Например, **Alt+F4** для команды **Заккрыть**.

У нижнего края окна находится строка состояния. Она содержит информацию о режимах работы приложения. Размеры окна можно изменять. Для этого нужно установить курсор мыши на границу окна, он примет вид двунаправленной стрелки, показывающей направления изменения размеров окна, затем нажать левую клавишу мыши и установить новые размеры окна.

Полоса прокрутки (вертикальная или горизонтальная) — это полоса вдоль нижней или правой границы окна. Она состоит из стрелок прокрутки, бегунка, полюсы, по которой движется бегунок прокрутки, и служит для просмотра содержимого окна, не уместящегося в его размерах. Окно документа может быть свернутым (при этом остается на экране лишь заголовок с названием документа), окном нормального размера (не развернутым на все Рабочее поле) или полностью развернутым в пределах Рабочего поля окна. Для некоторых приложений можно располагать на Рабочем поле несколько окон документов в различных состояниях. Документ в нормальном окне имеет собственную строку заголовка, содержащую имя документа. При увеличении размеров окна до максимального его имя появляется в строке заголовка приложения

после имени приложения. Окна документов, так же как и окна приложений, имеют кнопки для изменения вида окна. Установив указатель мыши на границу окна, можно изменить размеры окна. Вспомогательные окна используются для вывода сообщений, организации диалога с целью запроса параметров команды. Как правило, эти окна имеют фиксированный размер. Одновременно на Рабочем столе могут находиться несколько окон. Чтобы упорядочить окна, необходимо щелкнуть правой клавишей мыши на пустом месте Панели задач и выбрать один из способов упорядочивания: **Окна каскадом** (размещаются открытые окна «в стопку»), **Окна сверху вниз**, **Окна слева направо**. Выбор осуществляется одиночным щелчком левой клавишей. Чтобы переместить окно, надо установить курсор на строку заголовка окна, нажать левую кнопку и, не отпуская ее, переместить курсор в нужное место, отпустить кнопку мыши.

Главное системное меню. Щелчком мышью на кнопке **Пуск** можно вызвать **Главное системное меню**, запускающее программу, открывающую документ, вызывающее **Панель управления настройкой компьютера** и т. д. Вид **Главного системного меню** может быть различным на разных компьютерах. При стандартном меню на экране появятся пункты: **Программы**, **Документы**, **Настройка**, **Поиск**, **Справка**, **Выполнить**, **Завершение работы**.

Если у пункта есть подменю, то справа от названия появляется значок с изображением треугольника. Вызов подменю осуществляется мышью или нажатием клавиши.

Пункт **Программы** вызывает на экран список доступных приложений. Пункт **Документы** вызывает на экран список документов, с которыми недавно работали. Пункт **Настройка** имеет подменю из команд **Панель управления**, **Принтеры** и **Панель задач**, которые позволяют настроить Windows, внешний вид **Панели задач** и состав **Основного меню**. Пункт **Найти** осуществляет поиск файлов или папок. Пункт **Справка** используется для получения доступа к справочной системе. Пункт **Выполнить** позволяет запустить любое приложение. Для этого в окне **Запуск программы** необходимо ввести путь и название приложения. Если точно неизвестно местонахождение приложения, то можно выбрать кнопку **Обзор** и затем отыскать его. Пункт **Завершение работы** служит для выключения, перезагрузки компьютера.

Работа с папками (файлами). Перед переименованием, копированием или перемещением папки (файла) ее (его) выделяют в окне щелчком кнопки мыши по значку. Выделенная папка может быть переименована с помощью пункта контекстного меню **Переименовать**. Для копирования файлов (папок) используют разные способы:

- 1) буфер обмена (Clipboard) используется для промежуточного хранения копируемого файла, фрагмента данных, документа; для занесения информации в буфер и ее извлечения оттуда приложения Windows имеют в своих меню команды **Вырезать** (выде-

ление информации в буфер с удалением при этом ее оригинала), **Копировать** (помещение выделенной информации в буфер без изменения оригинала) и **Вставить** (извлечение информации из буфера, если она есть);

- 2) выбрать нужный файл или папку, выбрать команду главного меню **Правка**, пункт **Копировать** (можно использовать контекстное меню или панель инструментов), открыть диск и (или) папку, в которую предполагается переместить копию, выбрать команду главного меню **Правка**, пункт **Вставить** (можно использовать контекстное меню и Панель инструментов);
- 3) нажатием левой кнопки мыши на пиктограмме файла или папки и транспортировкой его (ее) в другую папку, на диск, дискету, **Рабочий стол**. На экране при этом должны быть одновременно представлены копируемая папка-источник и папка-приемник документа (файл-источник — файл-приемник) — для чего могут быть использованы два окна, **Мой компьютер**, и если диски файла (папки)-источника и файла (папки)-получателя не совпадают, то файл (папка) будет скопирован(а).

Справочная система Windows. Windows имеет развитую справочную систему. На **Рабочем столе** справка доступна из **Главного системного меню**. В результате вызова справки на экране появится окно **Справочной системы Windows**, содержащее три вкладки: **Содержание**, **Указатель**, **Поиск**.

Содержание отображает список всех разделов справки. **Указатель** дает возможность найти справку по ключевому слову (ввести первые буквы названия раздела). **Поиск** позволяет проводить поиск терминов справочной системы. Далее появляется прокручиваемый список предметного указателя с названиями разделов и терминов. Необходимо выбрать нужную строку и нажать на кнопку **Вывести**. Появится окно с текстом нужной справки. Оно имеет четыре кнопки: **Скрыть**, **Назад**, **Вперед**, **Параметры**. Нажатие кнопки **Параметры** вызывает меню, с помощью которого можно скопировать справку в буфер обмена, вывести справку на печать, изменить размер шрифта справки, установить закладку и т. д.

Инструментальная система — программный комплекс, среда для упрощения, удешевления, ускорения разработки программ (обработки данных) как системного, так и прикладного назначения (инструментарий). С помощью одних инструментальных систем можно разрабатывать другие.

ПРИМЕР

Одним из наиболее широко распространенных видов сервисного инструментария являются **программы-архиваторы**, которые предназначены для архивации (упаковки) любых файлов путем сжатия хранимой в них (битовой) информации. При сжатии информации в файле уменьшается (обычно всегда имеющаяся из-за стандартизации представления машинными словами) избыточность и соответственно уменьшается требуе-

мый объем памяти для хранения. Сжатие производится различными способами, например за счет упрощения кодов, исключения из них неизменных битов или представления повторяющихся символов и последовательностей символов в виде коэффициента повторения и соответствующего набора символов. **Архивация** (упаковка) — помещение исходных файлов в архивный файл в сжатом виде. **Разархивация** (распаковка) — восстановление файлов из архива в виде до сжатия, архивации. Большие по объему архивные файлы размещаются на нескольких дисках (томах) и называются многотомными. Существует много программ-архиваторов, различающихся перечнем функций, сервисом, методом сжатия и т. д.: ARJ, ZIP, RAR и др. Упаковку и распаковку файлов выполняют одной и той же программой.

В пакет прикладных программ (или ППП) входят:

- 1) информационная модель предметной области ППП или описание, представление класса задач, решаемых с помощью ППП;
- 2) системное наполнение ППП, т. е. комплекс программ, обеспечивающих построение прикладных программ ППП (технологической цепочки), контроль за исполнением, обеспечение работы различных объектов;
- 3) функциональное наполнение ППП, т. е. комплекс прикладных программ, обеспечивающих решение любой задачи из предметной области ППП (реализацию алгоритмов или их фрагментов для решения этих задач, сборки из них программ, реализующих алгоритмы решения задачи);
- 4) языковые средства ППП или языки реализации прикладных программ ППП (функционального наполнения) и запросов к ППП (входной язык ППП);
- 5) база данных для хранения данных, передача их модулям ППП;
- 6) монитор (управляющая программа) ППП для ввода задания (запроса), его понимания (расшифровки) и построения технологической цепочки из модулей ППП для решения расшифрованной задачи.

Состав ППП:

- предметно-ориентированное обеспечение (функциональное наполнение);
- системно-ориентированное обеспечение (системное наполнение);
- интерфейсно-ориентированное обеспечение (языковое обеспечение);
- информационно-ориентированное обеспечение (база данных и знаний).

Предметно-ориентированное обеспечение включает:

- программы решения задач предметной области ППП, т. е. программы, реализующие алгоритмы или их фрагменты;
- средства сборки программ, т. е. построения и исполнения программы.

Программная единица ППП (модуль), как правило, оформляется на каком-то языке программирования. Сборка может осуществляться вручную, т. е. явным указанием последовательности выполняемых модулей, или автоматически — по спецификации предметной области ППП, с помощью специальной программы ППП — Планировщик вычислений.

Системно-ориентированное обеспечение — совокупность системных средств (программы, данные), обеспечивающих определенную дисциплину работы Пользователя с ППП. В него входят:

- монитор для управления процессом взаимодействия всех компонент ППП;
- трансляторы с входных языков, на которых реализованы модули пакета;
- служебные модули, реализующие взаимодействие ППП с ОС;
- средства работы с информационно-ориентированным обеспечением.

Интерфейсно-ориентированное обеспечение — совокупность языковых средств описания требований к модулям и к построению прикладных программ пользователей ППП.

Информационно-ориентированное обеспечение используется для межмодульных передач и хранения данных, для ведения баз данных. Предметно-ориентированное обеспечение определяется классом задач, решаемых средствами ППП. Это достаточно широкий набор задач, объединенных общими алгоритмами, данными и отношениями между ними.

ПРИМЕР

В экономических проблемах часто используют пакет «1С-Бухгалтерия». Фрагмент демоверсии этого пакета приведен на рисунке 23.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое ПО? Перечислите элементы ПО.
2. Что такое ТО? Перечислите элементы ТО.
3. Что входит в понятие «вычислительная система»?
4. Каковы основные элементы, особенности ППП?
5. Что такое ОС, каковы ее функции и основные элементы?
6. Что такое планирование ОС работ при выполнении заданий?
7. Что такое инструментальная система? Приведите примеры.

ЗАДАЧИ

1. К каким типам программ можно отнести драйвер клавиатуры, программу распознавания текста для сканера, архиватор?
2. Приведите и охарактеризуйте примеры ПО и ТО различного назначения.
3. Из программ: 1) компилятор; 2) интерпретатор; 3) текстовый редактор; 4) редактор связей — укажите программы, которые:



Рис. 23. Иллюстрация возможностей пакета «1С-Бухгалтерия».

- а) обрабатывают входные данные к программе; б) обрабатывают саму программу; в) обрабатывают программу и входные данные к ней; г) ничего по отношению к программе и данным не осуществляют.
4. Какие режимы запуска Windows существуют? Выполните запуск в каждом режиме. Завершите корректно работу в Windows.
5. Каково назначение папки «Мой компьютер»? Как изменить вид отображения дисков, файлов, папок? Выполните действия с этой папкой.
6. Как создать файл или папку, их копирование, перемещение, переименование, удаление и восстановление файлов? Выполните действия.
7. Выполните операции архивации и разархивации файла, всех файлов в каталоге, отдельных файлов в каталоге.

ТЕСТ

К базовому программному обеспечению не относятся:

- а) Windows-NT, драйвер клавиатуры;
- б) клавиатурный тренажер, программа интегрирования;
- в) пакет прикладных программ решения систем уравнений, компилятор;

- г) Word, Excel, Access.
2. Операционная система — это:
 - а) система программ для автоматизации операций в офисе;
 - б) программа управления данными, заданиями;
 - в) система программ управления данными, заданиями, ресурсами ЭВМ;
 - г) система распределения памяти на ЭВМ.
 3. В списке (драйвер, загрузчик, монитор ОС, СУБД, транслятор, программа интегрирования, редактор текстов, графический редактор) элементов прикладного ПО указано:
 - а) 7; б) 5; в) 3; г) 1.
 4. В списке {драйвер, монитор ОС, ППП решения математических задач, СУБД, компилятор, программа интегрирования, редактор текстов, электронная таблица} элементов системного ПО указано:
 - а) 6; б) 5; в) 4; г) 3.
 5. ППП — это:
 - а) Персонал Пользователей Программ;
 - б) Пакет Пересылаемых Программ;
 - в) Пакет Прикладных Программ;
 - г) Пакет Поддержки Приложений.
 6. Основные функции ОС — это:
 - а) управление данными к программам;
 - б) управление самими программами;
 - в) управление ресурсами ЭВМ;
 - г) все перечисленное в а, б, в.
 7. К инструментальному ПО относится набор:
 - а) ОС, ППП, транслятор;
 - б) ОС, транслятор, драйвер;
 - в) транслятор, АРМ, редактор текстов;
 - г) редактор текстов, электронная таблица, САПР.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
б	в	г	г	в	г	г

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. История развития ПО.
2. История развития ТО.
3. Современные ОС (анализ).
4. Современные инструментальные системы (анализ).
5. Современные интегрированные ППП (анализ).
6. Файловая система.
7. Эволюция ОС Windows.

ФОРМАЛИЗАЦИЯ И АКСИОМАТИЗАЦИЯ, МОДЕЛЬНОЕ ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

При поиске наиболее удобных, рациональных средств и форм информационного обмена человек сталкивается с проблемой компактного, однозначного и достаточно полного представления информации.

Знания — система понятий и отношений для такого обмена.

Представление знаний — это процесс, конечная цель которого — представление информации (семантического смысла, значения) в виде информативных сообщений (синтаксических форм): фраз устной речи, предложений письменной речи, страниц книги, понятий справочника, объектов географической карты, мазков и персонажей картины и т. п.

Для этого необходимо пользоваться некоторой конструктивной системой правил для их представления и восприятия (прагматического смысла). Назовем такую систему правил **формализмом** представления знаний. Неформализуемые знания — это знания, получаемые с применением неизвестных или неформализуемых правил, например эвристики, интуиции, здравого смысла.

Человек пользуется естественным формализмом — языком. Язык развивается благодаря тому, что человеческие знания постоянно нуждаются в языковом представлении, выражении, сжатии, хранении, обмене. Мысль, которую нельзя выразить в языковой конструкции, не может быть включена в информационный обмен. Язык — форма представления знаний. Чем богаче язык народа, чем больше знаний он может отражать, тем богаче культура народа. В то же время предложения и слова языка должны иметь однозначный семантический смысл. Особую роль играет язык математик, наук (не только точных, но и гуманитарных), формализованных знаний. Свой язык имеют химия, физика, экономика, информатика и т. д. Языки наук часто пересекаются и взаимообогащаются при исследовании междисциплинарных проблем.

Использование языковых систем повышает надежность информационного обмена, снижая возможность неправильного истолкования передаваемой информации, шумы в сообщениях.

ПРИМЕР

Найдем и формализуем закономерность в последовательности 1, 10, 11, 100, 111, 1000, 1111, 10000, ... Из сравнения членов $A[i]$ ($i = 1, 2, \dots$) последовательности, стоящих на четных и нечетных местах, видно, что: 1) элемент на нечетном месте получается из элемента на предыдущем нечетном месте добав-

лением справа к нему единицы; 2) каждый элемент на четном месте получается из элемента на предыдущем четном месте добавлением справа к нему нуля. Это словесно описанное (неформализованное) правило можно записать на математическом языке. Получим отдельно для случаев 1 или 2: $A[2n] = 10 \cdot A[2n - 2]$, $A[2n - 1] = 10 \cdot A[2n - 1] + 1$, $n = 1, 2, \dots$. Можно записать формулу, объединяющую обе эти формулы: $A[2n + m] = 10 \cdot A[2n + n - 2] + m$, где $m = 0$ или $m = 1$. Такая формула записи лучшая. Существует лучшая форма записи: $A[2n + \text{mod}(n, 2)] = 10A[2n + \text{mod}(n, 2) - 2] + \text{mod}(n, 2)$.

ПРИМЕР

Формализуем закон формирования последовательности AB , AAB , ABV , $AAAB$, $ABVV$, $\dots$. Словесное описание правила: к слову, стоящему на очередном нечетном месте, добавляется справа символ B , а к слову, стоящему на очередном четном месте, добавляется слева символ A . «Формульная» запись правила: $X_{2n+1} = X_{2n-1} + "B"$, $X_{2n} = "A" + X_{2n-2}$, $n = 1, 2, 3, \dots$. Здесь операция «+» означает конкатенацию (присоединение текста к тексту справа), а X_n — элемент последовательности на n -м месте. Бывают формализмы и более сложного и универсального вида.

ПРИМЕР

Если формулы A и B тождественны ($A \equiv B$), а C — произвольная формула, то верны формализмы $A \equiv B$, $A \wedge C \equiv B \wedge C$, $C \wedge A \equiv C \wedge B$, $A \vee C \equiv B \vee C$, $C \Leftrightarrow A \equiv C \Leftrightarrow B$.

Можно условно классифицировать знания в конкретной предметной области на понятийные, конструктивные, процедурные, фактографические и метазнания.

Понятийные знания — набор понятий, используемых при решении данной задачи, например в фундаментальных науках и теоретических областях наук, т. е. это понятийный аппарат науки.

Конструктивные знания — наборы структур, подсистем системы и взаимодействий между их элементами, например в технике.

Процедурные знания — методы, процедуры (алгоритмы) их реализации и идентификации, например в прикладных науках.

Фактографические знания — количественные и качественные характеристики объектов и явлений, например в экспериментальных науках, медицине.

Метазнания — знания о порядке и правилах применения знаний (знания о знаниях).

Один из наиболее древних и «мощных» методов формализации знаний — аксиоматический. Аксиоматический метод построения научной теории зародился в Древней Греции. Его суть заключается в следующем: выделяются основные понятия, затем формулируются не доказываемые в рамках данной теории утверждения (аксиомы) теории, а все остальные утверждения и факты теории

выводятся логическим путем, с опорой на основные понятия, аксиомы и на ранее выведенные из них утверждения. Одни понятия опираются на другие, которые, в свою очередь, также определяются с помощью каких-то известных понятий, и т. д. Таким образом, мы приходим к элементарным понятиям (термам), которые нельзя определить через другие. Эти понятия и называются основными. Когда мы доказываем утверждение, то опираемся на другие утверждения, уже доказанные ранее, или приводим их к утверждениям, опирающимся логически только на аксиомы и на ранее доказанные утверждения, которые и принимаем без доказательства (пишем «... что и требовалось доказать»). Так устроены не только наши знания, но и природа: все состоит и образуется из веществ, они состоят из атомов и т. д.

«Математические истины являются производными небольшого числа аксиом, из которых они развиваются посредством ряда безупречных строгих умозаключений; они лежат не только в природе нашего познания, но и в существе самой природы» (А. Пуанкаре).

Набор аксиом, хотя и не является чем-то завершенным раз и навсегда, исключительно совершенным, тем не менее должен удовлетворять определенным свойствам аксиом, основными из которых являются:

1) **непротиворечивость** — невозможность вывода из этой системы аксиом двух каких-то противоречивых утверждений;

2) **независимость** — никакая аксиома из системы аксиом не может быть выведена из других (т. е. быть их следствием);

3) **полнота** — система аксиом должна иметь интерпретации, неразличимые по своим свойствам, реализациям (изоморфные, как говорят в математике).

Аксиоматизация широко используется в геометрии.

ПРИМЕР

В геометрии Евклида используется аксиома параллельности Евклида: через точку, не лежащую на данной прямой в плоскости, определяемой точкой и прямой, можно провести не более одной прямой, параллельной исходной прямой. В геометрии Лобачевского вместо этой аксиомы принимается следующая аксиома: через точку, не лежащую на данной прямой, можно провести по крайней мере две прямые, лежащие с данной прямой в одной плоскости и не пересекающие ее. Обе геометрии непротиворечивы, независимы, полны. Аксиомы и основные понятия составляют основы планиметрии. Основные понятия геометрии определяются как объекты любой природы, удовлетворяющие аксиомам этой геометрии, т. е. при аксиоматическом построении геометрии исходят из некоторой системы аксиом, или некоторой аксиоматики. В этих аксиомах описываются свойства основных понятий геометрической системы, и мы можем представить основные понятия в виде объектов любой природы, которые обладают свойствами, ука-

занными в аксиомах. После формулировки и доказательства первых геометрических утверждений становится возможным доказывать («разворачивать») новые утверждения с помощью других.

Аксиоматически строятся не только геометрические теории, но и многие математические и другие теории.

ПРИМЕР

Аксиоматически строятся теории множеств, теория высказываний и предикатов и другие теории.

«Все доказывающие науки применяют аксиомы. Аксиомы обладают наивысшей степенью общности и представляют начала всего. И если не дело философа, то чье же еще — рассмотреть, что по отношению к ним — правда и что — ложь» (Аристотель).

Основная черта современной математики и информатики — широкое использование аксиоматического метода.

Одним из важных форм (методов) формализации знаний является его представление классом (классификация).

Классификация — выделение некоторого критерия и группировка объектов или процессов таким образом, что в одну группу попадают лишь те объекты (процессы), которые удовлетворяют этому критерию. Это метод научной систематики, особенно важный на начальном этапе формирования базовых знаний научного направления. Классификация, установление эквивалентности объектов, систем позволяет решать такие важные задачи информатики, как фиксация знаний, поиск по образцу, сравнение и др.

ПРИМЕР

Такими системами являются классификационная система К. Линнея в ботанике, систематика живых организмов, таблица химических элементов Д. И. Менделеева, систематика экономических систем, Табель о рангах, введенная Петром I в 1722 г., которая подразделяла чины на 14 рангов. Каждому чину соответствовала определенная должность. Первые 6 рангов статской и придворной службы и первый обер-офицерский чин в армии давали право на получение потомственного дворянства, что способствовало формированию дворянской бюрократии. Таким образом, Табель о рангах выполняла социально стимулирующее упорядочивание.

Для компьютерного представления и обработки знаний и данных о предметной области (об объектах, процессах, явлениях, их структуре и взаимосвязях) они должны быть формализованы и представлены в определенном формализованном виде, с помощью определенной модели представления знаний.

Основные модели представления знаний — продукционные, фреймовые, сетевые и логические. Используя общие модели знаний и специальные правила вывода знаний (продукции), можно вывес-

ти решение для заданной конкретной ситуации, описываемой некоторыми исходными данными. Подобного рода логические выводы называются дедуктивными, т. е. от общего к частному. Цепочка логических выводов строится по мере приближения к решению, в зависимости от выведенных к этому шагу новых знаний.

Продукционная модель представления знаний наиболее распространена в приложениях. Модель реализуется правилами-продукциями типа:

если < условие > то < заключение > все.

В качестве условия может выступать любая совокупность суждений, соединенных логическими связками и, ^ (или, ∨).

ПРИМЕР

Продукцией будет следующее правило:

если (курс доллара растет) или (сезон — осень) и (число продавцов меньше)

то (рост рублевых цен на квартиры).

все.

Такого рода правила и знания о ценах, предложении и спросе на рынке жилья могут стать основой для базы знаний о рынке жилья и экспертной системы для риелторской фирмы.

Продукционные модели удобны для представления логических взаимосвязей между фактами, так как они более формализованные и достаточно строгие (теоретические), модульные (продукции явно между собой не связаны), соответствуют хранению знаний в долговременной памяти человека.

Представление знаний в виде семантической сети является одной из основных моделей представления знаний.

Семантическая сеть — структура из вершин и соединяющих их ребер. Вершина отображает некоторое понятие (объект, процесс, ситуацию), а ребро соответствует связи, отношению типа «...это есть...», «...принадлежит...», «...есть причина...», «...входит в...», «...состоит из...» и т. д.

ПРИМЕР

Рассмотрим факт: причиной неритмичной работы предприятия является старое оборудование, а причиной последнего — отсутствие оборотных средств. Семантическая сеть может содержать вершины «оборотные средства», «старое оборудование», соединяемые ребрами — отношениями типа «быть причиной».

Достоинство семантических сетей — наглядность представления знаний, с их помощью удобно представлять причинно-следственные связи между элементами (подсистемами), а также структуру сложных систем. Недостаток таких сетей — сложность вывода, поиска по запросу.

Фреймовая модель представления знаний задает остоу описания класса объектов и удобна для описания структуры и характе-

ристических одготипных объектов (процессов, событий, ситуаций), описываемых **фреймами** — специальными шаблонами понятий фреймовой сети (знания). Фрейм — модель кванта знаний (абстракции), активизация фрейма аналогична активизации этого кванта знаний — для объяснения, предсказания и т. п. Отдельные характеристики (элементы описания) объекта называются **слотами** фрейма.

Различают фреймы-образцы (прототипы), хранящиеся в базе знаний, и фреймы-экземпляры, создаваемые для отображения реальных ситуаций, для конкретных данных.

Фреймовое представление позволяет отобразить знания с помощью:

- фрейм-структур — для обозначения объектов и понятий;
- фрейм-ролей — для обозначения ролевых обязанностей;
- фрейм-сценариев — для обозначения поведения;
- фрейм-ситуаций — для обозначения режимов деятельности, состояний.

ПРИМЕР

Фрейм-структуры — это понятия «заем», «вексель», «кредит». Фрейм-роли — «кассир», «клиент», «сервер». Фрейм-сценарии — «страхование», «банкинг», «банкротство». Фрейм-ситуации — «эволюция», «функционирование», «безработица».

ПРИМЕР

Возьмем такое понятие, как «функция». Различные функции могут отличаться друг от друга, но можно указать некоторый набор формальных характеристик для описания любой функции (фрейм «Функция»): тип и допустимое множество изменений аргумента (область определения функции), тип и допустимое множество значений функции (множество значений функции), аналитический закон связи аргумента со значением функции. Соответственно могут быть определены фреймы «Аргумент», «Значение функции», «Закон соответствия». Далее можно определить фреймы «Тип аргумента», «Вычисление значения функции», «Операция» и др. Пример слотов для фрейма «Закон соответствия»: аналитический способ задания закона, сложность вычисления (реализации). Чтобы описать конкретное значение фрейма, необходимо каждому слоту придать конкретное значение, например:

Имя фрейма — Функция;

Аргумент — x ;

Значение функции — y ;

Закон соответствия — квадратичный.

Слоты:

Значение аргумента — R ;

Способ задания функции — $y = ax^2 + bx + c$;

Сложность вычисления — 7 (операций).

Логическая (предикатная) модель представления знаний основана на алгебре высказываний и предикатов, на системе аксиом и ее правилах вывода. Из предикатных моделей наибольшее распространение получила модель предикатов, базирующаяся на термах (аргументах предикатов — логических констант, переменных, функций), предикатах (выражениях с логическими операциями). Предметная область описывается предикатами и аксиомами.

ПРИМЕР

Возьмем утверждение: «Инфляция в стране превышает прошлогодний уровень в 2 раза». Запишем в виде логической модели: $r(\text{InfNew}, \text{InfOld}, n)$, где $r(x, y)$ — отношение вида $x = ny$; InfNew — текущая инфляция в стране; InfOld — инфляция в прошлом году. Можно рассматривать предикаты, например: $r(\text{InfNew}, \text{InfOld}, 2) = 1$, $r(\text{InfNew}, \text{InfOld}, 3) = 0$ и т. д.

Логические модели удобны для представления логических взаимосвязей между фактами, формализованы, строги, для их использования имеется удобный и адекватный инструментарий, например язык логического программирования Пролог.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое знания? Что такое представление знаний?
2. Что такое формализм? Приведите примеры.
3. Что такое классификация? Приведите пример классификации.
4. Что такое продукционная модель знаний?
5. Что такое семантическая сеть для представления знаний?
6. Что такое фреймовая модель знаний?
7. Что такое логическая модель знаний?

ЗАДАЧИ

1. Постройте одну классификационную систему.
2. Формализуйте понятие «решить задачу».
3. Постройте одну продукционную модель знаний.
4. Постройте одну семантическую модель знаний.
5. Постройте одну фреймовую модель знаний.
6. Постройте одну логическую модель знаний.
7. Формализуйте правило образования чисел 2, 8, 48, 384, ...

ТЕСТ

1. Основными моделями знаний являются:
 - а) продукционные, семантические, фреймовые, логические;
 - б) конструктивные, процедурные, графические, метазнания;
 - в) понятийные, непонятийные модели;
 - г) формализмы, классы, группы.
2. Формализм — это система правил для:
 - а) восприятия и представления знаний;

- б) необъективного, формального подхода;
 в) восприятия формы системы;
 г) всего, приведенного в а, б, в.
3. Классификация — группировка по критерию так, чтобы все:
 а) попали в одну группу;
 б) удовлетворяющие критерию попали в одну группу;
 в) не удовлетворяющие критерию попали в различные группы;
 г) группы были непустыми.
4. Продукционной моделью не является модель вида:
 а) если < условие > то < заключение >;
 б) если < условие > то < заключение > иначе < заключение >;
 в) если < условие > то < условие >;
 г) если < не условие > то < заключение >.
5. Модель знаний вида если < предикат > то < заключение > все является:
 а) семантической; в) логической;
 б) фреймовой; г) продукционной.
6. Семантической сети соответствует:
 а) множество вершин-событий;
 б) множество вершин-событий, отношений-ребер между ними;
 в) множество отношений-ребер;
 г) множество вершин-отношений и множество ребер-событий.
7. Если модель знаний описана лишь только вершинами (понятиями) и ребрами (отношениями), то это модель:
 а) семантическая; в) логическая;
 б) фреймовая; г) продукционная.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
а	а	б	в	г	б	а

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Методы формализации знаний.
2. Продукционные модели и их использование.
3. Семантические модели и их использование.
4. Фреймовые модели и их использование.
5. Логические модели и их использование.
6. Классификация, группировка и таксономия.
7. Принятие решений как наука и искусство.

18

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

Модель — объект, описание объекта, системы для замещения (при определенных гипотезах) одной системы (оригинала) другой системой для изучения оригинала или воспроизведения его свойств. Модель — результат отображения одной структуры на другую. Отображая физическую систему (объект) на математическую систему (например, математический аппарат уравнений), получим физико-математическую модель системы или математическую модель физической системы. Любая модель строится и исследуется при некоторых гипотезах.

ПРИМЕР

Рассмотрим физическую систему: тело массой m , которое скатывается по наклонной плоскости с ускорением a и на которое действует сила F . Закон Ньютона: $F = ma$. Это физико-математическая модель системы или математическая модель физической системы. При описании этой системы (построении этой модели) приняты следующие гипотезы: 1) поверхность идеальна (т. е. коэффициент трения равен нулю); 2) тело находится в вакууме (т. е. сопротивление воздуха равно нулю); 3) масса тела неизменна; 4) тело движется с одинаковым постоянным ускорением в любой точке.

ПРИМЕР

Система кровообращения человека подчиняется некоторым законам термодинамики. Описывая эту систему на физическом (термодинамическом) языке балансовых законов, получим физическую, термодинамическую модель физиологической системы. Если записать эти законы математически с помощью уравнений, получим математическую модель системы кровообращения. Эту модель можно назвать физиолого-физико-математической моделью или физико-математической моделью.

Информация — это абстракция. Модель — объект, система, которая позволяет этой информации обрести конкретное, например компьютерное, представление, содержание. Моделирование — процесс, метод, который позволяет осуществлять перенос информации от реальной системы к модели и наоборот.

По уровню, «глубине» моделирования модели бывают:

- эмпирические — на основе эмпирических фактов, зависимостей;
- теоретические — на основе математических, аналитических описаний;

— смешанные, полуэмпирические — использующие эмпирические зависимости и математические описания.

Проблема моделирования — триединая задача, которая включает подзадачи:

- построение модели (задача менее формализуема и конструктивна в том смысле, что нет алгоритма для построения моделей);
- исследование модели (задача более формализуема, имеются методы исследования различных классов моделей);
- использование модели (конструктивная и конкретизируемая задача).

Моделирование — универсальный метод получения, описания и использования знаний. Оно применяется в любой профессиональной деятельности. В современной науке и технологии роль и значение математического моделирования усиливаются, актуализируются. Математическое моделирование реальных и нелинейных систем живой и неживой природы позволяет перекидывать мостики между нашими знаниями и реальными системами, процессами.

Модель — статическая, если среди параметров, участвующих в описании модели, нет временного параметра. Она дает в каждый момент времени лишь «фотографию» системы, ее срез.

ПРИМЕР

$F = at$ — статическая модель движения материальной точки массой m с ускорением a . Не учитывает изменение ускорения от одной точки к другой.

Модель — динамическая, если среди параметров модели есть временной параметр, т. е. она отображает систему (процессы в системе) во времени.

ПРИМЕР

$S = gt^2/2$ — динамическая модель пути при свободном падении тела. Динамическая модель типа закона Ньютона: $F(t) = a(t)m(t)$. Еще лучше: $F(t) = s''(t)m(t)$.

Модель — дискретная, если она описывает поведение системы лишь в дискретные моменты времени.

ПРИМЕР

При $t = 0, 1, 2, \dots, 10$ (с) модель $S_i = gt^2/2$ или числовая последовательность $S_0 = 0, S_1 = g/2, S_2 = 2g, S_3 = 9g/2, \dots, S_{10} = 50g$ может служить дискретной моделью движения свободно падающего тела.

Модель — непрерывная, если описывает поведение системы для всех моментов времени из некоторого заданного промежутка времени.

ПРИМЕР

Модель $S = gt^2/2, 0 < t < 100$ непрерывна на промежутке времени $(0; 100)$.

Модель — имитационная, если предназначена для испытания, изучения, проигрывания возможных путей развития и поведения объекта путем варьирования некоторых (всех) параметров модели.

ПРИМЕР

Если модель экономической системы производства товаров двух видов 1 и 2 соответственно в количестве x_1 и x_2 единиц и стоимостью каждой единицы товара a_1 и a_2 на предприятии описана в виде соотношения $a_1x_1 + a_2x_2 = S$, где S — общая стоимость произведенной предприятием всей продукции (видов 1 и 2), то ее можно использовать в качестве имитационной модели, по которой можно планировать выпуск продукции с наибольшей общей стоимостью S .

Модель — детерминированная, если каждому входному набору данных соответствует однозначно определенный набор выходных параметров, иначе модель недетерминированная, стохастическая (вероятностная).

ПРИМЕР

Все перечисленные выше физические модели — детерминированные. Если в модели $S = gt^2/2$, $0 < t < 100$ мы учли бы случайный параметр — порыв ветра с силой p при падении тела, например $S(p) = g(p)t^2/2$, $0 < t < 100$, то получили бы стохастическую модель (уже не свободного!) надения.

Модель — теоретико-множественная, если представима с помощью некоторых множеств и отношений принадлежности множеств и элементов для них.

ПРИМЕР

Даны множество $X = \{\text{Николай, Петр, Николаев, Петров, Елена, Екатерина, Михаил, Татьяна}\}$ и отношения: Николай — супруг Елены, Екатерина — супруга Петра, Татьяна — дочь Николая и Елены, Михаил — сын Петра и Екатерины, семьи Михаила и Петра дружат друг с другом. Множество X и множество перечисленных отношений Y — теоретико-множественная «модель дружбы».

Модель — логическая, если представима предикатами, логическими функциями.

ПРИМЕР

Две логические функции $z = x \wedge \bar{y} \vee \bar{x} \wedge y$, $p = x \wedge y$ могут служить математической (логической) моделью одноразрядного сумматора компьютера.

Модель — игровая, если она описывает, реализует некоторую игровую ситуацию между участниками игры (лицами, группами, объектами, процессами).

