Л. Л. Босова
А. Ю. Босова

ИНФОРМАТИКА

8

класс

БАЗОВЫЙ УРОВЕНЬ

Л. Л. Босова, А. Ю. Босова

ИНФОРМАТИКА

8 класс

БАЗОВЫЙ УРОВЕНЬ

Учебник

Допущено
Министерством просвещения
Российской Федерации

5-е издание, переработанное

Москва
«Просвещение»
2023

УДК 373.167.1:004+004(075.3)
ББК 32.81я721
Б85

Учебник допущен к использованию при реализации имеющих государственную аккредитацию образовательных программ начального общего, основного общего, среднего общего образования организациями, осуществляющими образовательную деятельность, в соответствии с Приказом Министерства просвещения Российской Федерации № 858 от 21.09.2022 г.

Босова, Людмила Леонидовна.
Б85 **Информатика : 8-й класс : базовый уровень : учебник** / Л. Л. Босова, А. Ю. Босова. — 5-е изд., перераб. — Москва : Просвещение, 2023. — 272 с. : ил.
ISBN 978-5-09-102543-9.

Учебник входит в состав УМК по информатике для 7–9 классов, включающего авторскую программу, учебники, электронные приложения, методическое пособие, рабочие тетради, сборники задач и другие компоненты. УМК может использоваться после вводного курса информатики в 5–6 классах или полностью самостоятельно обеспечивать освоение обязательного курса информатики в 7–9 классах, поддерживая базовую (1 ч/нед.) модель изучения предмета на уровне основного общего образования. Содержание учебника структурировано по темам «Системы счисления», «Элементы математической логики» (тематический раздел «Теоретические основы информатики»), «Основы алгоритмизации», «Начала программирования» (тематический раздел «Алгоритмы и программирование»). Теоретический материал поддержан развёрнутым аппаратом организации усвоения изучаемого материала, направленным на достижение обучающимися личностных, метапредметных и предметных образовательных результатов, обеспечивающим подготовку школьников к государственной итоговой аттестации по информатике в форме основного государственного экзамена (ОГЭ). Выдержан принцип инвариантности к конкретным моделям компьютеров и версиям программного обеспечения. Соответствует ФГОС ООО и Примерной рабочей программе по информатике для основного общего образования.

УДК 373.167.1:004+004(075.3)
ББК 32.81я721

Учебное издание

Босова Людмила Леонидовна
Босова Анна Юрьевна

ИНФОРМАТИКА
8 класс
Базовый уровень
Учебник

Центр развития углублённого и профильного образования, функциональной грамотности, технологии и ИКТ-компетенций

Ответственный за выпуск *О. Полежаева*
Ведущий редактор *О. Полежаева*. Художественное оформление *Е. Чайко*
Технический редактор *Е. Денюкова*. Корректор *О. Шутова*
Компьютерная вёрстка *Л. Катуркиной*

Подписано в печать 01.11.2022. Формат 70×100/16. Гарнитура SchoolBookCSanPin.
Усл. печ. л. 22,1. Тираж экз. Заказ

Акционерное общество «Издательство «Просвещение», Российская Федерация, 127473, г. Москва, ул. Краснопролетарская, д. 16, стр. 3, этаж 4, помещение 1.

Адрес электронной почты «Горячей линии» — vopros@prosv.ru.

ISBN 978-5-09-102543-9

ВВЕДЕНИЕ

Уважаемые восьмиклассники!

Мы живём во время стремительных перемен, когда для человека важна способность к постоянному развитию, готовность к освоению новых, в том числе информационных, технологий. Необходимость подготовки к быстро наступающим переменам в окружающем мире требует от человека развитого мышления, умения организации собственной учебной деятельности, ориентации на деятельностную жизненную позицию. Формирование таких качеств личности невозможно без фундаментального базового образования.

В курсе информатики 8 класса большое внимание уделяется фундаментальным (теоретическим) вопросам информатики. Вы будете изучать математические основы информатики, освоите базовые алгоритмические конструкции, по своему выбору познакомитесь с одним из языков программирования (Python, Паскаль).

Мы надеемся, что знания и умения, полученные на уроках информатики, вы сможете применять в дальнейшем при решении разнообразных жизненных задач, предполагающем такие этапы, как:

- *целеполагание* — постановка задачи на основе соотнесения того, что уже известно, и того, что требуется установить;
- *планирование* — определение последовательности промежуточных целей с учётом конечного результата, разбиение задачи на подзадачи, разработка последовательности и структуры действий, необходимых для достижения цели с помощью фиксированного набора средств;
- *прогнозирование* — предвосхищение результата;
- *контроль* — интерпретация полученного результата, его соотнесение с имеющимися данными с целью установления соответствия или несоответствия (обнаружения ошибки);
- *коррекция* — внесение необходимых дополнений и исправлений в план действий в случае обнаружения ошибки;
- *оценка* — осознание человеком того, насколько качественно им решена задача.

На страницах учебника подробно рассмотрены решения типовых задач по каждой изучаемой теме. В конце каждой главы учебника приведены тестовые задания, которые помогут вам

оценить, хорошо ли вы освоили теоретический материал и можете ли применять свои знания для решения возникающих проблем.

Электронное приложение к учебнику, содержащее видеоролики, мультимедийные презентации, интерактивные тесты, ссылки на ресурсы Интернета, находится в авторской мастерской Л. Л. Босовой (**https://bosova.ru/**).

Изучая теоретический материал, работая с дополнительными материалами, отвечая на вопросы, решая задачи и выполняя практические задания на компьютере, вы сможете полностью подготовиться к государственной итоговой аттестации по информатике в форме основного государственного экзамена (ОГЭ), требования к которому размещены на сайте Федерального института педагогических измерений (**https://fipi.ru/**).

В работе с учебником вам помогут навигационные значки:

 — важное утверждение или определение;

 — интересная информация;

 — пример решения задачи;

 — информация, полезная для решения практических задач;

 — ссылка на ресурс в Интернете;

 — вопросы в тексте параграфа, вопросы и задания для самоконтроля;

 — задания для подготовки к итоговой аттестации;

 — домашний проект или исследование;

 — задания для выполнения на компьютере;

 — групповая работа;

 — межпредметные связи.

Желаем успехов в изучении информатики!

Глава 1
СИСТЕМЫ СЧИСЛЕНИЯ

§ 1.1
Общие сведения о системах счисления

Ключевые слова:

- система счисления
- цифра
- алфавит
- непозиционная система счисления
- позиционная система счисления
- основание
- развёрнутая форма записи числа
- свёрнутая форма записи числа

1.1.1. История систем счисления

Как только люди начали считать (а это произошло ещё в каменном веке), у них появилась потребность в записи чисел.

Система счисления — это знаковая система, определяющая правила записи чисел. Знаки, с помощью которых записываются числа, называются **цифрами**, а их совокупность — **алфавитом системы счисления**.

В любой системе счисления цифры служат для обозначения чисел, называемых *узловыми*; остальные числа (*алгоритмические*) получаются из узловых чисел в результате некоторых операций.

Системы счисления различаются выбором узловых чисел и способами образования алгоритмических чисел. Можно выделить следующие виды систем счисления:

1) унарная система;
2) непозиционные системы;
3) позиционные системы.

Находки археологов на стоянках первобытных людей свидетельствуют о том, что первоначально количество предметов отображалось равным количеством каких-либо значков: зарубок, чёрточек, точек. Такая система записи чисел называется *унарной* (единичной), так как любое число в ней образуется путём повторения одного знака, символизирующего единицу.

Отголоски унарной системы счисления встречаются и сегодня. Сами того не осознавая, ею пользуются малыши, показывая на пальцах свой возраст; унарная система счисления (счётные палочки) используется для обучения детей счёту; её можно разглядеть в индикаторах уровня звукового сигнала в современных аудиосистемах.

Унарная система — не самый удобный способ записи чисел. Записывать таким образом большие количества утомительно, да и записи при этом получаются очень длинными. С течением времени возникли иные, более экономичные системы счисления.

Примерно в третьем тысячелетии до нашей эры египтяне придумали числовую систему, в которой для обозначения узловых чисел 1, 10, 100 и т. д. были предусмотрены специальные значки — иероглифы. Вот они (рис. 1.1).

Рис. 1.1. Алфавит египетской системы счисления

Все остальные (алгоритмические) числа составлялись из этих узловых чисел при помощи операции сложения. Например, чтобы изобразить число 3252, рисовали три цветка лотоса (три тысячи), два свёрнутых пальмовых листа (две сотни), пять дуг (пять десятков) и два шеста (две единицы). Величина числа не зависела от того, в каком порядке располагались составляющие его знаки: их можно было записывать сверху вниз, справа налево или вперемешку.

Система счисления называется **непозиционной**, если в ней количественный эквивалент (значение) цифры не зависит от её положения (места, позиции) в записи числа.

Примером непозиционной системы счисления, которая сохранилась до наших дней, может служить система счисления, придуманная более двух с половиной тысяч лет назад в Древнем Риме.

В основе римской системы счисления лежат узловые числа 1, 5, 10, 50, 100, 500 и 1000, обозначаемые соответственно знаками I (один палец), V (раскрытая ладонь), X (две сложенные ладони), L, C, D и M (рис. 1.2).

Рис. 1.2. Римские цифры

Запись алгоритмических чисел получается путём сложения и вычитания узловых чисел с учётом следующего правила: значение каждого меньшего знака, поставленного слева от большего, вычитается из значения большего знака; значение каждого меньшего знака, поставленного справа от большего, прибавляется к значению большего знака.

Например, запись IX обозначает число 9, а запись XI — число 11. Десятичное число 99 имеет такое представление:

$$\mathrm{XCIX} = \underbrace{-10+100}\underbrace{-1+10}$$

В наши дни любую из римских цифр предлагается использовать в записи одного числа не более трёх раз подряд (табл. 1.1).

Таблица 1.1

Запись чисел в римской системе счисления

Единицы	Десятки	Сотни	Тысячи
1 I	10 X	100 C	1000 M
2 II	20 XX	200 CC	2000 MM
3 III	30 XXX	300 CCC	3000 MMM
4 IV	40 XL	400 CD	
5 V	50 L	500 D	
6 VI	60 LX	600 DC	
7 VII	70 LXX	700 DCC	
8 VIII	80 LXXX	800 DCCC	
9 IX	90 XC	900 CM	

Таблица 1.1 позволяет быстро и правильно записать в римской системе счисления любое целое число от 1 до 3999. Чтобы это сделать, для цифр, стоящих в разрядах тысяч, сотен, десятков и единиц исходного числа, по таблице подбираются и выписываются в порядке следования соответствующие кодовые группы. Для того

чтобы записать числа, бóльшие 3999, применяют специальные правила, но знакомство с ними выходит за рамки нашего курса.

Римскими цифрами пользовались очень долго. Ещё 200 лет назад в деловых бумагах числа должны были обозначаться римскими цифрами (считалось, что обычные арабские цифры легко подделать). Римская система счисления сегодня используется в основном для наименования знаменательных дат, томов, разделов и глав в книгах.

Наряду с иероглифическими в древности широко применялись алфавитные системы счисления, в которых числа изображались буквами алфавита. Так, в Древней Греции числа 1, 2, ..., 9 обозначали первыми девятью буквами греческого алфавита: $\alpha = 1$, $\beta = 2$, $\gamma = 3$ и т. д. Для обозначения десятков применялись следующие девять букв: $\iota = 10$, $\kappa = 20$, $\lambda = 30$, $\mu = 40$ и т. д. Для обозначения сотен использовались последние девять букв: $\rho = 100$, $\sigma = 200$, $\tau = 300$ и т. д.

Алфавитной нумерацией пользовались также южные и восточные славянские народы (табл. 1.2).

Таблица 1.2

Славянский цифровой алфавит

Буква	Название	Числовой эквивалент	Буква	Название	Числовой эквивалент	Буква	Название	Числовой эквивалент
А	Аз	1	І	И	10	Р	Рцы	100
В	Веди	2	К	Како	20	С	Слово	200
Г	Глаголь	3	Л	Люди	30	Т	Твердо	300
Д	Добро	4	М	Мыслете	40	У	Ук	400
Є	Есть	5	Н	Наш	50	Ф	Ферт	500
Ѕ	Зело	6	Ѯ	Кси	60	Х	Хер	600
З	Земля	7	О	Он	70	Ѱ	Пси	700
И	Иже	8	П	Покой	80	Ѡ	Омега	800
Ѳ	Фита	9	Ч	Червь	90	Ц	Цы	900

Над буквой, обозначавшей цифру (или над всем числом), ставился специальный значок (ти́тло). Например, если записать в славянской нумерации числа 55, 288, 1 и 498, то можно прочитать фразу:

Для записи тысяч, десятков тысяч, сотен тысяч и т. д. использовались те же буквы, что и для чисел 1–9, но с добавленными к ним специальными значками (рис. 1.3).

Рис. 1.3. Славянская алфавитная нумерация

В России славянская нумерация сохранялась до конца XVII века. При Петре I возобладала арабская нумерация, которой мы пользуемся и сейчас. Славянская нумерация сохранилась только в богослужебных книгах.

1.1.2. Позиционные системы счисления

Рассмотренные нами иероглифические и алфавитные системы счисления имели один существенный недостаток — в них было очень трудно выполнять арифметические операции. Этого неудобства нет у позиционных систем.

Система счисления называется **позиционной**, если количественный эквивалент (значение) цифры зависит от её положения (места, позиции, разряда) в записи числа.

Основные достоинства любой позиционной системы счисления — простота выполнения арифметических операций и ограниченное количество символов, необходимых для записи любых чисел.

Знаменитый французский математик, механик, физик и астроном Пьер-Симон Лаплас (1749–1827) такими словами оценил «открытие» позиционной системы счисления: «Мысль выражать все числа немногими знаками, придавая им значение по форме, ещё значение по месту, настолько проста, что именно из-за этой простоты трудно оценить, насколько она удивительна».

Десятичная система записи чисел, которой мы привыкли пользоваться в повседневной жизни, с которой мы знакомы с детства, в которой производим все наши вычисления, — пример позиционной системы счисления.

Алфавит десятичной системы составляют цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

У позиционной системы счисления есть основание. Основание десятичной системы — число 10; степени основания определяют «веса» позиций (разрядов) в записи числа. Из двух одинаковых цифр, находящихся в записи числа рядом, левая цифра имеет в 10 раз большее значение, чем правая.

Пример 1

В десятичном числе $355 = 3 \cdot 100 + 5 \cdot 10 + 5 \cdot 1$ цифры 5, находящиеся в разных позициях, имеют различные количественные значения: 5 десятков и 5 единиц.

Алгоритмические числа образуются в десятичной позиционной системе счисления следующим образом: значения цифр умножаются на «веса» соответствующих позиций (разрядов) и все полученные произведения складываются. Это отчётливо прослеживается в числительных русского языка, например: «три-ста пять-десят пять».

Арифметические действия над десятичными числами выполняются с помощью достаточно простых операций, в основе которых лежат известные каждому школьнику таблицы умножения и сложения, а также правило переноса: если в результате сложения двух цифр получается число, которое больше или равно 10, то оно записывается с помощью нескольких цифр, находящихся на соседних позициях.

Потребовалось много тысячелетий, чтобы люди научились называть и записывать числа так, как это делаем мы с вами. Начало этому было положено в Древнем Египте и Вавилоне. Дело в основном завершили индийские математики в V–VII веках нашей эры. Важным достижением индийской науки было введение особого обозначения для пропуска разрядов — нуля. Арабы, познакомившись с этой нумерацией первыми, по достоинству её оценили, усвоили и перенесли в Европу. Получив название арабской, эта система в XII веке нашей эры распространилась по всей Европе и, будучи проще и удобнее остальных систем счисления, быстро их вытеснила и стала использоваться повсеместно.

Существует множество позиционных систем счисления. Позиционная система счисления определяется основанием, в качестве которого можно использовать произвольное целое число $q > 1$. Алфавитом позиционной системы счисления с основанием q служат числа 0, 1, ..., $q - 1$, каждое из которых может быть записано с помощью одного уникального символа; младшей цифрой всегда является 0.

В позиционной системе счисления с основанием q любое неотрицательное целое число может быть представлено в виде

$$A_q = a_{n-1} \cdot q^{n-1} + a_{n-2} \cdot q^{n-2} + \ldots + a_0 \cdot q^0. \qquad (1)$$

Здесь:

A — число;

q — основание системы счисления;

a_i — цифры, принадлежащие алфавиту данной системы счисления;

n — количество разрядов числа;

q^i — «вес» i-го разряда.

Запись числа по формуле (1) называется **развёрнутой формой** записи. **Свёрнутой формой** записи числа называется его представление в виде $a_{n-1}a_{n-2}\ldots a_1a_0$.