ПРИМЕР

Игрок 1 — добросовестный налоговый инспектор, а игрок 2 — недобросовестный налогоплательщик. Идет процесс (игра) по уклонению от налогов, с одной стороны, и по выявлению сокрытия налогов, с другой стороны. Игроки выбирают натуральные числа i и j ($i, j \leq n$). Это или штраф игрока 2 за неуплату налогов при обнаружении факта неуплаты игроком 1, или оценка временной выгоды игрока 2 от сокрытия налогов. Рассмотрим игру с таблицей выигрышей A из элементов $a_{ij} = |i - j|$. Модель игры описывается этой матрицей и стратегией уклонения и поимки. Эта игра антагонистическая, бескоалиционная (эти формализуемые в математической теории игр понятия мы пока будем понимать содержательно, интуитивно).

Модель — алгоритмическая, если описана некоторым алгоритмом (комплексом алгоритмов), определяющим ее функционирование, развитие. Заметим, что не все модели могут быть исследованы или реализованы алгоритмически.

ПРИМЕР

Модель вычисления суммы бесконечного убывающего ряда чисел — алгоритм вычисления конечной суммы ряда до некоторой заданной степени точности. Алгоритмической моделью корня квадратного из числа x может служить алгоритм вычисления его приближенного, сколь угодно точного значения по известной рекуррентной формуле (см. выше).

Модель — языковая, лингвистическая, если представлена некоторым лингвистическим объектом, формализованной языковой структурой. Иногда такие модели называют вербальными, синтаксическими и т. п.

ПРИМЕР

Правила дорожного движения — языковая, структурная, вербальная модель движения транспорта и пешеходов. Пусть B — множество производящих основ существительных, C — множество суффиксов, P — множество прилагательных, «+» — операция конкатенации слов, «:=» — операция присваивания, « $\Rightarrow$ » — операция вывода (выводимости новых слов), Z — множество значений (смысловых) прилагательных. Языковая модель M словообразования: $\langle z_i \rangle \Leftarrow \langle p_i \rangle := \langle b_i \rangle + \langle s_i \rangle$. При b_i — «рыб(а)», s_i — «н(ый)», получаем по этой модели p_i — «рыбный», z_i — «приготовленный из рыбы».

Модель — визуальная, если она визуализирует отношения и связи моделируемой системы, особенно в динамике, анимацией.

ПРИМЕР

Виртуальная клавиатура на экране монитора часто используется в почтовых системах, как и приложение «Калькулятор».

Модель — **натурная**, если она материальная копия объекта моделирования.

ПРИМЕР

Глобус — натурная географическая модель земного шара.

Модель — **геометрическая**, графическая, если представима геометрическими образами и объектами.

ПРИМЕР

Макет дома — натурная геометрическая модель строящегося дома. Вписанный в окружность многоугольник дает модель окружности. Именно она используется при изображении окружности на экране компьютера. Прямая линия является моделью числовой оси, а параллелограмм часто изображает плоскость.

Модель — **клеточно-автоматная**, если она представляет систему с помощью клеточного автомата или системы клеточных автоматов. Клеточный автомат — это дискретная динамическая система, аналог физического (непрерывного) поля. Клеточно-автоматная геометрия — аналог евклидовой геометрии. Неделимый элемент евклидовой геометрии — точка, на основе ее строятся отрезки, прямые, плоскости и т. д. Неделимый элемент клеточно-автоматного поля — клетка, на основе ее строятся кластеры клеток и различные конфигурации клеточных структур. Это «мир» некоторого автомата, исполнителя. Представляется клеточный автомат равномерной сетью клеток («ячеек») этого поля. Эволюция клеточного автомата разворачивается в дискретном пространстве — клеточном поле. Такие клеточные поля могут быть вещественно-энергоинформационными. Законы эволюции локальны, т. е. динамика системы определяется задаваемым неизменным набором законов или правил, по которым осуществляется вычисление новой клетки эволюции и ее материально-энергоинформационной характеристики в зависимости от состояния ее соседей (правила «соседства» заданы). Смена состояний в клеточно-автоматном поле происходит одновременно и параллельно, а время идет дискретно. Несмотря на кажущуюся простоту их построения, клеточные автоматы могут демонстрировать разнообразное и сложное поведение. В последнее время они широко используются при моделировании не только физических, но и социально-экономических процессов.

Клеточные автоматы могут быть одномерными (1D-ячейки — отрезки прямых), двумерными (2D-ячейки — прямоугольники на плоскости), трехмерными (3D-ячейки — параллелепипеды, кубы в пространстве).

ПРИМЕР

Известна игра «Жизнь» Джона Конвея. Она описана во многих книгах. Мы рассмотрим аналогичную клеточно-автоматную модель загрязнения среды, диффузии загрязнителя в некоторой среде. 2D-клеточный автомат для моделирования загрязнения среды генерируем следующими правилами:

- плоскость разбивается на одинаковые клетки, каждая клетка может находиться в одном из двух состояний: состояние 1 — если в ней есть диффундирующая частица загрязнителя, и состояние 0 — если ее нет;
- клеточное поле разбивается на блоки 2×2 двумя способами, которые будем называть четным и нечетным разбиением (у четного разбиения в кластере или блоке находится четное число точек или клеток поля, у нечетного разбиения — нечетное число);
- на очередном шаге каждый блок четного разбиения поворачивается (по заданному правилу распространения загрязнения) на заданный угол (направление поворота выбираем случайно или по правилу);
- аналогичное правило определяется и для блоков нечетного разбиения;
- процесс продолжается до некоторого момента или до очищения среды.

На рисунке 24 показаны фрагменты работы программы клеточно-автоматной модели (размеры клеток увеличены).

Модель — **фрактальная**, если описывает эволюцию моделируемой системы эволюцией фрактальных объектов. Определим понятие фрактала. Если физический объект однородный (сплошной), т. е. в нем нет полостей, можно считать, что плотность не зависит от размера. Например, при увеличении R до $2R$ масса увеличится в R^2 раз (круг) и в R^3 раз (шар), т. е. $M(R) \sim R^n$ (связь массы и длины), n — размерность пространства. Объект, у которого масса и размер связаны этим соотношением, называется компактным. Его плотность

$$\rho \sim \frac{M}{R^n} \sim R^0 = \text{const.}$$

Если объект (система) удовлетворяет соотношению $M(R) \sim R^{f(n)}$, где $f(n) < n$, то такой объект называется **фрактальным**. Его плотность не будет одинаковой для всех значений R , а масштабируется так:

$$\rho(R) \sim \frac{M(R)}{R^n} \sim R^{f(n)-n}.$$

Так как $f(n) - n < 0$, то плотность фрактального объекта уменьшается с увеличением размера, а $\rho(R)$ является количественной мерой разреженности, ветвистости (структурированности) объекта.

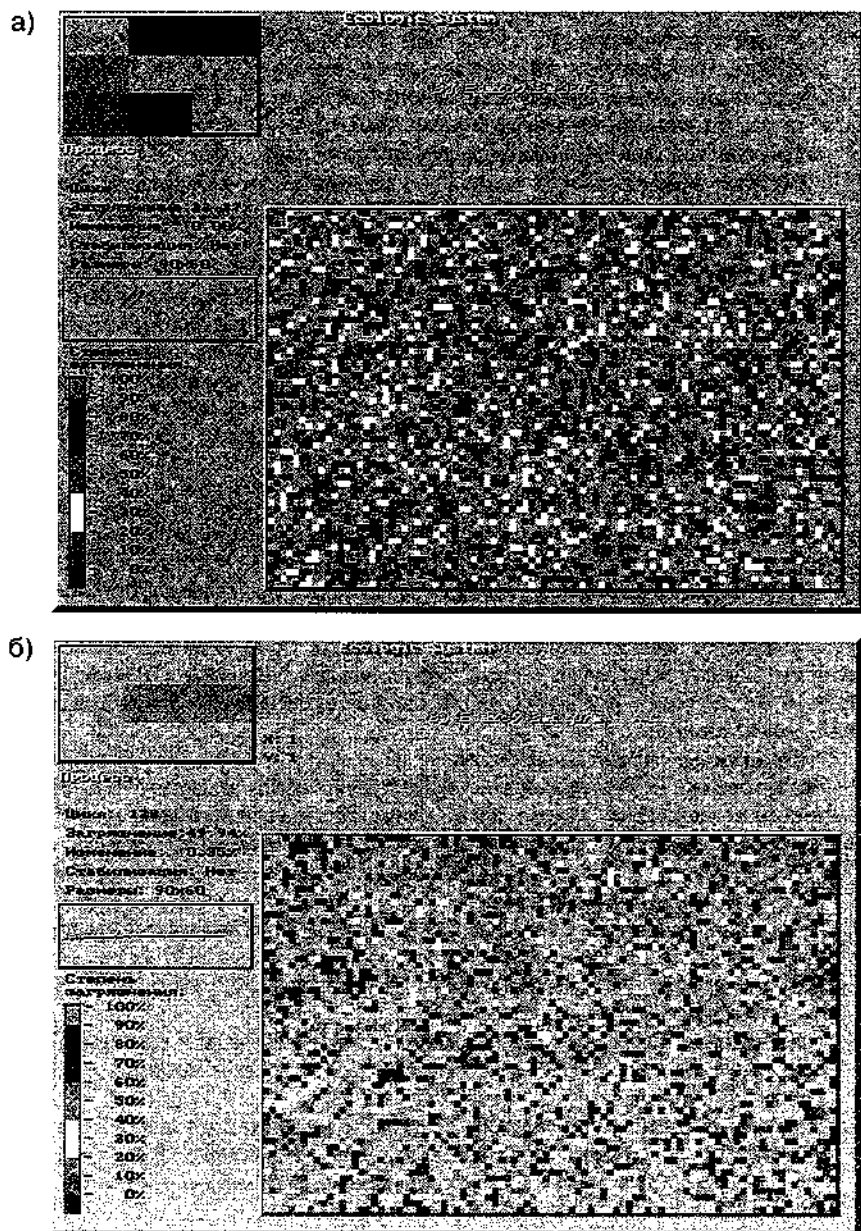


Рис. 24. Окно справа — состояние клеточного поля: а) исходное, слабо-загрязненное состояние; б) состояние после 120 циклов загрязнения. В левом верхнем углу — микроскоп, увеличивающий кластер поля, в середине слева — график динамики загрязнения, внизу слева — индикаторы загрязнения

ПРИМЕР

Разделим отрезок $[0; 1]$ на три части и выбросим средний отрезок. Оставшиеся два промежутка опять разделим на три части и выкинем средние промежутки и т. д. Получим множество, называемое множеством Кантора. В пределе получаем множество изолированных точек (рис. 25).

Можно показать, что если n — размерность множества Кантора, то $n = \ln 2 / \ln 3 \approx 0.63$, т. е. этот объект (фрактал) еще не состоит только из изолированных точек, хотя уже и не состоит из отрезка. Фрактальные объекты самоподобны, если они выглядят одинаково в любом пространственном масштабе или фрагменты структуры повторяются через определенные пространственные промежутки. Поэтому они очень подходят для моделирования нерегулярностей, так как позволяют описывать (например, дискретными моделями) эволюцию таких систем для любого момента времени и в любом пространственном масштабе.

Фрактальная модель применяется обычно тогда, когда реальный объект нельзя или трудно представить классической моделью, когда имеем дело с нелинейностью (многовариантностью путей развития) и недетерминированностью, хаотичностью и необратимостью эволюционных процессов.

Границы между моделями различного типа или же отнесение модели к тому или иному типу часто весьма условны. Лучше говорить о различных режимах использования моделей — имитационном, стохастическом и т. д.

Основные свойства модели:

- **конечность** — отображение оригинала лишь в конечном числе его отношений и, кроме того, ресурсы моделирования конечны;
- **упрощенность** — отображение лишь существенных сторон объекта и, кроме того, модель должна быть проста для исследования или воспроизведения;
- **приблизительность** — действительность отображается моделью грубо или приблизительно;
- **адекватность** — модель должна успешно описывать моделируемую систему;
- **наглядность, обозримость** основных ее свойств и отношений;
- **доступность и технологичность** — для исследования или воспроизведения;
- **информативность** — модель должна содержать достаточную информацию о системе (в рамках гипотез, принятых при построении модели) и должна давать возможность получить новую информацию;



Рис. 25. Множество Кантора для трех делений

- **сохранение информации**, содержащейся в оригинале (с точностью рассматриваемых при построении модели гипотез);
- **полнота** — в модели должны быть учтены все основные связи и отношения, необходимые для обеспечения цели моделирования;
- **устойчивость** — модель должна описывать и обеспечивать устойчивое поведение системы, даже если она вначале является неустойчивой;
- **замкнутость** — модель учитывает и отображает замкнутую систему необходимых основных гипотез, связей и отношений.

Жизненный цикл моделируемой системы:

- сбор информации об объекте, выдвижение гипотез, предмодельный анализ;
- проектирование структуры и состава подмоделей модели;
- разработка и отладка отдельных подмоделей, сборка модели в целом, идентификация (если это нужно) параметров моделей;
- исследование модели, выбор метода исследования и разработка алгоритма (программы) моделирования;
- исследование адекватности, устойчивости, чувствительности модели;
- оценка средств моделирования (затраченных ресурсов);
- интерпретация, анализ результатов моделирования и установление некоторых причинно-следственных связей в исследуемой системе;
- генерация отчетов и проектных (народно-хозяйственных) решений;
- уточнение, модификация модели, если это необходимо, и возврат к исследуемой системе с новыми знаниями, полученными при моделировании.

Основные функции компьютера при моделировании систем:

- вспомогательное средство для решения задач, решаемых обычными вычислительными средствами, алгоритмами, технологиями;
- средство постановки и решения новых задач, не решаемых традиционными средствами, алгоритмами, технологиями;
- средство конструирования обучающих и моделирующих сред типа «обучаемый — компьютер — обучающий», «обучающий — компьютер — обучаемый», «обучающий — компьютер — группа обучаемых», «группа обучаемых — компьютер — обучающий», «компьютер — обучаемый — компьютер»;
- средство моделирования для получения новых знаний;
- «обучение» новых моделей (самообучение модели).

Компьютерное моделирование — основа представления знаний в ЭВМ. Оно использует любую информацию, которую можно актуализировать с помощью ЭВМ.

Разновидность компьютерного моделирования — **вычислительный эксперимент**, осуществляемый экспериментатором над исследуемой системой или процессом с помощью компьютера, компью-

терной технологии. Вычислительный эксперимент позволяет находить новые закономерности, проверять гипотезы, визуализировать события и т. д.

ПРИМЕР

Модель повторяющейся эпидемии. Пусть существует группа из n контактирующих индивидуумов, в которой в момент времени t имеется x восприимчивых индивидуумов, y — источников инфекции и z — изолированных (выздоровевших и ставших невосприимчивыми к инфекции) индивидуумов. Таким образом, $x + y + z = n$. Определим исходные параметры системы: β — частота контактов между членами группы; γ — частота случаев удаления; μ — скорость пополнения восприимчивых индивидуумов извне (это скорость гибели индивидуумов, удаленных из популяции); y^* — критическое значение, при котором начинается эпидемия. После одной вспышки эпидемии, в результате которой плотность восприимчивых индивидуумов упадет ниже критического значения y^* , наступает период относительного затишья, длящийся до тех пор, пока снова не будет достигнуто значение y^* и не возникнет новая вспышка. За время Δt группа восприимчивых индивидуумов, с одной стороны, уменьшается на $\beta xy \Delta t$ за счет заражения части из них, а с другой — увеличивается на $\mu \Delta t$.

Математическая запись данной ситуации с помощью производных (динамическая непрерывная модель эпидемии):

$$\begin{cases} \frac{dx}{dt} = -\beta xy + \mu, \\ \frac{dy}{dt} = \beta xy - \gamma y, \\ \frac{dz}{dt} = \gamma y - \mu. \end{cases}$$

Здесь $\begin{cases} i=1, & y(t) < y^*, \\ i=2, & y(t) \geq y^*, \end{cases}$ где y^* — некоторое критическое значение (критический уровень больных; его определение — важная медицинская и социально-экономическая задача, связанная с объявлением карантина, привлечением дополнительных средств, расходами по оплате больничных листов и т. д.). При $t=0$ заданы начальные значения $x(0) = x_0$, $y(0) = y_0$, $z(0) = z_0$. Рассматривается развитие эпидемии на промежутке времени $t(0 \leq t \leq T)$.

При малых $\Delta t = 1$ (с) получим рекуррентные соотношения (дискретную модель эпидемии):

$$\begin{cases} x(t+1) = x(t) - \beta x(t)y(t) + \mu, \\ y(t+1) = y(t) + \beta x(t)y(t) - \gamma y(t), \text{ где } t = 1, 2, \\ z(t+1) = z(t) + \gamma y(t) - \mu, \end{cases}$$

Динамика развития эпидемии

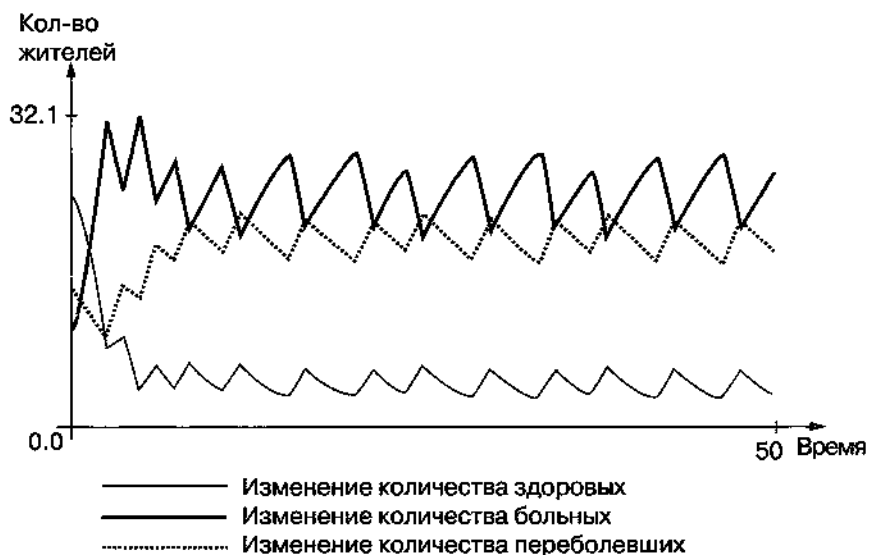


Рис. 26. Графики к эксперименту 1

Динамика развития эпидемии

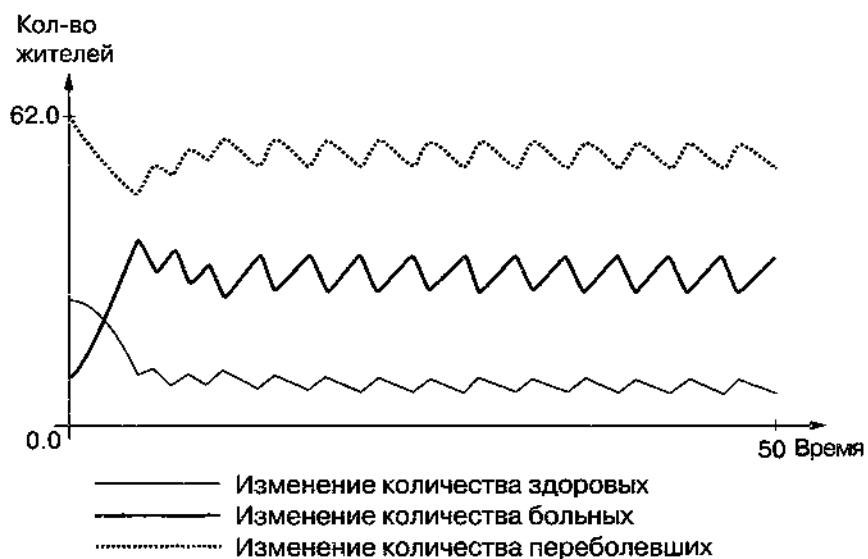


Рис. 27. Графики к эксперименту 2



Рис. 28. Графики к эксперименту 3

Прогноз по этой модели на компьютере осуществляется просто. Приведем результаты имитационных экспериментов.

Эксперимент 1. При $n = 50$, $x(0) = 25$, $y(0) = 10$, $z(0) = 15$, $\beta_1 = 0.04$, $\beta_2 = 0.01$, $\gamma_1 = 0.1$, $\gamma_2 = 0.3$, $\mu = 4$, $y^* = 27$, $T = 50$ получен результат: $x(50) = 3$, $y(50) = 28$, $z(50) = 18$. Графики течения (эволюции) эпидемии показаны на рисунке 26.

Эксперимент 2. При $n = 100$, $x(0) = 26$, $y(0) = 12$, $z(0) = 62$, $\beta_1 = 0.02$, $\beta_2 = 0.01$, $\gamma_1 = 0.1$, $\gamma_2 = 0.3$, $\mu = 6$, $y^* = 35$, $T = 50$ получен результат: $x(50) = 9$, $y(50) = 38$, $z(50) = 53$. Графики течения эпидемии показаны на рисунке 27.

Эксперимент 3. При $n = 100$, $x(0) = 50$, $y(0) = 15$, $z(0) = 35$, $\beta_1 = 0.02$, $\beta_2 = 0.01$, $\gamma_1 = 0.2$, $\gamma_2 = 0.3$, $\mu = 5$, $y^* = 45$, $T = 50$ получен результат: $x(50) = 10$, $y(50) = 24$, $z(50) = 66$ (рис. 28). Первые два случая демонстрируют периодичность эпидемии. В последнем случае имеет место стационарное состояние.

Компьютерное моделирование проходит следующие этапы:

1. Постановка задачи.
 - 1.1. Формулировка задачи.
 - 1.2. Определение цели моделирования.
 - 1.3. Сбор информации о системе, объекте моделирования.
 - 1.4. Описание данных (их структуры, диапазона, источника и т. д.).
2. Предмодельный анализ.
 - 2.1. Анализ существующих аналогов и подсистем.

- 2.2. Анализ технических средств моделирования (ЭВМ, периферия).
- 2.3. Анализ программного обеспечения (языки программирования, пакеты прикладных программ, инструментальные среды).
- 2.4. Анализ математического обеспечения (модели, методы, алгоритмы).
- 3. Анализ задачи (модели).
 - 3.1. Разработка структур данных.
 - 3.2. Разработка входных и выходных спецификаций, форм представления данных.
 - 3.3. Проектирование структуры и состава модели (подмоделей).
- 4. Исследование модели.
 - 4.1. Выбор методов исследования подмоделей.
 - 4.2. Выбор, адаптация или разработка алгоритмов, их псевдокодов.
 - 4.3. Сборка модели в целом из подмоделей.
 - 4.4. Идентификация модели, если в этом есть необходимость.
 - 4.5. Формулировка используемых критериев адекватности, устойчивости и чувствительности модели.
- 5. Программирование (проектирование программы).
 - 5.1. Выбор метода тестирования и тестов (контрольных примеров).
 - 5.2. Кодирование на языке программирования (написание команд).
 - 5.3. Комментирование программы.
- 6. Тестирование и отладка.
 - 6.1. Синтаксическая отладка.
 - 6.2. Семантическая отладка (отладка логической структуры).
 - 6.3. Тестовые расчеты, анализ результатов тестирования.
 - 6.4. Оптимизация программы.
- 7. Оценка моделирования.
 - 7.1. Оценка средств моделирования.
 - 7.2. Оценка адекватности моделирования.
 - 7.3. Оценка чувствительности модели.
 - 7.4. Оценка устойчивости модели.
- 8. Документирование.
 - 8.1. Описание задачи, целей.
 - 8.2. Описание модели, метода, алгоритма.
 - 8.3. Описание среды реализации.
 - 8.4. Описание возможностей и ограничений.
 - 8.5. Описание входных и выходных форматов, спецификаций.
 - 8.6. Описание тестирования.
 - 8.7. Описание инструкций пользователю.
- 9. Сопровождение.
 - 9.1. Анализ использования, периодичности использования, количества пользователей, типа использования (диалоговый, автономный и др.), анализ отказов во время использования модели.
 - 9.2. Обслуживание модели, алгоритма, программы и их эксплуатация.

9.3. Распирение возможностей: включение новых функций или изменение режимов моделирования, в том числе и под модифицированную среду.

9.4. Нахождение, исправление скрытых ошибок в программе, если таковые найдутся.

10. Использование модели.

Математическое и компьютерное моделирование подробно и поэтапно мы рассмотрим на примере простой модели производства. Рассмотрим укрупненные этапы моделирования производства.

Этап 1. Содержательная постановка задачи

Современное производство характерно тем, что часть производимой продукции (в стоимостном выражении) возвращается в виде инвестиций (т. е. части конечной продукции, используемой для создания основных фондов производства) в производство. При этом время возврата, ввода в оборот новых фондов может быть своим для различного рода производства. Необходимо промоделировать эту ситуацию и выявить динамику изменения величины основных фондов производства (капитала).

Этап 2. Формулировка гипотез, построение, исследование модели

Динамика изменения величины капитала определяется в основном в нашей модели простыми процессами производства и описывается так называемыми обобщенными коэффициентами амортизации (расхода фондов) и потока инвестиций (часть конечного продукта, используемого в единицу времени для создания основных фондов). Эти коэффициенты — относительные величины (за единицу времени). Необходимо разработать и исследовать модель динамики основных фондов. Считаем при этом допустимость конкретных гипотез, определяющих систему производства.

Пусть $x(t)$ — величина основных фондов (капитала) в момент времени t , где $0 \leq t \leq N$. Через промежуток времени Δt она будет равна $x(t + \Delta t)$. Абсолютный прирост равен $\Delta x = x(t + \Delta t) - x(t)$. Относительный прирост будет равен $\delta x = [x(t + \Delta t) - x(t)]/\Delta t$.

Примем следующие гипотезы:

1) социально-экономические условия производства достаточно хорошие и способствуют росту производства, а поток инвестиций задается в виде известной функции $y(t)$;

2) коэффициент амортизации фондов считается неизменным и равным m , и при достаточно малом значении Δt изменение основных фондов прямо пропорционально текущей величине капитала, т. е. $\delta x = y(t) - mx(t)$.

Считая $\Delta t \rightarrow 0$, а также учитывая определение производной, получим из предыдущего соотношения следующее математическое выражение закона изменения величины капитала — математическую модель (дифференциальное уравнение) динамики капитала:

$$x'(t) = y(t) - mx(t), \quad x(0) = x_0,$$

где $x(0)$ — начальное значение капитала в момент времени $t = 0$.

Эта простейшая модель не отражает того факта, что социально-экономические ресурсы производства таковы, что между выделением инвестиций и их введением и использованием в выпуске новой продукции проходит время. Учитывая это, можно записать модель в виде:

$$x'(t) = y(t - T) - mx(t), \quad x(0) = x_0.$$

Этой непрерывной дифференциальной, динамической модели можно поставить в соответствие простую дискретную модель: $x_{i+1} = x_i + y_j - mx_i$, $x_0 = c$, $i = 0, 1, 2, \dots, n$, $0 < j < n$, где n — предельное значение момента времени при моделировании.

Дискретная модель следует из непрерывной при $\Delta t = 1$, при замене производной $x'(t)$ на относительное приращение (из определения производной это справедливо при малых Δt).

Этап 3. Построение алгоритма моделирования

Рассмотрим для простоты режим моделирования, когда m , c известны и постоянны, а y увеличивается каждый следующий момент времени на 1%. Наиболее простой алгоритм моделирования в укрупненных шагах:

1. Ввод входных данных: $c = x(0)$ — начальный капитал; n — конечное время моделирования; m — коэффициент амортизации; s — единица измерения времени; y — инвестиции.
2. Вычисление x_i от $i = 1$ до $i = n$ по рекуррентной формуле, приведенной выше.
3. Поиск стационарного состояния — такого момента времени j , $0 \leq j \leq n$, начиная с которого все x_i , x_{j+1} , ..., x_n постоянны или изменяются на малую допустимую величину $\varepsilon > 0$.
4. Выдача результатов моделирования и графика.

Алгоритм, записанный на ПЯЯ, имеет вид:

алг Модель_производства (вещ m , c , цел n , вещ таб $x[0:366]$,
лит s , q);

дан | производство с основными фондами по закону:

$x[i+1] = x[i] + y - mx[i]$, $x[0] = c$, $i = 0, 1, 2, \dots, n$, $0 < j < n$,
 $t = i \cdot h$, $h = 1$ — шаг по времени (день),

i — текущий момент времени,

m — коэффициент амортизации,

$x[0] = c$ — заданная начальная величина капитала,

y — увеличиваемая на 1% величина инвестиций

надо | промоделировать динамику фондов, т. е. выяснить:

1) чему они равны на момент времени n ;

2) наступает ли гибель предприятия, т. е. обращаются ли основные фонды в нуль при некотором t , и найти это t ;

3) наступает ли ситуация, когда капитал стабилизируется

нач | начало тела алгоритма

цел i , | i — переменная цикла прогноза (текущее время)
 j , | j — задаваемая величина лага
 k , | k — момент гибели предприятия (если есть)
 y , | y — величина инвестиций, увеличиваемая $y := 1.01 * y$
ввод (m, n, c, y); | ввод исходных данных
 $s :=$ "предприятие не гибнет"; | задаем начальное значение s
 $q :=$ "капитал не стационарен"; | задаем начальное значение q
 $x[0] := c$; | начальное значение капитала (не нулевое)
 $i := 0$; | задаем начальный момент времени моделирования
нц пока ($i \leq n$) и ($x[i] > 0$) | заголовок цикла прогноза
 $x[i+1] = x[i] + y - mx[i]$ | прибыль в следующий момент
 $y := 1.01 * y$ | увеличиваем на 1%
если $x[i+1] \leq 0$ | проверка гибели
 то | блок обработки погибшего предприятия
 $s :=$ "предприятие гибнет" | заменяем значение s
 $k := i - 1$ | и фиксируем время гибели
 нц для j от k до n | цикл вычисления остальных
 $x[j] = 0$ | значений прибыли
 все | конец блока обработки погибшего предприятия
 если $x[i+1] = x[i]$ | проверка стационарности прибыли
 то $q :=$ "капитал стационарен" | заменяем старое q
 все | конец проверки стационарности
 кц | конец цикла прогноза
кон | конец алгоритма моделирования.

По этому алгоритму несложно разработать программу имитационного моделирования (реализуйте его).

Этап 4. Проведение вычислительных экспериментов

Эксперимент 1. Поток инвестиций постоянный и в каждый момент времени равен 1000. В начальный момент капитал — 100 000 р. Коэффициент амортизации — 0.025. Постройте модель динамики (посуточно) и найдите величину основных фондов через 20 сут., если лаг равен 5 сут.

Эксперимент 2. Основные фонды в момент времени $t = 0$ были равны 5000 р. Через какое время общая их сумма превысит 120 000 р., если поток инвестиций постоянный и равен 200, а $m = 0.02$, $T = 3$?

Этап 5. Модификация (развитие) модели

Модификация 1. Коэффициент амортизации можно взять в форме $m = r - sx(t)$, где r — коэффициент обновления фондов: s — коэффициент устаревания фондов, причем $0 \leq r, s \leq 1$. При этом модель примет вид $x'(t) = y(t - T) - rx(t) + sx^2(t)$, $x(0) = x_0$. Этой непрерывной дифференциальной динамической модели можно поставить в соответствие простую дискретную модель:

$$x_{i+1} = x_i + y_j - rx_i + sx_i^2, \quad x_0 = c, \quad i = 0, 1, 2, \dots, n, \quad 0 < j < n,$$

где n — предельное значение момента времени при моделировании. Поставьте цели и исследуйте непрерывную и дискретную модели.

Модификация 2. Одна из моделей математической экономики задается уравнением $dz/dt = ((1-c)z(t) + k(t-w) + a)l$, где $z(t)$ — функция, которая характеризует выпуск продукции; k — коэффициент капиталовложений; a — независимые расходы производства; l — скорость реакции выпуска на капиталовложения; c — постоянная спроса; w — запаздывание (лаг). Поставьте цели и исследуйте непрерывную и дискретную модели.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое модель, математическая модель?
2. Какая модель называется статической (динамической, дискретной, непрерывной, имитационной, детерминированной и т. д.)?
3. Перечислите три задачи моделирования и приведите примеры.
4. Перечислите свойства моделей. Приведите примеры, показывающие необходимость каждого из этих свойств.
5. Перечислите этапы жизненного цикла моделирования.
6. Что такое вычислительный (или компьютерный) эксперимент?
7. Перечислите задачи этапов компьютерного моделирования.

ЗАДАЧИ

1. Скорость выпуска продукции постоянна и в каждый момент времени равна 0.05, она пропорциональна выпускаемой продукции. В начальный момент было выпущено 10 000 единиц продукции. Постройте модель динамики выпуска продукции (посуточно) и найдите продукцию при $t = 5$ сут.
2. Продукция, выпущенная предприятием в момент времени $t = 0$, равна 50 000 единиц. Через какое время продукция превысит 1 200 000 единиц, если скорость выпуска равна 0.001? Постройте модель.
3. Найдите постоянный коэффициент выпуска продукции предприятием, если в начальный момент выпущено 20 000 единиц продукции, в следующий момент — 20 020 (темп выпуска продукции постоянен). Постройте соответствующую модель для решения этой задачи.
4. Чему равен выпуск продукции предприятием в момент времени $t = 5$ сут., если при $t = 0$ выпущено 10 000 единиц продукции, а скорость выпуска меняется по закону $a = 0.01t + 0.002$? Постройте соответствующую модель для решения этой задачи.
5. Модель популяции в условиях сбора урожая. Рассмотрим популяцию рыб, из которой в текущий момент времени изымается часть популяции (сбор урожая). Модель имеет вид $x_{i+1} = x_i + ax_i - kx_i$, $x_0 = c$, где k — коэффициент сбора урожая (скорость изъятия особей). Стоимость одной рыбы равна b р.

- Цель моделирования: а) прогноз прибыли от лова рыбы; б) прогноз прибыли от лова рыбы при заданной квоте вылова.
6. Модель «хищник — жертва». Алгоритм эволюции клеточного автомата, моделирующего систему типа «хищник — жертва», состоит из этапов:
- 1) задаются начальные распределения хищников и жертв (случайно или детерминированно);
 - 2) определяются законы «соседства» особей (правила взаимоотношений) клеток: например, соседями клетки с индексами (i, j) считаются клетки $(i - 1, j)$, $(i, j + 1)$, $(i + 1, j)$, $(i, j - 1)$;
 - 3) задаются законы рождаемости и смертности клеток: например, если у клетки меньше двух (больше трех) соседей, она отмирает «от одиночества» («от перенаселения»).
- Цель моделирования: определение эволюции хищников и жертв, т. е., используя заданные законы соседства и динамики дискретного развития (время дискретно), указывают число новых особей и число погибших особей; если достигнута заданная конфигурация клеток или развитие привело к исчезновению вида (циклу), то моделирование заканчивается.
7. Модель обслуживания в очереди. Имеется некоторая очередь на обслуживание на АЗС из n автомашин. В единицу времени поступает в среднем a вызовов. Средняя длительность обслуживания одного вызова равна b , среднее время ожидания в очереди равно $T = b/(c(n - ab))$, где $n > ab$ и $c = 1 + ab + (ab)^2/2! + (ab)^3/3! + \dots + (ab)^{n-1}/(n-1)! + (ab)^n/((n-1)!(n-ab))$. Цель моделирования: а) прогнозирование среднего времени ожидания; б) нахождение удельной загрузки обслуживающего по формуле $d = (ab)/n$.