Пример 2

Рассмотрим десятичное число 4351. Его свёрнутая форма записи настолько привычна, что мы не замечаем, как в уме переходим к развёрнутой записи, умножая цифры числа на «веса» разрядов и складывая полученные произведения:

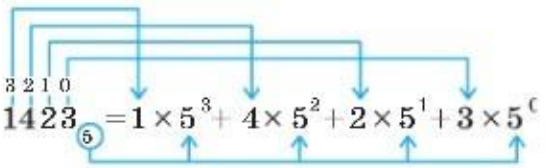
$$4 \cdot 10^3 + 3 \cdot 10^2 + 5 \cdot 10^1 + 1 \cdot 10^0.$$

Чтобы перевести число из позиционной системы счисления с основанием q в десятичную систему счисления, необходимо записать исходное число в развёрнутой форме и вычислить значение получившегося арифметического выражения.

Пример 3

Переведём в десятичную систему счисления число 1423_5, представленное в системе счисления с основанием 5.

Построим развёрнутую запись числа 1423_5:



Вычислим значение выражения:

$1 \cdot 5^3 + 4 \cdot 5^2 + 2 \cdot 5^1 + 3 \cdot 5^0 = 125 + 100 + 10 + 3 = 238.$

Итак, $1423_5 = 238_{10}$.

САМОЕ ГЛАВНОЕ

Система счисления — это знаковая система, определяющая правила записи чисел. Знаки, с помощью которых записываются числа, называются цифрами, а их совокупность — алфавитом системы счисления.

Система счисления называется непозиционной, если количественный эквивалент (значение) цифры не зависит от её положения (места, позиции) в записи числа.

Система счисления называется позиционной, если количественный эквивалент (значение) цифры зависит от её положения (места, позиции, разряда) в записи числа.

Позиционная система счисления определяется основанием, в качестве которого можно использовать произвольное целое число $q > 1$. Алфавитом позиционной системы счисления с основанием q служат числа 0, 1, ..., $q - 1$, каждое из которых может быть записано с помощью одного уникального символа; младшей цифрой всегда является 0.

В позиционной системе счисления с основанием q любое неотрицательное целое число может быть представлено в развёрнутой форме:

$A_q = a_{n-1} \cdot q^{n-1} + a_{n-2} \cdot q^{n-2} + \ldots + a_0 \cdot q^0.$

Здесь:
A — число;
q — основание системы счисления;
a_i — цифры, принадлежащие алфавиту данной системы счисления;
n — количество разрядов числа;
q^i — «вес» i-го разряда.

Чтобы перевести число из позиционной системы счисления с основанием q в десятичную систему счисления, необходимо записать исходное число в развёрнутой форме и вычислить значение получившегося арифметического выражения.

Вопросы и задания

1. Найдите дополнительную информацию об унарной, позиционных и непозиционных системах счисления. Чем они различаются? Приведите примеры.
2. Выясните с помощью Интернета, как слова «тьма», «легион», «леодр», «вран», «колода» связаны с темой «Системы счисления» и что они обозначают.
3. На постаменте памятника Петру I в Санкт-Петербурге римскими цифрами записан год открытия памятника: MDCCLXXXII. В каком году был открыт этот памятник?
4. Как вы считаете, почему позиционные системы счисления с основаниями 5, 10, 12 и 20 называют системами счисления анатомического происхождения?
5. Как от свёрнутой формы записи десятичного числа перейти к его развёрнутой форме?
6. Запишите в развёрнутой форме числа:
   а) 143511_{10};
   б) 143511_{8};
   в) 143511_{16}.
7. Вычислите десятичные эквиваленты следующих чисел:
   а) 172_{8};
   б) 219_{16};
   в) 101010_{2};
   г) 243_{6}.
8. Укажите, какое из чисел 110011_{2}, 111_{4}, 35_{8} и 16_{16} является:
   а) наибольшим;
   б) наименьшим.

9. Укажите минимальное основание системы счисления, в которой могут быть записаны числа 123, 222, 111, 241. Определите десятичный эквивалент данных чисел в найденной системе счисления.
10. Верны ли следующие равенства?
    а) $33_4 = 21_7$;
    б) $33_7 = 21_4$.
11. Найдите основание x системы счисления, если:
    а) $14_x = 9_{10}$;
    б) $2002_x = 130_{10}$.
12. Какое двузначное в десятичной системе счисления число окажется «круглым» (с двумя нулями в конце) в пятеричной системе счисления? Если таких чисел несколько, то перечислите их все.
13. Постройте граф, отражающий разновидности систем счисления.

§ 1.2
Двоичная система счисления

Ключевые слова:

- двоичная система счисления
- алфавит двоичной системы счисления
- «веса» двоичных разрядов
- двоичная арифметика

1.2.1. Двоичные числа и их перевод в десятичную систему счисления

Двоичной системой счисления называется позиционная система счисления с основанием 2.

Для записи чисел в двоичной системе счисления используются только две цифры: 0 и 1.

Для работы с двоичными числами надо знать «веса» двоичных разрядов (табл. 1.3).

Таблица 1.3

Степени числа 2

n	10	9	8	7	6	5	4	3	2	1	0
2^n	1024	512	256	128	64	32	16	8	4	2	1

Любое двоичное число можно записать в развёрнутой форме. Запишем, например, в развёрнутой форме двоичное число 10011_2:

$$\overset{4\ 3\ 2\ 1\ 0}{10011_2} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0.$$

Эту запись можно сделать проще, если не включать в сумму слагаемые с нулевыми сомножителями и не писать сомножители, равные 1. Другими словами, достаточно включать в развёрнутую запись только степени двойки, соответствующие единичным коэффициентам:

$$10011_2 = 2^4 + 2^1 + 2^0.$$

Такая форма записи «подсказывает» правило перевода целых двоичных чисел в десятичную систему счисления: необходимо вычислить сумму степеней двойки, соответствующих единицам в свёрнутой форме записи двоичного числа:

$$10011_2 = 2^4 + 2^1 + 2^0 = 16 + 2 + 1 = 19_{10}.$$

1.2.2. Перевод целых чисел из десятичной системы счисления в двоичную

Переводить небольшие целые числа (например, от 0 до 1024) в двоичную систему счисления можно с помощью таблицы степеней числа 2. Рассмотрим этот способ на примере числа 684.

Наша задача — представить число 684 в виде суммы степеней числа 2. Начнём с наибольшей степени числа 2, не превышающей исходное число. Очевидно, это 512. Запишем:

$$684 = 512 + 172.$$

Подберём для числа 172 наибольшую степень числа 2, не превышающую число 172. Очевидно, это 128. Запишем:

$$684 = 512 + 128 + 44.$$

Продолжив аналогичные рассуждения, получим:

$$684 = 512 + 128 + 32 + 8 + 4.$$

Воспользуемся уже знакомой вам таблицей степеней двойки и добавим в неё строку, в которую впишем 1, если соответствующее значение степени вошло в записанную выше сумму, и 0 — в противном случае.

n	10	9	8	7	6	5	4	3	2	1	0
2^n	1024	512	256	128	64	32	16	8	4	2	1
A_2	0	1	0	1	0	1	0	1	1	0	0

Выпишем все полученные нули и единицы слева направо — это и будет двоичная запись числа 648:

$A_2 = 01010101100_2$, или 1010101100_2.

Нули, стоящие левее первой единицы, являются незначащими, их можно удалять из записи двоичного числа.

Рассмотрим ещё один способ перевода целых чисел из десятичной системы счисления в двоичную, основанный на операции деления с остатком.

На основании формулы (1) для неотрицательных целых двоичных чисел можно записать:

$$a_{n-1}a_{n-2}\ldots a_1a_0 = a_{n-1}\cdot 2^{n-1} + a_{n-2}\cdot 2^{n-2} + \ldots + a_0\cdot 2^0. \qquad (1')$$

Разделим $a_{n-1}\cdot 2^{n-1} + a_{n-2}\cdot 2^{n-2} + \ldots + a_0\cdot 2^0$ на 2. Неполное частное будет равно $a_{n-1}\cdot 2^{n-2} + a_{n-2}\cdot 2^{n-3} + \ldots + a_1$, а остаток будет равен a_0.

Полученное частное опять разделим на 2, остаток от деления будет равен a_1.

Если продолжить этот процесс деления, то на n-м шаге получим набор цифр:

$$a_0, a_1, a_2, \ldots, a_{n-1},$$

которые входят в двоичное представление исходного числа и совпадают с остатками при его последовательном делении на 2. Но эти цифры в исходном числе были записаны справа налево. Мы получили правило перевода целых десятичных чисел в двоичную систему счисления.

Для перевода целого десятичного числа в двоичную систему счисления нужно последовательно выполнять деление данного числа и получаемых неполных частных на 2 до тех пор, пока не получим неполное частное, равное нулю. Представление исходного числа в двоичной системе счисления образуется путём последовательной записи полученных остатков, начиная с последнего.

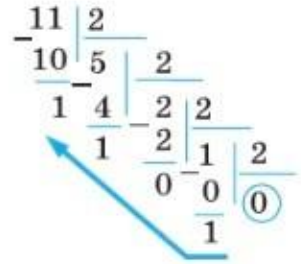
Переведём по этому правилу в двоичную систему счисления десятичное число 11:

$$\begin{aligned} 11 &: 2 = 5 \text{ (ост. } \mathbf{1}) \\ 5 &: 2 = 2 \text{ (ост. } \mathbf{1}) \\ 2 &: 2 = 1 \text{ (ост. } \mathbf{0}) \\ 1 &: 2 = 0 \text{ (ост. } \mathbf{1}) \end{aligned}$$

Выписав все остатки, начиная с последнего, получим двоичную запись исходного десятичного числа:

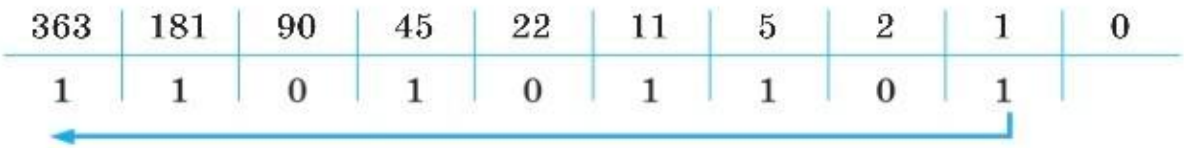
$11_{10} = 1011_2$.

Записывать рассмотренную выше последовательность действий (алгоритм перевода) можно и «уголком»:



Выписывая остатки от деления в направлении, указанном стрелкой, получим тот же результат: $11_{10} = 1011_2$.

Рассмотрим ещё один, более компактный способ записи перевода чисел из десятичной системы в двоичную. Его можно использовать при переводе больших (например, трёхзначных) чисел:



$363_{10} = 101101011_2$

1.2.3. Двоичная арифметика

Арифметика двоичной системы счисления основывается на использовании следующих таблиц сложения и умножения:

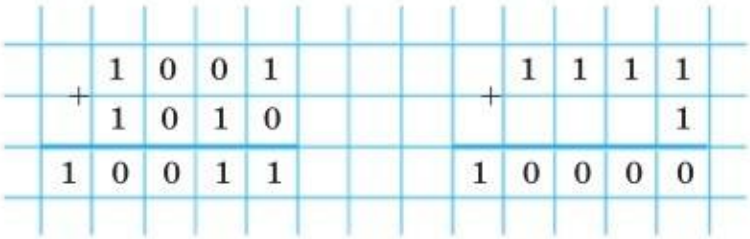
+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

Таблица двоичного сложения предельно проста. $1 + 1 = 10$, поэтому 0 остаётся в младшем разряде, а 1 переносится в старший разряд.

Пример 1

Рассмотрим примеры сложения чисел в двоичной системе счисления.

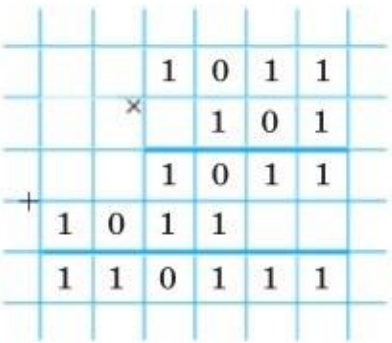


$1001_2 + 1010_2 = 10011_2$; $1111_2 + 1_2 = 10000_2$.

Операция умножения двоичных чисел выполняется по обычной схеме, применяемой в десятичной системе счисления, с последовательным умножением множимого на очередную цифру множителя.

Пример 2

Выполним умножение двоичных чисел 1011_2 и 101_2:



$1011_2 \cdot 101_2 = 110111_2$.

Таким образом, в двоичной системе счисления умножение сводится к сдвигам множимого и сложениям.

Вычитание, как и сложение, выполняется «столбиком». Очевидно, что:

$0 - 0 = 0$,
$1 - 0 = 1$,
$1 - 1 = 0$.

Но что получится, если необходимо вычесть из нуля единицу?

В аналогичной ситуации в десятичной системе счисления мы занимаем единицу в старшем разряде. Так же надо поступать и в двоичной системе счисления. При этом надо помнить, что в двоичной системе счисления единица старшего разряда образуется из двух единиц соседнего с нею младшего разряда.

Таблица вычитания в двоичной системе счисления будет иметь вид

–	0	1
0	0	1
1	1	0

Пример 3

Выполним вычитание двоичных чисел 10101_2 и 1010_2:

		1 1		1 1		
	1	0	1	0	1	
–		1	0	1	0	
		1	0	1	1	

$10101_2 - 1010_2 = 1011_2$.

Деление двоичных чисел сводится к операциям сравнения и вычитания и оформляется «уголком».

Пример 4

Разделим число 110110_2 на 110_2:

	1	1	0	1	1	0	1	1	0	
–	1	1	0				1	0	0	1
				1	1	0				
–				1	1	0				
					0					

$110110_2 : 110_2 = 1001_2$.

Правильность выполнения арифметических операций в двоичной системе счисления вы всегда можете проверить, переведя в десятичную систему двоичные операнды и полученный результат.

Как известно, компьютеры работают с информацией, представленной в двоичном коде. Исходя из этого, естественно было бы предположить, что именно конструкторы электронных вычислительных машин придумали и двоичную систему счисления. Но это не так. Двоичная система счисления с 0 и 1 была описана выдающимся немецким философом и математиком Готфридом Ле́йбницем (1646–1716) и несколько веков не представляла никакой практической ценности. Сегодня она является основой нашего цифрового окружения.

САМОЕ ГЛАВНОЕ

Двоичной системой счисления называется позиционная система счисления с основанием 2. Для записи чисел в двоичной системе счисления используются только две цифры: 0 и 1.

Для работы с двоичными числами надо знать «веса» двоичных разрядов — степени числа 2:

n	10	9	8	7	6	5	4	3	2	1	0
2^n	1024	512	256	128	64	32	16	8	4	2	1

Для перевода двоичных чисел в десятичную систему счисления достаточно вычислить сумму степеней двойки, соответствующих единицам в свёрнутой форме записи двоичного числа.

Для перевода целого десятичного числа в двоичную систему счисления нужно последовательно выполнять деление данного числа и получаемых неполных частных на 2 до тех пор, пока не получим частное, равное нулю. Представление исходного числа в двоичной системе счисления образуется путём последовательной записи полученных остатков, начиная с последнего.

Арифметические операции в двоичной системе счисления выполняются по тем же правилам, что и в десятичной системе счисления.

Вопросы и задания

1. Вычислите десятичные эквиваленты двоичных чисел:
   а) 111_2;
   б) 1010_2;
   в) 11011_2.
2. В некоторый момент времени в пробирке находилась 1 бактерия. Известно, что каждую секунду бактерий в пробирке становится вдвое больше. Сколько их будет через 12 секунд?
3. Переведите целые числа из десятичной системы счисления в двоичную:
   а) 89;
   б) 600;
   в) 2020.
4. Сколько единиц в двоичной записи десятичного числа?
   а) 128;
   б) 129;
   в) 255.
5. Сколько значащих нулей в двоичной записи десятичного числа?
   а) 126;
   б) 127;
   в) 128.
6. Найдите сумму двоичных чисел. Результат запишите в десятичной системе счисления.
   а) 101010 + 1101;
   б) 1010 + 1010;
   в) 10101 + 111.
7. Найдите произведение двоичных чисел. Результат запишите в десятичной системе счисления.
   а) 1010 · 11;
   б) 111 · 101;
   в) 1010 · 111.
8. Найдите разность двоичных чисел. Результат запишите в десятичной системе счисления.
   а) 10101 – 101;
   б) 10101 – 1101;
   в) 10101 – 1111.

9. Найдите частное двоичных чисел. Результат запишите в десятичной системе счисления.
   а) 101101 : 101;
   б) 1001000 : 1001;
   в) 1111001 : 1011.
10. Расставьте знаки арифметических операций так, чтобы были верны следующие равенства в двоичной системе:
   а) 1100 ? 11 ? 100 = 100000;
   б) 1100 ? 10 ? 10 = 100;
   в) 1100 ? 11 ? 100 = 0.
11. Какими преимуществами и недостатками обладает двоичная система счисления по сравнению с десятичной?

§ 1.3 Системы счисления, родственные двоичной

Ключевые слова:

- восьмеричная система счисления
- двоичная триада
- шестнадцатеричная система счисления
- двоичная тетрада

1.3.1. Восьмеричная система счисления

Восьмеричной системой счисления называется позиционная система счисления с основанием 8.

Для записи чисел в восьмеричной системе счисления используются цифры: 0, 1, 2, 3, 4, 5, 6, 7.

На основании формулы (1) для целого восьмеричного числа можно записать:

$$a_{n-1}a_{n-2}\ldots a_1a_0 = a_{n-1}\cdot 8^{n-1} + a_{n-2}\cdot 8^{n-2} + \ldots + a_0\cdot 8^0. \qquad (1'')$$

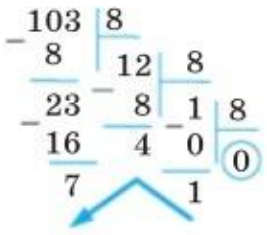
Например: $\overset{3\,2\,1\,0}{1063}_8 = 1\cdot 8^3 + 0\cdot 8^2 + 6\cdot 8^1 + 3\cdot 8^0 = 563_{10}$.

Таким образом, для перевода целого восьмеричного числа в десятичную систему счисления следует перейти к его развёрнутой записи и вычислить значение получившегося выражения.

Для перевода целого десятичного числа в восьмеричную систему счисления надо последовательно выполнять деление данного числа и получаемых неполных частных на 8 до тех пор, пока не получим неполное частное, равное нулю. Представление исходного числа в восьмеричной системе счисления образуется путём последовательной записи полученных остатков, начиная с последнего.

Пример 1

Переведём десятичное число 103 в восьмеричную систему счисления:



$103_{10} = 147_8$.

Между восьмеричной и двоичной системами счисления существует связь, основанная на том, что $8 = 2^3$.

Поставим в соответствие каждой восьмеричной цифре двоичную триаду — цепочку из трёх 0 и 1, имеющую такое же числовое значение, что и восьмеричная цифра (табл. 1.4).

Таблица 1.4

Восьмеричные цифры и их двоичное представление

Восьмеричная цифра	Двоичная триада
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Для перевода целого числа из восьмеричной системы счисления в двоичную достаточно заменить каждую восьмеричную цифру в записи числа на соответствующую ей двоичную триаду.

Пример 2

Переведём в двоичную систему счисления восьмеричное число 147_8:

$147_8 = 001\ 100\ 111_2$.

Отбросив незначащие нули, получим:

$147_8 = 1100111_2$.

В том, что перевод выполнен правильно, нетрудно убедиться, если перевести полученное число в десятичную систему счисления:

$$147_8 = 1100111_2 = 2^6 + 2^5 + 2^2 + 2^1 + 2^0 = \\ = 64 + 32 + 4 + 2 + 1 = 103_{10}.$$

Мы получили тот же результат, что и в примере 1.

Для перевода целого числа из двоичной системы счисления в восьмеричную достаточно разбить двоичную запись числа на триады справа налево и заменить каждую двоичную триаду на восьмеричную цифру.

Пример 3

Переведём в восьмеричную систему счисления двоичное число 1101101100_2.

Это десятизначное число; 10 знаков нельзя разбить на группы по 3 знака. Дополним исходную двоичную запись до 12 знаков двумя незначащими нулями:

$1101101100_2 = 001101101100_2$.

Разобьём двоичную запись на триады:

$001101101100_2 = 001\ 101\ 101\ 100_2$.

Заменим триады восьмеричными цифрами:

$001101101100_2 = 001\ 101\ 101\ 100_2 = 1554_8$.

1.3.2. Шестнадцатеричная система счисления

Шестнадцатеричной системой счисления называется позиционная система счисления с основанием 16.

Для записи чисел в шестнадцатеричной системе счисления используются цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Здесь только десять цифр из шестнадцати имеют общеизвестное обозначение 0, ..., 9. Для записи цифр с десятичными количественными эквивалентами 10, 11, 12, 13, 14, 15 используются первые пять букв латинского алфавита.

Таким образом, запись $3AF_{16}$ означает:

$$\overset{2\ 1\ 0}{3AF_{16}} = 3 \cdot 16^2 + 10 \cdot 16^1 + 15 \cdot 16^0 = 768 + 160 + 15 = 943_{10}.$$

Как видите, перевод шестнадцатеричного числа в десятичную систему счисления осуществляется через его развёрнутую запись. Обратный перевод (из десятичной системы счисления в шестнадцатеричную) осуществляется путём деления на 16.

Пример 4

Переведём десятичное число 154 в шестнадцатеричную систему счисления:

154	16	
144	9	16
10	0	
(A)	9	0

$154_{10} = 9A_{16}$.

Связь между шестнадцатеричной и двоичной системами счисления основана на том, что $16 = 2^4$.

Поставим в соответствие каждой шестнадцатеричной цифре двоичную тетраду — цепочку из четырёх 0 и 1, имеющую такое же числовое значение, что и шестнадцатеричная цифра (табл. 1.5).

Таблица 1.5

Шестнадцатеричные цифры и их двоичное представление

Шестнадцатеричная цифра	Двоичная тетрада
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A (10)	1010
B (11)	1011
C (12)	1100
D (13)	1101
E (14)	1110
F (15)	1111

Для перевода целого числа из шестнадцатеричной системы счисления в двоичную достаточно заменить каждую шестнадцатеричную цифру в записи числа на соответствующую ей двоичную тетраду.

Пример 5

Переведём в двоичную систему счисления шестнадцатеричное число $3AF_{16}$:

$3AF_{16} = 0011\ 1010\ 1111_2$.

Отбросив незначащие нули, получим:

$3AF_{16} = 1110101111_2$.

В том, что перевод выполнен правильно, нетрудно убедиться, если перевести полученное число в десятичную систему счисления:

$$3AF_{16} = 1110101111_2 = 2^9 + 2^8 + 2^7 + 2^5 + 2^3 + 2^2 + 2^1 + 2^0 =$$
$$= 512 + 256 + 128 + 32 + 8 + 4 + 2 + 1 = 943_{10}.$$

В начале пункта мы получили: $3AF_{16} = 943_{10}$.

Для перевода целого числа из двоичной системы счисления в шестнадцатеричную достаточно разбить двоичную запись числа на тетрады справа налево и заменить каждую двоичную тетраду на шестнадцатеричную цифру.

Пример 6

Переведём в шестнадцатеричную систему счисления двоичное число 11101101100110_2. Это четырнадцатизначное число; 14 знаков нельзя разбить на группы по 4 знака. Дополним исходную двоичную запись до 16 знаков двумя незначащими нулями:

$$11101101100110_2 = 0011101101100110_2.$$

Разобьём двоичную запись на тетрады:

$$11101101100110_2 = 0011\ 1011\ 0110\ 0110_2.$$

Заменим тетрады шестнадцатеричными цифрами:

$$11101101100110_2 = 0011\ 1011\ 0110\ 0110_2 = 3B66_{16}.$$

САМОЕ ГЛАВНОЕ

Восьмеричной системой счисления называется позиционная система счисления с основанием 8. Для записи чисел в восьмеричной системе счисления используются цифры: 0, 1, 2, 3, 4, 5, 6, 7.

Шестнадцатеричной системой счисления называется позиционная система счисления с основанием 16. Для записи чисел в шестнадцатеричной системе счисления используются цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (10), B (11), C (12), D (13), E (14), F (15).

Для перевода восьмеричного (шестнадцатеричного) числа в десятичную систему счисления следует перейти к его развёрнутой записи и вычислить значение получившегося выражения.

Для перевода целого десятичного числа в восьмеричную (шестнадцатеричную) систему счисления нужно последовательно выполнять деление данного числа и получаемых неполных частных на 8 (16) до тех пор, пока не получим частное, равное нулю. Представление исходного числа в новой системе счисления образуется путём последовательной записи полученных остатков, начиная с последнего.

Для переводов чисел между восьмеричной и двоичной, шестнадцатеричной и двоичной системами счисления используется замена восьмеричных цифр на двоичные триады и шестнадцатеричных цифр на двоичные тетрады.

Вопросы и задания

1. Переведите целые числа из десятичной системы счисления в восьмеричную:
   а) 55;
   б) 600;
   в) 2022.
2. Переведите целые числа из восьмеричной системы счисления в двоичную:
   а) 65_8;
   б) 123_8;
   в) 1756_8.
3. Переведите целые числа из двоичной системы счисления в восьмеричную:
   а) 11011011_2;
   б) 11101110111_2;
   в) 11001100110011_2.
4. Разработайте таблицы сложения и умножения для восьмеричной системы счисления.
5. Переведите целые числа из десятичной системы счисления в шестнадцатеричную:
   а) 55;
   б) 600;
   в) 2022.
6. Переведите целые числа из шестнадцатеричной системы счисления в двоичную:
   а) $A5_{16}$;
   б) $1C3_{16}$;
   в) $9A5E_{16}$.

7. Переведите целые числа из двоичной системы счисления в шестнадцатеричную:
   а) 11011011_2;
   б) 11101110111_2;
   в) 11001100110011_2.
8. Придумайте способ быстрого перевода целого числа из шестнадцатеричной системы счисления в восьмеричную. Переведите с его помощью числа:
   а) $A5_{16}$;
   б) $1C3_{16}$;
   в) $9A5E_{16}$.
9. Заполните в тетради таблицу, в каждой строке которой одно и то же число должно быть записано в системах счисления с основаниями 2, 8, 10 и 16.

Основание 2	Основание 8	Основание 10	Основание 16
101010			
	127		
		321	
			2A

10. Сравните двоичные числа x и y:
    а) $x = 1010101111010111$, $y = 101010111010111$;
    б) $x = 10101011010101101$, $y = 10101101010101101$.
11. Среди приведённых ниже трёх чисел, записанных в различных системах счисления, найдите наименьшее и запишите его в десятичной системе счисления.
    36_{16}, 64_8, 111010_2.
12. Среди приведённых ниже трёх чисел, записанных в различных системах счисления, найдите наибольшее и запишите его в десятичной системе счисления.
    36_{16}, 63_8, 111101_2.
13. Вычислите выражения:
    а) $(1111101_2 + AF_{16}) : 36_8$;
    б) $125_8 + 101_2 \cdot 2A_{16} - 141_8$.
    Ответ дайте в десятичной системе счисления.

14. Почему восьмеричная и шестнадцатеричная системы счисления считаются системами, родственными двоичной? Какая ещё позиционная система счисления может считаться родственной двоичной?
15. Вы умеете переводить целые числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную. Попробуйте сформулировать правило перевода целых десятичных чисел в систему счисления с основанием q. Проверьте справедливость сформулированного правила, осуществив с его помощью перевод числа 555 в систему счисления с основанием 5.

§ 1.4 Системы счисления и представление информации в компьютере

Ключевые слова:

- ячейка памяти
- разряд
- представление целых чисел
- двоичная система счисления
- шестнадцатеричная система счисления
- кодовая таблица

1.4.1. Представление целых чисел в компьютере

Оперативная память компьютера состоит из ячеек, каждая из которых представляет собой физическую систему, образованную некоторым числом однородных элементов. Эти элементы обладают двумя устойчивыми состояниями, одно из которых соответствует нулю, а другое — единице. Каждый такой элемент служит для хранения одного из битов — разряда двоичного числа. Именно поэтому каждый элемент ячейки называют **битом** или **разрядом** (рис. 1.4).

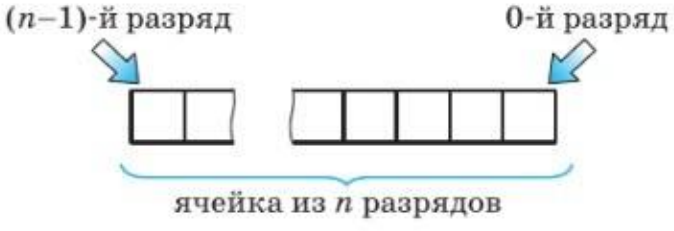


Рис. 1.4. Ячейка памяти

Для хранения целых чисел в памяти компьютера могут использоваться ячейки, содержащие 8, 16, 32 или 64 разряда. В них целые числа могут быть представлены беззнаковым или знаковым способом.

Беззнаковое представление используется для таких объектов, как адреса ячеек, всевозможные счётчики (например, число символов в тексте), а также числа, обозначающие дату и время, размеры графических изображений в пикселях и т. д.

Так как в каждый разряд ячейки можно записать одно из двух значений (0 или 1), то ячейка из n разрядов может иметь 2^n разных состояний. Это значит, что в ячейке из n разрядов можно представить 2^n различных целых чисел.

Минимальное число соответствует n нулям, хранящимся в n разрядах памяти, и равно нулю. Максимальное значение целого беззнакового (неотрицательного) числа достигается в случае, когда во всех разрядах ячейки хранятся единицы. Для n-разрядного представления оно будет равно $2^n - 1$.

В табл. 1.6 приведены максимальные значения для беззнаковых целых n-разрядных чисел.

Таблица 1.6

Диапазоны беззнаковых целых чисел

Количество бит	Минимальное значение	Максимальное значение
8	0	255 ($2^8 - 1$)
16	0	65 535 ($2^{16} - 1$)
32	0	4 294 967 295 ($2^{32} - 1$)

Для получения компьютерного представления беззнакового целого числа достаточно перевести число в двоичную систему счисления и дополнить полученный результат слева нулями до нужной разрядности.

Например, число $53_{10} = 110101_2$ в восьмиразрядном представлении имеет вид

0	0	1	1	0	1	0	1

Это же число 53 в шестнадцати разрядах будет записано следующим образом:

0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1

Во многих практических задачах, решаемых с помощью компьютера, появляется необходимость в использовании отрицательных чисел; надо, чтобы число было представлено вместе со своим знаком — плюсом или минусом.

При представлении со знаком самый старший (левый) разряд отводится под знак числа, остальные разряды — под само число. Если число положительное, то в знаковый разряд помещается 0; если число отрицательное, то 1. Такое представление чисел называется **прямым кодом**.

Ноль и положительные числа представляются в прямом коде. Для представления отрицательных чисел используется так называемый **дополнительный код**, позволяющий заменить операцию вычитания сложением.

Вы можете прочитать в Интернете, как образуется и применяется дополнительный код.

В ячейке из n разрядов можно представить 2^{n-1} отрицательных чисел, число 0 и $2^{n-1} - 1$ положительных чисел.

В табл. 1.7 приведены минимальные и максимальные значения для целых n-разрядных чисел со знаком.

Таблица 1.7

Диапазоны целых чисел со знаком

Количество бит	Минимальное значение	Максимальное значение
8	-128 (2^7)	127 ($2^7 - 1$)
16	$-32\,768$ (2^{15})	32 767 ($2^{15} - 1$)
32	2 147 483 648 (2^{31})	2 147 483 647 ($2^{31} - 1$)

1.4.2. «Компьютерные» системы счисления

В компьютерной технике используется двоичная система счисления, обладающая рядом преимуществ по сравнению с другими системами счисления:

- двоичные числа представляются в компьютере с помощью достаточно простых технических элементов с двумя устойчивыми состояниями;

- представление информации посредством только двух состояний надёжно и помехоустойчиво;
- двоичная арифметика наиболее проста;
- существует математический аппарат, обеспечивающий логические преобразования двоичных данных.

Обмен данными между компьютерными устройствами осуществляется путём передачи двоичных кодов. Пользоваться такими кодами из-за их большой длины и зрительной однородности человеку неудобно. Поэтому специалисты (программисты, инженеры) на некоторых этапах разработки, создания, настройки вычислительных систем заменяют двоичные коды на эквивалентные им представления в восьмеричной или шестнадцатеричной системе счисления. В результате длина исходного кода сокращается в три или четыре раза соответственно. Это делает информацию более удобной для рассмотрения и анализа.

1.4.3. Тайна кодовой таблицы

Вы знаете, что текст, как и любая другая информация, обрабатываемая компьютером, представляется в двоичном коде. При этом каждому символу, встречающемуся в тексте, ставится в соответствие некоторый двоичный код. Соответствие символов и двоичных кодов задаётся кодовой таблицей. Одна из самых известных кодовых таблиц представлена на рис. 1.5.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SQH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SQ	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Рис. 1.5. Кодировка ASCII

Посмотрите внимательно на эту таблицу. Видите ли вы в ней двоичные коды?

Двоичных кодов в явном виде в этой кодовой таблице нет, но их достаточно просто получить по шестнадцатеричным кодам. Последние образуются указанием шестнадцатеричных номеров строки и столбца для нужной вам буквы.

Например, буква «M» находится в строке с номером 4, в столбце с номером D; её шестнадцатеричный код 4D, что соответствует двоичному коду 01001101.

Использование шестнадцатеричных кодов связано с тем, что, с одной стороны, они значительно короче и поэтому проще воспринимаются человеком; с другой стороны, они очень быстро могут быть преобразованы в двоичное представление.

САМОЕ ГЛАВНОЕ

Для компьютерного представления целых чисел используется несколько способов, отличающихся друг от друга количеством разрядов (8, 16, 32 или 64) и наличием или отсутствием знакового разряда.

Для представления беззнакового целого числа его следует перевести в двоичную систему счисления и дополнить полученный результат слева нулями до нужной разрядности.

При представлении со знаком самый старший разряд отводится под знак числа, остальные разряды — под само число. Если число положительное, то в знаковый разряд помещается 0; если число отрицательное — то 1.