ТЕСТ

1. Математическая модель системы — это:
 - а) система программ для решения задач в системе;
 - б) система алгоритмов для решения задач системы;
 - в) математическое описание связей для исследования системы;
 - г) система математических уравнений для исследования системы.
2. Правильная последовательность этапов моделирования на ЭВМ — это:
 - а) алгоритм — тесты — программа — гипотеза — модель — решение;
 - б) гипотеза — модель — алгоритм — компьютерный эксперимент — тест;
 - в) гипотеза — модель — алгоритм — тесты — оценка адекватности;
 - г) гипотеза — модель — оценка адекватности — алгоритм — программа.

3. Математическая модель системы используется:
 - а) для вычисления по модели;
 - б) для управления системой с помощью модели;
 - в) для обучения моделям и моделированию;
 - г) во всех перечисленных пунктах — а, б, в.
4. Правильная последовательность этапов составления, отладки программы:
 - а) тесты — алгоритм — структура данных — тестирование;
 - б) структура данных — алгоритм — тесты — тестирование;
 - в) алгоритм — структура данных — тесты — тестирование;
 - г) тесты — структура данных — алгоритм — тестирование.
5. Модель движения материального тела $s = vt$, $0 \leq t \leq 10$, будет:
 - а) детерминированной, дискретной, статической;
 - б) детерминированной, динамической, дискретной;
 - в) детерминированной, непрерывной, динамической;
 - г) стохастической, непрерывной, динамической.
6. При описании свободного падения тела с учетом влияния порыва ветра полученная модель движения будет:
 - а) детерминированной, статической;
 - б) стохастической, динамической;
 - в) детерминированной, динамической;
 - г) стохастической, статической.
7. При скорости 0.05 прироста популяции пропорционально плотности численности и начальной плотности 1000, плотность в конце вторых суток равна:
 - а) 1050; б) 1102,5; в) 1175,5; г) 1250,5.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
в	в	г	б	в	б	б

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Математическое моделирование — метод и методология.
2. Компьютерное моделирование — метод и методология.
3. Вычислительный эксперимент: его значение.
4. Модели в макромире и микромире.
5. Моделирование — универсальный метод получения знаний.
6. Усиление роли моделирования в современном мире.
7. Функции компьютера и человека при моделировании.

Н **19** **НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

Процесс получения информации строится на основе упорядоченных действий по сбору, накоплению, отражению, преобразованию, актуализации данных. Такие процессы в информатике называются **информационными технологиями**. Их основными элементами являются технические средства и устройства: в телеграфе — телетайпное устройство, в телевидении — телевизор и т. д.

Новые информационные технологии — информационные технологии, базирующиеся на новых, компьютерных средствах получения, хранения, актуализации информации, информационных процессов в жизни общества.

Высокие технологии — технологии качественного (порядкового) изменения состава, характера, методов решаемых задач.

ПРИМЕР

Обычная («старая») технология вычислений ставит основную цель — найти решение задачи за приемлемое время и стоимость. Новая технология использования математических компьютерных пакетов ставит новую цель — найти решение достаточно быстро, точно и экономично. Высокая технология распределенных вычислений ставит цель — найти решение задачи, не решаемой (трудно решаемой) новыми или обычными технологиями.

В узком понимании новая информационная технология — использование вычислительной техники и систем связи для создания, сбора, передачи, хранения, обработки информации или часть информационного бизнеса.

Рассмотрим основные направления новых информационных технологий (математическое и компьютерное моделирование нами рассмотрено выше).

19.1. КОМПЬЮТЕРНЫЙ (КОМПЬЮТЕРИЗИРОВАННЫЙ) И ВИРТУАЛЬНЫЙ ОФИС

Компьютерный (компьютеризированный) офис — офис, в котором имеется высокий уровень компьютеризации, внедрения АРМ, систем делопроизводства и вся профессиональная деятельность которого автоматизирована. Высшая форма развития такого офиса — **виртуальный офис и виртуальная корпорация** или офис и корпорация, не существующие в обычном, классическом виде («с вывеской, штатом, зданием»). Они воображаемы, распределены как в пространстве, так и во времени (отделы и сотрудники могут находиться

на различных континентах, общаясь по работе с помощью ЭВМ и сетей связи).

Для автоматизации работ в офисе Microsoft разработала интегрированный пакет приложений к ОС Windows — Microsoft Office или семейство пакетов, имеющих единый пользовательский интерфейс, с помощью единого протокола связывания и вставки объектов OLE (Object Linking and Embedding).

ПРИМЕР

Текстовые документы, таблицы, графики можно, например, создавать как в текстовом редакторе Word, так и в электронной таблице Excel, хотя в первом удобнее создавать активные тексты, чем активные таблицы, а во втором — наоборот. Эти документы затем можно связывать с помощью OLE.

Microsoft Office имеет встроенный простой язык программирования Visual Basic for Applications (VBA). Этот язык позволяет создавать новые приложения или корректировать и связывать старые, выполняемые в среде Microsoft Office, а также подключать различные макросы (наборы команд, расширяющие стандартные возможности) к приложениям.

ПРИМЕР

Из Excel можно с помощью программ на Visual Basic обращаться к объектам Word: например, можно написать программу специальной обработки данных, полученных стандартными и ограниченными средствами Excel. Рассмотрим базовые, стандартные приложения Microsoft Office.

А. Редактор текстов Word. Как и любой редактор текстов, он позволяет выполнять основные функции редактора — набор, именование и сохранение текста, модификацию, переименование и перемещение текста или его отдельных фрагментов, вставку различных формул, графиков, таблиц, диаграмм и др.

Хотя Word и не обладает той гибкостью и качеством оригинал-макета, который можно получить, например, в среде редактора T_{EX}, все же Word является одной из наиболее часто используемых сред. Вводимый текст отображается на экране именно в том виде, в каком он будет выведен на печать: «что видите на экране, то и получите на бумаге».

Имеется встроенный редактор математических формул. Формулы хотя и занимают много памяти, громоздки для редактирования и неудобны для масштабирования, но при определенном опыте их использования можно добиться хорошего качества.

Можно автоматически создавать конверт, адрес для которого будет выбран из базы данных адресатов и автоматически напечатан на конверте. Есть встроенная программа проверки правописания, и динамически по ходу набора текста проверяется орфография. Текст можно форматировать и нумеровать автоматически, с переносами слов или без них. Можно использовать электронную почту из сре-

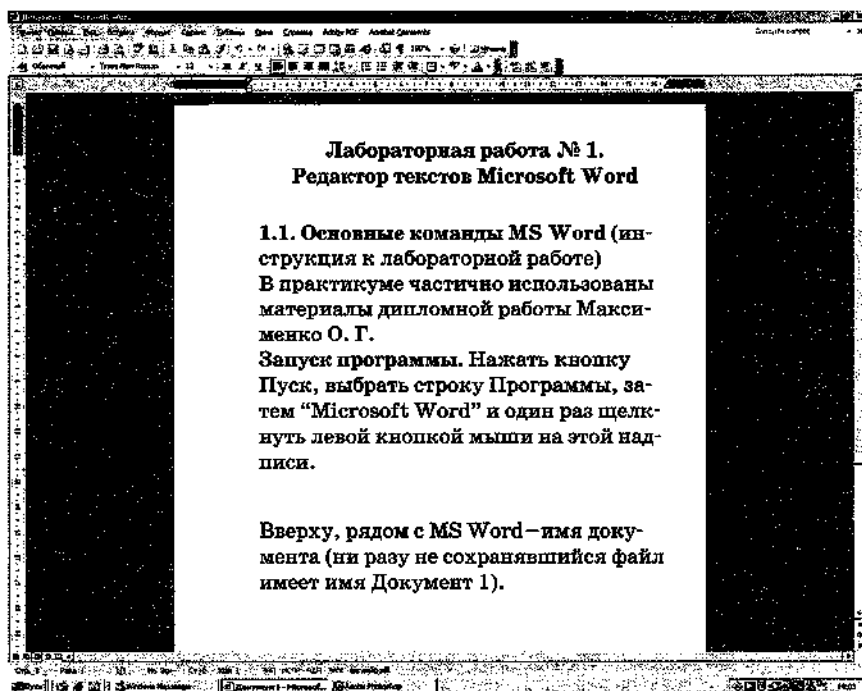


Рис. 29. Основное рабочее окно Word

ды редактора, например набранный текст можно сразу отослать по почте, следовательно, в письмо можно включать рисунки, графики, таблицы, диаграммы, что позволяет принимать на рассмотрение, согласование и подписание документации, т. е. вести электронный документооборот в офисе.

Далее мы везде упрощенно под «кнопкой» понимаем следующую процедуру: выбор соответствующей пиктограммы (рисунка), установка курсора на нее и нажатие (щелчок по левой кнопке мыши), т. е. реализацию всех действий, связанных с этой пиктограммой.

Запуск программы. Кнопка Пуск, выбрать из появившегося меню Программы, затем Word и щелкнуть левой кнопкой мыши на этой надписи.

Выход из программы. Выйти из MS Word можно: а) щелкнув на кнопке крестик справа вверху; б) выполнив команду **Выход**, меню **Файл**. Если текущий документ не сохранялся, появится окно с предложением сохранить документ. Если нужно сохранить, то щелкнуть на кнопке **Да**, не нужно сохранять — **Нет**, отмена команды выхода из Word — **Отмена**.

Рабочее окно MS Word. После запуска Word на экране появится окно вида, показанного на рисунке 29.

Слева вверху, рядом с MS Word — имя документа (не сохранявшиеся и пока «безымянные» новые файлы имеют имена Документ 1, Документ 2 и т. д.).

Горизонтальное (главное) меню имеет вид



Для открытия одного из меню необходимо один раз щелкнуть левой клавишей на заголовке, в который необходимо войти. Появится меню. При выборе команды из этого меню достаточно переместить указатель на нужную строку и нажать на кнопку мыши. В дальнейшем для краткой записи того, что надо, например, выбрать из меню **Файл** и выполнить команду **Открыть** появившегося меню (подменю), мы будем использовать запись «**Файл, Открыть**».

С файлами и документами помогают работать команды меню **Файл**:

- 1) **Создать** — создание нового документа.
- 2) **Открыть** — открытие документа, который хранится на диске.
- 3) **Сохранить** — сохранение текущего документа на диске с текущим именем.
- 4) **Сохранить как** — сохранение текущего документа с задаваемым именем.
- 5) **Параметры страницы** — установка полей печати, формы страницы и др.
- 6) **Предварительный просмотр** — показ документа, как он будет напечатан.
- 7) **Печать** — управление печатью документа.
- 8) **Перечень файлов** — список активных последних документов.
- 9) **Выход** — выход из Word.

Команды меню **Правка** служат для редактирования документа:

- 1) **Отменить** — отмена последней операции.
- 2) **Повторить** — повторение последней отмеченной операции.
- 3) **Вырезать** — удаление выделенного фрагмента с сохранением его в буфере обмена.
- 4) **Копировать** — копирование выделенного фрагмента текста в буфер обмена.
- 5) **Вставить** — вставка содержимого буфера в позицию, помеченную курсором.
- 6) **Удалить** — удаление выделенного фрагмента без сохранения в буфере обмена.
- 7) **Выделить все** — выделение всего текста.
- 8) **Найти** — нахождение нужного фрагмента текста в документе.
- 9) **Заменить** — замена одного фрагмента текста в документе на другой.

Изменяют структуру и вид экрана команды меню **Вид**:

- 1) **Обычный** — просмотр документа с визуализацией видов форматирования.

- 2) **Структура** — просмотр документа с созданием, изменением его структуры.
- 3) **Разметка страницы** — просмотр разметки страницы, как она будет напечатана.
- 4) **Во весь экран** — отображение документа во весь экран.
- 5) **Панели инструментов** — отображение или скрытие панелей инструментов.
- 6) **Линейка** — отображение или скрытие линейки.
- 7) **Колоды титулы** — создание, изменение текста сверху или снизу страниц.

Меню **Вставка** служит для вставки номера страницы, символа, рисунка и др.

Форматируют символы, таблицы, фрагменты команды меню **Формат**:

- 1) **Шрифт** — установка параметров шрифтового оформления.
- 2) **Абзац** — установка параметров оформления абзацев.
- 3) **Границы и заливка** — обрамление элементов текста и закрашивание их.
- 4) **Регистр** — переключение на строчные или прописные буквы.

Команды меню **Сервис** улучшают вид документа, печати, орфографию:

- 1) **Орфография** — грамматическая проверка документа.
- 2) **Расстановка переносов** — выбор вариантов расстановки переносов.
- 3) **Настройка** — работа с панелью настройки.
- 4) **Параметры** — установка параметров работы с документом и др.

Меню **Таблица** содержит команды работы с таблицами:

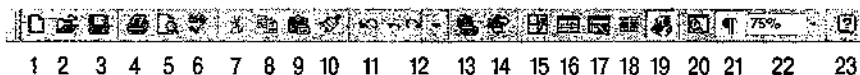
- 1) **Вставить таблицу** — вставка таблицы в позицию, где курсор.
- 2) **Вставить строки** — вставка новых строк в созданную таблицу.
- 3) **Удалить ячейки** — удаление ячейки из таблицы.
- 4) **Объединить ячейки** — объединение нескольких ячеек в одну.
- 5) **Разбить ячейки** — разбиение одной ячейки на несколько ячеек.
- 6) **Выделить строку** — выделение строки, где стоит курсор.
- 7) **Выделить столбец** — выделение столбца, где стоит курсор.
- 8) **Автоформат** — выбор формата таблицы.
- 9) **Высота и ширина ячейки** — установка параметров ячейки.
- 10) **Сортировка** — сортировка данных таблицы в заданном порядке.
- 11) **Формула** — актуализация формулы для вычисления.
- 12) **Сетка** — отображение или скрытие таблицы (ее границ).

Меню **Окно** содержит команды для работы с окнами:

- 1) **Новое** — создание нового окна, являющегося копией текущего.
 - 2) **Расположить все** — отображение без перекрытий окон на экране.
 - 3) **Разделить** — разбиение активного окна на два окна.
- Назначение команд меню ? — помощь при работе с Word (хелпер).
Панели инструментов.

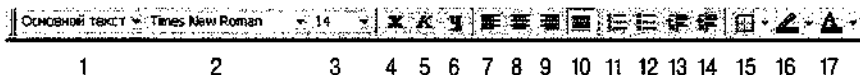
Панель — набор кнопок, с помощью которых выполняются операции для Word задачи. Рассмотрим четыре основные панели инструментов.

Панель инструментов «Стандартная» — для выполнения операций по загрузке, сохранению, редактированию, масштабированию, печати текста и т. д.:



- 1) создание нового документа;
- 2) открытие файла документа, который хранится на диске;
- 3) сохранение активного документа под именем, под которым он был создан;
- 4) управление печатью документа;
- 5) просмотр вида документа, в котором он будет напечатан;
- 6) грамматическая проверка документа;
- 7) удаление выделенного фрагмента в буфер обмена;
- 8) копирование выделенного фрагмента текста в буфер обмена;
- 9) вставка содержимого буфера обмена в позицию, в которой стоит курсор;
- 10) копирование форматов символов;
- 11) отмена последней операции;
- 12) повторение последней отмененной операции;
- 13) добавление гиперссылки;
- 14) панель WWW;
- 15) подключение панели инструментов «Таблицы и границы»;
- 16) вставка таблицы в позицию, в которой стоит курсор;
- 17) вставка листа MS Excel в позицию, помеченную курсором;
- 18) изменение количества колонок в документе;
- 19) подключение панели инструментов «Рисование»;
- 20) слева выводится схема документа (пункты и подпункты);
- 21) визуализация специальных символов форматирования;
- 22) масштабирование изображения документа на экране;
- 23) вызов мастера подсказок.

Панель инструментов «Форматирование» — для форматирования элементов текста:



- 1) выбор стиля форматирования;
- 2) выбор типа шрифта;
- 3) выбор кегля (высоты символов, букв);
- 4) написание букв **полужирным шрифтом**;
- 5) написание букв *курсивом*;
- 6) написание букв с подчеркиванием;

- 7) выравнивание текста по левому краю;
- 8) выравнивание текста по центру;
- 9) выравнивание текста по правому краю;
- 10) выравнивание текста по ширине (по обоим краям одновременно);
- 11) создание нумерованного списка (с автоматической нумерацией);
- 12) создание списка-бюллетеня с пометками различного типа;
- 13) уменьшение абзацного отступа;
- 14) увеличение абзацного отступа;
- 15) установка внешних границ;
- 16) выделение линий нужным цветом;
- 17) выделение букв нужным цветом.

Панель инструментов «Таблицы и границы» — для оформления текста:



- | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|--|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| 1) установка типа линии оформления элемента текста (абзаца, таблицы, кадра); | 2) установка (изменение) толщины линии; | 3) установка (изменение) цвета границы; | 4) установка оформления сверху, снизу, слева, справа, внутри и т. д.; | | | | | | | | | | | | | |
| 5) цвет заливки выделенной ячейки; | 6) объединение ячеек; | 7) разбиение ячеек; | 8) выравнивание текста по верхнему краю; | | | | | | | | | | | | | |
| 9) центрирование текста по вертикали; | | | 10) выравнивание текста по нижнему краю; | | | | | | | | | | | | | |
| 11) выравнивание высоты строк; | | | 12) выравнивание ширины столбцов; | | | | | | | | | | | | | |
| 13) автоформат таблицы; | | | 14) изменение направления текста в ячейке; | | | | | | | | | | | | | |
| 15) сортировка данных в строках таблицы по возрастанию; | | | 16) сортировка по убыванию; | | | | | | | | | | | | | |
| 17) автоматическое суммирование данных в строках или столбцах таблицы. | | | | | | | | | | | | | | | | |

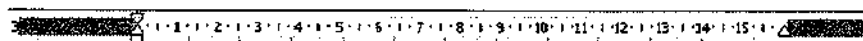
Панель инструментов «Рисование» — для создания рисунков и управления ими:



- | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|--|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|
| 1) действия над фигурами (группировка, сдвиг, изменение автофигуры); | | | | | | | | | | | | | | | | | |
| 2) выбор объекта; | | | | | | | | | | | | | | | | | |
| 3) свободное вращение автофигуры; | | | | | | | | | | | | | | | | | |
| 4) вычерчивание автофигур (линия, фигуры, стрелки, блок-схем и др.); | | | | | | | | | | | | | | | | | |

- 5) вычерчивание линий (вертикальных, горизонтальных, жирных и т. д.);
- 6) вычерчивание стрелок (левых, правых, жирных и т. д.);
- 7) вычерчивание прямоугольников или квадратов;
- 8) вычерчивание эллипсов или окружностей;
- 9) вставка надписи;
- 10) добавление объекта WordArt (библиотеки рисунков на различные темы);
- 11) заливка, закрашивание замкнутой поверхности;
- 12) установление цвета линии;
- 13) установление цвета шрифта;
- 14) установка толщины линии;
- 15) установка стиля линии (штрих, сплошная);
- 16) установление вида и направления стрелки;
- 17) настройка тени (от нарисованного объекта);
- 18) настройка объема (нарисованного объекта).

Линейки. Ниже изображена горизонтальная линейка, позволяющая быстро изменять отступы абзацев, поля страницы, ширину колонок текста и столбцов таблицы.



Есть и вертикальная линейка (слева), служащая для быстрого изменения верхних и нижних полей страницы и высоты строк таблицы.

Строка состояния. Содержит информацию о месте курсора, странице, операции и др.



Полосы прокрутки. Горизонтальная и вертикальная полосы прокрутки, размещенные по нижней и правой границам документа, используются для перемещения по документу (прокрутки) с помощью мыши. Есть два способа перемещения по документу: а) установив курсор на бегунок и передвигая его, нажав левую кнопку мыши (не отпуская); б) щелкнув на кнопке со стрелкой.

Просмотр документа. В левом нижнем углу окна — кнопки быстрого выбора режимов:



- 1) режим просмотра документа — визуализирует виды форматирования документа без деталей разметки страницы (режим выбирается кнопкой);
- 2) режим электронного документа — визуализирует слева элементы документа;
- 3) режим просмотра разметки страницы — позволяет получить изображение страницы в том виде, в каком она будет распечатана (рекомендуется);
- 4) режим просмотра структуры документа — в этом режиме можно создавать, просматривать или изменять структуру документа.

Масштабирование изображения текста. Изменить масштаб изображения текста на экране можно кнопкой с процентами: выбрав нужный процент или введя его в окно ввода.

Ввод текста. Ввод текста осуществляется с клавиатуры, как на печатающей машинке.

Клавиши	Назначение
Shift+Ctrl или Shift+Alt	Переключение на русский/английский алфавит
Shift+<буква>	Переключение на строчные/прописные буквы
Backspace	Стирание символа слева от курсора
Delete	Стирание символа справа от курсора
Caps Lock	Печать прописными буквами
Ctrl+Home	Перемещение курсора на первую букву текста
Ctrl+End	Перемещение курсора на последнюю букву текста
Ctrl+↑	Перемещение курсора на абзац вверх
Ctrl+↓	Перемещение курсора на абзац вниз
End	Перемещение курсора в конец строки
Home	Перемещение курсора в начало строки
PageUp	Листание кадров (экранных страниц) вверх
PageDown	Листание кадров (экранных страниц) вниз
←↑→↓	Перемещение курсора в направлении стрелки

Правила ввода текста: символ вводится в позицию, где мигает курсор; указатель мыши служит только для установки текстового курсора в нужную позицию; нажимать **Enter** надо в конце абзаца; не производить центрирование, установку абзаца и сдвиг текста с помощью пробелов; не нумеровать страницы вручную; после точки и запятой (а не до!) следует ставить пробел.

Форматирование текста. Для этого фрагмент нужно выделить.

Элемент	Способ выделения
Слово	Курсор — на любой символ слова и щелкнуть 2 раза
Предложение	Курсор — на любой символ предложения, нажать Ctrl и щелкнуть
Строка	Курсор — на полосу выделения напротив строки и щелкнуть; если, не отпуская левую кнопку мыши, перемещать ее указатель вверх или вниз, то выделяется несколько смежных строк
Абзац	Курсор — на полосу выделения напротив абзаца и 2 раза щелкнуть; если, не отпуская левую кнопку мыши, перемещать ее указатель вверх или вниз, то выделится несколько смежных абзацев
Весь текст	Курсор — на полосу выделения и 3 раза щелкнуть, или нажать клавишу Ctrl и щелкнуть, или Правка, Выделить все , или Ctrl+F
Рисунок	Щелкнуть на некотором месте рисунка
Таблица	Поставить курсор внутрь таблицы и выполнить Таблица, Выделить таблицу

Чтобы отменить выделение фрагмента текста, надо щелкнуть вне его. Далее используем кнопки панели инструментов «Форматирование» (см. выше). С помощью **Формат, Шрифт** можно установить: тип шрифта; начертание шрифта (**полужирный**, **обычный**, *курсив*); кегль шрифта (8—72); подчеркивание; цвет шрифта; эффекты (верхний индекс — x^3 , нижний индекс — x_3 и т. д.).

Редактирование текста. Операции редактирования текста позволяют изменить существующий документ добавлением, удалением или перестановкой его фрагментов, слиянием нескольких документов в один или разбиением документа на несколько более мелких.

Редактирование строки. Для вставки пустой строки установить курсор за последним символом строки и нажать **Enter**. Для слияния двух строк установить курсор в позицию, следующую за последним символом первой из двух соединяемых строк, и нажать **Delete**.

Удаление фрагментов документа. Нажатием клавиши **Backspace** удаляется символ слева от курсора, а нажатием клавиши **Delete** — справа от курсора. Удалить фрагмент можно одним из спо-

совов: а) выделить фрагмент и нажать **Delete**; б) выделить фрагмент и выполнить **Правка, Вырезать** (фрагмент при этом переместится в буфер обмена).

Перемещение и копирование фрагментов текста. Осуществляется одним из способов: а) выделив фрагмент текста, затем (для перемещения) нажав комбинацию клавиш **Shift+Delete**, затем (для копирования) нажав комбинацию клавиш **Ctrl+Ins**, затем установив курсор в точку вставки фрагмента и нажав **Shift+Ins** (фрагмент вставится в текст из буфера обмена); б) выделив фрагмент текста, затем (для перемещения) выполнив **Правка, Вырезать** или щелкнув на кнопке панели инструментов «Стандартная», затем (для копирования) выполнив **Правка, Копировать** или щелкнув на кнопке панели инструментов «Стандартная», потом, установив курсор в точку вставки фрагмента и выполнив **Правка, Вставить** или щелкнув на кнопке панели инструментов «Стандартная» (фрагмент вставится в текст из буфера обмена — Clipboard).

Создание таблицы. Выполнить **Таблица, Вставить таблицу**. Затем в диалоговом окне (рис. 30) установите в окнах «Число строк» и «Число столбцов» нужное количество строк и столбцов таблицы, нажмите **ОК**. В нужной ячейке набирайте текст. Для перехода в другую ячейку используйте клавиши управления курсором.

Б. Электронная таблица Excel. Электронная таблица — программная система, позволяющая обрабатывать, хранить и модифицировать (в отличие от обычных таблиц) в таблице не только числа, строки, столбцы, но и формулы, по которым динамически модифицируются числа, строки и столбцы.

В MS Office имеется наиболее используемая электронная таблица — Excel. Она позволяет вставлять в свои таблицы диаграммы, графики, объекты, а также вставлять Excel-объекты в другие объекты и приложения, например в объекты Word, PowerPoint и др. Вся работа с Excel осуществляется с помощью строки инстру-

ментов, в которой имеются соответствующие кнопки для манипуляций с элементами и формулами таблицы. Каждая строка нумеруется цифрой, а столбец — латинской буквой. В любую ячейку таблицы можно вводить числа и значения, вычисляемые по формуле. Для этого ячейка активизируется, выделяется более толстой рамкой. Данные в таблицу вводятся интерактивно, пользователь динамически и визуально отслеживает создаваемую таблицу.

В качестве параметров выражений, по которым должно

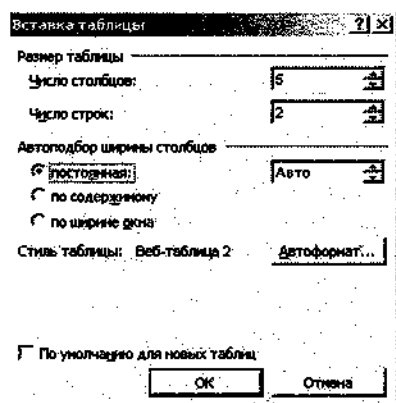


Рис. 30. Окно для создания таблиц

быть рассчитано содержимое тех или иных ячеек, используются как числа, так и номера (имена) ячеек, которые, в свою очередь, также могут быть рассчитаны по некоторым формулам (вложение формул). В Excel можно использовать несколько сотен математических, статистических, финансовых и других специализированных функций, несколько десятков встроенных типов диаграмм, а также создавать свои формулы и диаграммы.

Excel-книга содержит листы (таблицы). Каждая таблица состоит из 256 столбцов и 16 384 строк. Строки нумеруются числами, начиная с 1. Столбцы обозначаются латинскими буквами. Каждая ячейка таблицы имеет свой адрес: имя столбца и номер строки (D9 — адрес ячейки, расположенной на пересечении строки 9 и столбца D). Все данные таблицы размещаются в ячейках.

Запуск Excel: Пуск, Программы, Excel и Enter.

Заключить работу с Excel можно двумя способами: а) с помощью команды Выход меню Файл; б) активизацией кнопки с крестиком в верхней части окна. Если последняя редакция документа (рабочего листа) не была сохранена, а вы пытаетесь завершить работу с Excel, то программа выдает запрос о том, необходимо ли сохранить изменения в рабочем документе.

Рабочее окно MS Excel показано на рисунке 31.

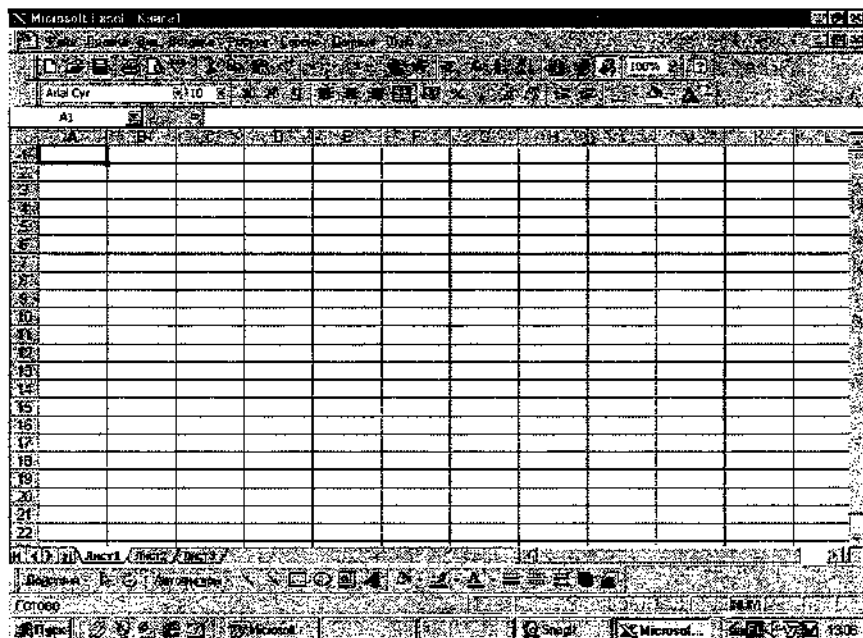


Рис. 31. Основное рабочее окно Excel

Большую часть экрана занимает окно документа (электронная таблица), расчерченная с помощью вертикальных и горизонтальных линий на ячейки. Таблица обрамлена рамкой, используемой для нумерации строк и наименования столбцов. Номер текущей электронной таблицы высвечивается в нижней части экрана. При заполнении таблицы и при ее просмотре табличный курсор (крест) высвечивает границу ячейки. Такая ячейка называется текущей.

Главное меню. В горизонтальном (главном) меню 8 пунктов (не считая Help). Оно почти полностью совпадает с горизонтальным меню MS Word. Это связано с единой технологией OLE, использованной в MS Office: интерфейсы приложений реализованы на одних и тех же принципах. Например, в Excel пункты меню Файл, Вид и Окно аналогичны Word. Есть и отличия, например, в пункте Правка есть команды Заполнить ячейки формулами, Очистить ячейки, в пункте Вставка — команды Ячейки, Строки, Функция и т. д.

Панели инструментов. Рассмотрим две панели: «Стандартная» и «Форматирование».

Панель инструментов «Стандартная». Кнопки, показанные на рисунке 32, аналогичны кнопкам Word: 1) создать новую рабочую книгу; 2) открыть существующий файл; 3) сохранить активный документ; 4) напечатать активный документ; 5) просмотреть перед печатью; 6) проверить правописание.

Группа кнопок на рисунке 33 содержит кнопки: 1) вырезать фрагмент; 2) копировать фрагмент; 3) вставить фрагмент; 4) копировать формат.

Кнопки (рис. 34) отменяют (1) и повторяют (2) последнюю команду.

С помощью кнопок, показанных на рисунке 35, выполняются операции, упрощающие ввод функций в ячейку электронной таблицы. Если щелкнуть на кнопке 1, то в текущей ячейке появится заготовка функции = SUM(...), аргументы которой Excel подставит по соглашению. Можно отредактировать эту функцию или зафиксировать в ячейке без редактирования. После щелчка на кнопке 2 на экран поступит диалоговое окно Мастера функций и можно будет выбрать и отредактировать функцию, вводимую в текущую ячейку.

Кнопки, показанные на рисунке 36, позволяют сортировать выделенные столбцы по строкам и возрастанию (1) или убыванию (2) кодов букв.

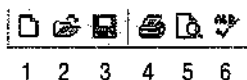


Рис. 32



Рис. 33

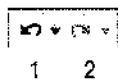


Рис. 34

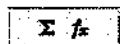


Рис. 35

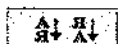
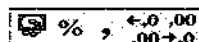


Рис. 36



Рис. 37

Рис. 38



СЗ



Рис. 39

Рис. 40

Рис. 41

Три кнопки рисунка 37 обслуживают графические операции Excel:

Кнопка 1 (Мастер диаграмм) позволяет «нарисовать» на рабочем листе прямоугольную рамку и в диалоговом режиме построить диаграмму.

Кнопка 2 позволяет создать на рабочем листе текстовое окно и заполнить его произвольным текстом и (или) рисунками.

Кнопка 3 — переключатель панели инструментов «Рисование», с помощью которой можно создать на рабочем листе рисованный объект, как и в Word.

В группе кнопок справа есть знакомый по Word раскрывающийся список масштабирования (для увеличения или уменьшения изображения электронной таблицы на экране). Если щелкнуть на кнопке переключателя панели подсказок-советов («лампочка»), то на экране появится хелпер по операциям Excel.

Панель инструментов «Форматирование». В группе кнопок этой панели (рис. 38) есть кнопки: выбора шрифта (1) и его размера (2), переключатели начертания Ж, К, Ч (3—5), полей выбора способа выравнивания текста — к левой границе ячейки (6), по центру (7), к правой границе ячейки (8); кнопка (9) выравнивает текст как целое по центру выделенной в строке группы ячеек.

Кнопки, показанные на рисунке 39, позволяют: 1 — пометить число в текущей ячейке знаком доллара; 2 — умножить число в текущей ячейке на 100 и пометить его знаком «%» (в процентах); 3 — установить разделитель тысяч в представлении числа; 4—5 — увеличить или уменьшить точность представления числа (изменить количество знаков после точки).

Группа кнопок (рис. 40) служит для оформления таблицы: 1 — рамка для окаймления выделенного блока ячеек; 2 — цвет фона выделенного блока ячеек; 3 — цвет символов в выделенном блоке.

Строка формул показана на рисунке 41 и предназначена для обработки содержимого ячеек. Кнопка 1 — для отображения содержимого текущей ячейки, которое можно редактировать с помощью кнопок 2—5, расположенных в центре строки. Редактирование данных можно производить непосредственно в самой ячейке или в строке формул. Для перехода в режим редактирования содержимого текущей ячейки в строке формул следует нажать клавишу F2 или выполнить щелчок в правой части строки формул. Если выполнить двойной щелчок на ячейке, то она станет текущей и произойдет переход в режим редактирования. Кнопки, представленные в строке формул, предназначены для обработки содержимого ячейки. Кноп-

ка 3 применяется для отмены последнего действия. После щелчка на этой кнопке последние изменения содержимого ячейки будут считаться недействительными, т. е. действие этой кнопки аналогично действию клавиши Esc. Кнопка 4 служит для подтверждения ввода данных или изменения содержимого ячейки и соответствует клавише Enter. С помощью кнопки 5 можно активизировать программу **Мастер функций** для работы с функциями. В правой части строки формул, в поле имени, указан адрес активной ячейки или размер выделенной области, например $2R \times 3C$ указывает на то, что эта область состоит из двух строк и трех столбцов.