Обмен данными между компьютерными устройствами осуществляется путём передачи двоичных кодов. Пользоваться такими кодами из-за их большой длины и зрительной однородности человеку неудобно. Поэтому специалисты на некоторых этапах разработки, создания, настройки вычислительных систем заменяют двоичные коды на эквивалентные им представления в восьмеричной или шестнадцатеричной системе счисления.

Вопросы и задания

1. Как в памяти компьютера представляются целые числа?
2. Представьте число 63_{10} в беззнаковом 8-разрядном формате.

3. Найдите десятичные эквиваленты чисел по их прямым кодам, записанным в 8-разрядном формате со знаком:
   а) 01001100;
   б) 11111000.
4. Какие из чисел 443_8, 101010_2, 256_{10} можно сохранить в 8-разрядном беззнаковом формате?
5. Декодируйте следующие текстовые сообщения, записанные в кодировке ASCII:
   а) 01000100 01010010 01000101 01000001 01001101;
   б) 01001101 01101111 01110011 01100011 01101111 01110111.
6. На листке в клетку запишите шестнадцатеричные числа в столбик: 1800, 1F00, 1980, 40C6, FFFF, 0066, 0FC0, 0E00. Декодируйте графическое изображение, заменяя каждую шестнадцатеричную цифру двоичной тетрадой и записывая её в заданной последовательности, правее соответствующего шестнадцатеричного числа.
7. Почему не только двоичная, но и восьмеричная, и шестнадцатеричная системы счисления считаются «компьютерными»?

Тестовые задания для самоконтроля

1. Совокупность знаков, с помощью которых записываются числа, называется:
   а) системой счисления
   б) цифрами системы счисления
   в) алфавитом системы счисления
   г) основанием системы счисления

2. Чему равен результат сложения двух чисел, записанных римскими цифрами: MCM + LXVIII?
   а) 1168
   б) 1968
   в) 2168
   г) 1153

3. Запись 301011 может соответствовать числу в системах счисления с основаниями:
   а) 2 и 10
   б) 4 и 3
   в) 4 и 8
   г) 2 и 4

4. Двоичное число 100110 в десятичной системе счисления записывается как:
   а) 36
   б) 38
   в) 37
   г) 46

5. В классе 110010_2% девочек и 1010_2 мальчиков. Сколько учеников в классе?
   а) 10
   б) 20
   в) 30
   г) 40

6. Сколько цифр 1 в двоичном представлении десятичного числа 15?
   а) 1 б) 2 в) 3 г) 4

7. Сколько единиц в двоичном представлении восьмеричного числа 36?
   а) 2 б) 3 в) 4 г) 5
8. Сколько единиц в двоичном представлении шестнадцатеричного числа 3B?
   а) 3 б) 4 в) 5 г) 6
9. Чему равен результат сложения чисел 110_2 и 12_8?
   а) 6_{10}
   б) 10_{10}
   в) 10000_2
   г) 17_8
10. Среди приведённых ниже четырёх чисел, записанных в различных системах счисления, найдите максимальное.
    а) 14_{16}
    б) 26_8
    в) 1000_2
    г) 17_{10}
11. Среди приведённых ниже четырёх чисел, записанных в различных системах счисления, найдите минимальное.
    а) 24_{16}
    б) 26_8
    в) 11000_2
    г) 27_{10}
12. Ячейка памяти компьютера состоит из однородных элементов, называемых:
    а) кодами
    б) разрядами
    в) цифрами
    г) коэффициентами
13. Количество разрядов, занимаемых двухбайтовым числом, равно:
    а) 8
    б) 16
    в) 32
    г) 64

14. В знаковый разряд ячейки для отрицательных чисел заносится:
    а) +
    б) –
    в) 0
    г) 1
15. Какое из чисел можно записать в восьмиразрядную ячейку в беззнаковом формате?
    а) 512
    б) 256
    в) 255
    г) –128

Глава 2
ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ

§ 2.1
Высказывания и логические связки

Ключевые слова:

- алгебра логики
- высказывание
- истина
- ложь
- высказывательная форма
- логическая связка НЕ
- логическая связка И
- логическая связка ИЛИ
- простое высказывание
- составное высказывание

2.1.1. Высказывания и высказывательные формы

Алгебра в широком смысле этого слова — это наука об общих операциях, аналогичных сложению и умножению, которые могут выполняться над разнообразными математическими объектами. Многие математические объекты (целые и рациональные числа, многочлены, векторы, множества) вы изучаете в школьном курсе алгебры, где знакомитесь с такими разделами математики, как алгебра чисел, алгебра многочленов, алгебра множеств и т. д.

Для информатики важен раздел математики, называемый **алгеброй логики**; объектами алгебры логики являются высказывания.

Высказывание — это предложение на любом языке, содержание которого можно однозначно определить как истинное или ложное.

Например, относительно предложений «*Великий русский учёный М. В. Ломоносов родился в 1711 году*» и «*Two plus six is eight*» можно однозначно сказать, что они истинны. Предложение «*Зимой воробьи впадают в спячку*» ложно. Следовательно, эти предложения являются высказываниями.

В русском языке высказывания выражаются повествовательными предложениями. Но не всякое повествовательное предложение является высказыванием.

Например, предложение «*Это предложение является ложным*» не является высказыванием, так как относительно него нельзя сказать, истинно оно или ложно, без того чтобы не получить противоречие. Действительно, если принять, что предложение истинно, то это противоречит сказанному. Если же принять, что предложение ложно, то отсюда следует, что оно истинно.

Относительно предложения «*Компьютерная графика — самая интересная тема в курсе школьной информатики*» также нельзя однозначно сказать, истинно оно или ложно. Подумайте сами почему.

Побудительные и вопросительные предложения высказываниями не являются.

Например, не являются высказываниями такие предложения, как «*Запишите домашнее задание*», «*Как пройти в библиотеку?*», «*Кто к нам пришёл!*».

Высказывания могут строиться с использованием знаков различных формальных языков — математики, физики, химии и т. п.

Примерами высказываний могут служить:

1) «*Na — металл*» (истинное высказывание);
2) «*Второй закон Ньютона выражается формулой* $F = m \cdot a$» (истинное высказывание);
3) «*Периметр прямоугольника с длинами сторон* a *и* b *равен* $a \cdot b$» (ложное высказывание).

Не являются высказываниями числовые выражения, но из двух числовых выражений можно составить высказывание, соединив их знаками равенства или неравенства. Например:

1) «$3 + 5 = 2 \cdot 4$» (истинное высказывание);
2) «$II + VI > VIII$» (ложное высказывание).

Не являются высказываниями и равенства или неравенства, содержащие переменные (предложения с переменными). Например, предложение «$X < 12$» становится высказыванием только при замене переменной каким-либо конкретным значением: «$5 < 12$» — истинное высказывание; «$12 < 12$» — ложное высказывание. Такие предложения с переменными иначе называют высказывательными формами.

Высказывательная форма — это повествовательное предложение, которое содержит хотя бы одну переменную и становится высказыванием, когда все переменные заменяются конкретными значениями.

Множество тех значений переменной, при которых получаются истинные высказывания, называют областью истинности высказывательной формы (предложения с переменной).

Все рассмотренные выше высказывания либо отражали отношения между двумя объектами, либо описывали некоторое свойство объекта. Такие высказывания называют **простыми** или **элементарными.**

Так, высказывание «*Na — металл*» описывает свойство («металл») объекта («Na»); высказывание «$3 + 5 = 2 \cdot 4$» отражает отношение («=») между двумя объектами («$3 + 5$», «$2 \cdot 4$»).

2.1.2. Логические связки НЕ, И, ИЛИ

Из простых высказываний с помощью логических связок НЕ, И, ИЛИ можно строить **составные** высказывания.

Рассмотрим использование каждой логической связки отдельно.

Составное высказывание со связкой НЕ содержит одно простое высказывание. Составное высказывание со связкой НЕ истинно, если содержащееся в нём простое высказывание ложно.

Например, простое высказывание «*Число 5 является чётным*» ложно; составное высказывание «*Число 5 НЕ является чётным*» истинно.

Составное высказывание со связкой НЕ ложно, если содержащееся в нём простое высказывание истинно. Например, простое высказывание «*Последняя буква в слове "логика" является гласной*» истинно; составное высказывание «*Последняя буква в слове "логика" НЕ является гласной*» ложно.

Работать со связкой НЕ достаточно сложно. В высказываниях на русском языке логическая связка НЕ ставится в середине высказывания, и не всегда бывает понятно, где именно её следует поставить. Например, в высказывании «*Петя решил правильно все задания контрольной работы*» для логической связки НЕ возможны такие места:

1) *Петя НЕ решил правильно все задания контрольной работы.*
2) *Петя решил НЕправильно все задания контрольной работы.*
3) *Петя решил правильно НЕ все задания контрольной работы.*

Очевидно, второе из этих высказываний имеет совсем иной смысл, чем первое и третье.

По сути, с помощью логической связки НЕ мы строим отрицание исходного высказывания.

Отрицанием высказывания «*У меня дома есть компьютер*» будет высказывание «*Неверно, что у меня дома есть компьютер*» или, что в русском языке то же самое, «*У меня дома нет компьютера*».

Отрицанием высказывания «*Я не знаю китайский язык*» будет высказывание «*Неверно, что я не знаю китайский язык*» или, что в русском языке то же самое, «*Я знаю китайский язык*».

Отрицанием высказывания «*Все юноши 8-х классов — отличники*» является высказывание «*Неверно, что все юноши 8-х классов — отличники*», другими словами, «*Не все юноши 8-х классов — отличники*».

Для того чтобы избежать двусмысленности при составлении высказываний со связкой НЕ, рекомендуется выбирать один из двух вариантов:

1) использовать речевой оборот «неверно, что» и всегда ставить его перед исходным высказыванием;
2) строить отрицание к сказуемому, добавляя к соответствующему глаголу частицу «не».

Сформулируйте отрицания высказываний «*Последняя буква в слове "логика" является гласной*», «*Число 5 является чётным*» с речевым оборотом «*Неверно, что*». Какой ещё вариант отрицания исходных высказываний вы можете предложить?

Составное высказывание со связкой И содержит два простых высказывания. Составное высказывание со связкой И истинно тогда и только тогда, когда истинны оба входящие в него простые высказывания.

Например, простые высказывания «*Число 324 делится на 7*», «*Число 324 делится на 5*» — оба ложны; простые высказывания «*Число 324 делится на 3*», «*Число 324 делится на 2*» — оба истинны. Рассмотрим образованные из них с помощью логической связки И составные высказывания:

1) «*Число 324 делится на 7 И на 5*» — ложное высказывание;
2) «*Число 324 делится на 7 И на 3*» — ложное высказывание;
3) «*Число 324 делится на 7 И на 2*» — ложное высказывание;

4) «*Число 324 делится на 5 И на 3*» — ложное высказывание;
5) «*Число 324 делится на 5 И на 2*» — ложное высказывание;
6) «*Число 324 делится на 3 И на 2*» — истинное высказывание.

Рассмотрите следующие высказывания: «*Основоположником алгебры логики является Джордж Буль*», «*Исследования Клода Шеннона позволили применить алгебру логики в вычислительной технике*», «*Основоположником алгебры логики является Джордж Буль, и исследования Клода Шеннона позволили применить алгебру логики в вычислительной технике*». Сделайте вывод об истинности третьего высказывания, предварительно выяснив с помощью Интернета истинность первых двух высказываний.

Составное высказывание со связкой ИЛИ содержит два простых высказывания. Составное высказывание со связкой ИЛИ ложно тогда и только тогда, когда ложны оба входящие в него простые высказывания.

Рассмотрим составные высказывания, образованные с помощью логической связки ИЛИ:

1) «*Число 324 делится на 7 ИЛИ на 5*» — ложное высказывание;
2) «*Число 324 делится на 7 ИЛИ на 3*» — истинное высказывание;
3) «*Число 324 делится на 7 ИЛИ на 2*» — истинное высказывание;
4) «*Число 324 делится на 5 ИЛИ на 3*» — истинное высказывание;
5) «*Число 324 делится на 5 ИЛИ на 2*» — истинное высказывание;
6) «*Число 324 делится на 3 ИЛИ на 2*» — истинное высказывание.

Рассмотрите следующие высказывания: «*Идея использования в логике математической символики принадлежит Готфриду Вильгельму Лейбницу*», «*Лейбниц является основоположником бинарной арифметики*», «*Идея использования в логике математической символики принадлежит Готфриду Вильгельму Лейбницу, или Лейбниц является основоположником бинарной арифметики*». Сделайте вывод об истинности третьего высказывания, предварительно выяснив с помощью Интернета истинность первых двух высказываний.

В обычной жизни связка ИЛИ зачастую трактуется в исключающем смысле. Вспомните, например, диагноз, который поставил лекарь Богомол Буратино: «*Одно из двух: или пациент жив, или он умер*». Что касается математической логики и программирования, то в них ИЛИ воспринимается в неисключающем смысле. Так, высказывание «*Иван может перекусить бутербродом или яблоком*» означает, что Иван может перекусить бутербродом или яблоком или и тем, и другим.

Составные высказывания можно строить не только из элементарных высказываний, но и из других составных высказываний.

Например: «*В слове "вентиль" первая буква согласная И вторая — гласная, ИЛИ в слове "кулер" шесть букв*» — истинное высказывание.

САМОЕ ГЛАВНОЕ

Высказывание — это предложение на любом языке, содержание которого можно однозначно определить как истинное или ложное.

Высказывательная форма — это повествовательное предложение, которое содержит хотя бы одну переменную и становится высказыванием, когда все переменные заменяются конкретными значениями.

Различают простые и составные высказывания. Простые высказывания отражают отношения между двумя объектами или описывают некоторое свойство объекта. Никакая часть простого высказывания сама не является высказыванием. Составные (сложные) высказывания строятся из простых с помощью логических связок И, ИЛИ, НЕ.

Составное высказывание со связкой НЕ истинно, если содержащееся в нём простое высказывание ложно.

Составное высказывание со связкой И истинно тогда и только тогда, когда истинны оба входящие в него простые высказывания.

Составное высказывание со связкой ИЛИ ложно тогда и только тогда, когда ложны оба входящие в него простые высказывания.

Вопросы и задания

1. Обсудите в группе, почему следующие предложения не являются высказываниями.
   а) *Какого цвета этот дом?*
   б) *Число X не превосходит единицы.*
   в) $4X + 3$.
   г) *Посмотрите в окно.*
   д) *Пейте томатный сок!*
   е) *Эта тема скучна.*
   ж) *Гарри Стайлс — самый популярный певец.*
   з) *Вы были в театре?*
2. Приведите по одному примеру простых истинных и ложных высказываний из биологии, географии, информатики, истории, математики, литературы. Обратите внимание на то, что отражают эти высказывания: свойство объекта или отношение между двумя объектами.
3. При каких значениях переменной высказывательная форма «*X — простое число*» становится истинным высказыванием?
   а) 2;
   б) 4;
   в) 5;
   г) 9.
4. При каких значениях переменной высказывательная форма «*В слове X вторая буква — гласная*» становится истинным высказыванием?
   а) автобус;
   б) аэропорт;
   в) встреча;
   г) метро.
5. Постройте отрицания следующих высказываний.
   а) *Сегодня в театре идёт опера «Евгений Онегин».*
   б) *Каждый охотник желает знать, где сидит фазан.*
   в) *Число 1 есть простое число.*
   г) *Натуральные числа, оканчивающиеся цифрой 0, не являются простыми числами.*
   д) *Неверно, что число 3 не является делителем числа 198.*
   е) *Наташа выполнила все задания домашней работы по информатике.*

ж) *В любой школе некоторые ученики интересуются спортом.*

з) *Некоторые млекопитающие не живут на суше.*

и) *Неверно, что для прогулки подойдёт не любое время после 16:00.*

к) $5 < 4$.

л) $12 > 100$.

м) *501 — чётное число.*

н) *А — согласная.*

6. В жизни мы часто пользуемся высказываниями с союзами «а», «но», «хотя». Например:

а) *«Зимой Ирина любит кататься на коньках, а Денис — на лыжах»*;

б) *«Денис выполнил задания самостоятельной работы, но забыл сдать на проверку тетрадь с решениями»*;

в) *«На день рождения Саше подарили планшет, хотя он мечтал о щенке».*

Выделите в этих утверждениях простые высказывания и переформулируйте их так, чтобы получились составные высказывания со связкой И.

7. Укажите такие значения переменной *X*, при которых высказывательная форма *(X < 20) И (X >= 10)* превратится в истинное высказывание:

а) 5;

б) 10;

в) 15;

г) 20.

8. Укажите такие значения переменной *X*, при которых высказывательная форма *(X > 20) ИЛИ (X <= 10)* превратится в ложное высказывание:

а) 5;

б) 10;

в) 15;

г) 20.

9. Для какого имени ложно высказывание *«НЕ (Первая буква гласная И Последняя буква гласная)»*?

а) Данила;

б) Ирина;

в) Светлана;

г) Максим.

§ 2.2 Логические операции и логические выражения

Ключевые слова:

- логическая переменная
- логическое значение
- логическая операция
- логическое отрицание
- логическое умножение
- логическое сложение
- приоритет логических операций
- логическое выражение

2.2.1. Логические переменные и логические значения

Обоснование истинности или ложности высказываний даётся теми науками, к сфере которых эти высказывания относятся. Алгебра логики отвлекается от смысловой содержательности высказываний. Её интересует только, истинно или ложно данное высказывание.

В алгебре логики высказывания обозначают буквами и называют **логическими переменными.** При этом если высказывание истинно, то значение соответствующей ему логической переменной обозначают единицей ($A = 1$), а если ложно — нулём ($B = 0$). 0 и 1, обозначающие значения логических переменных, называются **логическими значениями.**

Алгебра логики определяет правила записи, упрощения и преобразования составных высказываний и вычисления их значений.

Оперируя логическими переменными, которые могут быть равны только 0 или 1, алгебра логики позволяет свести обработку информации к операциям с двоичными данными. Именно аппарат алгебры логики положен в основу компьютерных устройств хранения и обработки данных. С применением элементов алгебры логики вы будете встречаться и во многих других разделах информатики.

2.2.2. Логические операции

Заменив высказывания логическими переменными, имеющими значение 0 или 1, можно рассматривать логические связки как **логические операции** над ними.

Для описания операции И можно использовать одну из следующих таблиц:

A	*B*	*A* И *B*
0	0	0
0	1	0
1	0	0
1	1	1

И	0	1
0	0	0
1	0	1

Первая таблица называется **таблицей истинности.** В таблице истинности перечисляются все возможные значения исходных высказываний (столбцы «*A*» и «*B*»), причём соответствующие им двоичные числа, как правило, располагаются в порядке возрастания: 00, 01, 10, 11. В последнем столбце записывается результат выполнения логической операции для соответствующих операндов.

Каждую из логических операций можно описать таблицей подобно тому, как мы описывали арифметические операции в двоичной системе счисления.

Вторая таблица полностью повторяет таблицу, описывающую операцию двоичного умножения. Именно поэтому операцию И очень часто называют **логическим умножением.** Есть у неё и ещё одно название — **конъюнкция.**

Конъюнкция — логическая операция, ставящая в соответствие двум высказываниям новое высказывание, являющееся истинным тогда и только тогда, когда оба исходных высказывания истинны.

Для описания операции ИЛИ можно использовать одну из следующих таблиц:

A	*B*	*A* ИЛИ *B*
0	0	0
0	1	1
1	0	1
1	1	1

ИЛИ	0	1
0	0	1
1	1	1

Вторая из этих таблиц очень похожа на таблицу, описывающую операцию двоичного сложения. Благодаря этому операцию ИЛИ очень часто называют **логическим сложением**. Ещё одно её название — **дизъюнкция**.

Дизъюнкция — логическая операция, ставящая в соответствие двум высказываниям новое высказывание, являющееся ложным тогда и только тогда, когда оба исходных высказывания ложны.

Операция НЕ может быть описана следующей таблицей истинности:

A	НЕ *A*
0	1
1	0

Другое название операции НЕ — **логическое отрицание** или **инверсия**.

Инверсия — логическая операция, ставящая в соответствие исходному высказыванию новое высказывание, значение которого противоположно значению исходного.

У хорошо известных всем арифметических операций есть обозначения. Например, операция сложения обозначается знаком «+», операция вычитания — знаком «−» и т. д. Есть несколько вариантов специальных обозначений и для логических операций. Наиболее распространённые из них представлены в табл. 2.1.

Таблица 2.1

Названия и обозначения логических операций

Логическая связка	Логическая операция	Другое название	Обозначение
НЕ	Логическое отрицание	Инверсия	$\neg$, $^-$
И	Логическое умножение	Конъюнкция	$\wedge$, $\cdot$, &
ИЛИ	Логическое сложение	Дизъюнкция	$\vee$, +, |

2.2.3. Логические операции и операции над множествами

Операции конъюнкция и дизъюнкция тесно связаны с такими известными вам операциями над множествами, как пересечение и объединение.

Пересечением двух множеств X и Y называется множество их общих элементов (рис. 2.1).

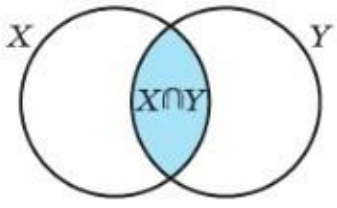


Рис. 2.1. Пересечение множеств

Объединением двух множеств X и Y называется множество, состоящее из всех элементов этих множеств и не содержащее никаких других элементов.

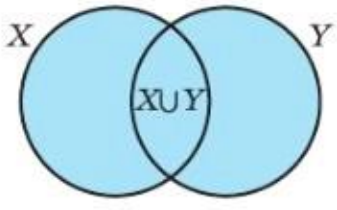


Рис. 2.2. Объединение множеств

Пусть X — множество веб-страниц, на которых встречается слово «*крейсер*»; Y — множество веб-страниц, на которых встречается слово «*линкор*».

Пересечением множеств X и Y будет множество страниц, на которых встречаются оба эти слова одновременно. Другими словами, $X \cap Y$ — это множество истинности высказывательной формы «*На веб-странице встречается слово "крейсер" и слово "линкор"*», представляющей собой конъюнкцию высказывательных форм «*На веб-странице встречается слово "крейсер"*», «*На веб-странице встречается слово "линкор"*».

Объединением множеств X и Y будет множество страниц, на которых встречается хотя бы одно из этих слов, а также оба эти слова одновременно. Другими словами, $X \cup Y$ — это множество истинности высказывательной формы «*На веб-странице встречается слово "крейсер" или слово "линкор"*», представляющей

собой дизъюнкцию высказывательных форм *«На веб-странице встречается слово "крейсер"»*, *«На веб-странице встречается слово "линкор"»*.

Логическими операциями мы пользуемся при поиске информации в сети Интернет. При решении задач, связанных с количественными характеристиками результатов поиска, используются операции над множествами.

Пример 1

В языке запросов поискового сервера для обозначения логической операции ИЛИ используется символ «|», а для обозначения логической операции И — символ «&».

В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Запрос	Найдено страниц (в тысячах)
крейсер	4800
линкор	4500
крейсер | линкор	7000

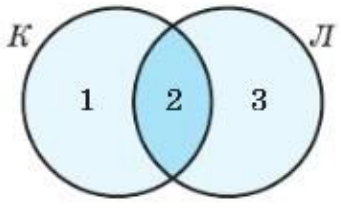
Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Какое количество страниц (в тысячах) будет найдено по запросу

крейсер & линкор ?

Решение

Изобразим условие задачи графически:



Здесь множества страниц, найденных по соответствующим запросам, обозначены двумя пересекающимися кругами; непересекающиеся области на полученной схеме обозначены цифрами 1, 2, 3.

По схеме видно, что количество элементов, образующих множество $К$, равно сумме элементов, входящих в области 1 и 2; количество элементов, образующих множество $Л$, равно сумме элементов, входящих в области 2 и 3; количество элементов, образующих множество $К \cup Л$, равно сумме элементов областей 1, 2 и 3. Наша задача — найти количество элементов, входящих в область 2.

Дополним этой информацией исходную таблицу.

Запрос	Найдено страниц (в тысячах)	Мощность множеств, соответствующих запросам
крейсер	4800	$N_1 + N_2$
линкор	4500	$N_2 + N_3$
крейсер | линкор	7000	$N_1 + N_2 + N_3$
крейсер & линкор	?	N_2

Если просто сложить мощности множеств $К$ и $Л$, то мы получим:

$$(N_1 + N_2) + (N_2 + N_3) = (N_1 + N_2 + N_3) + N_2.$$

Следовательно:

$$N_2 = (N_1 + N_2) + (N_2 + N_3) - (N_1 + N_2 + N_3) =$$
$$= 4800 + 4500 - 7000 = 2300.$$

Ответ: по запросу крейсер & линкор будет найдено 2300 страниц (в тысячах).

Самостоятельно найдите количество элементов, входящих: в область 1; в область 3. Какие запросы будут соответствовать каждой из этих областей?

2.2.4. Логические выражения

На уроках алгебры вы составляете, преобразуете и вычисляете значения арифметических выражений, состоящих из чисел, переменных, знаков арифметических операций и скобок. Точно так же из логических значений, логических переменных, логических операций и скобок можно составлять логические выражения.

Любое составное высказывание можно записать в виде логического выражения — выражения, содержащего логические

переменные, логические значения, знаки логических операций и скобки.

Логическое выражение — это запись составного высказывания, составленная из логических переменных, логических значений, знаков логических операций и скобок.

Логические операции в логическом выражении выполняются в следующей очерёдности: инверсия, конъюнкция, дизъюнкция. Изменить порядок выполнения операций можно с помощью расстановки скобок.

Логические операции при выполнении имеют следующий приоритет: инверсия, конъюнкция, дизъюнкция.

Для вычисления значения логического выражения необходимо:

1) вычислить значения выражений в скобках (при наличии скобок);
2) выполнить логические операции в соответствии с их приоритетом.

Рассмотрим некоторые логические выражения и вычислим их значения:

1) $(0 \vee 1) \wedge 1 = 1 \wedge 1 = 1$;
2) $\neg(0 \vee 1 \wedge 1) = \neg(0 \vee 1) = \neg 1 = 0$;
3) $(1 \vee 1) \wedge (0 \vee 1) = 1 \wedge 1 = 1$;
4) $1 \vee 1 \wedge 0 \vee 1 = 1 \vee 0 \vee 1 = 1$;
5) $\neg(0 \wedge 1) \wedge \neg 1 = \neg\ 0 \wedge \neg 1 = 1 \wedge 0 = 0$.

Ни одно из рассмотренных выше логических выражений 1–5 не содержало логических переменных, поэтому их можно было вычислить с помощью таблиц истинности логических операций.

Рассмотрим примеры логических выражений, содержащих логические переменные.

1) $A \vee 0 = A$ — нулевой операнд не может повлиять на результат логического сложения, который будет полностью зависеть от значения A;
2) $A \wedge 0 = 0$ — так как один из операндов равен 0, то результат логического умножения тоже будет равен 0, независимо от того, чему равно A;
3) $A \vee 1 = 1$ — так как один из операндов равен 1, то логическая сумма будет равна 1 при любом значении A;
4) $A \wedge 1 = A$ — единичный операнд не может повлиять на результат логического умножения, который будет полностью зависеть от значения A.

Равенства 1–4 называют иначе *законами операций с константами* 0 *и* 1; они бывают полезны при определении истинности логических высказываний.

Пример 2

Определим истинность высказывания *(X < 13) И НЕ (X < 2)* при $X = 0$.

Вначале определим истинность простых высказываний:

0 < 13 — истинное высказывание;

0 < 2 — истинное высказывание.

Запишем логическое выражение, соответствующее исходному высказыванию, и вычислим его значение:

$1 \wedge \neg 1 = 1 \wedge 0 = 0$.

Пример 3

Определим, при каких целых значениях X высказывание *(X < 13) И НЕ (X < 2)* будет истинным.

Для этого, в первую очередь, избавимся от отрицания: вместо *НЕ (X < 2)* запишем *(X >= 2)*.

Перейдём от записи *(X < 13) И (X >= 2)* к принятой в математике записи системы неравенств:

$$\begin{cases} X<13, \\ X>=2. \end{cases}$$

Изобразим решение этой системы неравенств графически:

Решением системы неравенств является множество из следующих одиннадцати чисел {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}; минимальным из них является 2, максимальным — 12.

При всех перечисленных значениях X исходное высказывание будет истинным.

2.2.5. Решение логических задач путём преобразования логических выражений

Задачи, в которых требуется определить истинность или ложность некоторых высказываний, принято называть логическими. Существуют разные способы решения логических задач, в том числе путём преобразования логических выражений.

Для логических операций справедливы многие законы, аналогичные законам для арифметических операций:

Закон	Арифметика	Математическая логика
Переместительный	$a + b = b + a$ $a \cdot b = b \cdot a$	$a \vee b = b \vee a$ $a \wedge b = b \wedge a$
Сочетательный	$(a + b) + c = a + (b + c)$ $(a \cdot b) \cdot c = a \cdot (b \cdot c)$	$(a \vee b) \vee c = a \vee (b \vee c)$ $(a \wedge b) \wedge c = a \wedge (b \wedge c)$
Распределительный	$a \cdot (b + c) = a \cdot c + a \cdot b$	$a \wedge (b \vee c) = a \wedge c \vee a \wedge b$

Покажем, как эти законы могут быть использованы при решении логических задач.

Задача

В соревнованиях по гимнастике участвуют Алла, Валя, Сима и Даша. Болельщики высказали предположения о возможных победителях:

1) Сима будет первой, Валя — второй;
2) Сима будет второй, Даша — третьей;
3) Алла будет второй, Даша — четвёртой.

По окончании соревнований оказалось, что в каждом из предположений только одно из высказываний истинно, другое — ложно. Какое место на соревнованиях заняла каждая из девушек, если все они оказались на разных местах?

Решение

Рассмотрим простые высказывания:

C_1 = «*Сима заняла первое место*»;
B_2 = «*Валя заняла второе место*»;
C_2 = «*Сима заняла второе место*»;
$Д_3$ = «*Даша заняла третье место*»;
A_2 = «*Алла заняла второе место*»;
$Д_4$ = «*Даша заняла четвёртое место*».

Так как в каждом из трёх предположений одно из высказываний истинно, а другое ложно, то можно заключить следующее:

1) $C_1 + B_2 = 1$, $C_1 \cdot B_2 = 0$;
2) $C_2 + Д_3 = 1$, $C_2 \cdot Д_3 = 0$;
3) $A_2 + Д_4 = 1$, $A_2 \cdot Д_4 = 0$.

Логическое произведение истинных высказываний будет истинным:

$$(C_1 + B_2) \cdot (C_2 + Д_3) \cdot (A_2 + Д_4) = 1.$$

На основании распределительного закона преобразуем левую часть этого выражения:

$$(C_1 \cdot C_2 + C_1 \cdot Д_3 + B_2 \cdot C_2 + B_2 \cdot Д_3) \cdot (A_2 + Д_4) = 1.$$

Высказывание $C_1 \cdot C_2$ означает, что Сима заняла и первое, и второе место. Согласно условию задачи, это высказывание ложно. Ложным является и высказывание $B_2 \cdot C_2$. Учитывая закон операций с константой 0, запишем:

$$(C_1 \cdot Д_3 + B_2 \cdot Д_3) \cdot (A_2 + Д_4) = 1.$$

Дальнейшее преобразование левой части этого равенства и исключение заведомо ложных высказываний дают:

$$C_1 \cdot Д_3 \cdot A_2 + C_1 \cdot Д_3 \cdot Д_4 + B_2 \cdot Д_3 \cdot A_2 + B_2 \cdot Д_3 \cdot Д_4 = 1.$$
$$C_1 \cdot Д_3 \cdot A_2 = 1.$$

Из последнего равенства следует, что $C_1 = 1$, $Д_3 = 1$, $A_2 = 1$. Это означает, что Сима заняла первое место, Алла — второе, Даша — третье. Следовательно, Валя заняла четвёртое место.

САМОЕ ГЛАВНОЕ

В алгебре логики высказывания обозначают буквами и называют логическими переменными. 0 и 1, обозначающие значения логических переменных, называют логическими значениями.

Заменив высказывания логическими переменными, можно рассматривать логические связки как логические операции над переменными.

Инверсия — логическая операция, ставящая в соответствие высказыванию новое высказывание, значение которого противоположно значению исходного.

Конъюнкция — логическая операция, ставящая в соответствие двум высказываниям новое высказывание, являющееся истинным тогда и только тогда, когда оба исходных высказывания истинны.

Дизъюнкция — логическая операция, ставящая в соответствие двум высказываниям новое высказывание, являющееся ложным тогда и только тогда, когда оба исходных высказывания ложны.

Логическое выражение — это запись составного высказывания, составленная из логических переменных, логических значений, знаков логических операций и скобок.

Вопросы и задания

1. В следующих составных высказываниях выделите простые высказывания, обозначив каждое из них буквой; запишите с помощью букв и знаков логических операций составное высказывание.
   а) *Число 376 чётное и трёхзначное.*
   б) *Зимой дети катаются на коньках или на лыжах.*
   в) *Новый год мы встречаем на даче или на Красной площади.*
   г) *Неверно, что Солнце движется вокруг Земли.*
   д) *Земля имеет форму шара, который из космоса кажется голубым.*
   е) *На уроке математики старшеклассники отвечали на вопросы учителя, а также писали самостоятельную работу.*
2. Пусть A = «*Ане нравятся уроки математики*», а B = «*Ане нравятся уроки химии*». Выразите следующие логические выражения на разговорном языке:
   а) $A \wedge B$; г) $A \vee B$; ж) $\overline{A \wedge B}$;
   б) $\overline{A} \wedge B$; д) $A \vee \overline{B}$; з) $\overline{A \vee B}$;
   в) $A \wedge \overline{B}$; е) $\overline{A} \vee B$; и) $\overline{A \wedge \overline{B}}$.
3. Найдите значения выражений:
   а) $(1 \vee 1) \vee (1 \vee 0)$;
   б) $(((1 \vee 0) \vee 1) \vee 1)$;
   в) $(0 \wedge 1) \wedge 1$:
   г) $1 \wedge (1 \wedge 1) \wedge 1$;
   д) $((1 \vee 0) \wedge (1 \wedge 1)) \wedge (0 \vee 1)$;
   е) $((1 \wedge 1) \vee 0) \wedge (0 \vee 1)$;
   ж) $((0 \wedge 0) \vee 0) \wedge (1 \vee 1)$;
   з) $(A \vee 1) \vee (B \vee 0)$;
   и) $((1 \wedge A) \vee (B \wedge 0)) \vee 1$;
   к) $1 \wedge A \wedge 0$.