Строка состояния. Она содержит индикатор состояния табличного процессора или расписывку текущей команды меню, используемого инструмента: например, «Ввод» — ввод данных; «Правка» — редактирование данных.

Адресация ячеек. Обозначение ячейки, составленной из номеров столбца и строки (A5, B7 и т. д.), называется относительным адресом или просто адресом. При некоторых операциях копирования, удаления, вставки Excel автоматически изменяет этот адрес в формулах. Иногда это служит источником ошибок. Чтобы отменить автоматическое изменение адреса данной ячейки, можно назначить ей абсолютный адрес. Для этого необходимо проставить перед номером столбца и (или) перед номером строки знак «\$». Например, в адресе \$A5 не будет меняться номер столбца, в адресе B\$7 — номер строки, а в адресе \$D\$12 — ни тот ни другой номер. Чтобы сослаться на диапазон ячеек (например, на группу смежных ячеек в строке), можно указать через двоеточие адреса начальной и конечной ячеек в диапазоне. Например, обозначение A7:E7 адресует ячейки A, B, C, D, E в строке 7, обозначение B3:B6 адресует ячейки 3, 4, 5, 6 в столбце B и т. п. Диапазоном можно обозначить и блок ячеек. Например, для ссылки на блок можно написать C6:G9.

Ввод данных в ячейку таблицы. Чтобы ввести данные в конкретную ячейку, необходимо выделить ее щелчком, а затем щелкнуть на строке формул (справа от вертикальной черты). Появится мерцающий курсор, и можно набирать информацию. Вводимые в строке формул данные воспроизводятся в текущей ячейке. При вводе данных в левой части строки формул возникают три кнопки, предназначенные для управления процессом ввода: с красным крестиком, зеленой галочкой и значком f_x . Если щелкнуть на кнопке с крестиком, ввод прекращается и данные, поступившие в ячейку в текущем сеансе ввода, удаляются. Если щелкнуть на кнопке со значком f_x , то на экране появится диалоговое окно **Мастера функций** и можно ввести функцию. Завершив ввод данных, нужно зафиксировать их в ячейке любым из трех способов: нажав клавишу Enter, или щелкнув на кнопке с галочкой, или щелкнув мышью на другой ячейке.

Для ввода данных можно также дважды щелкнуть мышью на выбранной ячейке, и текстовый курсор появится в ячейке. В этом случае допускается только ввод символов и удаление их клавишей

Backspace. Для фиксации данных можно либо воспользоваться одним из описанных выше трех способов, либо сместить курсорную рамку на другую ячейку любой из клавиш управления курсором. Для ввода новых или для исправления старых данных можно просто начать их набор в текущей ячейке. Ячейка очищается, появляется текстовый курсор, активизируется строка формул. Адреса ячеек и имена функций можно набирать в любом регистре — верхнем или нижнем. В любую ячейку электронной таблицы можно ввести число, формулу, текст по следующим правилам:

Число. Если набрана последовательность символов, в которую входят цифры, а также знаки «+», «-» (в начале последовательности) или «.» (как разделитель целой и дробной частей), то эта последовательность считается числом. Примеры чисел: 34; -26.3; +750.31. После фиксации числа Excel сдвигает его к правой границе ячейки. Можно выровнять его по центру ячейки или сместить к левой границе ячейки с помощью кнопок выравнивания. Кроме того, можно записать число в экспоненциальной форме (в виде мантиссы — дробной части числа и порядка — показателя степени 10). Например, число 48 900 можно представить как 4.89E+04. Чтобы число в ячейке выглядело на экране именно так, как вы его ввели, количество вводимых знаков не должно превышать некоторую величину, зависящую от задаваемой (или по умолчанию) ширины столбца (обычно 8—12 символов). Слишком большие и слишком малые числа Excel попытается представить экспоненциально, а при дальнейшем увеличении количества знаков в числе Excel отобразит число как цепочку символов # (# — «эхо» цифры числа, которую невозможно «втиснуть» в заданный формат изображения числа). Точность числа (число знаков после точки) можно регулировать кнопками панели инструментов «Форматирование».

Формула. Если набранная последовательность символов начинается со знака «=» (равно), Excel определяет ее как формулу, арифметическое выражение. В формулу могут входить только числа, адреса ячеек и функции, соединенные знаками арифметических операций. Например, если в ячейку B5 введена формула =A2+C3*F7, значением этой ячейки будет число, равное произведению чисел, записанных в C3 и F7, сложенному с числом из ячейки A2. В зависимости от значения переключателя Формулы вкладки Вид команды Сервис—Параметры в ячейке на экране может отображаться как формула, так и результат вычислений по данной формуле.

Текст. Если набранная последовательность в Excel не является ни числом, ни формулой, она считается текстом и не подвергается преобразованиям. Например, последовательность 365.5 является числом, а 365,5 — текстом. Последовательность =A2+A4 — формула, а A2+A4 — текст. Если текст превысит по длине видимую ширину столбца, и если следующие ячейки в строке пусты, то тогда визуально текст накроет эти ячейки, а если в следующих ячейках есть данные, правая часть текста скроется за ячейками.

Создание диаграммы. В Excel предусмотрены различного типа диаграммы и функции для работы с диаграммами. Можно воспользоваться также панелью инструментов **Диаграмма**, в которой «кнопочно» представлены наиболее важные команды. Они доступны пользователю после активизации листа диаграммы в рабочей книге или выделения диаграммы, расположенной в рабочем листе. Диаграмму можно создать на отдельном листе или как графический объект в рабочем листе и вывести на печать вместе с таблицей. Такая диаграмма будет сохраняться в документе вместе с другими элементами рабочей книги. Диаграмма, созданная на отдельном листе, может быть сохранена в виде отдельного документа. Рассмотрим диаграмму, созданную на основе данных по «Заболеваемости гепатитом «В» в городе N». В этой диаграмме основу ряда данных составляют показатели с 1996 по 2000 г. Создаем таблицу вида

Ситуация по вирусному гепатиту «В»

	1996	1997	1998	1999	2000
Всего больных гепатитом «В»	677	439	368	211	177
из них наркоманов	37	43	65	85	25

По данным этой таблицы составим диаграмму:

- 1) выделим диапазон A3—G5, данные из которого должны быть представлены в диаграмме; выделенная область должна содержать ячейки с метками строк и столбцов, используемые в качестве меток оси и для именования диаграммы;
- 2) активизируем Мастер диаграмм, нажав кнопку на панели инструментов «Стандартная»;
- 3) создаем диаграмму с его помощью:
 - в диалоговом окне выбираем тип диаграммы («Гистограмма») и нажимаем Далее;
 - открывается второе окно Мастера диаграмм, в котором задаем интервалы ячеек, данные из которых представляются диаграм-



Рис. 42. Столбиковая пространственная Excel-диаграмма (гистограмма)

мой. Так как интервалы выделены, то значения в поле **Диапазон** будут внесены автоматически, нажимаем **Далее**;

- в окне **Параметры диаграммы** прием текущие параметры Excel (**Далее**);
- вводим с помощью **Далее** и **Легенда** названия столбиков;
- в следующем окне из предлагаемых вариантов активизируем кнопку **Готово**.

Результат построения приведен на рисунке 42.

Электронные таблицы — мощный и удобный инструмент для составления различного рода отчетов, сводок, таблиц и т. д. Кроме повышения производительности труда, электронные таблицы помогают хранить в электронной форме и модифицировать данные и таблицы.

ПРИМЕР

Приведем пример бухгалтерского документа, подготовленного в среде Excel. Этот документ был создан (заполнена таблица, подсчитаны все строки и столбцы таблицы) в Excel, а затем документ был вставлен (импортирован) в книгу средствами Word. Приведенный пример достаточно прост и нагляден, на практике же электронная среда Excel используется эффективно для таблиц более сложного вида. Электронные таблицы помогают модифицировать данные таблицы динамически, соединять их с графиками и диаграммами, вести документооборот.

Номер строки	Показатель	Условное обозначение	Июль	Август	Изменение	
					абсолют.	в %
A	B	C	D	E	F	G
1	Валовой оборот, тыс. р.	ВО	1333	1418	85	106,4
2	Валовая продукция, тыс. р.	ВП	741	746	5	100,7
3	Товарная продукция, тыс. р.	ТП	767	736	-31	96
4	Отгруженная продукция, тыс. р.	ОП	765	740	-25	96,7
5	Реализованная продукция, тыс. р.	РП	763	765	2	100,3
6	Коэффициент соотношения валовой продукции и валового оборота (стр2: стр1)	ВП:ВО	0,556	0,526	-0,030	94,6

Номер строки	Показатель	Условное обозначение	Июль	Август	Изменение	
					абсолют.	в %
A	B	C	D	E	F	G
7	Коэффициент товарности (стр3: стр2)	ТП:ВП	1,035	0,987	-0,048	95,4
8	Коэффициент отгрузки (стр4: стр3)	ОП:ТП	0,997	1,005	0,008	100,7
9	Коэффициент реализации (стр5: стр4)	РП:ОП	0,997	1,034	0,036	103,7

В. Система управления базами данных (СУБД) Microsoft Access. В состав Microsoft Office входит СУБД Microsoft Access. Мы несколько отвлечемся и рассмотрим банки, базы данных и СУБД, так как эти объекты важны.

Банком данных называется совокупность наборов данных, связанных отношениями файлов, удовлетворяющих условиям:

- данные должны составлять единое целое, связанное некоторым классификационным критерием;
- данные и средства работы с ними должны быть актуализируемы определенным числом пользователей.

База данных (БД) — достаточно большие наборы структурированных сообщений (данных) некоторой предметной области, представленных на машинных носителях и имеющих общую и удобную структуру, организацию, средства поддержки и доступа для различных программ пользователей.

Данные, хранящиеся в БД, организуются с помощью файловой системы. Файлы могут быть различной структуры и назначения.

ПРИМЕР

Файл book.txt содержит материалы книги в текстовом виде, запись — строка текста (последовательность символов): имя файла — book, «добавка» txt (называемая расширением файла) указывает на тип данных в файле — текстовые данные; на внешнем носителе (диске) текст книги (записи файла) представляется в закодированном двоичном виде. Файл input.dat — файл с входными данными (например, числовыми) к некоторой программе, запись состоит из чисел (на внешнем носителе в двоичном виде).

Для построения БД выясняем общую структуру данных, описываем ее на логическом уровне: формально, математически описываем все типы данных, типы их структур. Затем переводим логические описания в физическое представление в соответствии с выбранным способом и технологией хранения данных базы.

ПРИМЕР

Спроектируем БД о пенсионерах пенсионного фонда. Для разработки БД необходимо определить структуру данных базы (от качества выполнения этой работы зависит скорость обработки данных, типы и формы запросов к базе данных и др.). Для каждого пенсионера будем хранить следующую информацию:

- 1) Регистрационный номер пенсионера (не более 7 цифр).
- 2) Ф. И. О. пенсионера (не более 30 символов).
- 3) Домашний адрес пенсионера (не более 30 символов).
- 4) Возраст пенсионера (3 цифры).
- 5) Дата начисления пенсии (8 символов).
- 6) Текущий размер пенсии (не более 4 цифр).
- 7) Специальные отметки (не более 20 символов).

Запись базы будет выглядеть, например, следующим образом:
976 Иванов Николай Федорович Горького 373—42 67 12.04.06 2560 ВОВ.

БД могут быть реализованы различным образом:

- таблично — в виде таблиц, которые содержат данные по всем записям; их можно просматривать, прокручивать на экране;
- странично — в виде последовательно листаемых страниц, где каждая страница содержит, как и в картотеках, данные об одной только записи;
- смешанное, таблично-страничное представление.

ПРИМЕР

Табличная форма реализации БД пенсионеров:

Номер	Ф.И.О.	Дом. адрес	Возраст	Дата	Размер пенсии	Спец. отм.
1	Абрамова Надежда Львовна	Ленина 5-4	62	17.09.93	367.5	ВОВ 57
2	Авалов Игорь Петрович	Речная 23А	78	23.06.87	245.0	ПП 1
...	...	...	...	...	...	...
36000	Яровая Елена Степановна	Колхозная 45	91	30.12.72	345.0	НС 1

Страничная форма реализации БД пенсионеров:

36000 Яровая Колхозная 45 91 30.12.72 345.0 НС							
Елена 1							
2 Авалов Речная 23А 78 23.06.87 245.0 ПП							
1							
Абрамова Ленина 5-4 62 17.09.93 367.5 ВОВ							
Надежда 57							
Львовна							

В любой БД задается порядок на множестве записей, например ключевыми полями, содержимое которых нумеруемо, лексикографически упорядочено (т. е. упорядочено по символам алфавита). Таких полей может быть несколько, и при сортировке (выборке, модификации) данных записи ищут сначала по одному ключу, затем — по другому и т. д., пока не будет совпадения или несовпадения требуемых полей. Остальные поля при этом не сравниваются. Такой процесс называется сортировкой или поиском, сравнением по ключу (ключам). Кроме поиска по ключу, можно искать и по значению, перебирая все записи БД. Этот процесс более длителен и часто требует построения вспомогательных индексных таблиц для хранения подходящих по поисковому значению записей (если такие есть).

В последнее время распространяется технология удаленных БД. Она базируется на коллективном доступе пользователей к информационным ресурсам, сосредоточенным на едином компьютере (хост-компьютере, host computer), в диалоговом режиме по сетям передачи данных. Информационными продуктами здесь выступают БД разных предметных областей, а также различные директории, рубрикаторы и другие данные, облегчающие поиск по БД. Информационные услуги предоставляются благодаря разнообразным средствам поиска, обработки и выдачи информации. Информационные продукты и программные средства являются главными элементами БД или автоматизированных банков данных (АБД) — основной организационной формы, в которой развиваются современные технологии коммерческого распространения информации.

Основными особенностями данной технологии, определяющими ее достоинства и недостатки, являются:

- предоставление пользователю только информационных услуг, а не непосредственно информационных продуктов, в результате чего он получает (оплачивает) только нужную информацию;
- полнота информации, связанная с загрузкой на мощные хост-компьютеры больших массивов данных;
- высокая скорость обновления, модификации и перемещения информации;
- развитое ПО, позволяющее не только находить и получать информацию, но и при необходимости осуществлять ее графическую и другую обработку.

- Интерактивные услуги АБД могут предоставляться в режимах:
- локальном, когда работа пользователя осуществляется с терминала, подключенного к хост-компьютеру;
 - удаленном, когда работа пользователя осуществляется с физически удаленного от хост-компьютера терминала по сетям связи.

СУБД (DBMS — DataBase Management System), или система управления базами данных, — это программная система, обеспечивающая общение (интерфейс) программ пользователя и данных из БД. СУБД должна иметь средства, позволяющие сформулировать запрос к БД (поиск, сортировка и т. д.) на языке, близком естественному и понятном пользователю, но формальном, реализованном на ЭВМ. Такие языки называются **языками запросов к базам данных** и языками непроцедурного типа (на них формулируют «что делать», а не «как делать», как на процедурных языках).

Основные функции СУБД:

- управление данными во внешней памяти;
- управление буферными областями памяти;
- управление транзакциями, последовательностями операций над БД, рассматриваемыми СУБД как одна макрооперация;
- поддержание надежности хранения данных в БД и др.

Итак, вернемся, наконец к Microsoft Access. Это мощная СУБД, доступная любому пользователю и позволяющая быстро и эффективно обрабатывать большие массивы информации. В Access можно создавать приложения исключительно встроенными средствами Access, что привлекательно для неспециалистов. Если нужно организовать управление БД сложным, нестандартным образом, то для этих целей имеется язык программирования Visual Basic Application. Можно организовывать сложные связи таблиц без дублирования информации в них. Например, для создания базы данных необходимо выбрать соответствующий пункт в меню или воспользоваться 20—30 шаблонами, например:

- адресная книга — создает базу данных типа адресной книги;
- библиотека — создает базу данных типа библиотеки;
- контакты — создают базу данных типа контактных связей;
- каталог видеозаписей и др.

После создания БД можно воспользоваться мастерами:

- анализ таблиц — анализ элементов базы (таблиц) с целью повышения эффективности структуры базы данных путем минимизации количества повторяющихся данных;
- импорт — анализ данных при импорте текста, данных из Excel в Access;
- экспорт — анализ данных при экспорте данных из Access в Excel;
- запрос (выборки) — автоматическое создание запроса к базе из Access;
- защита данных — копирование и шифрование базы, установки и снятия защиты на доступ;

Microsoft Access [Журнал учета приема больных - таблица]

Файл Вид Запросы Мастер Таблицы Сервис Справка

№ № Серв Серв

№	Фамилия	Имя	Паспорт	Страховой полис	Дата рождения	Место работы
1	Осипова	Иван	М-ВЕ №457689	сер.003 №056680	12.12.52	ЖЭК №4, мастер
2	Перова	Елена	М-ГН №326790	сер.001 №779211	31.08.67	с/школа №105, преподаватель
3	Львова	Татьяна	87-93 №590341	сер.004 №566333	23.03.70	Химзавод, бухгалтер
4	Иванов	Евгений	У-РП №645391	сер.001 №570426	01.12.59	МГУ, лаборант
5	Смирнов	Владислав	Х-МК №732456	сер.004 №803467	15.04.65	ОАО "Телеком", мастер

Рис. 43. Основное рабочее окно Access

- диспетчер таблиц — переключение с одной базы на другую (изменения связей таблиц);
- диаграммы — создание диаграмм;
- создание отчета — формирование отчета или формы по запросу и др.

Запускать и завершать работу MS Access можно любым из стандартных способов, предусмотренных в среде Windows.

Рабочее окно MS Access такое же, как и в Word, Excel (рис. 43).

Создание базы данных типа таблицы. Базу данных (например, учет больных) удобнее создавать в режиме таблицы с помощью Мастера таблиц. Он предложит несколько эталонов таблиц, из которых выбираем подходящую таблицу с нужными полями. Для создания базы данных необходимо:

- 1) Выполнить **Файл, Создать**.
- 2) В окне **Создание** активизировать вкладку **Общие**, выбрать вариант **Новая база данных** и **ОК**.
- 3) В окне **Файл** новой базы данных в разделе **Имя файла** ввести имя файла будущей базы (расширение **mdb**) и, если необходимо, установить директорию, в которой будет находиться файл; затем **Создать**.
- 4) В окне **База данных** активизировать **Таблица** и **Создать**.
- 5) В окне **Новая таблица** выбрать вариант **Мастер таблиц** и **ОК**.
- 6) В первом окне Мастера таблиц в разделе **Образцы таблицы** выбрать подходящий образец (например, вариант **Сотрудники**), при этом в разделе **Образцы полей** появятся имена полей; щелкая на кнопке «>», перенести нужные имена полей в раздел **Поля новой таблицы** (например, поля **Фамилия**, **Дата оформления**, **Паспорт**, **Страховой полис**, **Дата рождения**, **Место работы**, **Место жительства**, **Примечания**); затем **Далее**.
- 7) Во втором окне Мастера таблиц ввести имя будущей таблицы (оно обязательно должно совпадать с именем самой базы данных) и **Далее**.
- 8) В третьем окне Мастера таблиц выбрать режим **Непосредственный ввод данных в таблицу** (т. е. после создания таблицы мы начнем вводить в нее данные) и **Готово**.

Теперь можно начать ввод данных в ячейки базы-таблицы. Перемещение между ячейками базы производится с помощью клавиш управления курсором или **Tab**. Ввод данных в ячейку завершается

№	Фамилия	Имя	Паспорт	Страховой полис	Дата рождения	Место работы
1	Сидоров	Иван	М-ВЕ №457689	сер.003 №066688	12.12.52	ЖЭК №4, мастер
2	Порова	Елена	И-ГН №326790	сер.001 №779211	31.08.67	с/школа №105, преподаватель
3	Львова	Татьяна	67-93 №590341	сер.004 №565333	23.03.70	Минздрав, санитар
4	Иванов	Борис	У-РП №645391	сер.001 №670426	01.12.58	МГУ, лаборант
5	Смирнов	Вячеслав	Х-МК №732456	сер.004 №003467	15.04.65	ОАО "Телеком", мастер

Место жительства	Кем доставлен	Предварительный диагноз	Примечания
г.Москва, ул.Семеновская 72-90	врачом скорой помощи	I стадия Гипертонический криз	
г.Москва, ул.Уссурийская 15-20	по направлению	Пилонефрит	
г.Митищи, пер.Садовый 23-69	по направлению	Бронхиальная астма	
г.Шелковс, пл.Серова 99-76	врачом скорой помощи	Хронический панкреатит	
г.Москва, ул.Черная 65-74	по направлению	Холецистит	

Рис. 44. База данных «Журнал учета поступления больных»

при нажатии **Enter** или перемещении в другую ячейку. Если содержимое ячейки введено неверно, следует щелкнуть в ячейке, отредактировать запись и нажать **Enter**. Если текст не уместается целиком в поле-ячейке, следует переместить мышью границу ячейки заголовка таблицы, содержащей название этого поля, расширив тем самым весь столбец. Сохранить таблицу в файле следует с помощью **Файл, Сохранить**.

На рисунке 44 показана база данных «Журнал учета поступления больных».

Редактирование структуры базы данных (в режиме таблицы). До редактирования структуры таблицы-базы следует переключиться в режим конструктора для внесения изменений в структуру командами **Вид, Конструктор таблиц**. Возвращение в режим таблицы осуществляется командами **Вид, Режим таблицы**. Окно режима конструктора состоит из двух частей. Вверху — имена полей базы данных и типы данных этих полей (текстовый, денежный и т. п.), внизу — параметры текущего поля (того, в котором находится курсор). Существуют следующие варианты редактирования структуры базы данных.

Изменение имени поля. Щелкнуть на нужном имени поля в верхней части окна конструктора, изменить имя и нажать **Enter**.

Изменение типа данных. Щелкнуть в разделе **Тип** верхней части окна конструктора против нужного имени поля. Раскрыть, щелкнув на кнопке с направленной вниз стрелкой, список типов данных и выбрать нужный тип.

Изменение шаблона данных. Щелкнуть на нужном имени поля в верхней части окна конструктора. Щелкнуть в нижней части окна в разделе **Маска ввода** и изменить шаблон данных.

Добавление нового поля справа. Щелкнуть в строке, расположенной непосредственно под последним именем поля в списке имен полей в верхней части окна конструктора, ввести имя нового поля, тип данных этого поля и, если нужно, внизу шаблон данных.

Добавление нового поля внутри базы. Выделить в списке имен полей строку с именем поля, слева от которого надо вставить новое

№	2	КемДоставл	по направлению
Фамилия	Перова	Предварител	Пиелонефрит
Имя	Елена	Примечания	
Паспорт	III-ГН №326790		
СтраховойПс	сер.001 %779211		
Дата рожден	31.08.67		
МестоРаботы	с/школа №105,препо		
Место Жител	г.Москва, ул. Уссурийская		

Рис. 45. Карточка страничной базы данных

поле (для этого следует щелкнуть мышью слева от имени поля). Выполнить **Вставка, Поле**. Щелкнуть в появившейся пустой строке, ввести имя нового поля, тип данных и, если нужно, шаблон данных.

Представление базы данных в режиме страницы (формы). Иногда удобнее вводить новые записи в базу, когда она представлена в виде отдельных карточек. Это страничный режим представления базы (режим формы). Чтобы представить нашу таблицу-базу в страничном (рис. 45) режиме, следует:

- 1) Командами **Файл, Открыть** активизировать окно База данных с базой.
- 2) В окне База данных активизировать вкладку **Форма и Создать**.
- 3) В окне Новая форма выбрать вариант **Мастер форм**; из списка **Выберите в качестве источника**, щелкнув на кнопке со стрелкой, направленной вниз, щелчком выбрать имя созданной нами таблицы; щелкнуть **ОК**.
- 4) В первом окне **Мастера форм** с помощью кнопки «>>» добавить в картотеку все поля таблицы-базы или с помощью кнопки «>» — нужные поля (форма может содержать не все поля таблицы, а лишь некоторые из них) и **Далее**.
- 5) Во втором окне **Мастера форм** выбрать **В один столбец** и **Далее**.
- 6) В третьем окне **Мастера форм** выбрать нужный вариант оформления карточки и **Далее**.
- 7) В четвертом окне **Мастера форм** ввести имя будущей формы (необязательно совпадает с именем самой базы и с именем таблицы) и **Готово**.

Сортировка данных (в режиме таблицы). Сначала следует выделить поле, по которому надо произвести сортировку данных (т. е. надо выделить весь столбец нашей таблицы), для чего достаточно щелкнуть на имени поля в заголовке таблицы. Затем нужно щелкнуть на одной из двух кнопок алфавитного упорядочивания текстовых данных: **А→Я** или **Я→А**.

Фильтрация данных (в режиме таблицы). Иногда необходимо видеть на экране не все записи базы данных (строки), а те из них,

которые отвечают каким-то условиям (фильтрации, селекции, выборки). Чтобы осуществить выборку необходимых данных, следует:

- 1) Выполнить **Записи, Фильтр, Изменить фильтр**.
- 2) Щелкнуть на имени требуемого поля в списке и ввести условие (маску) выбора.
- 3) Выполнить **Фильтр, Применить фильтр**. На экране появится таблица, которая будет содержать строки, данные которых отвечают указанному условию. Чтобы вновь увидеть всю таблицу целиком, следует выполнить **Записи, Удалить фильтр**. Используют равенства и неравенства в условиях.

Создание отчета. Перед выводом базы данных на печать следует установить, в каком виде (таблица, страница) она должна быть представлена на бумаге, какие в ней должны присутствовать поля и записи. Это процесс создания отчета. Для создания отчета следует:

- 1) В окне **База данных** активизировать вкладку **Отчет и Создать**.
- 2) В окне **Новый отчет** выбрать вариант **Мастер отчетов**; из списка **Выберите в качестве источника**, щелкнув на кнопке со стрелкой, направленной вниз, выбрать имя созданной нами таблицы и **ОК**.
- 3) В первом окне **Мастера отчетов** с помощью кнопки «>» выбрать имена полей из списка, которые следует включить в будущий отчет, и **Далее**.
- 4) Во втором окне **Мастера отчетов** произвести, если необходимо, группировку полей и **Далее**.
- 5) В третьем окне **Мастера отчетов**, щелкнув на кнопке со стрелкой, направленной вниз, выбрать щелчком имя поля, по которому следует провести сортировку данных (если нужно), и **Далее**.
- 6) В четвертом окне **Мастера отчетов** выбрать подходящий вариант оформления отчета и ориентацию листа бумаги в принтере (книжная — строки печатаются вдоль узкой стороны листа, альбомная — вдоль широкой стороны) и **Далее**.
- 7) В пятом окне **Мастера отчетов** выбрать внешний вид и **Далее**.
- 8) В шестом окне **Мастера отчетов** ввести заголовок и **Готово**.

Созданный отчет можно напечатать: **Файл, Печать**. Если щелкнуть на кнопке **W**, расположенной на Панели инструментов, отчет будет перенесен в MS Word, где его можно дополнительно редактировать средствами MS Word.

Г. Система презентаций PowerPoint. В MS Office есть пакет PowerPoint, позволяющий создавать электронные презентации (слайд-шоу), красочные пленки для проектора, раздаточные печатные материалы. С помощью PowerPoint готовят презентации, выполняющиеся автоматически.

Презентация — слайд-фильм, в котором можно использовать текстовую и графическую, звуковую и видеoinформацию, готовые варианты дизайна. Созданная презентация может быть распечатана на бумаге, подключена к другой презентации, выведена на экран ЭВМ или с помощью компьютерного проектора на электронную доску аудитории и др.

Для создания презентации возможны действия:

- создание слайдов презентации, ввод и редактирование текста, изменение порядка следования слайдов, цвета и фона графических объектов;
- создание сценария презентации;
- создание и модификация объектов презентации, их анимация;
- подготовка раздаточных материалов;
- определение и формирование интервалов между слайдами;
- демонстрация презентации.

Запуск программы. Выбрать команды **Пуск, Программы, Microsoft PowerPoint**. Появляется окно (рис. 46) и возможности выбора: **Мастер автосодержания; Шаблон презентаций; Пустая презентация**. **Мастер автосодержания** помогает быстро создать презентацию. Обычная информация на титульном листе презентации: имя, тема, пояснение и др.

Тип создаваемой презентации выбирается из вариантов:

- 1) **Рекомендация стратегии** (цели, характер, возможности, рекомендации).
- 2) **Продажа продукции, услуги или идеи** (преимущества перед аналогами).
- 3) **Обучение** (метод организации процесса обучения).
- 4) **Отчет о достижениях** (оценка работы, текущее состояние, его развитие).
- 5) **Сообщение о плохих новостях** (что, почему, кто виноват, как исправить).
- 6) **Стандартный** (стандартная презентация для доклада).

Если нужной темы нет в этом списке, то, щелкнув мышью на кнопке **Другой**, получим список, в котором, кроме уже перечисленных, будут строки: **Главная часть; Лучшая десятка; Общее собрание; Ориентировка персонала; Отчет о положении дел; План продаж; План работы; Предлагаем стратегию**.

Шаблоны презентаций. Они используются тогда, когда тема презентации достаточно стандартна и для нее уже существует гото-

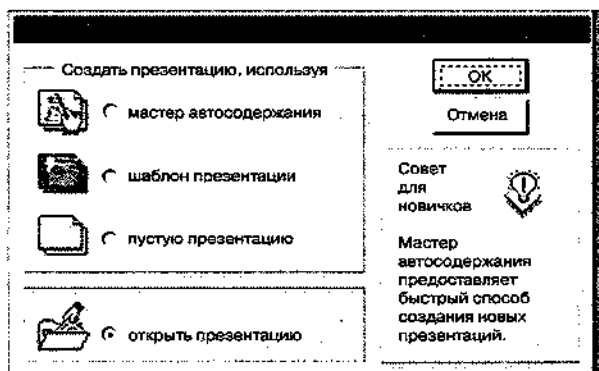


Рис. 46. Основное рабочее окно PowerPoint

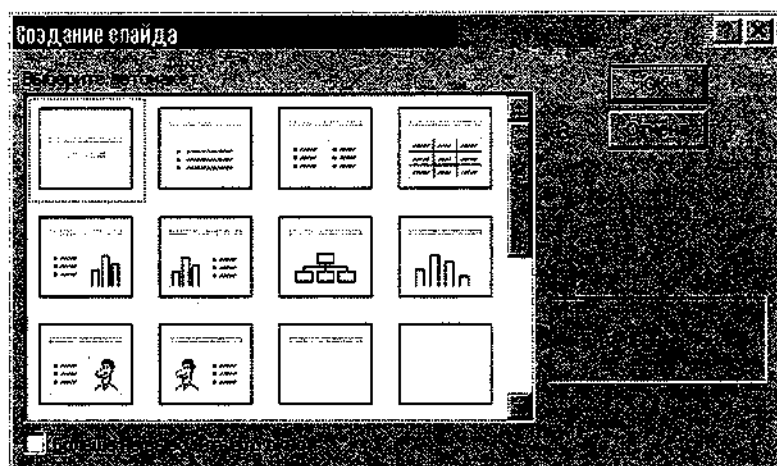


Рис. 47



Рис. 48

вый шаблон, а время создания презентации ограничено. Для выбора шаблона презентации воспользуйтесь командой **Файл, Создать** и в появившемся окне **Создать презентацию** активизируйте вкладку **Презентации** (рис. 47).

Элементы окна. В нижней части окна PowerPoint расположена строка состояния с номером слайда на экране и именем шаблона (рис. 48).

Остальные элементы окна программы аналогичны строке заголовка, панелям инструментов **Стандартная** и **Форматирование**, полосам прокрутки в других программах MS Office.

Два первых и последний разделы меню (**Файл**, **Правка** и **Окно**) практически не отличаются от Word, Excel. Раздел **Вид** — для выборки того, что мы хотим увидеть на экране:

- 1) **Слайды** — крупное изображение слайда, в этом режиме слайд редактируем.
- 2) **Структура презентации** — ее план, отдельные слайды сведены в один текст.
- 3) **Сортировщик слайдов** — на экране, насколько хватает на нем места, помещаются уменьшенные изображения всех слайдов данной презентации.
- 4) **Страницы заметок** — отдельная страница документа, в верхней половине которой расположен рисунок слайда, а в нижней есть место для заметок.
- 5) **Демонстрация** — запуск презентации, главный итог всей работы.

Четвертый раздел **Вставка** — для внесения дополнений в презентацию:

- 1) **Создать слайд** — выбрать его внешний вид из предлагаемого набора.
- 2) **Дата и время** — указать на слайде дату и время его создания.
- 3) **Номер слайда** — поместить на слайде его номер в презентации.
- 4) **Слайд из файла** — вставить созданный ранее слайд из файлов.
- 5) **Слайд из структуры** — вставить фрагмент документа.
- 6) **Графика** — добавляет готовую иллюстрацию.
- 7) **Рисунок** — вставляет созданный ранее рисунок из файла.
- 8) **Кино** — включение анимационного файла (видеоролика).
- 9) **Звук** — из звукового файла.
- 10) **Microsoft Graph** — добавляет графики и диаграммы.
- 11) **Таблица Microsoft Word** — включение таблиц.

Раздел **Формат** отвечает за общее оформление слайда, размеры и форму надписей, цветовую гамму. Работа с надписями происходит так же, как и во всех других программах: надо выделить текст и указать, какие внести изменения (размер букв, их цвет и т. д.). Можно изменить весь дизайн презентации (пункт меню **Применить шаблон дизайна**).

В разделе **Сервис** есть традиционные возможности: **Орфография**, **Замена шрифтов**, **Автозамена**, **Автографика** (автоматический подбор графических образов для презентации по ключевым словам из текста слайда) и специфические возможности PowerPoint для аудио- и видеовставок:

- 1) **Переход слайда** — задает эффекты для смены слайдов при демонстрации.

- 2) **Скрыть слайд** — для пропуска отмеченных слайдов при демонстрации.
- 3) **Построение текста слайда** — установка режима появления текста на экране (сбоку и т. п.).
- 4) **Настройка анимации** — установка, как вести себя объектам (тексты, графика) во время демонстрации: влетать, проявляться, изменять цвет, сопровождать свое появление звуком.
- 5) **Интерактивные настройки** — установка, как должны вести себя объекты при щелчке на них мышью во время демонстрации (перейти на другой слайд, воспроизвести звук и т. д.).

Создание презентации урока «Системы счисления» (рис. 48). Выбираем **Шаблон презентаций**. Начнем с того, что определимся с внешним оформлением слайдов — это подраздел **Дизайн презентации**. Предлагаемые шаблоны можно просмотреть на уменьшенной картинке в правой части окна. Для этого достаточно один раз щелкнуть мышью на соответствующем значке. Выберем, например, стиль **Современный**. Выделим его в перечне и щелкнем **ОК**. Далее выберем расположение надписей на титульном листе.