4. Пусть A = *«Первая буква имени — гласная»*, B = *«Четвёртая буква имени — согласная»*. Найдите значение логического выражения $\overline{A} \vee B$ для следующих имён:
   а) ЕЛЕНА;
   б) ВАДИМ;
   в) АНТОН;
   г) ФЁДОР.
5. Пусть A = *«$X < 3$»*, B = *«$X >= 5$»*. Найдите значение логического выражения $\overline{A} \wedge \overline{B}$ для следующих значений числа X:
   а) 2;
   б) 3;
   в) 4;
   г) 5;
   д) 6.
6. Пусть $M = \{1, 2, 3, 4, 5, 6\}$, $K = \{1, 3, 5\}$, $P = \{2, 4, 6, 7, 8\}$. Запишите в фигурных скобках области истинности следующих высказывательных форм:
   а) $(x \in M) \wedge (x \in P)$;
   б) $(x \in K) \wedge (x \in P)$;
   в) $x \in M \cap P$;
   г) $x \in K \cup P$.
7. В языке запросов поискового сервера для обозначения логической операции ИЛИ используется символ «|», а для обозначения логической операции И — символ «&». В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Запрос	Найдено страниц (в тысячах)
крейсер | линкор	3700
крейсер & линкор	400
линкор	1800

   Какое количество страниц (в тысячах) будет найдено по запросу

   крейсер ?

8. Определите наименьшее целое число X, для которого истинно высказывание: *НЕ ($X < 59$) И НЕ (X — чётное)*.

9. Определите наибольшее целое число X, для которого истинно высказывание: *НЕ (X >= 60) И НЕ (X — нечётное)*.
10. Алёша, Боря и Гриша нашли в земле старинный сосуд. Рассматривая удивительную находку, каждый высказал по два предположения.

    1) Алёша: «Это сосуд греческий и изготовлен в V веке».

    2) Боря: «Это сосуд финикийский и изготовлен в III веке».

    3) Гриша: «Это сосуд не греческий и изготовлен в IV веке».

    Учитель истории сказал ребятам, что каждый из них прав только в одном из двух предположений. Где и в каком веке изготовлен сосуд?

§ 2.3 Таблицы истинности логических выражений

Ключевые слова:

- таблица истинности

2.3.1. Построение таблиц истинности для логических выражений

Вы уже знакомы с таблицами истинности логических операций. В них установлено соответствие между всеми возможными наборами операндов и результатами выполнения соответствующих операций.

Таблица истинности логического выражения показывает, какие значения принимает выражение при всех наборах значений входящих в него переменных.

Для построения таблицы истинности следует:

1) подсчитать n — число переменных в выражении;
2) установить последовательность выполнения логических операций с учётом скобок и приоритетов;
3) подсчитать общее число логических операций в выражении;
4) определить число столбцов в таблице: число переменных + + число операций;

5) заполнить шапку таблицы, включив в неё переменные и операции в соответствии с последовательностью, установленной в п. 2;
6) определить число строк в таблице (не считая шапки таблицы): $m = 2^n$;
7) выписать наборы входных переменных с учётом того, что они представляют собой ряд целых n-разрядных двоичных чисел от 0 до $2^n - 1$;
8) провести заполнение таблицы по столбцам, выполняя логические операции в соответствии с установленной последовательностью.

Пример 1

Построим таблицу истинности для логического выражения $A \vee A \wedge B$.

В нём две переменные, две операции, причём сначала выполняется конъюнкция, а затем — дизъюнкция. Всего в таблице будет четыре столбца:

A	B	$A \wedge B$	$A \vee A \wedge B$

Наборы переменных — это целые числа от 0 до 3, представленные в двухразрядном двоичном коде: 00, 01, 10, 11.

Заполненная таблица истинности имеет вид:

A	B	$A \wedge B$	$A \vee A \wedge B$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Обратите внимание, что последний столбец (результат) совпал со столбцом A. В этом случае можно сказать, что логическое выражение $A \vee A \wedge B$ **равносильно** логической переменной A.

Таблицу истинности можно построить для логического выражения с любым числом переменных n, главное — правильно записать все 2^n комбинаций 0 и 1, являющихся значениями переменных. Сделать это можно так: в столбце для последней переменной чередуйте 0 и 1, для предпоследней — 00 и 11, затем — 0000 и 1111 и т. д.

С помощью таблиц истинности можно доказывать различные свойства логических операций, называемые также **законами алгебры логики.**

Пример 2

Докажем распределительный закон для логического сложения:

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C).$$

A	B	C	$B \wedge C$	$A \vee (B \wedge C)$	$A \vee B$	$A \vee C$	$(A \vee B) \wedge (A \vee C)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Совпадение значений в выделенных столбцах, соответствующих логическим выражениям в левой и правой частях равенства, доказывает справедливость распределительного закона для логического сложения.

2.3.2. Использование таблиц истинности для решения логических задач

Вы уже знаете, что решать логические задачи можно, составляя, упрощая и анализируя логические выражения. Рассмотрим ещё один способ решения логических задач — с помощью таблиц истинности.

Задача

Коля, Вася и Серёжа гостили летом у бабушки. Однажды один из мальчиков нечаянно разбил любимую бабушкину вазу. На вопрос, кто разбил вазу, они дали такие ответы:

Серёжа: 1) я не разбивал; 2) Вася не разбивал.

Вася: 3) Серёжа не разбивал; 4) вазу разбил Коля.

Коля: 5) я не разбивал; 6) вазу разбил Серёжа.

Бабушка знала, что один из её внуков, назовём его правдивым, оба раза сказал правду; второй, назовём его шутником, оба раза сказал неправду; третий, назовём его хитрецом, один раз сказал правду, а другой раз — неправду.

Назовите имена правдивого, шутника и хитреца. Кто из внуков разбил вазу?

Решение

Пусть K = «*Коля разбил вазу*», B = «*Вася разбил вазу*», C = «*Серёжа разбил вазу*». Для решения задачи можно составить таблицу истинности, в которой представить все возможные варианты значений высказываний каждого мальчика. Но так как ваза разбита одним внуком, то достаточно фрагмента таблицы истинности, содержащего наборы значений переменных: 001, 010, 100.

K	B	C	Утверждения Серёжи		Утверждения Васи		Утверждения Коли	
			$\overline{C}$	$\overline{B}$	$\overline{C}$	K	$\overline{K}$	C
0	0	1	0	1	0	0	1	1
0	1	0	1	0	1	0	1	0
1	0	0	1	1	1	1	0	0

Исходя из того, что знает о внуках бабушка, следует искать в таблице строку, содержащую в каком-либо порядке три комбинации значений: 00 (слова шутника), 11 (слова правдивого внука), 01 или 10 (слова хитреца). Такая строка выделена жирной рамкой. Согласно этой строке, вазу разбил Серёжа, он же оказался хитрецом. Шутником оказался Вася. Имя правдивого внука — Коля.

САМОЕ ГЛАВНОЕ

Таблица истинности логического выражения показывает, какие значения принимает логическое выражение при всех наборах значений входящих в него переменных.

Таблицу истинности можно построить для логического выражения с любым числом переменных n; такая таблица будет содержать 2^n строк.

Вопросы и задания

1. Постройте таблицы истинности для следующих логических выражений:
   а) $B \wedge (A \vee B)$;
   б) $A \wedge (B \vee \overline{B})$;
   в) $B \wedge (A \vee B \vee C)$;
   г) $\overline{A \wedge B \vee \overline{C}}$.

2. С помощью таблиц истинности докажите справедливость следующих тождеств:
   а) $\overline{A} \vee \overline{B} = \overline{A \wedge B}$;
   б) $\overline{A} \wedge \overline{B} = \overline{A \vee B}$.

3. Сколько строк содержат таблицы истинности для следующих выражений? Для каких наборов значений переменных эти выражения истинны?
   а) $A \wedge B \wedge C$;
   б) $A \wedge B \wedge C \wedge D$;
   в) $A \vee B \vee C$;
   г) $A \vee B \vee C \vee D$.

   Ответьте на вопросы, не прибегая к заполнению таблиц истинности.

4. В школьной олимпиаде по информатике приняли участие три ученика 8 класса: Александр, Иван и Мария. Перед олимпиадой их друзья высказали три предположения.
   1) Александр сможет пройти на городской тур олимпиады, или Иван не сможет пройти на городской тур олимпиады.
   2) Иван сможет пройти на городской тур олимпиады.
   3) Неверно, что Мария и Александр смогут пройти на городской тур олимпиады.

   Кто из ребят прошёл на городской этап олимпиады, если все предположения оказались истинными высказываниями?

§ 2.4 Логические элементы

Ключевые слова:

- логический элемент
- инвертор
- конъюнктор
- дизъюнктор
- цифровая схема

2.4.1. Логические элементы

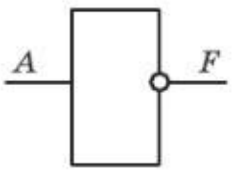
Алгебра логики — раздел математики, играющий важную роль в конструировании автоматических устройств, разработке аппаратных и программных средств информационных и коммуникационных технологий.

Вы уже знаете, что любая информация может быть представлена в дискретной форме — в виде фиксированного набора отдельных значений. Устройства, которые обрабатывают такие значения (сигналы), называются дискретными. Дискретный преобразователь, который выдаёт после обработки двоичных сигналов значение одной из логических операций, называется логическим элементом.

Логический элемент — устройство, которое обрабатывает двоичные данные и выдаёт после их обработки значение одной из логических операций.

Рассмотрим логические элементы, реализующие инверсию, логическое умножение и логическое сложение.

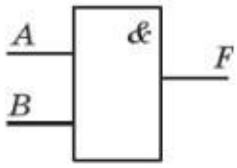
Логический элемент НЕ **(инвертор)** реализует операцию отрицания. Если на входе элемента будет 0, то на выходе будет 1, и наоборот. Схема и таблица истинности инвертора представлены на рис. 2.3.



A	*F*
0	1
1	0

Рис. 2.3. Инвертор

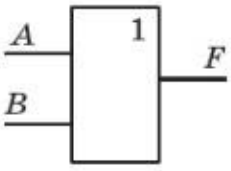
Логический элемент И **(конъюнктор)** реализует операцию логического умножения. Единица на выходе этого элемента появится только тогда, когда на всех входах будут единицы. Схема и таблица истинности конъюнктора представлены на рис. 2.4.



A	*B*	*F*
0	0	0
0	1	0
1	0	0
1	1	1

Рис. 2.4. Конъюнктор

Логический элемент ИЛИ **(дизъюнктор)** реализует операцию логического сложения. Если хотя бы на одном входе будет единица, то на выходе элемента также будет единица. Схема и таблица истинности дизъюнктора представлены на рис. 2.5.



A	*B*	*F*
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 2.5. Дизъюнктор

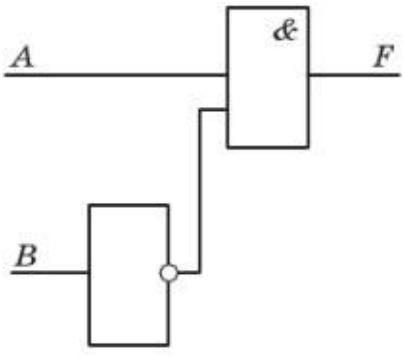
Компьютерные устройства (цифровые микросхемы), производящие операции над двоичными данными, и ячейки, хранящие данные, состоят из множества логических элементов.

2.4.2. Анализ и синтез цифровых схем

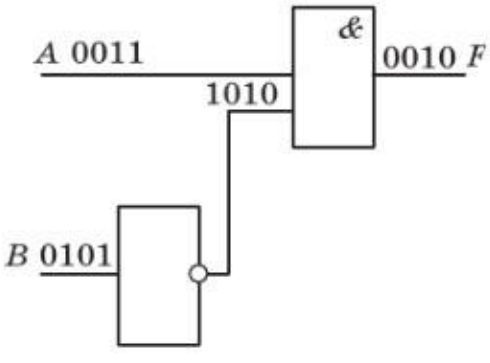
Цифровая схема — это схема, состоящая из нескольких логических элементов.

Пример 1

Проанализируем цифровую схему, т. е. выясним, какой сигнал будет на выходе F при каждом возможном наборе сигналов A и B на входах.



Все возможные комбинации сигналов A и B на входах внесём в таблицу истинности. Проследим преобразование каждой пары сигналов при прохождении их через логические элементы и запишем полученный результат в таблицу. Заполненная таблица истинности полностью описывает рассматриваемую цифровую схему.



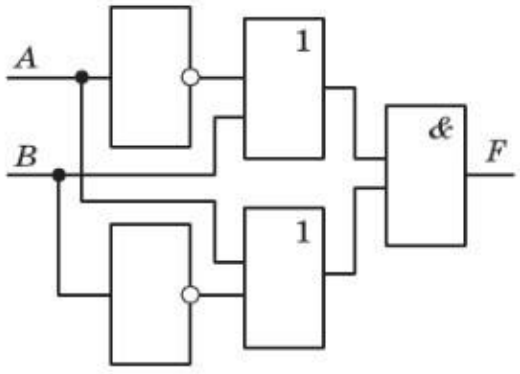
A	B	F
0	0	0
0	1	0
1	0	1
1	1	0

Составим логическое выражение, соответствующее рассматриваемой схеме. Последний логический элемент в рассматриваемой схеме — конъюнктор. В него поступают сигналы от входа A и от инвертора. В свою очередь, в инвертор поступает сигнал от входа B. Таким образом: $F = A \,\&\, \overline{B}$.

Пример 2

Построим цифровую схему для логического выражения $(\overline{A} \vee B) \;\&\; (A \vee \overline{B})$.

В логическом выражении присутствуют две операции инверсии, две операции дизъюнкции и одна операция конъюнкции. Для соответствующей цифровой схемы нам потребуются логические элементы: два инвертора, два дизъюнктора, один конъюнктор. Порядок выполнения логических операций определяет порядок соединения логических элементов.



A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Цифровые микросхемы очень сложны. Вот так, например, выглядит электронная схема одноразрядного сумматора, выполняющего сложение младших разрядов двух двоичных чисел.

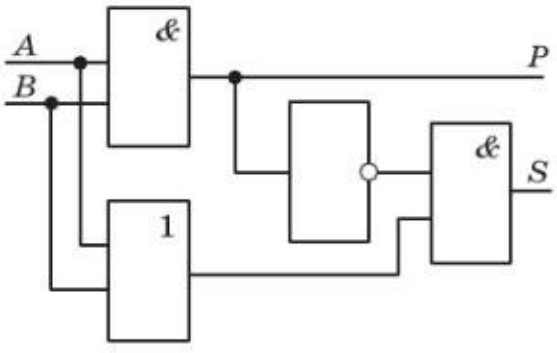


Таблица истинности этого устройства имеет вид

A	B	S	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Работу этого устройства можно описать логическими выражениями:

$S = (A \vee B)\ \&\ (\overline{A\ \&\ B})$;

$P = A\ \&\ B$.

При разработке цифровых схем их работу первоначально описывают с помощью таблиц истинности, указывая значения выходных сигналов, которые должны получаться при разных наборах входных сигналов. Затем, по определённым правилам, записывают логическое выражение, соответствующее полученной таблице истинности. Полученное логическое выражение пытаются упростить и только после этого на его основе строят цифровую схему.

САМОЕ ГЛАВНОЕ

Логический элемент — устройство, которое обрабатывает двоичные данные и выдаёт после их обработки значение одной из логических операций.

Логический элемент НЕ (инвертор) реализует операцию отрицания. Если на входе элемента будет 0, то на выходе будет 1, и наоборот.

Логический элемент И (конъюнктор) реализует операцию логического умножения. Единица на выходе этого элемента появится только тогда, когда на всех входах будут единицы.

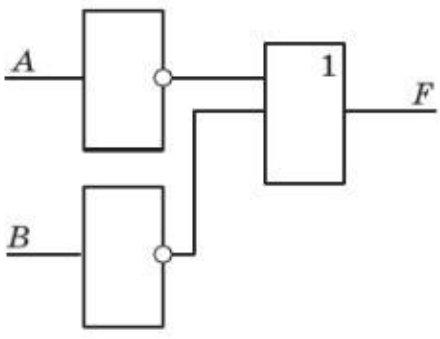
Логический элемент ИЛИ (дизъюнктор) реализует операцию логического сложения. Если хотя бы на одном входе будет единица, то на выходе элемента также будет единица.