Выделим первый вариант и **ОК**. На нем две надписи: «Щелчок добавляет заголовок» и «Щелчок добавляет подзаголовок». Щелкнем мышью на первой фразе и вместо нее запишем: «Тема урока». Вместо второй надписи наберем тему урока «Системы счисления». Выделим последнее предложение, увеличим шрифт и добавим «жирности». При желании предложение можно переместить. Приступим к созданию следующего слайда. Найдем кнопку **Создать слайд** и щелкнем по ней мышью. Из предложенной в ответ **Авторазметки** возьмем второй слайд и запишем определение системы счисления, увеличим шрифт. Слайд 3 создаем аналогично слайду 2, в нем запишем наиболее используемые в информатике системы счисления и примеры перевода в десятичную систему. Слайд 4 содержит процедуру перевода десятичных чисел в r -ю систему счисления и создается так же, как и предыдущие. Слайд 5 — это пример перевода из десятичной системы в двоичную. Так как пример деления непосредственно в PowerPoint мы набрать не сможем, можно воспользоваться Word (вставляем пример в PowerPoint следующим приемом: скопируем пример в буфер обмена, перейдем в PowerPoint и воспользуемся **Правка, Вставка**). Сдвинуть включенный текст в пределах слайда можно следующим образом: поместить курсор мыши «внутри» картинки, нажать кнопку мыши и, держа ее нажатой, переместить текст. Потянув мышью по углам и в середине, растягиваем текст. Слайд 6 содержит таблицу (набран в текстовом редакторе). Несколько слайдов с номерами 2 (а), 5 (б), 6 (в) готовой презентации показаны на рисунке 49.

Организуем переход от слайда к слайду с помощью раздела **Сервис**. Вызовем всю презентацию (**Вид — Сортировщик слайдов**). Отметим щелчком слайд 1. Теперь можно задать эффекты, сопровождающие его появление на экране (**Сервис — Переход слайда**). Выберем подходящее и **ОК**. На экране под уменьшенным изображе-

ОПРЕДЕЛЕНИЕ

Алфавит X из p символов и правила записи (изображения) чисел с помощью символов этого алфавита называется *системой счисления* с основанием p . Все системы счисления строятся по общему принципу: выбирается p – символ основания системы, а любое число x записывается в виде комбинации степеней p от 0-й до $(p - 1)$ -й степени, т. е.:

$$x = x_p p^d + x_{p-1} p^{d-1} + \dots + x_1 p^1 + x_0 p^0$$

Для перевода из 2-й системы счисления в 8-ю и наоборот, из 2-й в 16-ю и наоборот, из 8-й в 16-ю и наоборот используется таблица вида:

Основание системы			
10	2	8	16
0	0	000	0000
1	1	001	0001
2		010	0010
3		011	0011
4		100	0100
5		101	0101
6		110	0110
7		111	0111
8			1000
9			1001
10			1010
11			1011
12			1100
13			1101
14			1110
15			1111

ПРИМЕР

Пример перевода из десятичной системы счисления в двоичную числа $13,4_{10} = ?_2$:

1) переводим сначала целую часть числа $13,4_{10}$:

$$\begin{array}{r} 13 \overline{) 2} \\ 12 \quad 6 \overline{) 2} \\ 1 \quad 6 \quad 3 \overline{) 2} \\ 0 \quad 2 \quad 1 \\ 1 \end{array} \qquad 13_{10} = 1101_2$$

2) переводим мантиссу числа 0,4:

$$0,4 \cdot 2 = 0,8; 0,8 \cdot 2 = 1,6; 0,6 \cdot 2 = 1,2; 0,2 \cdot 2 = 0,4; 0,4_{10} = 0,0110..._2$$

3) записываем результат перевода: $13,4_{10} = 1101,0110..._2$

Рис. 49. Фрагменты презентации урока «Системы счисления»

нием появился значок с загнутой стрелкой. Если щелкнуть на нем, то увидим этот слайд при демонстрации. Во время репетиции (**Вид — Демонстрация**) слайды разворачиваются на весь экран, их переходы сопровождаются заданными эффектами. Если репетиция удачна, то измеренные интервалы времени можно сохранить для запуска, затем для презентации в режиме **Автоматическая смена кадров**. При ручном режиме для перехода от одного слайда к другому надо будет щелкать левой клавишей мыши. Во время демонстрации достаточно немного провести мышью, и в левом нижнем углу экрана возникает кнопка, щелчок на ней мышью вызывает выпадающее меню (это меню можно вызвать в любом месте экрана правой клавишей мыши): **Вперед**; **Назад**; **Перейти** — переход к заданному слайду; **Записная книжка** — вызов записной книжки; **Хронометр**; **Стрелка**; **Перо** — карандаш, которым можно во время демонстрации что-то пометить на слайде или подчеркнуть текст (нажата левая клавиша мыши); **Указатель** — убирает с экрана карандаш или его цвет; **Экран** — очистка экрана от пометок карандашом; **Завершить демонстрацию**.

При создании слайдов можно изменять цвет, форматы и образцы шаблонов, которые определяют размещение текстовых и графических объектов, шрифты, фон слайдов и др.

Можно вставлять в презентацию тексты, таблицы, рисунки, графики и диаграммы, аудиофайлы и видеоролики.

Электронная презентация демонстрируется на мониторе компьютера или проецируется на внешний экран электронным способом. Можно распечатать слайды (все или часть их) презентации и раздать участникам.

Более совершенные версии Microsoft Office-97, Microsoft Office-2000, кроме приведенных стандартных приложений, имеют и следующие приложения:

- Office Assistant — помощник для подсказок;
- HTML- и web-поддержку (Internet Assistants);
- различные программы-помощники (Graph — графическое представление данных, Organization Chart — создатель штатного расписания офиса, Equation Editor — редактор формул, WordArt — создатель логотипов, заголовков, ClipArt Gallery — библиотека рисунков и др.).

В состав Microsoft Office-2000 и ее дальнейшие модификации входят возможности одновременного показа презентаций по локальной сети, использование возможностей системы распознавания речи, визуальные среды разработки различных офисных приложений (например, заполнения платежных поручений), сайты рабочих групп, системы визуализации данных, система сканирования и ввода данных и др.

19.2. БАЗЫ ЗНАНИЙ И ЭКСПЕРТНЫЕ СИСТЕМЫ

База знаний — специальным образом формализованные и представленные в ЭВМ знания (модели) предметной области (предметных областей), которые можно корректировать, обобщать, сужать и расширять и, что наиболее важно, извлекать на их основе новые знания (или моделировать рассуждения по заложенным, формализованным правилам, например правилам логики).

Хотя база знаний имеет много похожего на БД, она, в отличие от БД, имеет более сложную логическую структуру (и сами знания сложнее данных) и динамическую структуру (переплетение знаний более изменчиво, чем переплетение данных); база знаний при каждом обращении к ней может изменять свои знания, в отличие от обращений к базе данных, которые часто используют данные базы пассивно, например, только извлекая и пересылая их.

Экспертная система — компьютерная система на основе базы знаний экспертов-профессионалов в некоторой предметной области, которая в удобной форме (интерфейс системы похож на интерфейс общения с профессионалом в этой предметной области) может консультировать, подсказывать, имитировать интеллектуальное поведение в различных, достаточно сложных ситуациях.

Экспертная система, кроме знаний, может использовать умения и навыки, опыт и интуицию экспертов (все, что можно формализовать и представить в ЭВМ). Знания в экспертной системе представляются в виде фактов, характерных для рассматриваемой предметной области, процедур и правил соотнесения фактов, их применений и знаний о том, когда и как их применять.

Основные функции экспертной системы:

- 1) накопление, получение знаний в предметной области системы и их актуализация при решении конкретных задач;
- 2) обучение (объяснение, тренировка) по способам получения результатов решения задач и оценки этих решений;
- 3) интерактивное общение, адаптируемое к уровню знаний пользователя, профессиональное общение с пользователем при решении задач;
- 4) запоминание и фиксация эффективных способов решения и результатов решения задач и их использование в новых ситуациях.

Пользователь на языке запросов (близком к профессиональному языку, но тем не менее формальном) сообщает условие задачи, которое затем грамматически анализируется. Затем, используя заложенные в базу знаний (и приобретенные при решении аналогичных задач) знания о методах решения задач, корректное описание задачи преобразуется в достаточную для решения данной задачи цепочку (процедуру) исполняемых модулей системы. Алгоритм, соответствующий этой процедуре, выполняется управляющей программой (монитором) системы, результат преобразуется в профессионально привычную форму (таблицы, графики и т. д.) и выдается пользователю вместе с пояснениями, необходимыми для принятия решений.

Обычно в экспертную систему включаются подсистемы:

- 1) **База знаний** отвечает за модели знаний.
- 2) **Интерфейсная система** отвечает за обеспечение диалога с пользователем в удобной, комфортной и интеллектуальной форме, часто визуальной в виде графиков, схем, рисунков и т. д.
- 3) **Система логического вывода** отвечает за нахождение структуры, цепочки формализованных правил и знаний из базы знаний, наиболее подходящей, адекватной запросу к системе, оценивает причинно-следственные связи и обращается к базе знаний до тех пор, пока не найдет приемлемое решение.
- 4) **Система объяснения** отвечает за выдачу пользователю объяснений по поводу того, как и почему была выдана такая консультация, такие рекомендации и советы; на основе этих объяснений пользователь вправе отказаться или принять данные рекомендации, например в медицинских системах.

Структура экспертной системы изображена на рисунке 50.

Экспертная система может рассуждать (имитировать рассуждения) и настраиваться на предметную область.

ПРИМЕР

Рассмотрим базу знаний и экспертную систему «Социально-экологическая система», которая построена в 1995 г. и позволяет оценивать (в том числе качественно) социально-экологическое состояние некоторой среды по задаваемым пользователем (экспертом) количественным оценкам тех или иных

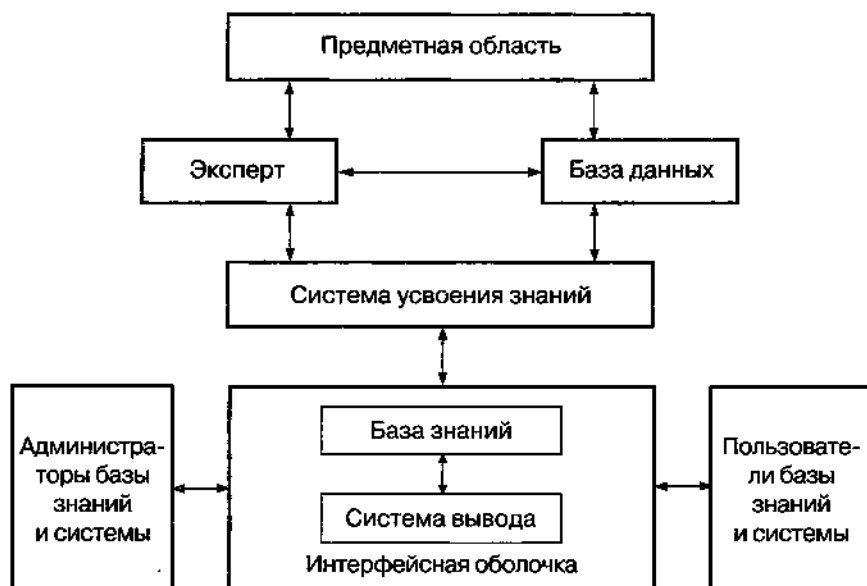


Рис. 50. Структура экспертной системы

параметров среды (выбираемых из базы знаний системы). Для каждого входного фактора в диалоговом режиме задаются относительные (от 0 до 1) оценки влияния этого фактора (вес фактора). После анализа этих данных (этой экологической обстановки) система принимает на основе базы знаний решение о состоянии социоэкономико-экологической среды, используя количественную оценку (от 0 до 1) и десятибалльную (0—9) качественную систему оценок. Фрагмент работы экспертной системы приведен на рисунке 51.

Протокол диалога (23.02.1998 — Понедельник, 11:23:37)

Входные данные

1. Контроль над эрозией: 0.6
2. Сооружения для отдыха: 0.1
3. Иригация: 0.9
4. Сжигание отходов: 1.0
5. Строительство мостов и дорог: 0.6
6. Искусственные каналы: 0.5
7. Плотины: 0.3
8. Туннели и подземные сооружения: 0.9
9. Взрывные и буровые работы: 0.45667
10. Открытая разработка: 0.567
11. Вырубка лесов: 0.345
12. Охота и рыболовство: 0.234
13. Растениеводство: 0.678

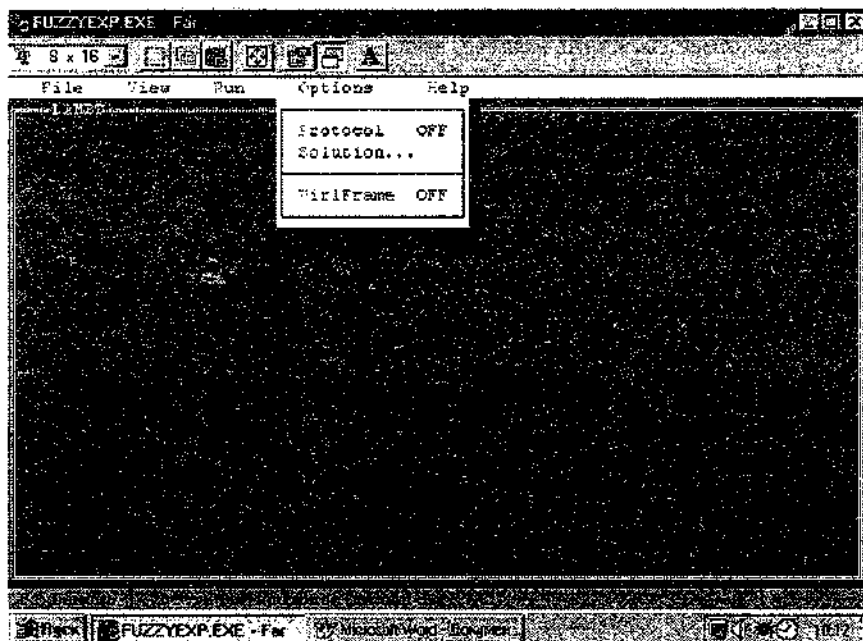


Рис. 51. Принятие решения и запись отчета (протокола) в файл

14. Скотоводство: 0.648
15. Химическая промышленность: 0.2456
16. Лесопосадки: 0.54846
17. Удобрения: 0.6
18. Регулирование диких животных: IGNORE (игнорируется фактор)
19. Автомобильное движение: 0.6
20. Трубопроводы: 0.0
21. Хранилища отходов: 0.0
22. Использование ядохимикатов: 0.2
23. Течи и разливы: 0.0

Принятие решения о социоэкономико-экологической обстановке

1. Состояние почвы: 0.55177 (слабое, положительное)
2. Состояние поверхностных вод: 0.52969 (слабое, положительное)
3. Качественный состав вод: 0.62299 (некоторое, положительное)
4. Качественный состав воздуха: 0.61298 (некоторое, положительное)
5. Температура воздуха: 0.48449 (слабое, отрицательное)
6. Эрозия: 0.59051 (слабое, положительное)
7. Деревья и кустарники: 0.54160 (слабое, положительное)
8. Травы: 0.59051 (слабое, положительное)
9. Сельхозкультуры: 0.51698 (слабое, положительное)
10. Микрофлора: 0.48702 (слабое, отрицательное)
11. Животные суши: 0.59804 (слабое, положительное)

12. Рыбы и моллюски: 0.51525 (слабое, положительное)
13. Насекомые: 0.56000 (слабое, положительное)
14. Заболачивание территории: 0.50000 (слабое, положительное)
15. Курорты на суше: 0.52729 (слабое, положительное)
16. Парки и заповедники: 0.54668 (слабое, положительное)
17. Здоровье и безопасность: 0.62870 (некоторое, положительное)
18. Занятость людей: 0.51196 (слабое, положительное)
19. Плотность населения: 0.55539 (слабое, положительное)
20. Соленость воды: 0.48750 (слабое, отрицательное)
21. Солончаки: 0.57000 (слабое, положительное)
22. Заросли: 0.62935 (некоторое, положительное)
23. Оползни: 0.70588 (выраженное, положительное)

19.3. ЭЛЕКТРОННАЯ ПОЧТА И ТЕЛЕКОММУНИКАЦИОННЫЙ ДОСТУП

Электронная почта и телекоммуникационный доступ к удаленной от пользователя информации носителю информации, собеседнику — человеку или компьютеру (передача сообщения с помощью компьютера-отправителя и его прием с помощью компьютера-получателя; при этом сообщение преобразуется из цифровых кодов, например, в коды электромагнитных колебаний, передаваемых по телефонным каналам, а ЭВМ адресата производит обратное преобразование). Развитие сетей связи — виртуальные локальные вычислительные сети, объединяющие пользователей по профессиям.

Электронная почта — аналог классической почты по результату, но отличный по скорости, возможностям, удобству; эта почта пересылается от компьютера одного пользователя (отправителя почтового сообщения) к компьютеру другого (адресата) с помощью средства кодирования и передачи сообщений по каналам связи (телефонной, радиорелейной, космической и др.). Для того чтобы написать и отправить электронное письмо, необходимы:

- электронный почтовый адрес адресата;
- почтовая программа, например The Bat;
- компьютер отправителя, на котором он набирает текст сообщения, указывает адрес адресата и дает команду почтовой программе отправить почту;
- компьютер адресата с директориумом на диске — почтовый ящик адресата;
- канал связи с компьютером адресата;
- письмо — текст сообщения, набранное на компьютере отправителя.

Телеконференции объединяют средства почты и теледоступа.

Телеконференция — обмен сообщениями (докладами) между участниками (подписчиками) конференции, объявленной на специальной доске объявлений в сети, в частности на электронной доске объявлений BBS — Bulletin Board System. Телеконференция представляет собой технологию на базе программных средств интерак-

тивного доступа к ресурсам сети и предназначена для обсуждения какой-либо тематики. С их помощью можно проводить консалтинг, обучение, совещание, автоматизацию офиса и др. Базовая система видеоконференций включает:

- мощную рабочую мультимедийную станцию;
- видеокамеру и плату для сжатия видеoinформации;
- микрофон и видеоманитофон;
- средства сопряжения с используемой сетью.

Качество связи зависит от канала: для оптоволоконных линий оно выше, чем для стандартных линий (Ethernet). Имеется стандарт ISDN (Integrated Services Digital Network) для совместной передачи различных видов информации (видео, аудио, цифровая).

Телеконференции могут проводиться как в режиме обмена письмами по электронной почте (режим почтового подключения), так и в режиме терминального интерактивного подключения через телекоммуникационные сети. В режиме терминального подключения пользователь может иметь доступ (подписку) к системе телеконференций, но, в отличие от режима почтового подключения, можно подключаться к конференции непосредственно в сети с помощью специальных программ, управляющих телеконференциями. Эти программы позволяют выполнять следующие манипуляции: найти конференцию; зарегистрироваться; перейти в конференцию; послать отклик (доклад); получить отклик; закрыть подписку и др.

ПРИМЕР

Рассмотрим медицинские видеоконференции (убедительный и яркий социально-экономический пример использования телеконференции). В крупных больницах и клиниках сейчас имеется современное медицинское оборудование — томографы, эхокардиографы и др., а также достаточно высококвалифицированный медицинский персонал, с помощью которого в режиме видеодialoga (конференции) врачи из региональных (вплоть до районных) медицинских учреждений могут обсудить результаты диагностики больного, диагноз, методы и стратегии лечения. Проблема приближения этих средств и кадров особенно актуальна для нашей страны с ее большой территорией. Такие конференции могут использоваться для первичного консультирования дооперационных больных и уточнения предварительного диагноза, анализа обследования; послеоперационных консультаций; неотложных консультаций больных в критических ситуациях; консилиумов и консультаций, обмена мнениями врачей. Технология проведения видеоконференции:

- согласование времени видеоконференции (сеанса связи);
- подготовка информации о пациенте (файлов записей из истории болезни, статических данных, например рентгенограммы, эхокардиограммы и др., и динамических, например видеозаписи операций и результатов анализов);
- предварительная пересылка данных по электронной почте;

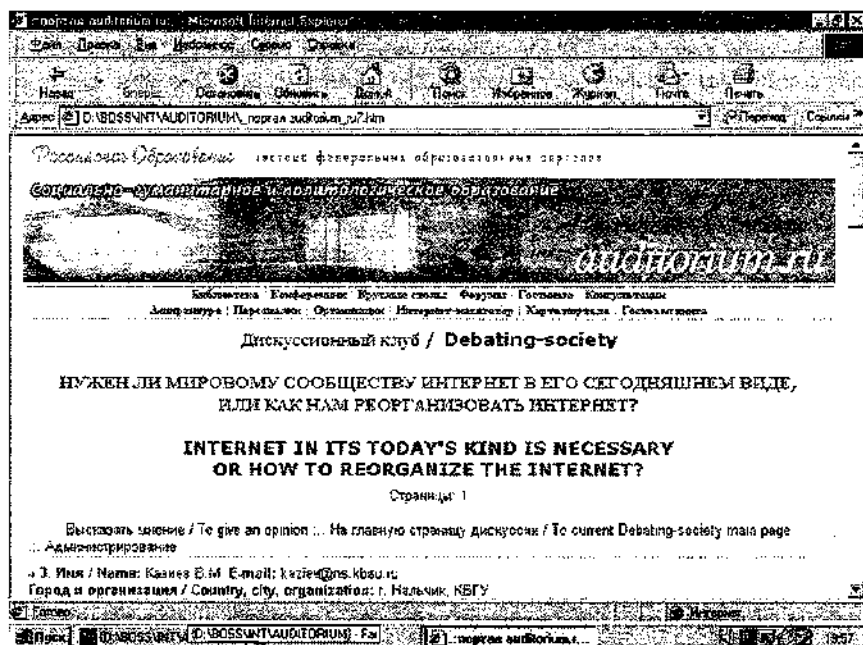


Рис. 52. Фрагмент дискуссии на портале <http://www.auditorium.ru>

- обсуждение в режиме видеодиалога информации о больном;
- принятие решения, а также документирование результатов обсуждения.

В Научном центре сердечно-сосудистой хирургии им. А. Н. Бакулева РАМН в рамках проекта «Москва — регионы России» проведено много плановых и экстренных видеоконсультаций. Экономическая и социально-медицинская выгода от таких видеоконференций в 6—10 раз выше, чем от классической технологии проведения консультаций с выездом в клинику (иногда это и невозможно). В режиме телеконференции проходят дискуссионные клубы, форумы.

ПРИМЕР

В ноябре—декабре 2003 г. автор выступал модератором (модератор — ведущий) дискуссионного клуба на тему «Нужен ли мировому сообществу Интернет в его сегодняшнем виде, или Как нам реорганизовать Интернет?» (см. форму на рисунке 52 и содержание на портале <http://www.auditorium.ru>).

19.4. АРМ, САПР И ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

АРМ (автоматизированная система профессиональных работ, автоматизированные рабочие места), САПР (система автоматизированного проектирования), ИС (интеллектуальные системы управления производством) — человеко-машинные системы для вы-

полнения ежедневных, зачастую рутинных, профессионально выполняемых на рабочем месте сотрудника работ, например проектировщика, с целью уменьшения времени, ошибок и обеспечения оперативной связи с другими сотрудниками; интеллектуальные системы имеют также способность к перестройке технологической цепочки, способность к обучению.

ПРИМЕР

АРМ на базе программной системы «1-С Бухгалтерия» автоматизации работы бухгалтерии. Примером САПР может быть САПР для автоматизированного проектирования мелиоративных коммуникаций, водоканалов (автор участвовал в создании такой большой системы).

Автоматизированное рабочее место (АРМ) — компьютерная система, устанавливаемая непосредственно на рабочем месте специалиста и предназначенная для автоматизации профессиональных работ (сидящего за этим рабочим столом сотрудника). Несмотря на различный характер задач, решаемых с помощью АРМ, принципы создания АРМ должны быть общими:

- системность подхода к проектированию и решению задач и возможность работы в составе сети, системы;
- гибкость, адаптируемость к изменениям задач;
- устойчивость, восстанавливаемость при сбоях;
- эффективность по затратам, производительности труда;
- быстрота отклика (минимум времени на шаг диалога);
- полнота выполняемых функций, решаемых задач;
- интерактивность (возможность вмешиваться в диалог, выбирать следующий шаг диалога, например, в форме команд на специальном командном языке, в форме выбора объектов, в форме меню, в смешанной форме);
- функциональность, дружелюбность (эргономические характеристики и удобство использования, в частности ориентация на непрофессионала в области компьютерной подготовки) и др.

Пользовательский интерфейс с АРМ часто организуется с помощью понятия Рабочего стола на экране. Экран делится на три части (три объекта). Первая — строка меню, с ее помощью осуществляется доступ к другим объектам. Вторая часть называется строкой состояния, с ее помощью быстро вызываются часто используемые объекты или отображается важная текущая информация. Третья часть называется рабочей поверхностью стола, с ее помощью отображаются все объекты, вызываемые из меню или из строки состояния. Такая форма организации диалога человека и машины наиболее удобна, и многие программы используют ее. Программные средства АРМ — часть инструментального ПО.

ПРИМЕР

АРМ банковского служащего и банковские системы наиболее развиваемые системы. Они включают программное и техниче-

ское обеспечение как специального назначения (например, для банковских расчетов и операции с банкоматами), так и для обеспечения безопасности таких систем. Банковские системы используют локальные вычислительные сети, специализированные бизнес-серверы, ЭВМ общего назначения, технологию «клиент—сервер» и часто ОС UNIX, объединение нескольких локальных сетей, межсетевой обмен и удаленный доступ филиалов банка к ресурсам центрального офиса банка для выполнения электронных платежей, транзакций. Банковские системы должны иметь средства адаптации к конкретным условиям эксплуатации. Для поддержки оперативной работы банков банковские системы должны функционировать в режиме реального времени OLTP (On-Line Transaction Processing, Онлайнная Обработка Состыковок). Основные их функции:

- автоматизация всех ежедневных внутрибанковских операций, ведение бухгалтерии и составление сводных отчетов;
- обеспечение связей с филиалами и отделениями;
- автоматизированное взаимодействие с клиентами (система «банк—клиент»);
- анализ деятельности банка и выбор оптимальных в данной ситуации решений;
- автоматизация розничных операций — применение банкоматов и кредитных карточек;
- межбанковские расчеты;
- автоматизация деятельности банка на рынке ценных бумаг (слежение за курсами, объемами сделок и т. д.);
- оперативная информационная поддержка финансовой и кредитной политики банка.

Примерами популярных банковских систем являются IBS-90 Winter, Bankier CSBI, IBIS Bruce Payne, Bankware Interlog. Отметим повышенные требования к надежности, защищенности и стоимости банковских систем.

Необходимо отметить, что автоматизация учреждения, организации — это обеспечение работников АРМ не только локальной сетью, электронной почтой и т. п., но и новой технологией совместной работы и управления с целью обеспечения эффективной работы всего учреждения. Это своего рода информационная, экономическая (корпоративная) культура совместной работы, использования профессиональных знаний, умений и инноваций. При этом знания каждого сотрудника могут быть актуализированы другими сотрудниками, обсуждены (проголосованы) и применены ими.

ПРИМЕР

По зарубежным оценкам, наличие такой культуры может дать повышение эффективности работы учреждения в целом на 20% и выше.

Автоматизированная обучающая система — комплекс из научно-методических, программных и технических средств (включая

мультимедиа, гипертекст и гипермедиа) для построения новых технологий обучения и контроля, повышения эффективности учебного процесса, индивидуализации и активизации самостоятельной работы обучаемого, повышения отдачи преподавания.

Процесс обучения, самообучения — системный и алгоритмический процесс независимо от того, что, где и как, какой технологией изучается. Обучающие и тестирующие среды лучше подходят под самообучение: они в отличие от «статичных» книг более динамичны, структурированы сами и структурируют процесс обучения (при использовании средств мультимедиа это проявляется особенно отчетливо). Когда обучаются с помощью компьютерных сред, обучаемые развивают свои мыслительные способности и знания. Предметное наполнение таких систем способствует обдумыванию, решению различных проблем предметной области, а также созданию деловых ситуаций, моделированию, решению задач и построению новых задач. При этом возможно образование различных учебных групп для моделирования той или иной предметной ситуации, решения как учебного, так и реального характера. Жизненная ситуация, в которую погружается обучаемый и в которой он может моделировать, проигрывать свое поведение, — лучший учитель.

Компьютерные обучающие среды (КОС) — программные комплексы для изучения дисциплин (разделов) с помощью компьютера в интерактивном режиме. Наличие обратной связи в КОС способствует формированию модели самим учащимся и делает возможным более полное и самостоятельное усвоение материала. Это эффективно для дисциплин с большим объемом теоретического материала и небольшим объемом практических занятий и самообучения.

Гораздо эффективнее в методическом, технологическом и психофизиологическом отношении использование технологий гипермедиа-обучения. Гипермедиа актуализирует больше чувств восприятия (зрение, слух, пространственное воображение, подсознание и др.) и повышает качество и скорость навигации (нахождения необходимых сведений в информационных материалах большого объема) за счет использования контекстных гиперссылок. Это позволяет сопровождать лекцию контекстными переходами в смежные разделы, а также параллельно с лекцией представлять в дополнительных окнах на экране сопутствующий разъяснительный материал.

ПРИМЕР

Для изучения языка Паскаль автором разработана КОС (рис. 53—55) <http://pascal.hotbox.ru>.

Старые специальности (старые информационные технологии) отмирают, появляются новые (новые информационные технологии). Запоздывание последних может привести к безработице из-за несоответствия спроса и предложения рабочей силы. Выход — развитие системы дополнительного образования, структурная перестройка. Поэтому в последнее время большое внимание уделяется системам дистанционного образования. Начав

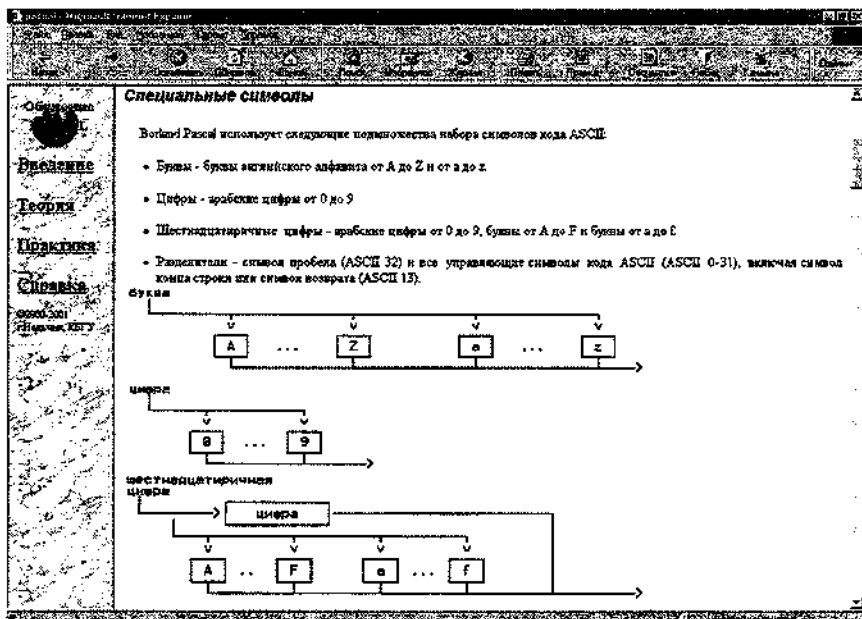


Рис. 53. Фрагмент учебного материала

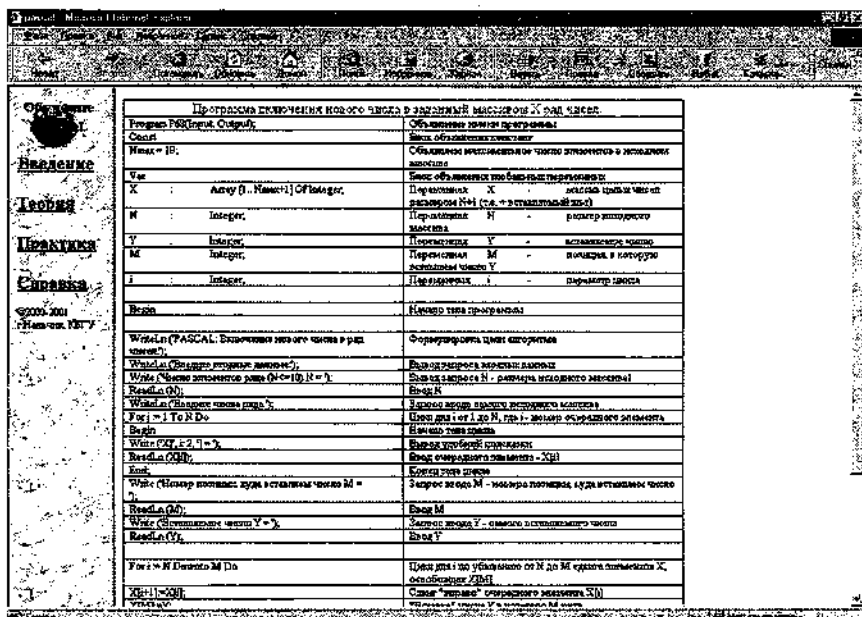


Рис. 54. Фрагмент демонстрации программы решения типовой задачи

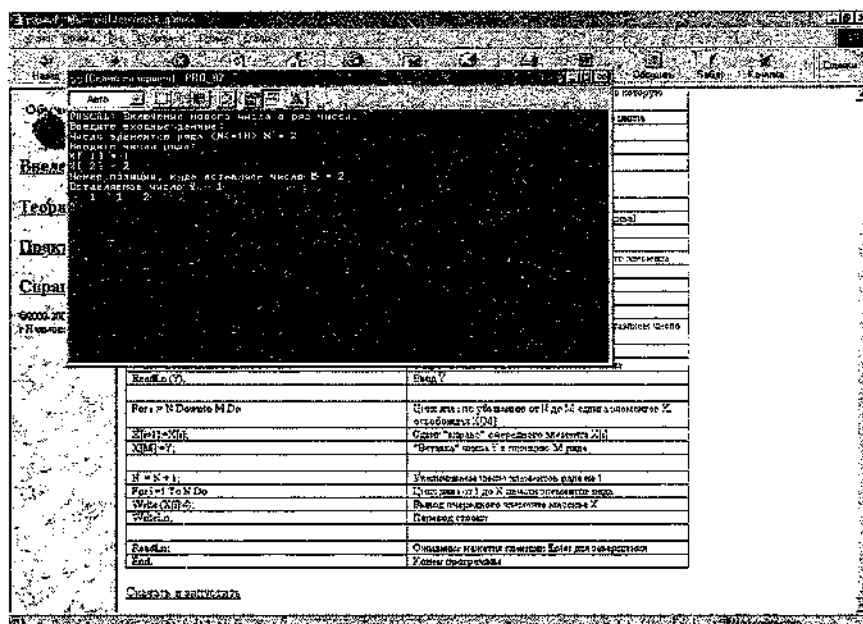


Рис. 55. Запуск продемонстрированной программы на выполнение

с разработки отдельных обучающих программ и курсов, система развилась сейчас до такой степени, что позволяет овладеть всем комплексом знаний, необходимых для получения диплома о высшем образовании. Базовое программное обеспечение для дистанционного образования включает инструментальные средства, план, лекции, лабораторные и практические занятия, проверку и тестирование по пройденному материалу, базу данных по курсу и методике. Общение с преподавателем происходит либо по электронной почте, либо с помощью телеконференций и обеспечивает индивидуальное обучение.