Компьютерные устройства (цифровые микросхемы), производящие операции над двоичными данными, и ячейки, хранящие данные, состоят из множества логических элементов.

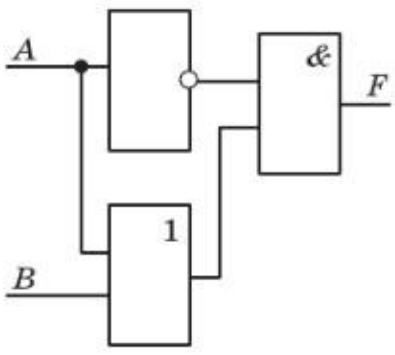
Вопросы и задания

1. Какие сигналы будут на выходе цифровой схемы при всех возможных наборах сигналов на входах? Составьте таблицу работы схемы. Каким логическим выражением описывается схема?

а)



б)



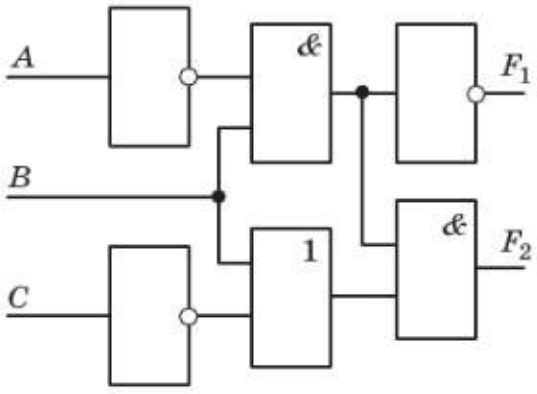
2. Какие сигналы будут на выходах F_1 и F_2 схемы, если на её входы A, B и C поданы следующие сигналы?

   а) $A = 0$, $B = 0$, $C = 1$;

   б) $A = 0$, $B = 1$, $C = 1$;

   в) $A = 1$, $B = 0$, $C = 0$;

   г) $A = 1$, $B = 1$, $C = 1$.



3. Постройте цифровые схемы по следующим логическим выражениям:

   а) $\overline{A} \vee \overline{B}$;

   б) $\overline{A \wedge B}$.

   Выясните, какие сигналы будут на выходе каждой цифровой схемы при всех возможных наборах сигналов на входах. Составьте таблицы, описывающие работу схем. Сравните получившиеся таблицы.

Тестовые задания для самоконтроля

1. Какое предложение НЕ является высказыванием?
   а) *Никакая причина не извиняет невежливость.*
   б) *Обязательно стань отличником.*
   в) *Логика — наука о законах и формах человеческого мышления.*
   г) $1011_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$.

2. Какое высказывание является ложным?
   а) *Знаком $\vee$ обозначается логическая операция ИЛИ.*
   б) *Логическую операцию ИЛИ также называют логическим сложением.*
   в) *Дизъюнкцию называют логическим сложением.*
   г) *Знаком $\vee$ обозначается логическая операция конъюнкция.*

3. Для какого из указанных значений числа X истинно высказывание $((X < 5) \vee (X < 3)) \wedge ((X < 2) \vee (X < 1))$?
   а) 1
   б) 2
   в) 3
   г) 4

4. Для какого символьного выражения верно высказывание: «*НЕ (Первая буква согласная) И НЕ (Вторая буква гласная)*»?
   а) abcde
   б) bcade
   в) babas
   г) cabab

5. Сколько всего целых значений X, для которых истинно следующее высказывание: $НЕ\ (X < 5)\ И\ НЕ\ (X > 9)$?
   а) 3
   б) 4
   в) 5
   г) 8

6. В языке запросов поискового сервера для обозначения логической операции ИЛИ используется символ «|», а для обозначения логической операции И — символ «&». В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Запрос	Найдено страниц (в тысячах)
крейсер	4700
линкор	2800
крейсер & линкор	1200

Какое количество страниц (в тысячах) будет найдено по запросу

крейсер | линкор ?

а) 7500
б) 6300
в) 5900
г) 4000

7. На перекрёстке произошло дорожно-транспортное происшествие, в котором участвовали автобус (*А*), грузовик (*Г*), легковой автомобиль (*Л*) и маршрутное такси (*М*). Свидетели происшествия дали следующие показания. Первый свидетель считал, что первым на перекрёсток выехал автобус, а маршрутное такси было вторым. Другой свидетель полагал, что последним на перекрёсток выехал легковой автомобиль, а вторым был грузовик. Третий свидетель уверял, что автобус выехал на перекрёсток вторым, а следом за ним — легковой автомобиль. В результате оказалось, что каждый из свидетелей был прав только в одном из своих утверждений. В каком порядке выехали машины на перекрёсток? В вариантах ответов перечислены подряд без пробелов первые буквы названий транспортных средств в порядке их выезда на перекрёсток:

а) *АМЛГ*
б) *АГЛМ*
в) *ГЛМА*
г) *МЛГА*

8. Какому логическому выражению соответствует следующая таблица истинности?

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

а) $A \wedge B$ б) $A \vee B$ в) $\overline{A \wedge B}$ г) $\overline{A} \wedge \overline{B}$

9. Для скольких наборов значений переменных ложно выражение $A \wedge B \wedge C \wedge D$?

а) 1 б) 4 в) 15 г) 16

10. Разбирается дело Джона, Брауна и Смита. Известно, что один из них нашёл и утаил клад. На следствии каждый из подозреваемых сделал два заявления.

Смит: «Я не делал этого. Браун сделал это».

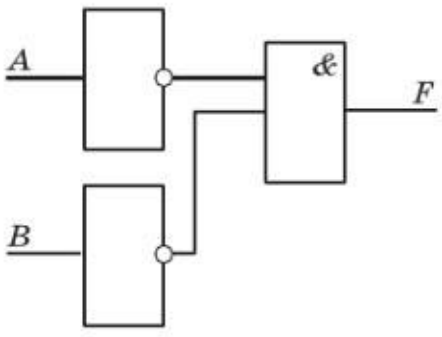
Джон: «Браун не виновен. Смит сделал это».

Браун: «Я не делал этого. Джон не делал этого».

Суд установил, что один из них дважды солгал, другой дважды сказал правду, третий один раз солгал, один раз сказал правду. Кто из подозреваемых должен быть оправдан?

а) Смит
б) Джон и Браун
в) Джон и Смит
г) Браун и Смит

11. Какое логическое выражение соответствует следующей схеме?



а) $A \,\&\, B$
б) $A \vee B$
в) $\overline{A \,\&\, B}$
г) $\overline{A} \,\&\, \overline{B}$

Глава 3
ОСНОВЫ АЛГОРИТМИЗАЦИИ

§ 3.1
Алгоритмы и исполнители

Ключевые слова:

- алгоритм
- свойства алгоритма
  - ✓ дискретность
  - ✓ понятность
  - ✓ определённость
  - ✓ результативность
  - ✓ массовость
- исполнитель
- характеристики исполнителя
  - ✓ круг решаемых задач
  - ✓ среда
  - ✓ режим работы
  - ✓ система команд
- формальное исполнение алгоритма

3.1.1. Понятие алгоритма

Каждый человек в повседневной жизни, в учёбе или на работе решает огромное количество задач самой разной сложности. Сложные задачи требуют длительных размышлений для нахождения решения; простые и привычные задачи человек решает, не задумываясь, автоматически. В большинстве случаев решение каждой задачи можно разбить на простые этапы (шаги). Для многих таких задач (установка программного обеспечения, сборка шкафа, создание сайта, эксплуатация технического устройства, покупка авиабилета через Интернет и т. д.) уже разработаны и предлагаются пошаговые инструкции, при последовательном выполнении которых можно прийти к желаемому результату.

Пример 1

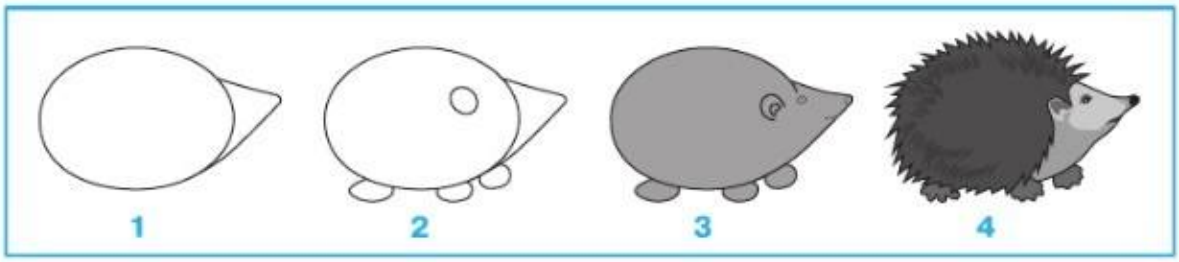
Решение задачи «Зарегистрировать аккаунт в VK» можно осуществить за шесть шагов:

1. Зайти на сайт **vk.com**.
2. Нажать на кнопку **Зарегистрироваться**.

3. Ввести номер телефона.
4. Подтвердить вход (ввести в окно ввода код из СМС).
5. Ввести информацию о себе (имя, фамилия, дата рождения, пол).
6. Подтвердить ввод данных.

Пример 2

Этапы решения задачи «Нарисовать весёлого ёжика» представлены графически:



Пример 3

Задача «Найти наибольший общий делитель (НОД) двух целых чисел» может быть решена так:

1. Разложить первое число на простые множители.
2. Разложить второе число на простые множители.
3. Выписать все простые множители, общие для двух чисел.
4. Вычислить НОД путём перемножения общих простых множителей.

Пример 4

Самое простое решение задачи «Сортировка мусора» может быть таким:

1. Органические отходы (частицы пищи, растения) поместить в бак серого цвета.
2. Неорганические отходы (пластик, бумага, картон, стекло, металл) поместить в бак синего цвета.
3. Ртутные градусники, люминесцентные лампы, батарейки и аккумуляторы сложить отдельно, в небольшую коробку.

Регистрация аккаунта, рисование ёжика, нахождение НОД и сортировка мусора — на первый взгляд, совершенно разные

процессы. Но у них есть общая черта: каждый из этих процессов описывается последовательностью кратких указаний, точное следование которым позволяет получить требуемый результат. Последовательности указаний, приведённые в примерах 1–4, являются алгоритмами решения соответствующих задач. Исполнитель этих алгоритмов — человек.

Алгоритм может представлять собой описание некоторой последовательности вычислений (пример 3) или шагов нематематического характера (примеры 1, 2, 4). Но в любом случае перед его разработкой должны быть чётко определены начальные условия (исходные данные) и то, что предстоит получить (результат). Можно сказать, что **алгоритм** — это описание последовательности шагов в решении задачи, приводящих от исходных данных к требуемому результату.

В общем виде схему работы алгоритма можно представить следующим образом (рис. 3.1).

Рис. 3.1. Общая схема работы алгоритма

Алгоритмами являются изучаемые в школе правила сложения, вычитания, умножения и деления чисел, многие грамматические правила, правила геометрических построений и т. д.

Пример 5

Следующий алгоритм приводит к тому, что из одного натурального числа получается другое натуральное число:

1. Исходное число переводится в двоичную систему счисления.
2. Подсчитывается число единиц в полученной двоичной записи исходного числа.
3. Если число единиц в двоичной записи нечётное, то к этой записи справа приписывается 1, если чётное — 0.
4. Число, соответствующее дополненной двоичной записи, переводится в десятичную систему счисления.

Получившееся таким образом число является результатом работы алгоритма.

Так, если исходным было число 21, то результатом работы алгоритма будет число 43, а если исходным числом было 15, то результатом работы алгоритма будет число 30.

3.1.2. Исполнитель алгоритма

Каждый алгоритм предназначен для определённого исполнителя.

Исполнитель — это некоторый объект (человек, животное, техническое устройство), способный выполнять определённый набор команд.

Различают формальных и неформальных исполнителей. Формальный исполнитель не вникает в смысл того, что он делает, и не рассуждает, почему он поступает так, а не иначе. Одну и ту же команду формальный исполнитель всегда выполняет одинаково. Неформальный исполнитель может выполнять команду по-разному.

Рассмотрим более подробно множество формальных исполнителей. Формальные исполнители необычайно разнообразны, но для каждого из них можно указать следующие характеристики: круг решаемых задач (назначение), среду, систему команд и режим работы.

Круг решаемых задач. Каждый исполнитель создаётся для решения некоторого круга задач — построения цепочек символов, выполнения вычислений, построения рисунков на плоскости и т. д.

Среда исполнителя. Область, обстановку, условия, в которых действует исполнитель, принято называть средой данного исполнителя. Исходные данные и результаты любого алгоритма всегда принадлежат среде того исполнителя, для которого предназначен алгоритм. Среду можно рассматривать как полный набор характеристик, описывающих состояние исполнителя.

Система команд исполнителя. Совокупность всех команд, которые могут быть выполнены некоторым исполнителем, образует систему команд данного исполнителя (СКИ). Алгоритм составляется с учётом возможностей конкретного исполнителя, иначе говоря, в системе команд исполнителя, который будет его выполнять.

Режимы работы исполнителя. Для большинства исполнителей предусмотрены *режимы непосредственного (ручного) управления и программного управления.* В первом случае исполнитель немедленно выполняет каждую поступившую команду. В таком режиме работает пульт кондиционера или телевизора. Во втором случае исполнителю сначала задаётся полная последовательность команд (программа), а затем он выполняет все эти команды в автоматическом режиме. Например, в память стиральной машины

заложены разные достаточно сложные программы стирки, каждая из которых предполагает ряд последовательных действий.

Рассмотрим примеры исполнителей.

Пример 6

Исполнитель **Черепаха** перемещается на экране компьютера, оставляя след в виде линии. Система команд Черепахи состоит из следующих команд:

`Вперёд n` (где `n` — целое число) — вызывает передвижение Черепахи на `n` шагов в направлении движения — в том направлении, куда развёрнуты её голова и корпус;

`Направо m` (где `m` — целое число) — вызывает изменение направления движения Черепахи на `m` градусов по часовой стрелке.

Запись

```
Повтори k [<Команда 1> <Команда 2> … <Команда n>]
```

означает, что последовательность команд в скобках повторится `k` раз.

Подумайте, какая фигура появится на экране после выполнения Черепахой следующего алгоритма:

```
Повтори 12 [Направо 45 Вперёд 20 Направо 45]
```

Пример 7

Система команд исполнителя **Вычислитель** состоит из двух команд, которым присвоены номера:

```
1. вычти 1
2. умножь на 3
```

Первая из них уменьшает число на 1, вторая увеличивает число в 3 раза. При записи алгоритмов для краткости указываются лишь номера команд.

Например, алгоритм `21212` означает следующую последовательность команд:

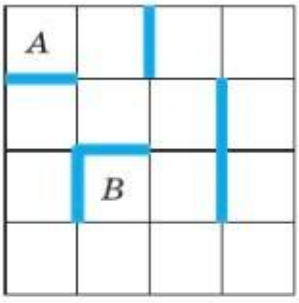
```
умножь на 3
вычти 1
умножь на 3
вычти 1
умножь на 3
```

С помощью этого алгоритма число 1 будет преобразовано в 15: $((1 \cdot 3 - 1) \cdot 3 - 1) \cdot 3 = 15$.

Пример 8

Исполнитель **Робот** действует на клетчатом поле, между соседними клетками которого могут стоять стены. Робот передвигается по клеткам поля и может выполнять следующие команды, которым присвоены номера:

```
1. вверх
2. вниз
3. вправо
4. влево
```



При выполнении каждой такой команды Робот перемещается в соседнюю клетку в указанном направлении. Если же в этом направлении между клетками стоит стена, то Робот разрушается.

Что произойдёт с Роботом из примера 8, если он выполнит последовательность команд 32323 (здесь цифры обозначают номера команд), начав движение из клетки *A*? Какую последовательность команд следует выполнить Роботу, чтобы переместиться из клетки *A* в клетку *B*, не разрушившись при этом от столкновения со стеной?

Пример 9

К пятизначному натуральному числу применяется следующий алгоритм:

1. Вычислить сумму первых трёх цифр.
2. Вычислить сумму последних двух цифр.
3. Записать полученные два числа друг за другом в порядке возрастания (неубывания).

Пример работы алгоритма для числа 56789:

1. $5 + 6 + 7 = 18$.
2. $8 + 9 = 17$.
3. Упорядочив, получаем 1718.

Выясним наименьшее и наибольшее пятизначные числа, в результате применения к которым этого алгоритма получается такой же результат.

В старших разрядах наименьшего пятизначного числа должны быть самые маленькие цифры из возможных; первая цифра

должна быть как можно меньше и т. д. В нашем случае это: $17 = 1 + 7 + 9$.

Есть единственный вариант, позволяющий получить вторую сумму из двух цифр: $18 = 9 + 9$.

Составим наименьшее пятизначное число: 17999.