ПРИМЕР

Оформление системы дистанционного обучения изображено на рисунке 56. Там видны возможности системы и стандартный набор предлагаемого сервиса: заголовок курса, функциональные клавиши «Информация о курсе», «Материалы курса», «Глоссарий», «Библиотека», «Письмо тьютору», «Конференция», «Тесты», «Помощь», которые актуализируют соответствующие возможности, и основное окно с кадром материала курса.

ПРИМЕР

Интернет-Университет Информационных технологий (<http://www.intuit.ru>) ведет доступное открытое обучение по различным дисциплинам, являясь эффективным виртуальным учебным заведением. Там можно ознакомиться с полезным

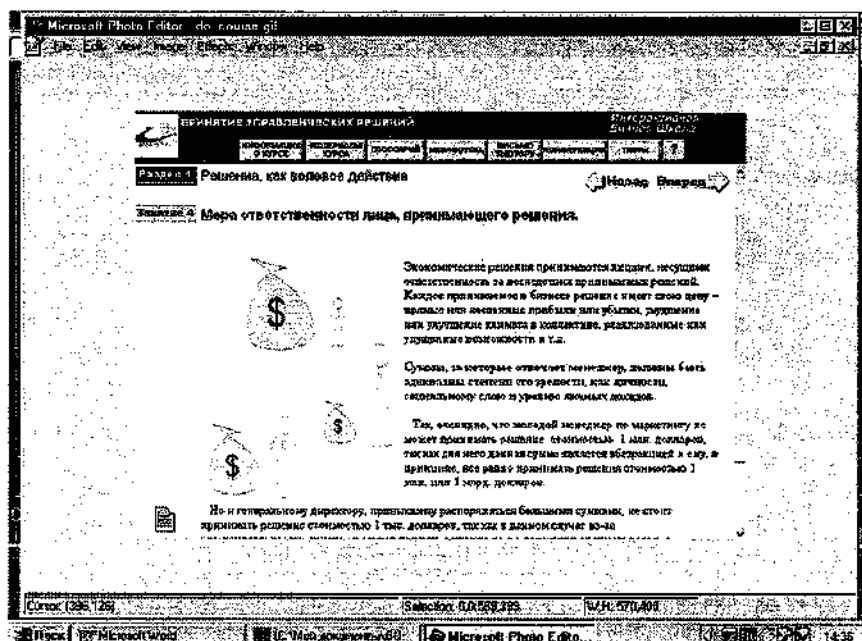


Рис. 56. Типовые возможности системы ДО

курсом «Введение в анализ, синтез и моделирование систем» автора.

Рост разнообразия — всеобщий закон прогрессивной эволюции. Знания и профессионализм становятся капиталом. Появляются автоматизированные системы профессиональной ориентации: САПР — системы автоматизации проектирования; АРМ — автоматизированные рабочие места; СОПР — системы обеспечения принятия решений и др. Рабочее место переносится в жилище человека, где основным орудием труда становится персональная ЭВМ (удаленный терминал центральной ЭВМ). Само жилище становится «умным» — «умный дом» включает или отключает тепло, свет, экономит энергию, открывает и закрывает двери и т. д.

19.5. ИНТЕГРИРОВАННЫЕ ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

Интегрированные пакеты прикладных программ — технологии на базе специальным образом организованных комплексов программ для решения различных классов однотипных и часто встречающихся задач из разных предметных областей.

Рассмотрим пакет программ для перевода с одного языка на другой — переводчик PROMT. Заметим, что переводчик PROMT явно не входит в стандартное ядро MS Office.



Рис. 57. Рабочее окно системы PROMT

Компоненты главного окна системы PROMT изображены на рисунке 57.

Можно выполнять интерактивное редактирование и перевод документа:

- 1) просматривать и видоизменять исходный текст или текст перевода;
- 2) переводить исходный текст полностью или по абзацам;
- 3) возвращаться к уже переведенным фрагментам исходного текста, редактировать их или корректировать словарь, после чего заново перевести отредактированный текст.

В рабочее окно документа могут одновременно выводиться либо исходный текст и его перевод (при этом окно документа разбивается на подокна — горизонтально (кнопка на рисунке 58) или вертикально (кнопка на рисунке 59), либо исходный (переведенный) текст. Можно выбрать различные режимы вывода текста в окне с помощью команд из меню Вид.



Рис. 58



Рис. 59

Горизонтальное разбиение. Верхнее подокно — исходный текст, нижнее — его перевод (рис. 60).

Вертикальное разбиение. Левое подокно — исходный текст, правое — его перевод.

Без разбиения. В полное окно документа выводится только текущий текст. Текущим является текст, в котором установлен курсор. В зависимости от того, какой текст вы хотите видеть в полном окне документа, выбирайте Оригинал или Перевод из меню Вид.

Если нужно набрать текст непосредственно в PROMT с целью

In spite of immediate response to antibiotic response to antibiotic therapy and prognosis of uneventful recovery, this child, demonstrated anxiety in his behaviour.

Десятилетний мальчик был госпитализирован из-за умеренной коленной инфекции, которая развивалась от царапины, подвергся при игре. Его отец положил его в больницу и оставил его там.

Рис. 60

его дальнейшего перевода, выполните одно из следующих действий:

нажать кнопку  либо выбрать Создать в меню Файл. В окне

«Направление перевода» укажите, с какого языка и на какой собираетесь переводить текст, и нажмите ОК. В любой момент можно изменить направление перевода, выбрав Изменить направление в меню Перевод. Появится пустое окно документа с мигающим курсором на первой позиции. После того как текст или его часть набрана, можно приступить к выполнению перевода.

Открытие файла с исходным текстом. Для открытия существующего файла нужно выполнить одно из действий:

1) Нажать кнопку .

2) Выбрать команду Открыть в меню Файл, затем в окне выбрать путь к каталогу, где хранится документ, имя файла и нажать кнопку Открыть. Появится окно документа, в котором будет текст из открываемого файла.

Перевод документа. В PROMT можно переводить как тексты из файлов, так и непосредственно набираемые. Система выполняет перевод текста по абзацам. Абзац — это фрагмент текста, который завершается нажатием клавиши Enter. В PROMT начало каждого абзаца отмечено специальным символом (флажком). Если одно предложение поделено на несколько абзацев, то для выполнения связанного перевода необходимо удалить разделители абзацев, «склеив» предложение в один абзац. Цвет метки абзаца (флажка) зависит от действий, выполненных над абзацем:

▷ (белый) Непереведенный абзац.

▶ (синий) Переведенный абзац.

► (черный) Абзац, который редактировался после перевода.

▶ (зеленый) Зарезервированный абзац — остается без перевода.

Каждый абзац исходного текста связан с соответствующим абзацем перевода. Если один из них является текущим (с курсором),

то выделяются они голубой полосой, каждый в своем подокне. Если в окне документа не видны метки начала абзацев, то выберите в меню **Сервис** команду **Параметры**, вкладку **Цвета** и опцию **Использовать условные значки**.

Последовательность действий при выполнении перевода. Чтобы перевести текст, необходимо выполнить следующие действия:

1. Открыть файл с исходным текстом или создать новый документ, если вы хотите набрать новый текст непосредственно в PROMT. Исходный текст появляется в окне документа.
2. Отредактировать исходный текст, если необходимо, разбив на абзацы.
3. Подключить словари, которые будут использоваться при переводе текста. Это позволит получить качественный перевод текста. В том случае если не подключен ни один словарь, при переводе будет использоваться только общий лексический **генеральный словарь**.
4. Для того чтобы некоторые слова не переводились и оставались в тексте перевода так, как они встречаются в исходном тексте, их необходимо зарезервировать. Также в системе предусмотрена возможность пометки целых абзацев, которые не требуют перевода.

Все команды, выполняющие перевод текста, находятся в меню **Перевод**. Можно переводить текст по абзацам или весь текст. Слова, не найденные в подключенных словарях, выводятся в тексте документа красным цветом, а зарезервированные слова — зеленым. Весь список незнакомых слов можно увидеть на информационной панели (**Незнакомые слова**). Если незнакомые слова далее переводятся, введите их в свой пользовательский словарь.

Сохранение всего документа в формате PROMT. Если нужна далее работа с документом в системе PROMT, то необходимо сохранить документ в файле формата std. Сохраняются результаты работы: исходный текст, полученный перевод, список незнакомых и зарезервированных слов. Чтобы сохранить документ в PROMT:

- 1) Выберите **Сохранить Документ** в меню **Файл**.
- 2) Задайте путь и имя файла.

Сохранение только текста перевода в файле. Если хотите сохранить в файле только текст перевода, нужно выбрать в меню **Файл** команду **Сохранить Перевод**. Формат файла может быть: MS DOS Текст, Только текст, Форматированный текст (rtf), Word.

Сохранение только исходного текста в файле. Если хотите сохранить в файле только текст перевода, нужно выбрать в меню **Файл** команду **Сохранить Исходный текст**. Формат файла может быть: MS DOS Текст, Только текст, Форматированный текст (rtf), Word.

Сохранение списка незнакомых слов. Список слов, не найденных системой при переводе в подключенных словарях, можно увидеть в окне документа (вкладка **Незнакомые слова**).

Сохранение списка зарезервированных слов. Список зарезервированных слов хранится вместе с PROMT-документом, для которого он был создан, и автоматически подгружается при открытии этого документа. Список можно увидеть в окне документа (вкладка **Зарезервированные слова**). Если нужно использовать этот список также при переводе и других документах, то сохраните его в файле wwt-формата или в текстовом формате. Чтобы сохранить список зарезервированных слов, необходимо:

- 1) Активизировать вкладку **Зарезервированные слова**, щелкнув на ней левой кнопкой.
- 2) Щелкнуть **Зарезервированные слова** правой кнопкой; появится меню.
- 3) Выбрать **Сохранить список**.
- 4) Указать формат, путь и имя файла.

Некоторые словари PROMT. В системе PROMT есть словари: генеральный словарь, специализированный словарь, пользовательский словарь. Словари содержат переводы слов и словосочетаний, грамматическую информацию, необходимую для получения перевода. Для пополнения словарного запаса системы нужно создать свой пользовательский словарь, в который можно вводить новые слова (словосочетания). Для просмотра содержимого словаря можно открыть пользовательский или специализированный словарь. В окне будут выведены все словарные статьи, находящиеся в словаре.

Общая схема ввода слова (словосочетания) в словарь. Если нужно ввести новое слово или словосочетание в словарь или изменить перевод слова, то выполните действия:

- 1) Выбрать **Словарная статья** в меню **Перевод**.
- 2) Набрать слово или словосочетание, которое хотите ввести в словарь (правила ввода слова можно узнать, нажав **Справка**).
- 3) Выбрать пользовательский словарь, где будет сохранена словарная статья.
- 4) Указать часть речи входного слова; если вводится словосочетание, указать часть речи для главного слова в словосочетании.
- 5) Программа автоматически просклоняет (проспрягает) входное слово.
- 6) Появится окно **Словарная статья**, содержащее словарную статью входного слова; если этого слова нет ни в одном из подключенных словарей, то в словарной статье список переводов пуст; в противном случае открывается словарная статья слова, взятого из первого по списку подключенного словаря.
- 7) Можно приступить к вводу (редактированию) грамматической информации.
- 8) Ввести перевод слова, нажав **Добавить** или **Правка**.
- 9) Нажать **Сохранить**: словарная статья будет сохранена в пользовательском словаре, который был указан, когда открывали словарную статью.

Приведем общую характеристику и примеры современных ППП. Они имеют диалоговую, интерактивную обратную связь с

пользователем в процессе постановки задачи, решения и анализа результатов. «Интеллектуальность» ППП — возможность постановки задачи содержательно, не указывая алгоритма ее решения. Построение алгоритма решения и сборка целевой программы производятся автоматически, скрытно от пользователя. Предметное обеспечение ППП — база знаний о методах, алгоритмах решения задачи и о самих задачах. Программирование осуществляется в терминах предметной области, ЭВМ используется уже на этапе постановки задачи, решения задач с помощью автоматического построения цепочки программ, по ходу накопления знаний о решаемой задаче, т. е. возможно пополнение базы знаний ППП. Используются инструментальные ППП, которые ускоряют и упрощают процесс создания ППП и снижают его стоимость. Высокая интеграция: наполнение ППП само состоит из ППП различного назначения.

19.6. СРЕДСТВА И СИСТЕМЫ МАШИННОЙ ГРАФИКИ

Средства и системы машинной графики (системы рисования и черчения различных графических объектов и образов с помощью, например, графопостроителей-плоттеров); особо следует отметить средства анимации — оживления изображений на экране, т. е. методы и средства создания динамических изображений (компьютерных мультфильмов).

Машинная графика — раздел информатики, изучающий и применяющий модели, алгоритмы, программные системы, компьютерные технологии анимации, оживления, визуализации и динамической активизации различных объектов и систем, от простейших примитивов (точка, отрезок, круг и т. п.) до сложнейших чертежей и геометрических объектов. В основе систем машинной графики лежат соответствующие математические алгоритмы моделирования примитивов и операций их преобразования, использования библиотек таких объектов и преобразования.

При решении задач разного рода результаты иногда представляются в пространстве в виде объемного изображения. Большей частью это какие-либо поверхности, заданные разнообразными периодическими функциями, например $\cos x$, $\sin x$ и др. Для таких случаев на ЭВМ требуется построение поверхностей с учетом объемности и пропорций представления в реальных координатах.

Объемная поверхность задается математической формулой от двух переменных $z = f(x, y)$, например $z = \cos(\sqrt{x^2 + y^2})$. Для построения объемных поверхностей используют несколько методов, например каркасный график (метод плавающего горизонта) и полутонный график (метод художника). **Каркасный график** — это совокупность отрезков, соединяющих точку графика с ее соседними точками. При построении обычно шаг между соседними линиями по оси x и по оси y одинаковый и равен const . При таком построении поверхность как бы покрыта двухцветной сеткой: для отображения верхней части поверхности более ярким цветом и нижней —

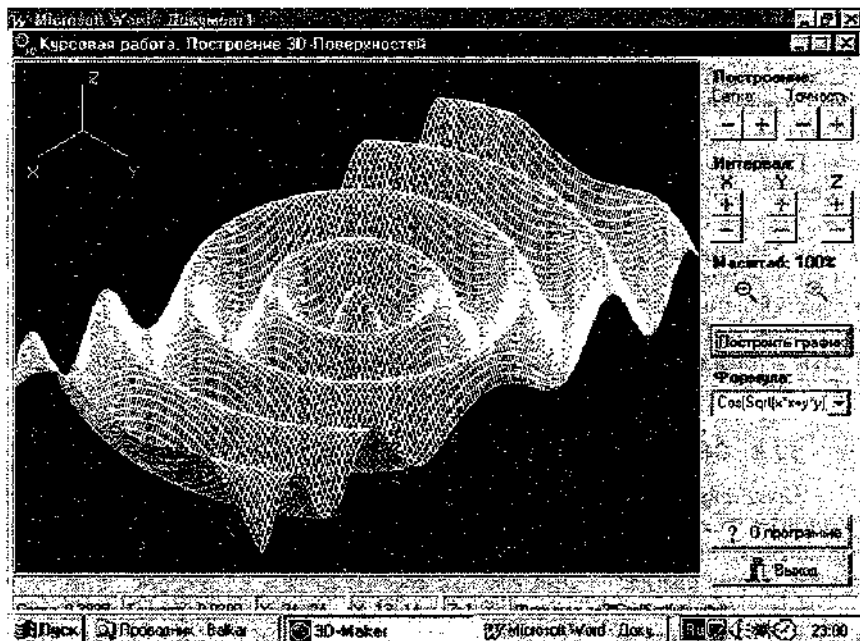


Рис. 61. График функции $z = f(x, y)$ с частой сеткой

более темным. Главное преимущество таких изображений в том, что их можно вывести на любое внешнее печатающее устройство (принтер, плоттер и др.) практически без потери качества.

При построении также учитывается удаление невидимых линий. Для этого можно применять алгоритм плавающего горизонта. При выводе очередной точки проекции на картинную плоскость анализируется видимость текущей точки ребра при помощи нижней и верхней границ уже полученной к этому моменту части изображения. Эти границы называются нижним и верхним плавающими горизонтами (они плавают с учетом видимости или невидимости очередной сбрасываемой на экран точки проекции ребра). При этом верхний и нижний горизонты можно представить в виде двух целочисленных массивов $\min[l]$ и $\max[l]$, $l = 0, 1, \dots, L$; l — количество точек раstra по горизонтали. Алгоритм достраивает изображение вне области, определяемой верхними и нижними горизонтами, если ордината изображаемой точки находится между $\max[l]$ и $\min[l]$.

Рассмотрим программу рисования 3D-графиков (рис. 61, 62).

Одной из наиболее используемых систем машинной графики является пакет программ 3D-Studio. Это качественная программная среда для построения трехмерных объектов, сцен в динамике, анимации, роликов и др.

Пакет 3D-Studio позволяет создавать трехмерные сцены, использовать эти сцены при реализации компьютерных анимаций

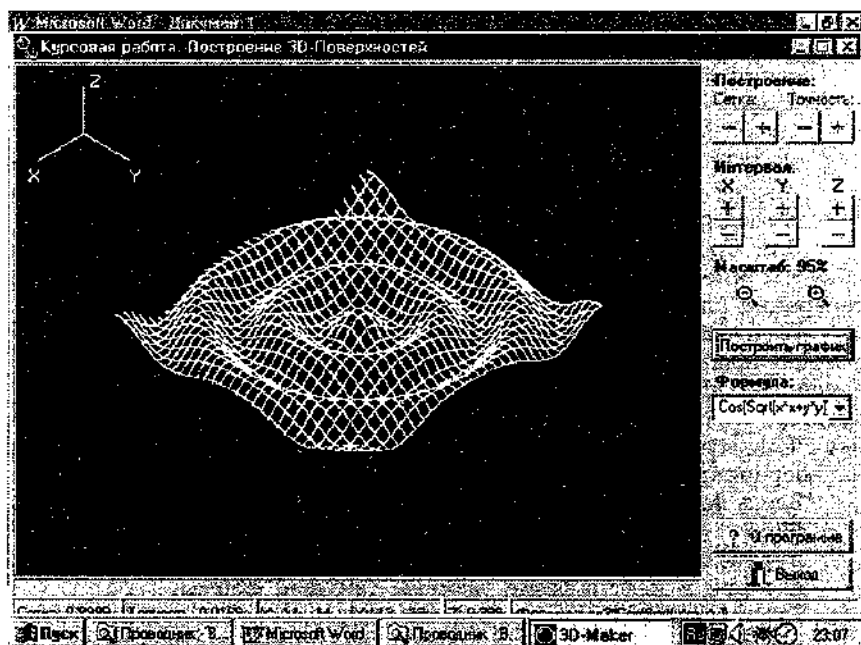


Рис. 62. График функции $z = f(x, y)$ с более редкой сеткой и уменьшением x, y, z

(мультипликаций) с использованием различных графических форматов, что позволяет применять при разработке мультфильмов известные графические пакеты CorelDraw, PhotoPaint и др.

3D-Studio имеет модульную структуру, состоящую из пяти модулей, за каждым из которых закреплен конкретный тип задач, решаемых в строгой последовательности.

Первый модуль (2D-Shaper) является основным инструментом создания и редактирования плоских фигур, а также снабжения других модулей особыми геометрическими структурами, формами и траекториями.

Функции модуля 2D-Shaper разбиты на пять основных групп:

- Create — команды создания 2D-фигур. 3D-Studio считает все фигуры многоугольниками, которые состоят из сегментов и могут быть сглажены достаточно точно сплайном (особого вида полином, определяемым на каждом промежутке с помощью своих коэффициентов). Например, круг представляется как многоугольник со «сглаженными углами».
- Select — команды создания набора выбранных объектов модуля 2D-Shaper (объединения их в единый кластер), после чего весь набор можно рассматривать как единый целостный объект с выделением его свойств.
- Modify — набор функций, изменяющих геометрические харак-

теристики существующих объектов модуля 2D-Shaper. Сюда входят, например, функции масштабирования по осям, выполнения наклонов и др.

- Shape — команды осуществления сборки геометрической формы из готовых геометрических фигур (геометрических примитивов).
- Display — команды для отображения на экран всевозможных параметров создаваемых объектов.

Для преобразования плоских фигур в трехмерные каркасные объекты имеется модуль 3D-Lofter, в который включены средства генерации сложных пространственных форм и структур.

Подготовленные двумерные планы моделей отображаются («выдавливается») в третье измерение по специально заданным траекториям. Модуль 3D-Lofter снабжен средствами деформации, например, по осям, что позволяет создавать трехмерные объекты более сложных форм. Можно построить 3D-фигуру по трем проекциям на координатные плоскости.

Здесь также имеется пять наборов основных функций:

- Shapes — содержит средства, манипулирующие загрузкой, копированием, перемещением форм объектов, определяющие траектории и изменяющие размеры текущей формы.
- Path — средства «вытягивания» формы в трехмерный объект, который позволяет использовать различные модификации заданной траектории и задавать новые.
- Deform — средства создания трехмерного объекта, которые позволяют динамически менять сами формы (положения) относительно траекторий при вытягивании с использованием аппарата генерации 3D-объекта из трех ортогональных проекций на объект (спереди, сверху и сбоку).
- Display — средства управления визуализацией объекта на экране.
- Objects — средства установления параметров генерации трехмерного объекта и регулирования степени детализации исходной двумерной формы и способа вытягивания (чем выше детализация исходной формы, тем более плотным и гладким будет выглядеть объект; регулирование можно осуществлять визуально — путем просмотра генерируемого объекта).

19.7. МУЛЬТИМЕДИА, ГИПЕРТЕКСТ, ГИПЕРМЕДИА

Средства мультимедиа (multimedia), медиа — «среда или носитель информации», многосредность (актуализация различных сред и чувств восприятия информации: средства озвучивания, оживления — мультипликация, графического и наглядного представления входных и выходных данных задачи и сценариев решения или даже самого решения).

ПРИМЕР

Примерами средств мультимедиа могут служить звуковые карты (Sound Blaster) для генерирования на ЭВМ достаточно широкого диапазона звуков, активные звуковые колонки для их передачи и устройства считывания информации с компакт-дисков — CD-ROM, позволяющие считывать большие объемы информации, например некоторую сложную и длительную музыкальную композицию, а затем ее воспроизводить с использованием предыдущих двух средств мультимедиа.

Гипертекст (гипертекстовые структуры, средства) — программные средства для быстрого и эффективного создания, хранения, поддержки, использования информационных систем различной сложности. Это средство (способ) существенного увеличения производительности труда при разработке и актуализации информационных систем, а также более полного и надежного представления данных (знаний) в таких системах. Достигается все это с помощью единой формы организации и структурирования информации и высокой степени автоматизации представления разнотипных сообщений — текстовых, цифровых, графических, которые могут быть разными по предметной области.

При работе с гипертекстовой информационной системой пользователь имеет возможность просматривать документы (страницы текста) в том порядке, в котором ему это больше нравится, а не последовательно, как это принято при чтении книг, т. е. гипертекст — нелинейная структура. Достигается это путем создания специального механизма связи страниц текста при помощи гипертекстовых ссылок, т. е., кроме линейных ссылок обычного текста типа «текст-предшественник — текст-преемник», для гипертекста можно построить еще сколь угодно много других динамических ссылок, ассоциированных с документом в целом или только с отдельными его фрагментами, т. е. контекстные ссылки.

Глобальной гипертекстовой системой является WWW (World Wide Web — Всемирная паутина) — гипертекстовая система навигации, поиска и доступа к мультимедийным ресурсам Интернета в реальном масштабе времени. Глобальной ее можно считать потому, что в отличие от обычного (локального) гипертекста ссылка на документ в нем (осуществляемая одним или несколькими щелчками мыши) может привести не только к другому документу (как в локальном гипертексте), но и к другому компьютеру (www-серверу), возможно, в другом полушарии. Работа ведется с помощью универсальной программы-клиента, которая позволяет объединить в единое целое клиента и сервер, удаленных во времени и в пространстве.

Для доступа к www-серверу необходимо знать адрес этого сервера, например <http://www.mark-itt.ru> — адрес сервера со списком российских www-серверов, <http> (HyperText Transfer Protocol) — протокол работы с гипертекстом. Информацию, считываемую с ЭВМ по введенному адресу, можно сохранять специальными командами. Имеется система автоматического поиска по определенным ключам

(запросам, разделам). Информация в WWW представлена в виде гипертекстового документа, включающего в себя различные типы данных (текст, графика, видео, аудио, ссылки на другие гипертекстовые документы и т. д.). Такие документы называют **www-страницами** (**www-pages**). Эти страницы просматриваются с помощью **браузеров** — специальных программ для навигации по сети. Страницы хранятся на компьютерах-узлах, которые называют **сайтами** (**site**). Каждый компьютер имеет свой уникальный IP-адрес **URL** (**Uniform Resource Locator** — Универсальный Локатор Ресурсов), с помощью которого браузер знает, где находится информация и что с ней надо делать. Страница — основной элемент WWW. На страницах находится та информация, которую мы ищем в сети, или ссылки на нее. Страницы, гипертекст — это легкая и быстрая в использовании, чрезвычайно мощная система связанных ключевых слов и фраз (ссылки), позволяющая ссылаться на другие ключевые слова и фразы с других страниц. Эти ссылки обычно выделены цветом, и достаточно просто щелкнуть мышью по выделенной ссылке, чтобы перейти к информации, на которую делается эта ссылка. Для создания гипертекстовых приложений (например, личной **www-страницы**) используется специальный язык **HTML** (**HyperText Markup Language**), позволяющий создавать гипертекстовый документ в любом текстовом редакторе формата **ASCII** с подключением графических файлов двух основных форматов **GIF**, **JPEG**. Язык разметки гипертекста определяет формальный набор правил и процедур для электронной интерпретации и представления подготавливаемого текста. Взять документ, созданный с помощью, например, **Word**, добавить в него команды **HTML** — и создан фрагмент страницы.

ПРИМЕР

Пример **html**-разметки двух предыдущих предложений:

```
<HTML><HEAD><META HTTP-EQUIV=«Content-Type» CONTENT=
=«text/html; charset=windows-1251»> <TITLE> Пример </TITLE>
</HEAD><BODY><FONT FACE=«Times New Roman» SIZE=2><P>
Язык разметки гипертекста определяет формальный набор пра-
вил и процедур для электронной интерпретации и представле-
ния подготавливаемого текста. Взять документ, созданный с по-
мощью, например, Word, добавить в него команды </FONT>
<FONT SIZE=2>HTML</FONT><FONT FACE=«Times New Roman»
SIZE=2>, и </FONT><FONT SIZE=2>—</FONT><FONT FACE=
=«Times New Roman» SIZE=2> — и создан фрагмент страницы.
</FONT><FONT SIZE=2>Web</P></FONT></BODY></HTML>
```

HTML обеспечивает только форматирование страницы и выделение гиперсвязей. Если пользователю понадобится какая-либо информация, не содержащаяся в документе или содержащаяся в какой-либо базе данных, то тут приходит на помощь **CGI** (**Common Gateway Interface** — Стандартный Интерфейс Шлюза), который пре-

доставляет механизмы для выбора, обработки и форматирования документов. CGI применяется для подготовки или запроса информации. В отличие от HTML, CGI не является языком описания документов. Собственно, это и не язык вообще, а стандарт оформления. При написании программы шлюза CGI можно использовать многие языки, например C++.

Средства гипермедиа — hypermedia = hypertext + multimedia (средства на основе синтеза концепции гипертекста и мультимедиа, т. е. в гипертекстовые фрагменты могут быть «встроены» звуковое, мультипликационное, графическое или мультимедийное сопровождение, приложение).

ПРИМЕР

Примером гипермедиа может служить система WWW (World Wide Web или Всемирная паутина) в Интернете. Пример гипермедийной технологии — разработка и тиражирование музыкально-познавательных электронных журналов с различными аудиовидеороликами, статьями о музыке и т. д.

По мнению ряда исследователей, следующей после Web формой коллективного сосуществования компьютеров будет Grid, которая даст пользователям больше возможностей для работы с удаленными машинами. Если World Wide Web можно сравнить с аналоговой телефонной сетью, способной передавать тексты, аудио и видео, то Grid подобна современной системе электроснабжения, предоставляющей потребителям столько вычислительных ресурсов, сколько им необходимо. В настоящее время в ЦЕРНе ведутся работы по определению стандартов для Grid. Так же как и WWW, новая концепция в первую очередь будет востребована в исследовательских кругах. Ученые с помощью Grid будут получать доступ к ресурсам, необходимым для решения их задач. Архитектура Grid трехслойна: интерфейс, слой приложений и операционная система Grid, позволяющая подключить пользователей к распределенным ресурсам.

Все общедоступные ресурсы WWW предназначены для всех, но нужен браузер, который преобразует сложно структурированный гипертекстовый документ в документ, читаемый на экране. Есть различные такие программы, например Netscape Navigator, Internet Explorer и др.

19.8. ТЕХНОЛОГИЯ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

Технология виртуальной реальности, или виртуальная реальность, — технологии актуализации различных гипотетических сред и ситуации не существующих реально и возможных как варианты развития реальных аналогов, систем реального мира; эти технологии и системы позволяют управлять виртуальным объектом, системой путем моделирования законов пространства, протяженности, времени, взаимодействия, инерции и др.

ПРИМЕР

Тренажеры для обучения пилотов самолетов, водителей автомобилей, капитанов судов позволяют помещать обучаемого в соответствующие воображаемые ситуации (включая и аварийные). Эффекты виртуальной реальности создаются часто за счет одновременного воздействия на различные органы чувств, включая подсознание, сенсомоторику. Интересны проекты создания хирургических тренажеров с использованием методов и средств виртуальной реальности. Важной формой виртуальной реальности (виртуального понятия) является рынок. Если раньше под рынком понималось реальное место встречи продавцов и покупателей, то теперь это понятие состоит из экономических, коммерческих, производственных и коммуникационных отношений и систем; они теперь могут встречаться и реализовывать свои функции в компьютерных системах.

Виртуальная реальность — система воображаемых объектов, процессов или средств, помещающих человека в среду (моделирующую с помощью описанных выше компьютерных технологий эту среду и поведение человека в ней), адекватную настоящей или реальной (возможной) среде или обстановке. Это не только техническое восприятие среды (обеспечиваемое устройствами обмена с окружающей средой, например, виртуальным шлемом, перчатками), но и психофизиологическое восприятие среды, актуализирующее эту среду таким образом, чтобы человек сам мог бы участвовать в происходящих в этой среде событиях. При этом имитируются реальные пространственные, временные и организационные связи с окружающим миром.

При виртуализации исследователь или проектировщик концентрирует свое внимание на сходстве фундаментальных, родовых свойств различных объектов некоторого класса, абстрагируясь от менее значимых в данный момент свойств. Но динамическое перепорядочивание связей в системе может изменять приоритеты, и фундаментальность того или иного свойства становится условной. При виртуализации сохраняется не какой-то отдельный объект, а целый класс объектов, для чего достаточно, чтобы в каждый момент времени объекты класса обладали хотя бы одним свойством.

Виртуализация среды тесно связана с трехмерной визуализацией и аудированием (т. е. звуковым стереосопровождением) этой среды и ее объектов, причем техническая реализация системы для реальной визуализации (восприятия в реальном масштабе времени), разрешающей способности сетчатки глаза (около $2^{18} \times 2^{18}$ точек) не является технически непреодолимой, например, для нейрокompьютеров или системы параллельно работающих компьютеров (процессоров). Кроме визуализации и аудирования, в виртуальных средах с помощью специальных устройств (например, специальных костюмов операторов) имитируются динамические нагрузки и сопротивление, эффекты, а также ощущение прикосновения к объекту, воздействия на объект (его реакции) с помощью движения головы и тела.

ПРИМЕР

Примерами систем виртуальной реальности могут быть различные имитационные системы, адаптирующие, в том числе и психологически, человека (оператора) к различным реальным действиям (в частности, дистанционно управляемый робот-манипулятор в радиационной опасной местности), к виртуальным ситуациям, спроектированным по некоторому интерактивному сценарию (в частности, тренажер пилота самолета), а также системы гибридного типа, сочетающие обычную и виртуальную реальность (в частности, компьютерная томография в реальном режиме времени).

ПРИМЕР

Для формирования трехмерных зрительных образов разрабатываются лазерные микросканеры высокого разрешения в реальном масштабе времени, а разветвленная параллельная архитектура компьютера сможет моделировать визуализацию окружающего пространства. Для создания объемного звука и аудирования создаются библиотеки из готовых звуков. Более сложная проблема — формирование тактильных ощущений: чувства сопротивления внешних предметов и чувства фактуры (шероховатости, влажности и т. д.). Имитация сопротивления достигается перчатками и костюмами с вмонтированными в них специальными тягами, звуковыми динамиками, элементами, которые регулируют жесткость и давление перчатки или костюма с учетом строения мышц человека, чтобы обеспечить реакцию на усилия всех их групп. Для воспроизведения ощущения фактуры используются высокочастотные токи, микролазеры и база данных (знаний) базовых элементарных тактильных ощущений, где в основном применяется комбинация различных давлений и создается ощущение наличия объекта в руках. Для перемещения изображения с поворотом головы, передвижения объекта и т. д. используется информация о суставных углах пальцев, локтя, повороте шеи и т. п., а также трехкнопочная мышь с шестью степенями свободы, графические ускорители, специальные базы объектов и пакеты программ.

Виртуальная реальность и виртуализация — сложная комплексная проблема, поэтапное решение которой будет оказывать серьезное воздействие на науку, промышленность, социальную сферу.

Сферы использования систем виртуальной реальности могут быть самые различные: проектирование, разработка, эксплуатация, управление, обучение в реальных и виртуальных системах, рационализация и оптимизация при исследовании таких систем старыми методами и технологиями (например, по использованию ресурсов всех типов — материальных, энергетических, человеческих, информационных, организационных и технологических). Важная сфера их применений — разработка новых методов, технологий постановки и решения новых проблем.

ПРИМЕР

Имея виртуальную систему для дизайна и проектирования автомобиля, можно проектировщикам, гонщикам и механикам вместе создавать новую модель автомобиля, изменяя как внутренние схемы и узлы, так и детали салона, узлов, «продувая» модель в аэродинамической трубе. Особо значимой становится сейчас возможность проведения виртуальных конференций, виртуальной рекламы, виртуальных покупок и продаж, виртуальных военных игр и ситуаций, учений, виртуального оперирования больных, виртуальных посещений выставок и презентаций, виртуальное художество, в частности живопись, и др. В перспективе — создание виртуальных систем интерфейса «человек—компьютер», например ввода-вывода данных, минуя классические технические средства ввода-вывода.

Виртуальная корпорация и распределенная корпорация — элементы новой инфраструктуры экономики, производства и потребления, виртуального бизнеса, особенно информационного. Виртуализация свойств и атрибутов корпорации, динамическое (а иногда и виртуальное!) выделение их общих фундаментальных (родовых, классовых) свойств и их описание, использование в рамках единой технологии позволяет сократить промежуток между прогнозируемым (или имитируемым) состоянием корпорации и его реальным состоянием. У корпорации общие интересы появляются, актуализируются, виртуализируются на тот период, когда они служат общей цели. Успех виртуальной корпорации (ее разработки, внедрения и сопровождения) зависит от полноты и качества информационных потоков между объектами корпорации. При этом сотрудники новой корпорации имеют дело в первую очередь с информационными моделями, например с моделями склада сырья, изделия, менеджера, поставщика, банка (банковских расчетов). Таким образом, виртуальные компьютерные корпорации поддерживают широкий спектр работ и услуг — соединение предприятий по ресурсам, производству, сбыту, снабжению, управлению, информационному обеспечению и т. д., обеспечивая базовую компетентность — совокупность знаний и умений по организации, координации и согласованию общих организационных, материальных, информационных ресурсов элементов корпорации, поддержке и развитию ее инфраструктуры (архитектуры, инноваций, активов, рекламы и др.) и обеспечению корпоративных функций системы с меньшими затратами и более качественно, уменьшая бюрократизм и уровни иерархии в системе, а также время реакции на изменения на рынке.

Основные характеристики виртуальной корпорации:

- наличие основного вида бизнеса (деятельности) для подсистем;
- концентрация пользователей около этого вида деятельности;
- общие (связанные) цели, планирование и ресурсообеспечение;
- общие (интегрированные) стратегии поведения и актуализации ресурсов;

— общая (интегрированная) технология актуализации ресурсов.

В виртуальной корпорации «новые рабочие или служащие корпорации» будут в основном иметь дело с инфологическими моделями рабочих мест, инструментов, сырья, поставщиков, рынков сбыта и услуг, «новые менеджеры корпорации» будут принимать оперативные и более интеллектуальные решения по гораздо более широкому кругу вопросов, а «новое руководство» корпорации будет занято многокритериальными стратегическими проблемами. Следовательно, от них требуется качественно новый уровень профессионализма, ответственности и стремление к самообразованию.

ПРИМЕР

Виртуальная маркетинговая корпорация «Да Винчи» объединяет ряд горнорудных месторождений, производственные (машиностроительные и строительные), транспортные, инвестиционные, экологические системы. Подсистемы «Да Винчи» поставляются без доработок под конкретный объект (как детские конструкторы сборно-разборного типа). Один из сценариев, предлагаемых в проекте (Venture Management Model), моделирует следующую ситуацию. Горнодобывающая компания ведет разработки в Новой Гвинее. Построенный в этой местности отель может быть расширен для обслуживания растущего потока деловых клиентов этой компании, а также туристов. Консорциуму, имеющему бизнес в сфере коммуникаций и гостиничных услуг, предлагается долевое участие в развитии этой местности и эксплуатации отеля. Для снижения накладных расходов на расширение отеля и инфраструктуры туризма привлекаются крупные строительные компании (на условиях долевого участия в прибыли). Отметим при этом, что критерии эффективности бизнеса в таком составе различны и процесс принятия стратегических решений сопряжен с конфликтными интересами партнеров, динамически изменяющейся их картиной. Для реализации этой корпорации имеются электронная (мультимедийная) почта для поддержки процессов принятия решений первыми лицами, средства телеконференций для функциональных подразделений и аналитиков, геоинформационная система, САПР, взаимодействующая с СУБД через структуру данных с пространственной привязкой, система компьютерного делопроизводства на всех этапах. Используются современные технологии типа «клиент—сервер» и объектно-ориентированные под Windows NT, Windows-95 (рабочие места), Unix (сервер), полные версии MS Office и компьютерный документооборот. В системе электронного документооборота используются полнотекстовый поиск, доступ к проектной документации на всех этапах жизненного цикла проекта, подготовка интерактивной технической документации. Документ может содержать текст, например html-документ, иллюстрации в одном или нескольких слоях, редакторские правки и коммента-

рии участников различных рабочих групп, участвующих в проекте, трехмерные объекты из программ САПР, подключаемые к документу видео- и аудиофайлы.

Важное значение имеет интеграция задач автоматизации проектирования и оценки решений с компонентами системы.

ПРИМЕР

Взаимодействие между сервером базы данных и САПР в «Да Винчи» происходит с помощью интерфейса удаленного вызова функций, например: модули САПР из специализированного инструмента проектирования крупных архитектурных объектов становятся компонентами пользовательского интерфейса рабочих станций, доступного любому участнику проекта.

Базовым элементом систем виртуальной реальности является система интерактивной стереоскопической визуализации виртуального пространства и перемещающихся в нем создаваемых реалистических или существующих реально объектов. Она в отличие от систем интерактивной машинной графики обеспечивает полное погружение в компьютерную модель и выполнение прямых манипуляций с моделью в виртуальном трехмерном аудиальном и визуальном пространстве.

ПРИМЕР

Если в будущем создадут распределенную параллельную архитектуру компьютера, позволяющую каждой палочке и колбочке глазного яблока соотнести свой процессор, то станет возможной реализация принципа полного управления визуальным окружающим пространством. Необходимы также библиотеки объемного звука в трехмерном пространстве и (или) его синтезаторов. Сложная проблема — формирование тактильных ощущений (сопротивления внешних предметов), фактуры (шероховатости, влажности и т. д.). Имитация сопротивления достигается с помощью перчаток и костюмов со специальными тягами, звуковыми динамиками, пьезоэлементами, сплавами с памятью и др. Имитация ускорений (перегрузок, силы тяжести) достигается, например, управляемой центрифугой с независимым вращением по трем осям.

Возможности виртуальной реальности можно использовать в управлении организационными и техническими системами, в проектировании изделий, медицинских исследованиях, градостроительстве, обучении эксплуатации различных устройств и т. д. Основное достоинство технологии виртуальной реальности — уменьшение стоимости решения инфологических задач за счет сокращения натуральных объектов, а также обеспечение возможности решения новых задач, не решаемых обычной реальностью.

ПРИМЕР

Таковы многие задачи с эффектом телеприсутствия — телеконференции, виртуальные операции, штабные учения, доступ к зараженной или опасной местности, посещение выставок и музеев, реклама, дизайн и архитектура (архитектор и заказчик совместно создают проект дома, изменяя внутреннее оформление помещения, двигая в нем мебель, прогуливаясь по комнатам или коридору), виртуальное кино.

19.9. ТЕХНОЛОГИИ ИНФОРМАЦИОННОГО РЕИНЖИНИРИНГА

Технологии информационного реинжиниринга — методы и средства коренного переосмысления и/или перепроектирования информационных сетей и процессов с целью достижения резких, например «порядковых», улучшений в ключевых показателях информационных сетей и систем, в частности в показателях типа «производительность — стоимость», «время — объем информации», «функциональность — документация», «технология — удобство» и др.

ПРИМЕР

Сокращение длительности принятия решения и цикла подготовки и подписания документов, например, средствами компьютерного делопроизводства, сетями связи и экспертными системами, обеспечивающими доступ руководителя, принимающего решение, ко всем этапам, узлам и инструментариям подготовки решения; перенос акцента с проблемы «как делать?» на проблему «что делать?». Отметим, что популярную в информационных системах, сетях технологию «клиент—сервер» можно считать реинжиниринговой. Реинжиниринг системы подготовки докладов, отчетов, например, может свести цикл подготовки отчета с 20—30 операций до 5—10. Простое усовершенствование не может дать таких результатов. «Навешивание компьютеров на старые информационные структуры не есть компьютеризация, а тем более информатизация!»

Технология «Рабочая группа» — технология совместной работы нескольких связанных между собой общими информационными ресурсами компьютеров (рабочей группы), объединенных для решения какой-либо общей задачи.

ПРИМЕР

Примеры рабочих групп: «Дирекция», «Бухгалтерия», «Канцелярия». Компьютерная сеть организации может объединять несколько рабочих групп. У каждого компьютера рабочей группы имеется идентификатор, имя в группе, например, по ФИО человека, на нем работающего. В рабочей группе «Бухгалтерия» может существовать компьютер (рабочее место) «Главбух» или «Иванов Сергей Николаевич».

Рабочая группа может быть и временной — для работы над конкретным проектом в пределах определенного промежутка времени.

ПРИМЕР

Можно организовать рабочую группу «Презентация фирмы», состоящую из компьютеров сотрудников фирмы, подготавливающих презентацию своей фирмы или «Годовой отчет» — для подготовки годового финансового отчета фирмы. Все эти люди могут работать в разных отделах, но они составляют временную рабочую группу, чтобы было легче обмениваться информацией общего доступа при работе над отчетом.

Обмен информацией может осуществляться и между рабочими группами. Для этого не нужно физически перемещать компьютеры, чтобы сформировать рабочую группу: достаточно присвоить всем компьютерам группы ее имя.

ПРИМЕР

Операционная система Windows for Workgroups позволяет производить выделение компьютеров в рабочие группы при ее установке. Изменять состав и структуру рабочей группы затем можно из Панели управления, запустив прикладную программу Network (сеть). При этом все компьютеры одной сети независимо от их объединения в рабочие группы имеют доступ к общим принтерам и общим файлам, а такие приложения, как Mail (Электронная почта), Shedule+ (Ежедневник), работают только в пределах одной рабочей группы. Передача почты через Mail возможна только в пределах одной рабочей группы. Как правило, в небольших фирмах имеется одна рабочая группа.

Технология (модель взаимодействия) «клиент—сервер» — это технология взаимодействия компьютеров в сети, в которой каждый из компьютеров имеет свое рабочее назначение. Один, более мощный компьютер (сервер) в сети владеет и распоряжается информационными и аппаратными ресурсами (мощный процессор, файловая система, почтовая служба, база данных и др.), другой, менее мощный (клиент) имеет доступ к этим ресурсам лишь через сервер.

Этот принцип распространяется и на взаимодействие программ и информационных сред. Программа (среда), выполняющая предоставление соответствующего набора услуг, — сервер, а программа (среда), пользующаяся этими услугами, — клиент. Технология традиционной модели «клиент—сервер» модернизируется и совершенствуется.

ПРИМЕР

Сейчас говорят уже о принципиально иной концепции взаимодействия между элементами сети peer-to-peer (P2P), позволяющей отдельным компьютерам работать друг с другом напрямую.

19.10. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ И СРЕДО-ОРИЕНТИРОВАННЫЕ ТЕХНОЛОГИИ

Объектно-ориентированные технологии — технологии представления и актуализации информации, информационных процессов, систем как совокупностей объектов и классов с использованием следующих понятий: **объект**, **экземпляр класса** — все то, что может быть полно описано некоторыми атрибутами состояния; **класс** — совокупность объектов с одинаковыми атрибутами; **инкапсуляция** — скрывание внутренней (см. ниже) информации, т. е. возможность отделения объектов и классов от внешнего мира; **наследование** — возможность создавать из классов-родителей новые классы-потомки, сохраняющие атрибуты и свойства родителей; **полиморфизм** — способность объектов выбирать метод представления на основе типов данных, актуализируемых сообщений.

ПРИМЕР

Объектно-ориентированные среды программирования, например C++, Smalltalk; объектно-ориентированный инжиниринг, или набор приемов и методов проектирования бизнеса, наиболее эффективно обеспечивающего заданные цели и прибыль; объектно-ориентированный пользовательский интерфейс, использующий, например, понятия «класс описаний», «класс языков», «класс операционных сред» и др. При объектно-ориентированном программировании в среде языка APL, например, инструкции выполняются в соответствии с логикой и инструкциями некоторой программы, которая определяет последовательность и содержание действий; выполнение этой программы инициируется с помощью сообщения, посылаемого заданному объекту пользователем, другой программой или объектом. Получатель сообщения решает, какая программа будет выполнена.

Средо-ориентированные технологии — технологии, обычно интерактивные, для проектирования, разработки, актуализации информационных систем, в которых сначала строится нужная среда (инструментарий), а затем происходит ее автоматизированная настройка с помощью выполнения процедур типа переместить, вставить, удалить, указать, активизировать и др.; готовые среды, как кубики, во-первых, объединяются в нужные структуры, а во-вторых, настраиваются на конкретные классы проблем и/или пользователей, причем при изменении одних из них могут изменяться и другие.

ПРИМЕР

Средо-ориентированные системы программирования, в которых часто используются «оконный интерфейс», «оконная среда», основываются на понятиях «окно», «рамка», «фрейм», «рисунок на экране», «оконное меню» и др., каждое из которых связывается с наиболее подходящей инструментальной средой: текст-

ты — с текстовым процессором, таблицы — с электронной таблицей, графики — со средой деловой графики и т. д. К этим типовым средам могут быть добавлены также и разработанные самим программистом среды, а также библиотеки сред. Отметим, что интерактивное планирование позволяет находить оптимальные структуры и набор ресурсов для достижения поставленной цели.

HTML — статичное средство. Чтобы «оживить» содержимое Web, сделать интерактивные html-страницы, используется среда JavaScript. Видимо, одной из наиболее важных для информатики объектно-ориентированных систем является java-система, сред — java-интерпретирующая машина, технологий — java-технология. Рассмотрим их кратко. В их основе лежит язык программирования Java, ориентированный на сеть Internet и серверы WWW. Язык Java произошел от языка программирования Oak с синтаксисом, близким к синтаксису языка C++. Средствами языка Java можно разрабатывать приложения для различных платформ: Intel Pentium, Macintosh, Sun и др. Java-программы бывают автономного использования (выполняемые в режиме интерпретации на конкретной компьютерной платформе) и апплеты — applets (выполняемые в режиме интерпретации виртуальной java-машиной, встроенной практически во все современные браузеры). Апплеты Java встраиваются в документы HTML, хранящиеся на сервере WWW. С помощью апплетов можно сделать страницы сервера Web динамичными и интерактивными. Все данные для обработки апплеты могут получить только от сервера Web.

Язык Java является объектно-ориентированным и имеет объемную библиотеку классов, значительно упрощающих разработку приложений, так как программист больше внимания может уделить функциональной части приложения, а не организации интерфейса, динамических массивов и т. п. В широком смысле слова Java — это технология, изначально рассчитанная на интеграцию с сетевой web-средой, полностью независимой от платформы.

Виртуальная java-машина — машина, на которой исходные java-программы интерпретируются в коды этой машины. Это делает java-среду мощным и удобным средством разработки клиентских web-приложений.

ПРИМЕР

В java-среде пользователь может осуществлять динамическую загрузку объектов из сети, т. е. не нужны дорогостоящие работы по наладке, администрированию клиентских java-систем, так как для обеспечения работы клиента на новой версии достаточно загрузить ее на сервере.

Java-программа компилируется в двоичный код, состоящий из команд виртуального процессора Java и предназначенный для выполнения java-интерпретатором. Разрабатываются аппаратные реализации, физические процессоры для выполнения этого кода, хотя

на всех основных компьютерных платформах имеются свои виртуальные java-процессоры. Java-приложение не обращается напрямую к интерфейсу операционной системы, а пользуется стандартными библиотеками организации пользовательского интерфейса, обращения к файлам, работы в сети и т. д. Загрузочные модули этих библиотек поставляются вместе с виртуальной java-машиной, поэтому программисту не нужно об этом заботиться. Если web-страница содержит java-апплеты, передаваемые по сети, то нужна компактность java-кода и быстрота его загрузки. Имеются инструментальные среды, например Java Studio, позволяющие проектировать приложения вообще без программирования, из готовых компонент, устанавливая между ними связи и отношения в соответствии с внутренней логикой приложения. Для повышения производительности java-приложений в браузерах используется компиляция Just-In-Time compilation (на лету). При первой загрузке апплета его код транслируется в обычную исполняемую программу, которая сохраняется на диске и запускается. В результате общая скорость выполнения апплета увеличивается в несколько раз.

В последние годы возрос интерес к распределенным системам — программным комплексам, составные части которых функционируют на разных компьютерах в сети, используя при взаимодействии технологии различного уровня — от непосредственного использования пакетов TCP/IP до технологий с высоким уровнем абстракции, таких, например, как CORBA. Эти системы обеспечивают следующие требования, не выполнимые при использовании традиционных технологий:

- масштабируемость, т. е. эффективное обслуживание разного числа клиентов одновременно;
- надежность создаваемых приложений, т. е. устойчивость не только к ошибкам пользователей, но и к сбоям в системе коммуникаций;
- непрерывную длительную работу (режим 24×7, т. е. 24 ч в течение 7 дней недели);
- высокий уровень безопасности системы, т. е. защиты и отслеживания информации на всех этапах функционирования;
- высокую скорость разработки приложений и простоту их сопровождения и модификации (достаточен средний уровень программиста).

Технология CORBA создавалась как универсальная технология создания распределенных систем с использованием языков Java, C, Ada, Smalltalk, Delphi, Perl, Python и др. Клиентская часть может быть написана на любом языке программирования, поддерживающем CORBA. CORBA — стандартная инфраструктура разработки и использования различных платформ, ОС и приложений, поэтому все спецификации CORBA являются открытыми. CORBA реализует высокий уровень абстракции — все проблемы и описания взаимодействия с операционной системой или сетевыми средствами осуществляются на низком уровне и скрытно от прикладного программиста.

Высокий уровень абстракции достигается за счет отображения инструкций на языке спецификаций на конкретный язык программирования. CORBA может передавать данные различных типов: структуры, объединения и др. Предусмотрена система описания и контроля типов. Для каждого языка используется свое отображение данных на языке спецификаций. CORBA поддерживает статический и динамический способ организации удаленных вызовов и имеет развитые средства получения информации о серверах. CORBA имеет высокий уровень устойчивости к сбоям за счет большей изоляции клиентов и серверов, автоматического сохранения состояния объектов, более мощной и продуманной схемы управления транзакциями. Управление транзакциями берет на себя так называемый Сервис Управления Транзакциями CORBA (Object Transaction Service). CORBA обеспечивает высокий уровень безопасности: предусмотрена идентификация пользователя, списки прав доступа к ресурсам, система аудита и многое другое.

19.11. CASE-ТЕХНОЛОГИИ

CASE-технологии (Computer-Aided System Engineering) — автоматизированное проектирование информационных систем или технологий, позволяющие автоматизировать основные этапы и процедуры жизненного цикла информационных систем — от анализа исходного состояния и целей до проектирования интерфейсов, привычных проектировщику, пользователю, и основных процедур функционирования системы; чем больше этапов и процедур автоматизируется, тем более развитой получается информационная система и тем шире обычно ее приложения.

ПРИМЕР

Технология STRADIS (STRategic Architecture for the Deployment of Information Systems — стратегическая архитектура для развертывания информационных систем) определяет и поддерживает основные этапы жизненного цикла системы: цели, их приоритеты, требования к ресурсам, распределению работ, составу и содержанию проектной документации, методику выполнения процедур проектирования и программирования, тестирования и управления. Включает в себя следующий инструментарий: графический редактор (графическая среда), СУБД, средства описания сценариев диалога с системой, выходных документов и др.

19.12. НЕЙРОТЕХНОЛОГИИ

Нейротехнологии — технологии на базе моделей, методов, алгоритмов, программ, моделирующих, имитирующих нейронные сети и процессы решения задач искусственного интеллекта, позволяют эффективно реализовывать параллелизм, самообучение, распознавание и классификацию, адаптивность, перестройку структуры, топологии.

ПРИМЕР

Одной из распространенных зарубежных нейросистем является пакет Brain Maker. Пусть необходимо решить задачу прогноза цены закрытия на сегодняшних торгах по валютным трехмесячным фьючерсным контрактам. Пусть нас устраивает точность прогноза, при которой правильно указывается ценовой тренд (подъем, спад) и изменение цены с точностью не ниже 90% от последнего скачка. Применение нейронной сети начинается с подготовки входных данных: курс доллара, индекс инфляции, ставка межбанка, биржевые индексы, объем торгов, количество сделок, максимальные и минимальные цены и др. После настройки сети начинается итерационный процесс обучения, в результате которого нейросеть настраивает свою логическую структуру для точной реакции рынка на те или иные воздействия. Для этого в пакете Brain Maker предусмотрен мощный аналитический блок, который позволяет увидеть, какие параметры оказывают позитивное влияние на ситуацию, а какие — негативное. Затем сеть снова обучается, далее тестируется на качество и адекватность и после удачного тестирования используется для прогнозов. За десять биржевых дней сеть ни разу не ошиблась в знаке отклонения фьючерсных котировок, а девять дней из десяти отклонение прогноза от реальной цены составило менее 10 р. В этом пакете реализована многослойная нейронная сеть с возможностью настройки всех параметров нейронных сетей и алгоритмов обучения.

19.13. КОГНИТИВНЫЕ ТЕХНОЛОГИИ

Когнитивные технологии — методы, средства и приемы, обеспечивающие визуальное, гипермедийное представление условий задач и/или предметной области, которое помогает находить стратегию решения или само решение либо позволяет оценивать и сравнивать пути решения, принять то или иное адекватное решение.

ПРИМЕР

Когнитивная графика, позволяющая геометрически, образно представлять предметную среду и построить, исходя из этого, требуемый графический объект, в частности пространственное представление этого объекта. Есть и средства, и методы визуального программирования (проектирования программ), в частности среда Visual-C. Когнитивные методы принятия решений в области бизнеса позволяют принимать решения и определять стратегии поведения на основе качественных данных, личностных суждений (эффективно для ликвидации неопределенностей). Например, модель принятия решений Института США и Канады РАН, используя блоки, подмодели типа «Мир», «Ценность», «Средство», «Интерес», «Стереотип», «Цель», «Сцена-

рий», «Проблема», позволяет изменять содержательное наполнение этих блоков, генерировать новые цели и сценарии, используя старые.

19.14. НЕЧЕТКИЕ ТЕХНОЛОГИИ

Нечеткие технологии — технологии обработки данных и вывода знаний, принятия решений на основе описания систем аппаратом нечетких множеств и нечеткой логики.

ПРИМЕР

Медицинский диагноз часто основан на нечетких, не определенных явно связях симптомов и болезней, их нечеткой зависимости, поэтому для компьютерной постановки диагноза, построения экспертной системы постановки диагноза эта технология особо эффективна, так как позволяет делать нечеткие выводы, которые затем могут быть проверены. Проблемы дактилоскопии также могут быть решены эффективно с помощью нечетких систем распознавания отпечатков.

Все новые информационные технологии так или иначе используют методы и проблематику искусственного интеллекта, инженерии знаний.

Проблематику искусственного интеллекта составляют знания, информация о проблемной области, которые пока объективно непонятны, неточны, неформализуемы, неструктурируемы, неактуализируемы доступными средствами (и могут стать такими в процессе функционирования системы, приобретения знаний).

Инженерия знаний — наука, изучающая проблемы выявления, структурирования, формализации и актуализации знаний для разработки различного типа интеллектуальных систем, технологий.

Все информационные технологии — способ актуализации информации, основа мышления.

Основные тенденции развития новых информационных технологий независимо от сферы их использования:

- возрастание роли и актуальности информационного ресурса;
- появление стандартов взаимодействия типа «человек—технология—человек», «система—технология—система» и др.;
- устранение этапов и функций посредников информационного обмена, ликвидация промежуточных функций внутри компаний и между ними, более широкое их распространение, упрощение доступа, снижение цен и т. д.;
- глобализация и интеграция;
- формирование рынка новых информационных технологий.

ПРИМЕР

В 1990 г. около 40% интеллектуальных работников в США использовали на своих рабочих местах новые информационные технологии; в частности, концерн Microsoft инвестирует в новые медиапроекты до 20% своего научного бюджета, выпус-

кая энциклопедии и справочники на CD, открывая в Интернете свои мультимедийные журналы, например Slate (<http://www.slate.com>) и др. Выполнение японской программы создания компьютеров пятого поколения сдерживается тем, что новая архитектура программного обеспечения пока не сочетается с существующими центрами искусственного интеллекта, новые протоколы не могут быть использованы в старых системах связи, а новые машинные языки не подходят для старых систем и т. д. В банковской сфере уменьшается роль мелких банков, так как на внедрение информационных технологий требуются значительные ресурсы. Имея доступ к терминалам общего пользования, можно заказать товары и получить электронные купоны. Примером глобальной программы является программа ESPRIT (Европейская стратегическая программа исследований в области НИТ).

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое АРМ, САПР? Приведите примеры их применения.
2. Что такое электронная почта, теледоступ? Приведите примеры их применения.
3. Что такое база данных, база знаний, экспертная система? Приведите примеры их применения.
4. Что такое машинная графика? Приведите примеры ее применения.
5. Что такое мультимедиа, гипертекст, гипермедиа? Приведите примеры их применения.
6. Что такое объектно-ориентированная и средо-ориентированная технологии? Приведите примеры их применения.
7. Что такое виртуальная реальность? Приведите примеры ее применения.

ЗАДАЧИ

1. Постройте базу данных табличного типа о юридических лицах и налоговых платежах с них.
2. Постройте базу данных страничного типа о сделках купли-продажи акций ряда банков, выпущенных в валютном исчислении, и осуществите с помощью этой базы запись в базу данных сумм налогов с каждого из участников сделки.
3. Выполните операции сортировки данных базы, по возможности упростив структуру базы данных.
4. Постройте несколько макетов (логических моделей) баз знаний по социально-экономической предметной области.
5. Постройте несколько макетов (логических моделей) экспертной системы по социально-экономической проблеме. Приведите примеры проблем, которые можно решить эффективно с помощью экспертной системы.

6. Осуществите постановку некоторых задач, которые можно решить с помощью базы знаний и (или) экспертной налоговой системы.
7. Постройте несколько сценариев проведения телеконференций по различным налоговым проблемам. Опишите работу организатора, модератора и пользователя телеконференции.

ТЕСТ

1. «НИТ — технология ...»:
 - а) обработки информации;
 - б) получения информации;
 - в) передачи информации;
 - г) получения, обработки, использования информации.
2. НИТ не является:
 - а) радио; б) e-commerce; в) мультимедиа; г) e-mail.
3. НИТ является:
 - а) Word; б) Excel; в) PowerPoint; г) e-commerce.
4. Обычная для работы рабочей группы в компьютерном офисе технология:
 - а) «клиент—сервер»;
 - б) «сервер—сервер»;
 - в) «клиент—клиент»;
 - г) автономная работа каждого пользователя.
5. АРМ — это:
 - а) автоматизация работ менеджера;
 - б) автоматизированное рабочее место;
 - в) автоматизированная регистрация модулей;
 - г) автоматическое рабочее место.
6. База данных — это большой набор:
 - а) разнообразных и разнородных данных на диске;
 - б) структурированных данных на диске;
 - в) структурированных данных с СУБД;
 - г) структурированных данных в ОЗУ.
7. Электронная таблица — это программа:
 - а) рисования шаблонов таблиц;
 - б) обработки данных табличной структуры;
 - в) заполнения и распечатки таблицы данных;
 - г) выдачи данных в табличной форме.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
г	а	г	а	б	б	б

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Компьютерный и виртуальный офис. Организация и управление, взаимодействие сотрудников и посетителей.
2. Теле- и видеоконференции. Организация телеработы и телеприсутствия.
3. Модели и базы данных. Организация удаленного доступа к серверу баз данных. Интеллектуальный анализ данных.
4. Модели представления и актуализации знаний.
5. Экспертные системы и их приложения. Организация взаимодействия разработчиков и пользователей.
6. Системы машинной графики и их использование. Основные формы и методы (алгоритмы) машинной графики.
7. Когнитивные технологии. Визуальные системы и среды.

20

КОМПЬЮТЕРНЫЕ СЕТИ И СИСТЕМЫ

Развитие компьютерных сетей, их интеграция с существующими международными телекоммуникационными сетями — важнейшая задача национального масштаба. В последнее время в мире наблюдается устойчивая тенденция ускоренного развития индустрии телекоммуникаций и инвестиций в нее, ибо именно информация становится основным стратегическим ресурсом страны, фактором социально-экономического роста. Кто полнее использует современные средства компьютерных телекоммуникаций и новые информационные технологии, их сетевые приложения (электронную почту, дистанционное обучение, мультимедиа, телевидение, телеконференции, визуализацию, моделирование, компьютерную графику и др.), тот будет опережать и в остальных сферах. Доступная для оперативного воспроизводства средствами компьютерной обработки информация превращается в важнейший фактор социального развития общества. Развивая глобальные системы телекоммуникаций и средства мобильных вычислений, общество создает новую универсальную информационную среду — инфосферу.

Термин **сеть** означает совокупность компьютеров, соединенных между собой в соответствии с одной из стандартных типовых топологий (форм соединения) и использующих для передачи пакетов данных, сообщений общую базовую сетевую технологию.

Основные функции любой сети:

- обеспечение доставки пакетов сообщений в сети;
- обеспечение надежной структуры и автоматического конфигурирования сети, соответствующей структуре информационных потоков, с помощью мостов и маршрутизаторов, а также надежного трафика (загрузки) и сервиса сети, учитывающего интенсивность обмена данными между сегментами;
- согласование и поддержка различных протоколов (принципов передачи информации) разного типа, в частности базовых технологий (Ethernet, Token Ring, FDDI) и образования транспортной системы, объединяющей несколько сетей с различными протоколами между конечными узлами.

Реализация протокола сетевого уровня осуществляется в сети с помощью специального устройства — **маршрутизатора**, объединяющего отдельные сети в составную сеть. К каждому маршрутизатору может присоединяться не менее двух сетей. В сложных составных сетях почти всегда существует несколько альтернативных маршрутов для передачи пакетов между двумя конечными узлами. Задачу выбора оптимального в каком-то смысле (обычно по времени, с минимумом узлов) маршрута из нескольких возможных решают маршрутизаторы и конечные узлы. **Маршрут** — это последова-

тельность маршрутизаторов, которые должен пройти пакет от отправителя до пункта назначения. Внутри одной сети сегменты не разделяются маршрутизаторами, иначе это была бы не одна сеть, а несколько сетей. Маршрутизатор соединяет несколько сетей в интернет. Чтобы иметь информацию о текущей конфигурации сети, маршрутизаторы обмениваются маршрутной информацией между собой по специальному протоколу обмена маршрутной информацией, или протоколу маршрутизации. Для того чтобы доставить удаленному маршрутизатору пакет протокола обмена маршрутизации, используется протокол сетевого уровня, так как только он может передать информацию между маршрутизаторами, находящимися в разных сетях. С помощью таких протоколов маршрутизаторы составляют карту межсетевых связей и принимают решение о том, какому следующему маршрутизатору в маршруте нужно передать пакет для образования рационального пути.

Для построения сетей используются концентраторы (хабы), к которым подсоединяются все устройства сети независимо от их назначения. Подсоединение ведется либо для связи «вверх» (с хабом более высокого уровня), либо для связи «вниз» (с хабом более низкого уровня). Для обеспечения защиты данных есть два режима работы каждого порта: конфиденциальный (каждый порт может получать лишь те сообщения, которые адресованы непосредственно ему) и публичный (может получать все сообщения, а этот режим используется для подключения, например, маршрутизаторов). Когда конечный узел готов передать пакет сообщения, он отправляет хабу запрос обычного или высокого приоритета. Если узлу нечего передать, он отправляет сигнал «свободен». Хаб циклически спрашивает подключенные узлы, выясняет их готовность к передаче и анализирует их запросы, опираясь на два критерия — приоритет запроса и физический номер порта, к которому присоединен передающий узел. Вначале обрабатываются запросы высокого приоритета, например от приложений, требующих немедленную реакцию, в частности мультимедиа-приложения. После этого обрабатываются запросы с обычным приоритетом в соответствии с физическим адресом порта. Чтобы обеспечить гарантированный ответ, запросу с ожиданием в несколько сотен миллисекунд присваивается высокий приоритет. При опросе порта, к которому подключен хаб нижнего уровня, опрашиваются его порты и только затем возобновляется опрос портов старшего хаба. Таким образом, все конечные узлы опрашиваются последовательно, независимо от уровня хаба, с которым они соединены.

Рассмотрим ряд других наиболее известных и используемых компьютерных сетей.

ПРИМЕР

LAN, или **ЛВС**, — локальная вычислительная (компьютерная) сеть (Local Area Network) состоит из компьютеров, удаленных в пределах одного здания друг от друга. Связанные между со-

бой кабелем компьютеры и периферийные устройства обмениваются данными. Преимущество перед сетью с сервером: если из строя выходит один из компьютеров, то это никак не отражается на работе остальных компьютеров. ЛВС широко применяется в системах автоматизированного проектирования, управления производством и технологическими процессами, в офисных системах и т. д. ЛВС объединяет компьютеры, расположенные в офисе, на предприятии, примерно на расстоянии до нескольких километров. Сеть MAN (Metropolitan Area Network) — городская или региональная сеть, т. е. сеть в пределах города, области, неразветвленной корпорации. Системы, в которых средства передачи данных принадлежат одной компании и используются только для нужд этой компании, называются **корпоративными сетями** (Enterprise Network). Основное назначение ЛВС — распределение ресурсов: программ, периферийных устройств, терминалов, памяти. ЛВС основывается на следующих принципах: единая передающая среда; единый метод управления; единый протокол связи; гибкая модульная организация; информационная, программная и (частично) аппаратная совместимость.

ПРИМЕР

FidoNet — любительская компьютерная сеть, в которой для организации компьютерного узла достаточно компьютера, модема и простого программного обеспечения. Несмотря на эти скромные ресурсы, FidoNet имеет разветвленную структуру. Используются существующая телефонная сеть и BBS (Bulletin Board System), электронная доска объявлений — место (ящик), где пользователи могут оставлять свои письма, файлы. FidoNet — иерархическая структура: на нижнем уровне — узел, в городе (небольшом регионе) — сеть с сетевым координатором, в государстве (большом регионе) — региональное объединение с региональным координатором, на материке — зона с зональным координатором. Основные зоны в FidoNet — Северная Америка, Европа, Австралия, Латинская Америка, Африка и Азия. Существуют и номерные FidoNet-подобные сети с условными номерами (сети, зоны), например зона 105, 092 и т. д. Общей работой FidoNet управляет международный координатор. Имеется свод основных законов работы и управления в FidoNet, а также правила передачи и протоколы сообщений. Это более дешевая и нетребовательная к знаниям и умениям сеть (форма коммуникации). Информацию о режимах работы BBS можно получить из специального списка BBS (NodeList) — еженедельно обновляемого файла с адресами всех узлов FidoNet. Каждый пользователь имеет определенный уровень доступа, от которого зависит количество выделяемого времени для работы, лимит на переписывание файлов, режим доступа и т. д.

ПРИМЕР

VPN — виртуальные частные сети (Virtual Private Network), или некоторый кластер компьютеров, использующих в качестве каналов передачи информации уже существующие сети, в частности Интернет. Эта виртуальная сеть организуется соглашением о передаче информации в зашифрованном виде, что выделяет (виртуально) эти компьютеры в сеть. На каждом компьютере сети устанавливается приложение-клиент для шифровки информации и организации защиты, а какой-либо компьютер сети выполняет функции сервера. Компьютеры такой сети могут быть географически размещены где угодно. Кроме того, появляется дополнительная VPN-защита.

В информационном обществе потребность высока, много потребителей информации, и без надежной и оперативной информации нельзя идти в ногу со временем, развивать науку и технику на уровне лучших мировых образцов, т. е. все мы — потенциальные пользователи Интернета.

Интернет ведет свою историю от исследовательского проекта американского оборонного ведомства DARPA (Defence Advanced Research Projects Agency) по разработке эффективного метода связывания ряда военных и правительственных сетей с целью расширения их сервисных возможностей. Интернет должен был объединить, а не заменить существующие сети (например, ARPANET, ARPA) с помощью коммуникационных процессоров и шлюзов, запоминающих, обрабатывающих и направляющих сообщения в соответствии с сетевым адресом их получателя и текущей структурой сети по принципу «запомнить и передать» (store and forward).

Так как заказчиком прототипа Интернета являлось военное ведомство США, то его позиция определила приоритеты целей при проектировании сети (перечисляем их по степени важности в задании DARPA):

- **выживаемость сети** — взаимодействие через Интернет продолжается и в том случае, когда выйдут из строя отдельные участки сети или часть коммуникационных процессоров;
- **поддержка различных типов стандартного сервиса** — удаленный доступ, передача файлов, разделение сетевых ресурсов;
- **интеграция различных типов сетей;**
- **распределенное оперативное управление ресурсами** при централизованном контроле за распределением сетевых адресов;
- **эффективность разработки** по времени и стоимости;
- **простота подключения** отдельных устройств и пользователей;
- **оперативный учет всех ресурсов сети.**

Выживаемость — самая главная эволюционная цель любой системы.

ПРИМЕР

При уничтожении командного пункта запуска стратегической ракеты должна была быть обеспечена возможность запуска ее с другого, удаленного командного пункта, несмотря на возмож-

ное частичное разрушение сети и выход из строя части коммуникационных процессоров.

В процессе реализации проекта Интернета оказалось, что предложенная модель взаимодействия между компьютерами может эффективно применяться и для создания сетей различного назначения — образовательных, коммерческих, научных, исследовательских и др., т. е. для создания на базе Интернета технологии глобальной информационной инфраструктуры, в результате чего Интернет стал международной средой. Сегодня Интернет объединяет более 100 стран, но при этом США сохраняют контроль над использованием базовых ресурсов этой системы, например системы распределения адресов ее абонентов (выделение 32-разрядного адреса каждому подключаемому компьютеру).

Интернет — система, объединяющая множество взаимосвязанных сетей со стандартизированными едиными протоколами внутри- и межсистемного обмена и доступа к информации в режиме реального времени. Это «сеть сетей», включающая государственные, региональные, ведомственные, образовательные, коммерческие и другие сети.

Интернет — организация с добровольным участием, явно управляемая подобием Общественного совета директоров. Составляющие сети могут иметь своих президентов, но в Интернете нет такой должности и личности. Высшая власть остается за ISOC (Internet Society) — добровольным обществом, способствующим глобальному обмену информацией через Интернет. Оно отвечает за техническую политику, поддержку и управление в Интернете, определяет и распределяет ресурсы, например адреса, способы общения между компьютерами и прикладными программами, стандарты (протоколы) общения; когда требуется стандарт, совет рассматривает проблему, принимает стандарт и по сети оповещает о нем мир.

ПРИМЕР

Любой компьютер в Интернете имеет свой уникальный 32-разрядный двоичный адрес (IP-адрес). Функции присваивания адресов по определенным правилам (разработанным ISOC) осуществляет IAB (Internet Architecture Board), или Совет по архитектуре Интернета.

Рабочие группы ISOC имеют различные функции: выпуск документации, выработка стратегии действий, перспективные исследования, разработка новых и модификация используемых стандартов и протоколов. Если какая-то сеть присоединяется к разработанным положениям среды Интернета и захочет считать себя ее частью, то она может это осуществить, пока не сделает ничего такого, что может навредить Интернету (в последнем случае она может быть временно или постоянно исключена из Интернета).

ПРИМЕР

Часто нарушаемые правила, положения интернет-этики:

— коммерческое использование сетей и ресурсов, некоммерческих

- по организации, назначению, субсидиям, например образовательных;
- нарушение экспортных, государственных, политических, морально-этических законов и норм при пересылке информационных ресурсов через национальные границы, например, вредоносных для страны;
 - нарушение авторских или лицензионных соглашений и права интеллектуальной собственности, как, например, использование нелегального программного обеспечения.

Интернет не собирает плату (нет просто организации, которая могла бы этим официально заниматься). Каждая сеть взимает свою плату за обеспечение связи, а каждый пользователь платит за использование такой конкретной сети. Представители сетей собираются вместе (реально или виртуально) и решают, как им соединяться друг с другом и поддерживать в актуальном режиме эти связи. Корпорация платит за ее подключение к некоторой, например, региональной сети, которая, в свою очередь, платит за свой доступ сетевому владельцу государственного масштаба. Конечный пользователь платит (если он должен платить) уже за связь корпорации.

Для того чтобы воспользоваться услугами Интернета, не нужно знать, как взаимодействуют и поддерживают межсетевые связи различные сети. Каждая сеть имеет свой собственный сетевой эксплуатационный центр. Каждый такой центр связан с другими и знает, как разрешать разные проблемы. Архитектура сетевых протоколов TCP/IP (Transmission Control Protocol/Internet Protocol), на базе которых построен Интернет, предназначена специально для разнородных подсетей, соединенных друг с другом шлюзами. В качестве подсетей могут выступать разные локальные сети, различные национальные, региональные и специализированные сети, другие глобальные сети. К этим сетям могут подключаться машины совершенно разных типов. Каждая подсеть может принять информацию и доставить ее по указанному адресу в этой подсети. Два компьютера одной подсети могут напрямую обмениваться, но если возникает необходимость передать сообщение компьютеру другой подсети, то, используя межсетевой язык — протокол IP, они передают сообщение по определенной цепочке шлюзов и подсетей.

Протокол TCP обеспечивает надежный поток данных между двумя узлами сети, а протокол IP обеспечивает базовые блоки (дейтограммы), из которых могли бы строиться другие типы транспортного сервиса.

Доступ в Интернет обычно получают через поставщиков информационных и коммуникационных услуг (провайдеров).

Современные сети построены по многоуровневому принципу. Чтобы организовать связь двух компьютеров, требуются правила и язык их общения, обмена сигналами. Эти правила и определения называются протоколом. Для работы сетей есть множество различных протоколов: управления физической связью, установления сетевой связи, доступа к различным ресурсам и т. д. Многоуровневая

структура спроектирована с целью упрощения и упорядочивания этих отношений как на виртуальном, так и на реальном (физическом) уровне.

ПРИМЕР

Числовой адрес компьютера в Интернете аналогичен почтовому индексу отделения связи. Первые цифры индекса говорят о регионе (имеется соответствующая таблица кодов регионов на указанных компьютерах), последние две цифры — это номер почтового отделения в городе, области, районе. Промежуточные цифры могут относиться как к региону, так и к отделению, в зависимости от территориального деления и вида населенного пункта. Адрес в Интернете имеет вид: 125.47.66.89 или 91.87.654.2. Правый конец адреса говорит сети, какой компьютер (хост) должен получить сообщение. Интернет знает (т. е. это знают специализированные ЭВМ в Интернете, которые открыты для доступа и поиска), где искать указанную сеть, а сама сеть знает, где в ней находится конкретный компьютер сети. Для определения, где в локальной сети находится компьютер с данным числовым IP-адресом, локальные сети используют собственные протоколы сетевого уровня, протоколы различения адресов.

Протокол TCP в Интернете делит передаваемую информацию на части и нумерует каждую часть, чтобы позже восстановить порядок. Чтобы пересылать эту нумерацию вместе с данными, он сопровождает каждую часть информации конвертом, который содержит соответствующую информацию. Это и есть TCP-конверт. TCP-пакет помещается в отдельный IP-конверт, и получается IP-пакет, который и передает сеть. Получатель при получении IP-конвертов распаковывает их и видит TCP-конверты, распаковывает затем и эти конверты, размещая данные в последовательности полученных частей в соответствующие места по прислапным порядковым номерам. Если чего-то недостает, сеть требует заново переслать этот кусок. Таким образом, информация собирается в нужном порядке и полностью, после чего пересылается выше к пользователю (например, на экран, на диск). При искажениях данных (из-за помех на линиях связи) TCP использует собственную систему кодов, исправляющих ошибки, например код четности с добавлением к каждому пакету контрольной суммы (и к каждому байту бита проверки на четность). При помещении в TCP-конверт вычисляется контрольная сумма, которая записывается в TCP-заголовок. Если при приеме вычисленная сумма не совпадает с той, что указана на конверте, то, значит, в пути были искажения, и этот пакет пересылается снова.

Предоставление услуг Интернета построено по технологии «клиент—сервер». Роль клиента выполняет компьютер пользователя, а роль сервера — компьютер достаточной мощности в межсетевой связи. Клиент и сервер являются, по сути, частями системы,

взаимодействующими виртуально в сети. Сервер по командам клиента выполняет определенные действия с информацией, например пересылает клиенту файл. Взаимодействие клиента и сервера описывается соответствующими стандартными протоколами, поэтому клиент и сервер могут работать и на разнородных компьютерах (Macintosh, IBM и др.).

В приложениях Интернета пользуются системными именами вместо числовых адресов.

Доменная система имен — система имен, создаваемых путем передачи сетевым группам ответственности за их подмножество имен. Каждый уровень системы называется доменом.

Домены в именах отделяются друг от друга точками. В имени может быть различное количество доменов, но практически их 2—4. По мере движения по доменам слева направо в имени количество имен, входящих в соответствующую группу, возрастает.

Первым в имени стоит название рабочего места (рабочей машины) — реального компьютера с IP-адресом. Группа входит в более крупное подразделение, которое, в свою очередь, является частью национальной, государственной сети. Для США наименование страны по традиции опускается, вместо него имеются домены сетей образовательных (edu), коммерческих (com), государственных (gov), военных (mil) учреждений, а также домены сетей других организаций (org) и сетевых ресурсов (net). Каждая страна имеет свой домен: Канада — ca, США — us и т. д.

ПРИМЕР

Рассмотрим доменное имя ns.kbsu.ru. По адресу ns.kbsu.ru любой компьютер может запросить помощь у сервера-узла, обладающего соответствующей базой данных, в число обязанностей которых входит обслуживание такого рода запросов. В приведенном доменном имени это сервер (новостей) КБГУ, домен ns. Сервер начинает обработку имени с правого его конца и двигается по нему влево (от домена ru части национальной сети, например России, до домена kbsu интернет-центра КБГУ, постепенно сужая поиск до домена ns, который, собственно, определяет конкретную ЭВМ — почтовую ЭВМ в интернет-центре).

Части доменного имени говорят о том, кто ответствен за поддержку этого имени, но они могут вообще ничего не сообщать о владельце компьютера, соответствующего этому IP-адресу, или даже (несмотря на код страны) где же эта машина находится. Можно иметь в Африке машину с именем ns.kbsu.ru.

ПРИМЕР

Одна из версий домашней страницы автора была размещена по адресу <http://kaziev.w3.to> с доменом to африканской страны Того, но сервер находился, разумеется, не в этой стра-

не. Другая версия уже была расположена в России (<http://www.kaziev.pp.ru>, <http://www.kaziev.by.ru>).

Части доменного имени даже не всегда указывают локальную сеть, в которой находится компьютер. Часто доменные имена и сети перекрываются, и жестких связей между ними нет: две ЭВМ одного домена могут не принадлежать одной сети, у ЭВМ может быть много имен.

Наиболее распространенные стандартные возможности Интернета:

- удаленный доступ (**remote login**) с помощью протокола эмуляции удаленного терминала (**telnet**) — работа на удаленном компьютере в режиме, когда ваш компьютер эмулирует (имитирует) терминал удаленного компьютера, т. е. доступно все, что можно делать с терминала удаленной машины;
- передача файлов (**ftp**) — передача файлов любой структуры и содержания с одного компьютера на другой в реальном масштабе времени, поиск файла на удаленной машине, т. е. переход из директории в директорию, просмотр содержимого файлов; пересылка данных в файлах либо как двоичной информации, либо как текста с возможностью автоматической перекодировки данных при пересылке текста на компьютер с другой кодировкой, что сохраняет прежний читаемый вид текста, и т. д.;
- электронная почта (**e-mail**) — распространение информации (почты) от отправителя к адресату без учета реального времени (без синхронизации отправки и получения по времени и пространству); с ее помощью можно посылать сообщения, получать их в своем электронном почтовом ящике, отвечать на письма отправителей автоматически, рассылать копии письма, перенаправлять полученное письмо на другой адрес и т. д.

Основные нестандартные и популярные возможности Интернета:

- виртуальные магазины и торговые площадки (порталы);
- виртуальные биржи;
- виртуальный маркетинг;
- виртуальный банкинг;
- виртуальное, дистанционное обучение и консалтинг;
- виртуальные библиотеки (медиатеки) и журналы, книги, подписки;
- виртуальный поиск работы и партнеров;
- виртуальное интерактивное общение на темы по интересам (чаты).

Экспорт информации в Интернете подпадает под ответственность и контроль соответствующего отдела по экспортным ограничениям, хотя в Интернете вроде бы и нет таможенной подсистемы. Тем не менее экспорт данных требует лицензии, и, например, пересылка файла, содержащего данные, которые нельзя, в частности, вычитать в публичных библиотеках, электронной почтой должна быть прикрыта экспортной лицензией. Что касается экспортируемо-

го и импортируемого документа, необходимо прежде всего узнать правообладателя этой информации и иметь соответствующее разрешение. Даже если вы не знаете, чьего разрешения вам добиваться, необходимо помнить, что права собственности могут стать проблемой даже при использовании общих файлов (public, publicly available files). Некоторые программы, доступные в Интернете, например обновления антивирусных программных систем с помощью (анонимного) ftp, позволяют получать эти обновления, но вы должны иметь лицензию на их использование.

Развитие Интернета, информационные технологии, разработанные для Интернета, и их успешное применение в рамках локальных сетей организаций (ЛВС) привели к появлению термина Интранета и вызвали рост корпоративных Интранет-сетей. Этому способствовала и необходимость развертывания новых инструментов для повышения эффективности глобальных коммуникаций. В Интранет-сетях используется то же аппаратное и программное обеспечение, те же протоколы и принципы, что и в Интернете. По существу, Интранет — это мини-Интернет. Хорошо спроектированная Интранет-сеть предоставляет широкие корпоративные возможности работы с информацией, способна значительно повысить эффективность и прибыльность корпоративной деятельности за счет того, что имеет такие возможности (приложения), как сетевые электронные доски, различные групповое программное обеспечение, потоковые аудио-, видео- и анимационные ресурсы и др.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое компьютерная сеть? Приведите примеры сетей.
2. Что такое Интернет? Приведите примеры его использования.
3. Что такое Интранет? Приведите примеры его использования.
4. Что такое e-mail, ftp, telnet?
5. Как контролируется (управляется) Интернет?
6. Что такое доменная система имен? Приведите пример.
7. Какие нестандартные возможности Интернета вы знаете?

ТЕСТ

1. Наиболее распространенный способ электронного общения:
а) e-mail; б) web-chat; в) ftp; г) IP.
2. В списке {www.hotbox.ru, hot@box.ru, ru@air, 256@56.789.ru, kvm@ttt, filli@hhh@gif.ru, hhh.fgfg@arc.hty.com} адресами e-mail могут быть всего:
а) 1; б) 2; в) 3; г) 4.
3. Стандартные возможности Интернета:
а) telnet, e-mail, TCP/IP; в) telnet, ftp, e-banking;
б) e-mail, telnet, ftp; г) e-mail, e2e, p2p.
4. WWW является для Интернета:
а) основной гипертекстовой средой;

- б) поисковой программой;
 - в) прокси-сервером;
 - г) средством пересылки электронной почты.
5. Нестандартной возможностью Интернета не является:
- а) виртуальная биржа;
 - б) виртуальный консалтинг;
 - в) виртуальное производство услуг;
 - г) виртуальное производство товаров.
6. К основным функциям любой сети не относится:
- а) доставка пакетов данных;
 - б) автоматическое конфигурирование;
 - в) выход в Интернет;
 - г) поддержка протокола.
7. Маршрутизатор — устройство:
- а) для реализации протокола сетевого уровня;
 - б) для проверки протоколов сети на сбои;
 - в) для определения маршрута движения конвертов электронной почты;
 - г) для передачи данных от одного компьютера ко всем компьютерам сети.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
а	в	б	а	г	в	а

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Защита в компьютерных сетях.
2. Коммерческие возможности Интернета.
3. Образовательные возможности Интернета.
4. Развлекательные (досуговые) возможности Интернета.
5. Спам и борьба с ним в Интернете.
6. Интернет-2. Как нам реорганизовать Интернет?
7. Виртуальные частные сети.

И **21** ИНФОРМАТИЗАЦИЯ ОБЩЕСТВА

Информатизация — это процесс применения достижений информатики в предметных областях, в обществе, это процесс превращения достижений теоретической информатики в адекватные достижения практической и технической информатики, в практически значимые решения во всех сферах человеческого общества. Информатизация страны — комплексный процесс, отражающий научно-технические, социально-экономические, технологические, историко-логические, культурные, традиционные и другие особенности страны.

Информатизация общества базируется на трех китах:

- компьютеризации — процессе совершенствования компьютерных средств и систем сбора, хранения и актуализации информации;
- мультимедиазации — процессе совершенствования средств и систем приведения в активное состояние чувств и способов восприятия и актуализации информации;
- интеллектуализации — процессе совершенствования способности и систем восприятия и актуализации новой информации на основе целеполагающей деятельности, знаний, опыта, интуиции.

Информатизация усиливает демократические процессы в обществе и невозможна без них, ибо эти процессы выступают в качестве социального заказчика совершенствования информационной среды общества. Информатизация способствует гуманитаризации, гармонизации и гуманизации общества.

Общество, в котором информация становится ресурсом, определяющим развитие производственных отношений и производительных сил, и которое имеет все необходимые для этого (качественно и количественно) информационные магистрали и информационные потоки, ресурсы, методы, технологии, средства, т. е. имеет самоорганизующуюся (важно!) инфраструктуру для удовлетворения как индивидуальных, так и общественных потребностей в информации, совершенствования интеллектуального, социально-экономического, гуманитарного, физиолого-психологического состояния человека и общества, — **информационное общество** (или общество, развивающееся на основе информационных технологий и решений).

Информатизация — государственная политика, но может решаться на различных уровнях — государства, региона, отрасли и т. д. Программы информатизации разработаны и реализуются во всех более или менее развитых странах. Любая страна окажется на задворках мира без реализации такой программы.

ПРИМЕР

В СССР в 1989 г. была разработана Концепция информатизации общества, которая должна быть реализована ориентировочно к 2050 г. В США такая программа ориентирована на завершение к 2020 г., в Японии и развитых странах Западной Европы — к 2040 г. XX век должен быть веком информационного (единого) общества, и начнется развитие постинформационного общества, общества развитой единой инфосферы.

Необходимой предпосылкой информационного общества является наличие в обществе развитой системы социально-экономических, правовых, гуманитарных и технологических связей, направленных на развитие всех институтов общества, социально-экономических, правовых, технологических, интеллектуальных и гуманитарных потребностей общества, повышение уровня жизни членов общества.

Информационное общество возникло на определенном этапе истории развития общества объективным образом.

Уровень развития общества определяется не только развитием средств производства и производственных процессов, но и развитием информационных процессов и отношений, информационных систем и технологий, причем развитие информационных процессов и систем общества определяющим образом воздействует на развитие как производственных сфер, так и непроизводственных сфер, в целом на общество.

Информатизация страны состоит в информатизации следующих основных систем нашего общества (перечень не претендует на полноту, хотя и охватывает основные системы).

1. Информатизация банковских систем. Основные задачи: развитие банковских расчетов и межбанковских обменов, в том числе и зарубежных, компьютерных безналичных расчетов на основе кредитных карточек; компьютеризация и использование типовых компьютерных сетей и систем банковского учета и контроля; разработка и использование экономико-математических моделей различных банковских ситуаций и др.

ПРИМЕР

Прогноз банковского кредитного риска; прогноз рейтинга кредитоспособности банков на основе статистических данных и экспертных процедур, суждений; математическое моделирование надежности банков и прогноз индексов надежности; разработка и использование АРМ банковского работника и др.

2. Информатизация систем рыночной экономики. Основные задачи: обеспечение занятости населения с помощью банков данных; разработка и использование экономико-математических моделей оперативного и долгосрочного прогноза экономических ситуаций, процессов; внутрихозяйственные экономические расчеты и планирование и др.

ПРИМЕР

Прогноз соотношения спроса и предложения на рынке; построение и исследование математических моделей максимума прибыли в зависимости от различных факторов; моделирование поведения на рынке; разработка и использование АРМ работника рыночной экономики и др.

3. Информатизация систем социального обеспечения. Основные задачи: развитие пенсионных расчетов и их автоматизация; компьютерная, информационная поддержка инвалидов, престарелых, пенсионеров; автоматизация учета и планирования социальной помощи; разработка и использование экономико-математических моделей динамики социальных ситуаций в обществе (особенно критических) и др.

ПРИМЕР

Прогноз, анализ инфляционного ожидания в личном страховании; моделирование социального партнерства и принятие решений в различных социально-экономических и социально-культурных ситуациях, в частности при катастрофах; разработка и использование АРМ социального работника и др.

4. Информатизация систем налоговой службы. Основные задачи: технологическое обеспечение и компьютерная поддержка налоговых служб; автоматизация налогообложения (ведение баз данных по налогооблагаемым лицам и др.); разработка, использование экономико-математических моделей налогооблагаемости, налогообложения и др.

ПРИМЕР

Анализ и регулирование изменения коэффициента сбора налогов в зависимости от коэффициента финансовой самостоятельности, ликвидности и других факторов; моделирование и прогнозирование тяжести налогового бремени и финансовых последствий тех или иных решений в системе налогообложения; расчет оптимальных ставок налогообложения различных структур, например банковских; разработка и использование АРМ работника налоговой службы и др.

5. Информатизация систем биржи и биржевой деятельности. Основные задачи: внедрение типовых компьютерных сетей и средств, обеспечение совместимости с зарубежными системами; обеспечение оперативного реагирования на биржевые ситуации; компьютерное ведение потоков информации на бирже; разработка и использование экономико-математических моделей ситуаций на бирже и др.

ПРИМЕР

Прогноз динамики курса ценных бумаг, валют; моделирование потоков (товаров, ценных бумаг, платежей, услуг и других ресурсов) на бирже; инвестиционные решения и их оценка (обоснование) с учетом инфляции; моделирование и прогнозирование аномалий, катастроф на бирже; разработка и использование АРМ работника биржи (брокера) и др.

6. Информатизация систем промышленности. Основные задачи: организация маркетинга, использование гибких автоматизированных линий и производств (ГАП), различного типа робототехнических систем; автоматизированное проектирование производимых изделий и автоматизированное производство (производство «АП-АП», или «автоматизированное проектирование — автоматизированное производство»); компьютеризированный контроль, наладка и учет промышленных изделий; разработка и использование экономико-математических моделей систем и ситуаций на производстве (в сбыте) и др.

ПРИМЕР

Прогноз производительности труда; моделирование рентабельности и прибыльности, финансовой устойчивости и платежеспособности предприятий; обоснованное принятие решений по составу и структуре производства, поставок, сбыта; разработка и использование САПР и АРМ; визуальное и когнитивное проектирование на ЭВМ, сборка с помощью ЭВМ и др.

7. Информатизация систем транспорта и связи. Основные задачи: разработка и внедрение автоматизированных систем продажи билетов; автоматизация контроля и планирования транспортных перевозок; использование компьютерных систем разработки и внедрения новых транспортных средств и услуг; автоматизация и компьютеризация сетей связи, включая спутниковую связь; разработка и использование экономико-математических моделей транспортных потоков, каналов и систем связи и др.

ПРИМЕР

Выбор оптимального маршрута движения транспорта, трафика сетей связи; оптимальное диспетчерское управление транспортными потоками и средствами; компьютеризация навигации транспортных средств; разработка и использование АРМ работника транспорта, связи и др.

8. Информатизация систем топливно-энергетического комплекса. Основные задачи: организация маркетинга, автоматизация разведки, добычи и переработки нефти, газа или угля; учет и контроль состояния энергоресурсов; разработка экономико-математических моделей оптимального объема добычи и использования различных энергоресурсов и др.

ПРИМЕР

Распознавание нефтеносного пласта и запасов нефти в нем; автоматизация систем распределения и учета энергии; разработка, использование АРМ работника комплекса и др.

9. Информатизация систем строительного комплекса. Основные задачи: информационное обеспечение инвестиционных потоков (стыковка финансовых потоков, их согласование, исключение противоречий, дублирования); создание и ведение баз данных архитектур-

ных конструкций, примитивов, строительных норм и правил, норм расценок, картографических и геологических данных; разработка и использование экономико-математических моделей (методов) для решения задач строительства; инженерно-технические расчеты материалов и конструкций; автоматизация проектных, чертежных и картографических работ; оперативное диспетчерское управление; автоматизированный учет, контроль, управление и др.

ПРИМЕР

Оптимизация парка строительных машин, механизмов и их простоя; учет строительных материалов; выбор с помощью информационно-поисковых систем и баз данных строительных материалов и конструкций с необходимыми параметрами, их автоматизированная, визуальная компоновка (например, прокладка с использованием пакета программ AutoCAD водопроводных, тепловых, электрических коммуникаций); расчет надежности и долговечности строительных конструкций и др.

10. Информатизация систем правительственных услуг и права. Основные задачи: автоматизация подготовки, передачи, контроля исполнения документов в различного типа структурах (центральных, региональных, муниципальных и др.); повышение оперативности и обеспечение аналитическими данными для принятия решений на всех уровнях правительственной власти; автоматизация правовых услуг; разработка и использование экономико-правовых и социально-гуманитарных математических моделей принятия решений и др.

ПРИМЕР

Разработка консультирующих правовых компьютерных систем; разработка моделей прогноза динамики правонарушений и их предупреждения в зависимости от различных факторов; разработка и/или использование различных систем консультирования по вопросам права и государства; разработка и использование АРМ работника правительственной службы, права и др.

11. Информатизация систем здравоохранения и медицины. Основные задачи: компьютерное ведение историй болезней; компьютеризация постановки диагноза болезни с использованием соответствующих экспертных или интеллектуальных систем; компьютерная томография; медико-математическое моделирование и прогнозирование; постоперационное компьютерное наблюдение за различными физиологическими и психологическими характеристиками больного и др.

ПРИМЕР

Прогнозирование эпидемий и различных медико-социально-экономических ситуаций; разработка и использование автоматизированных систем «Поликлиника», «Скорая помощь» и др.; разработка, использование АРМ терапевта, хирурга, кардиолога, медсестры, другого медицинского работника и др.

12. Информатизация систем экологии. Основные задачи: осуществление контроля и управления экологическими системами; прогноз состояния окружающей среды на основе эколого-математических моделей прогноза экологических ситуаций, например моделирования и управления основными экологическими факторами для адаптивного и рационального природопользования; моделирование и прогнозирование различных экологических катастроф и др.

ПРИМЕР

Прогноз загрязнения водного и воздушного бассейнов; моделирование и прогнозирование экосистем; разработка экспертных систем и баз знаний для принятия экологически обоснованных решений; разработка и использование АРМ эколога и др.

13. Информатизация систем сельского хозяйства. Основные задачи: создание условий для программирования урожайности сельскохозяйственных культур и продукции животноводства; планирование агротехнических мероприятий и управление ими; оптимизация внесения минеральных удобрений и воды; автоматизация различных мелиоративных, животноводческих систем; разработка и использование экономико-математических моделей в сельском хозяйстве и др.

ПРИМЕР

Выбор оптимального режима полива и подкормки растений; автоматизация учета и хранения сельскохозяйственной продукции; моделирование роста и развития сельхозкультур; принятие экологически обоснованных решений; разработка АРМ работника сельского хозяйства и др.

14. Информатизация систем образования и образовательных услуг. Основные задачи: развитие алгоритмического стиля и культуры мышления; выработка умений и навыков описания систем, объектов в целом и связей между ними; выработка умений и навыков планирования ресурсов, необходимых для достижения поставленной цели; автоформализация профессиональных знаний; подготовка преподавателей, готовых и умеющих решать проблемы информатизации образования; профессиональная подготовка, переподготовка кадрового состава предприятий и организаций; телекоммуникационные средства обучения; воспитание в личности терпения, аккуратности, порядка в логике мышления и в работе; усиление междисциплинарных связей; использование средств мультимедиа; разработка и использование обучающих систем и сред и др.

ПРИМЕР

Разработка и использование различных программных, методических продуктов педагогического характера; дистанционное обучение; актуализация и/или визуализация междисциплинарных связей, в частности, с помощью компьютерных лабораторных работ по физике, химии; разработка АРМ директора, за-

вуча, заведующего кабинетом; автоматизация составления расписания занятий и др.

15. Информатизация систем военного дела, военной политики, всего военного искусства. Основные задачи: повышение обороноспособности страны; моделирование и обеспечение военных игровых ситуаций и состояний; кодирование и др.

ПРИМЕР

Обеспечение автоматизированных систем управления огнем; моделирование поведения на театре военных действий; оптимальный выбор рациона питания армии; моделирование тактики поведения военных кораблей, самолетов, танков, в частности уклонения их от нападения; разработка АРМ военного и др.

16. Информатизация систем безопасности. Основные задачи: обеспечение безопасности различного уровня и назначения — от безопасности офиса до безопасности государственной, национальной, ноосферы; моделирование и прогнозирование опасности; разработка технических систем обеспечения безопасности и др.

ПРИМЕР

Разработка надежных средств шифрования и передачи данных; разработка систем защиты и обеспечения надежности сетей и систем ЭВМ; разработка новых криптографических систем; обеспечение защиты от помех и перехвата; разработка АРМ работника системы безопасности и др.

17. Информатизация систем делопроизводства. Основные задачи: автоматизированное создание, оформление, прием, передача, подписание, согласование, утверждение, регистрация, хранение, перевод текста, контроль исполнения различных документов и др.

ПРИМЕР

Использование электронных словарей и переводчиков, например программной системы Stylus; распознавание текстов и их ввод в компьютер, например, с помощью сканера; подготовка текстов документов, таблиц, презентаций баз данных с помощью MS Office и др.

В последнее время употребляются термины **региональная информатизация, региональные проблемы информатизации** как совокупность проблем 1—17, возникающих в регионах (областях и республиках, субъектах государства) и учитывающих национально-региональные предпосылки постановки и соответствующие ресурсы при решении указанных выше проблем информатики, уровень информационно-технологического развития региона, исторические, географические, национальные, демографические, социально-экономические, культурные предпосылки, связи региона.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое информатизация?
2. В чем состоит информатизация отраслей народного хозяйства?
3. Укажите основные задачи компьютеризации.
4. Укажите основные задачи мультимедиазации.
5. Укажите основные задачи интеллектуализации.
6. Что такое информационное общество?
7. Каковы основные условия построения информационного общества?

ЗАДАЧИ

1. Приведите и поясните пример информатизации банковской системы.
2. Приведите и поясните пример информатизации образования.
3. Приведите и поясните пример информатизации экологии.
4. Приведите и поясните пример информатизации медицины.
5. Приведите и поясните пример информатизации транспорта.
6. Приведите и поясните пример информатизации производства.
7. Приведите и поясните пример информатизации социальной сферы.

ТЕСТ

1. Информатизация — процесс использования ...:
 - а) компьютеров в общественной жизни;
 - б) информационных систем в обществе;
 - в) новых информационных технологий в обществе;
 - г) всего перечисленного в а, б, в.
2. Не являются атрибутами информационного общества:
 - а) электронный документооборот и электронное делопроизводство;
 - б) превращение информации в товар;
 - в) информационная грамотность населения;
 - г) общедоступность любой информации.
3. Свойство, являющееся атрибутом информационного общества:
 - а) ведение всех деловых отношений в Интернете;
 - б) государственная поддержка доступа к компьютерным сетям;
 - в) виртуальные офисы и корпорации — основной вид работы;
 - г) перенос проблем информационной безопасности на фирмы.
4. Глобальным ресурсом информационного общества не является:
 - а) вещество;

- б) энергия;
- в) AVR;
- г) организация.

5. Неверно утверждение:

- а) вещественно-энергетические ресурсы имеют тенденцию истощаться;
- б) информационные ресурсы имеют тенденцию насыщаться;
- в) чем меньше энергоресурсов, тем больше информационных;
- г) вещественно-энергетические ресурсы имеют тенденцию насыщаться.

6. Верно утверждение:

- а) вещественные ресурсы имеют тенденцию увеличиваться;
- б) информационные ресурсы имеют тенденцию насыщаться;
- в) чем меньше энергоресурсов, тем больше информационных;
- г) чем больше информационных ресурсов, тем больше энергетических.

7. Информатизация базируется на:

- а) компьютерах, мультимедиа, медиатации;
- б) компьютеризации, мультимедиатации, интеллектуализации;
- в) компьютеризации, автоматизации, урбанизации;
- г) компьютеризации, теоретизации, численности населения.

Таблица правильных ответов к тесту

1	2	3	4	5	6	7
г	г	б	в	в	б	б

ТЕМЫ ИССЛЕДОВАНИЙ И РЕФЕРАТОВ

1. Интернет-торговля.
2. Интернет-обучение.
3. Интернет-платежи.
4. Коммуникационная культура общества и информатизация.
5. Информатизация и ее негативные последствия для общества.
6. Информатизация и ее позитивные стороны для общества.
7. Информатизация и эволюция общества.

- Андерсон Р. Доказательство правильности программ/Р. Андерсон. — М.: Мир, 1982.
- Андреева Е., Фалина И. Системы счисления и компьютерная арифметика/Е. Андреева, И. Фалина. — М.: Лаборатория базовых знаний, 1999.
- Бауэр Ф., Гооз Г. Информатика: вводный курс. В 2 ч./Ф. Бауэр, Г. Гооз. — М.: Наука, 1989.
- Брой М. Информатика: основополагающее введение. В 4 ч./М. Брой. — М.: Диалог-МИФИ, 1996.
- Верецагин Н. К., Шень А. Х. Лекции по математической логике и теории алгоритмов/Н. К. Верецагин, А. Х. Шень. — М.: МЦНМО, 1999—2000.
- Вирт Н. Алгоритмы + структуры данных = программы/Н. Вирт. — М.: Мир, 1990.
- Гилмор Ч. Введение в микропроцессорную технику/Ч. Гилмор. — М.: Мир, 1984.
- Гулд Х., Тобочник Я. Компьютерное моделирование в физике. В 2 ч./Х. Гулд, Я. Тобочник. — М.: Мир, 1990.
- Информатика/под ред. Н. В. Макаровой. — М.: Финансы и статистика, 1998.
- Информационные технологии в бизнесе: энциклопедия/под ред. М. Железны. — СПб.: Питер, 2002.
- Казиев В. М., Казиев К. В. Информатика: задачи и тесты/В. М. Казиев, К. В. Казиев. — М.: Просвещение, 2004.
- Колмогоров А. Н. Теория информации и теория алгоритмов/А. Н. Колмогоров. — М.: Наука, 1987.
- Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ/Т. Кормен, Ч. Лейзерсон, Р. Ривест. — М.: МЦНМО, 2001.
- Лихтарников Л. М. Первое знакомство с математической логикой/Л. М. Лихтарников. — СПб.: Лань, 1997.
- Райли Д. Абстракция и структуры данных: вводный курс/Д. Райли. — М.: Мир, 1993.
- Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы: теория и практика/Э. Рейнгольд, Ю. Нивергельт, Н. Део. — М.: Мир, 1980.
- Хакен Г. Информация и самоорганизация/Г. Хакен. — М.: Мир, 1991.
- Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов/Дж. Хопкрофт, Дж. Ульман. — М.: Мир, 1979.