В старших разрядах наибольшего пятизначного числа должны быть самые большие цифры; первая цифра должна быть как можно больше и т. д. В нашем случае это $18 = 9 + 9 + 0$.

Вторую сумму из двух цифр можно получить только так: $17 = 9 + 8$.

Составим наибольшее пятизначное число: 99098.

При разработке алгоритма:

1) выделяются фигурирующие в задаче объекты, устанавливаются свойства объектов, отношения между объектами и возможные действия с объектами;
2) определяются исходные данные и требуемый результат;
3) определяется последовательность действий исполнителя, обеспечивающая переход от исходных данных к результату;
4) последовательность действий записывается с помощью команд, входящих в систему команд исполнителя.

Можно сказать, что алгоритм — план управления исполнителем.

По ссылке **http://gotourl.ru/11950** вы можете скачать систему программирования КуМир (**К**омплект **у**чебных **Мир**ов) со встроенными исполнителями Робот, Чертёжник, Водолей и др. КуМир работает в операционных системах Windows и Linux.

3.1.3. Свойства алгоритма

Не любая инструкция, последовательность предписаний или план действий может считаться алгоритмом. Каждый алгоритм обязательно обладает следующими свойствами: дискретностью, понятностью, определённостью, результативностью и массовостью.

Свойство дискретности означает, что путь решения задачи разделён на отдельные шаги (действия). Каждому действию соответствует предписание (команда). Только выполнив одну команду, исполнитель может приступить к выполнению следующей команды.

Свойство понятности означает, что алгоритм состоит только из команд, входящих в систему команд исполнителя, т. е. из таких команд, которые исполнитель может воспринять и по которым может выполнить требуемые действия.

Свойство определённости означает, что в алгоритме нет команд, смысл которых может быть истолкован исполнителем неоднозначно; недопустимы ситуации, когда после выполнения очередной команды исполнителю неясно, какую команду выполнять следующей. Благодаря этому результат алгоритма однозначно определяется набором исходных данных: если алгоритм несколько раз применяется к одному и тому же набору исходных данных, то на выходе всегда получается один и тот же результат.

Свойство результативности означает, что алгоритм должен обеспечивать получение результата после конечного, возможно очень большого, числа шагов. При этом результатом считается не только обусловленный постановкой задачи ответ, но и вывод о невозможности по какой-либо причине продолжения решения данной задачи.

Свойство массовости означает, что алгоритм должен обеспечивать возможность его применения для решения любой задачи из некоторого класса задач. Например, алгоритм нахождения корней квадратного уравнения должен быть применим к любому квадратному уравнению, алгоритм перехода улицы должен быть применим в любом месте улицы, алгоритм приготовления лекарства должен быть применим для приготовления любого его количества и т. д.

Пример 10

Рассмотрим один из методов нахождения всех простых чисел, не превышающих некоторое натуральное число n. Этот метод называется «решето Эратосфена» по имени предложившего его древнегреческого учёного Эратосфена (III в. до н. э.).

Для нахождения всех простых чисел, не бóльших заданного числа n, следуя методу Эратосфена, нужно выполнить такие шаги:

1. Выписать подряд все натуральные числа от 2 до n (2, 3, 4, ..., n).
2. Заключить в рамку 2 — первое простое число.
3. Вычеркнуть из списка все числа, делящиеся на последнее найденное простое число.

4. Найти первое неотмеченное число (отмеченные числа — зачёркнутые числа или числа, заключённые в рамку) и заключить его в рамку — это будет очередное простое число.
5. Повторять шаги 3 и 4 до тех пор, пока не останется неотмеченных чисел.

Рассмотренная последовательность действий является алгоритмом, так как она удовлетворяет свойствам:

- дискретности — процесс нахождения простых чисел разбит на шаги;
- понятности — каждая команда понятна ученику 8 класса, выполняющему этот алгоритм;
- определённости — каждая команда трактуется и выполняется исполнителем однозначно; имеются указания об очерёдности выполнения команд;
- результативности — через некоторое число шагов достигается результат;
- массовости — последовательность действий применима для любого натурального n.

Рассмотренные свойства алгоритма позволяют дать более точное определение алгоритма.

Алгоритм — это предназначенное для конкретного исполнителя описание последовательности действий, приводящих от исходных данных к требуемому результату, которое обладает свойствами дискретности, понятности, определённости, результативности и массовости.

3.1.4. Возможность автоматизации деятельности человека

Разработка алгоритма, как правило, трудоёмкая задача, требующая от человека глубоких знаний, изобретательности и больших временны́х затрат.

Решение задачи по готовому алгоритму требует от исполнителя только строгого следования заданным предписаниям.

Пример 11

Из кучи, содержащей любое, большее 3, количество каких-либо предметов, двое играющих по очереди берут по одному или по два предмета. Выигрывает тот, кто своим очередным ходом сможет забрать все оставшиеся предметы.

Рассмотрим алгоритм, следуя которому первый игрок наверняка обеспечит себе выигрыш.

1. Если число предметов в куче кратно 3, то уступить ход противнику, иначе начать игру, взяв 1 или 2 предмета так, чтобы осталось количество предметов, кратное 3.
2. Своим очередным ходом каждый раз дополнять число предметов, взятых соперником, до 3 (число оставшихся предметов должно быть кратно 3).

Предложите сыграть в эту игру кому-нибудь из своих родных, друзей или знакомых. Действуйте строго в соответствии с приведённым выше алгоритмом. Убедитесь, что алгоритм «работает»!

Исполнитель может не вникать в смысл того, что он делает, и не рассуждать, почему он поступает так, а не иначе, т. е. он может действовать формально. Способность исполнителя действовать формально обеспечивает возможность автоматизации деятельности человека. Для этого:

1) процесс решения задачи представляется в виде последовательности простейших операций;
2) создаётся машина (автоматическое устройство), способная выполнять эти операции в последовательности, заданной в алгоритме;
3) человек освобождается от рутинной деятельности, выполнение алгоритма поручается автоматическому устройству.

Мир, окружающий современного человека, с каждым днём наполняется всё более совершенными формальными исполнителями (цифровыми устройствами), «берущими» на себя многие рутинные виды деятельности, ранее выполнявшиеся человеком. Представления об алгоритмах позволяют понять, как работают цифровые устойства, какие ещё работы можно на них переложить, с какими трудностями при этом можно столкнуться.

САМОЕ ГЛАВНОЕ

Исполнитель — некоторый объект (человек, животное, техническое устройство), способный выполнять определённый набор команд.

Формальный исполнитель одну и ту же команду всегда выполняет одинаково. Для каждого формального исполнителя можно указать: круг решаемых задач, среду, систему команд и режим работы.

Алгоритм — предназначенное для конкретного исполнителя описание последовательности действий, приводящих от исходных данных к требуемому результату, которое обладает свойствами дискретности, понятности, определённости, результативности и массовости.

Способность исполнителя действовать формально обеспечивает возможность автоматизации деятельности человека.

Вопросы и задания

1. Что называют алгоритмом?
2. С помощью поиска в сети Интернет выясните происхождение термина «алгоритм».
3. Подберите синонимы к слову «предписание».
4. Приведите примеры алгоритмов, изучаемых вами в школе.
5. Кто может быть исполнителем алгоритма?
6. Приведите пример формального исполнителя. Приведите пример, когда человек выступает в роли формального исполнителя.
7. От чего зависит круг решаемых задач исполнителя «компьютер»?
8. Рассмотрите в качестве исполнителя текстовый процессор, имеющийся на вашем компьютере. Охарактеризуйте круг решаемых этим исполнителем задач и его среду.
9. Что такое команда, система команд исполнителя? Какие команды должны быть у робота, выполняющего функции: а) кассира в магазине; б) дворника; в) охранника?

   Обсудите эти вопросы в группе.
10. Исследуйте один из исполнителей системы программирования КуМир. Охарактеризуйте его назначение, среду, СКИ, возможности ручного и программного управления. Используйте встроенную в систему справочную информацию.
11. Перечислите основные свойства алгоритма.
12. К чему может привести отсутствие какого-либо свойства у алгоритма? Приведите примеры.
13. В чём важность возможности формального исполнения алгоритма?
14. Последовательность чисел строится по следующему алгоритму: первые два числа последовательности принимаются равными 1; каждое следующее число последовательности при-

нимается равным сумме двух предыдущих чисел. Запишите 10 первых членов этой последовательности. Выясните, как называется эта последовательность.

15. Некоторый алгоритм получает из одной цепочки символов новую цепочку следующим образом. Сначала записывается исходная цепочка символов, после неё записывается исходная цепочка символов в обратном порядке, затем записывается буква, следующая в русском алфавите за той буквой, которая в исходной цепочке стояла на последнем месте. Если в исходной цепочке на последнем месте стоит буква «Я», то в качестве следующей буквы записывается буква «А». Получившаяся цепочка является результатом работы алгоритма. Например, если исходная цепочка символов была «ДОМ», то результатом работы алгоритма будет цепочка «ДОММОДН». Дана цепочка символов «КОМ». Сколько букв «О» будет в цепочке символов, которая получится, если применить алгоритм к данной цепочке, а затем ещё раз применить алгоритм к результату его работы?
16. Найдите в сети Интернет анимацию шагов алгоритма Эратосфена. С помощью алгоритма Эратосфена найдите все простые числа, не превышающие 50.
17. Что будет результатом исполнения Черепахой (см. пример 6) следующего алгоритма?

```
Повтори 8 [Направо 45 Вперед 45]
```

18. Запишите алгоритм для исполнителя Вычислитель (см. пример 7), содержащий не более 5 команд:
    а) получения из числа 3 числа 16;
    б) получения из числа 1 числа 25.
19. Система команд исполнителя Конструктор состоит из двух команд, которым присвоены номера:

```
1. приписать 2
2. разделить на 2
```

По первой из них к числу приписывается справа 2, по второй число делится на 2.

а) Как будет преобразовано число 8, если исполнитель выполнит алгоритм `22212`?
б) Составьте алгоритм в системе команд этого исполнителя, по которому число 1 будет преобразовано в число 16 (в алгоритме должно быть не более 5 команд).

20. В какой клетке (*A* или *B*) должен находиться исполнитель Робот из примера 8, чтобы после выполнения алгоритма 3241 (где цифры — это номера команд Робота) в неё же и вернуться?
21. К пятизначному натуральному числу применяется следующий алгоритм:
    1. Вычислить сумму первых двух цифр.
    2. Вычислить сумму последних трёх цифр.
    3. Записать полученные два числа друг за другом в порядке возрастания (неубывания).

    Выясните наименьшее и наибольшее пятизначные числа, в результате применения к которым этого алгоритма получится число 1215.
22. К четырёхзначному натуральному числу применяется следующий алгоритм:
    1. Вычислить сумму первых двух цифр.
    2. Вычислить сумму последних трёх цифр.
    3. Записать полученные два числа друг за другом в порядке возрастания (неубывания).

    Выясните, какие из приведённых ниже чисел могут получиться в результате работы этого алгоритма: 2118, 1818, 1718, 1214, 123.
23. Все алгоритмы, которые мы рассматривали до этого, можно считать алгоритмами последовательными. Подумайте сами почему. Вместе с тем в реальной жизни очень много принципиально иных алгоритмов. Параллельный алгоритм — алгоритм, который может быть реализован по частям на множестве различных исполнителей с последующим объединением полученных результатов и получением корректного результата. Приведите 2–3 примера параллельных алгоритмов из окружающего нас мира.
24. Три актёра готовятся к спектаклю. С ними работают два опытных гримёра. Каждый актёр должен быть накрашен и причёсан. Макияж у каждого актёра продолжается полчаса, а причёсывание — только 10 минут. Спланируйте работу гримёров так, чтобы актёры как можно быстрее подготовились к выходу на сцену. Сколько для этого потребуется времени?
25. Группа из четырёх туристов должна пройти по мосту в темноте. Идти по мосту одновременно могут не более двух туристов. При этом они могут пользоваться только одним

фонарём. Перебросить фонарь с одного берега на другой нельзя, поэтому кто-то из них должен вернуться с фонарём. Аня проходит через мост за 1 минуту, Борис — за 2 минуты, Тимур — за 5 минут и Дана — за 10 минут. Какое наименьшее время требуется туристам, чтобы все они перешли по мосту на другой берег?

§ 3.2 Способы записи алгоритмов

Ключевые слова:

- словесное описание
- построчная запись
- блок-схема
- Школьный алгоритмический язык
- псевдокод

Существуют различные способы записи алгоритмов. Основными среди них являются:

- словесные (на естественных языках);
- графические;
- на языках программирования.

Теоретические исследования нашего соотечественника Андрея Андреевича Маркова (младшего) (1903–1979), выполненные в середине прошлого века, показали, что в общем случае алгоритмы должны содержать предписания двух видов:
1) предписания, направленные на непосредственное преобразование информации (функциональные операторы);
2) предписания, определяющие дальнейшее направление действий (логические операторы).
Именно эти операторы положены в основу большинства способов записи алгоритмов.

3.2.1. Словесные способы записи алгоритма

Словесное описание. Самой простой является запись алгоритма в виде набора высказываний на обычном разговорном языке. Словесное описание имеет минимум ограничений и является наименее формализованным. Однако все разговорные языки облада-

ют неоднозначностью. Чтобы избежать двусмысленности, тексты алгоритма приходится делать очень подробными. Алгоритм в словесной форме может оказаться очень объёмным и трудным для восприятия.

Пример 1

Словесное описание алгоритма нахождения наибольшего общего делителя (НОД) пары натуральных чисел (алгоритм Евклида):

«Чтобы найти НОД двух чисел, составьте таблицу из двух столбцов и назовите столбцы X и Y. Запишите первое из заданных чисел в столбец X, а второе — в столбец Y. Если данные числа не равны, замените большее из них на результат вычитания из большего числа меньшего. Повторяйте такие замены до тех пор, пока числа не окажутся равными, после чего число из столбца X считайте искомым результатом».

Построчная запись. Это запись на естественном языке, но с соблюдением некоторых дополнительных правил:

- каждое предписание записывается с новой строки;
- предписания (шаги) алгоритма нумеруются;
- исполнение алгоритма происходит в порядке возрастания номеров шагов, начиная с первого (если не встречается никаких специальных указаний).

Кроме слов естественного языка, предписания могут содержать математические выражения и формулы.

Пример 2

Построчная запись алгоритма Евклида:

1. Обозначить первое из заданных чисел X, второе обозначить Y.
2. Если $X = Y$, то перейти к п. 8.
3. Если $X > Y$, то перейти к п. 4, иначе перейти к п. 6.
4. Заменить X на $X - Y$.
5. Перейти к п. 2.
6. Заменить Y на $Y - X$.
7. Перейти к п. 2.
8. Считать X искомым результатом.

Построчная запись алгоритма позволяет избежать ряда неопределённостей; её восприятие не требует дополнительных знаний. Вместе с тем использование построчной записи требует от человека большого внимания.

3.2.2. Блок-схемы

Наибольшей наглядностью обладают графические способы записи алгоритмов; самый распространённый среди них — блок-схема.

Блок-схема представляет собой графическое изображение, дающее представление о порядке работы алгоритма. Здесь предписания изображаются с помощью различных геометрических фигур, а последовательность выполнения шагов указывается с помощью линий, соединяющих эти фигуры. Линии связи *справа налево* и *снизу вверх* изображаются *со стрелками*. Направления линий связи *слева направо* и *сверху вниз* считаются стандартными, эти линии можно изображать *без стрелок*.

Рассмотрим некоторые условные обозначения, применяемые в блок-схемах.

Выполнение алгоритма всегда начинается с блока начала и оканчивается при переходе на блок конца (рис. 3.2, *а*). Из начального блока выходит одна линия связи; в конечный блок входит одна линия связи.

Внутри блока данных (рис. 3.2, *б*) перечисляются величины, значения которых должны быть введены (исходные данные) или выведены (результаты) в данном месте алгоритма. В блок данных входит одна линия связи, и из блока выходит одна линия связи.

В блоке обработки данных (рис. 3.2, *в*) содержится описание тех действий, которые должны быть выполнены при переходе на этот блок (выполнение определённой операции или группы операций, приводящее к изменению значения, формы или размещения информации). В блок обработки данных входит одна линия связи, и из блока выходит одна линия связи.

Проверка условия изображается с помощью блока принятия решения, внутри которого записывается это условие (рис. 3.2, *г*). В блок принятия решения входит одна линия, а выходят две линии, около которых записываются результаты проверки условия.

Комментарии (рис. 3.2, *д*) используются для добавления пояснительных записей, делающих блок-схему более понятной.

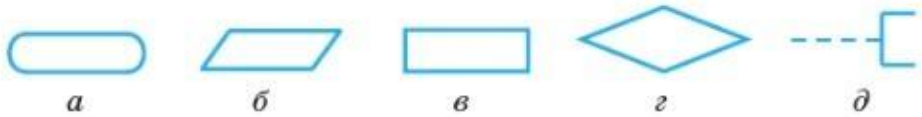


Рис. 3.2. Обозначения на блок-схемах

Пример 3

Приведём запись алгоритма Евклида с помощью блок-схемы (рис. 3.3).

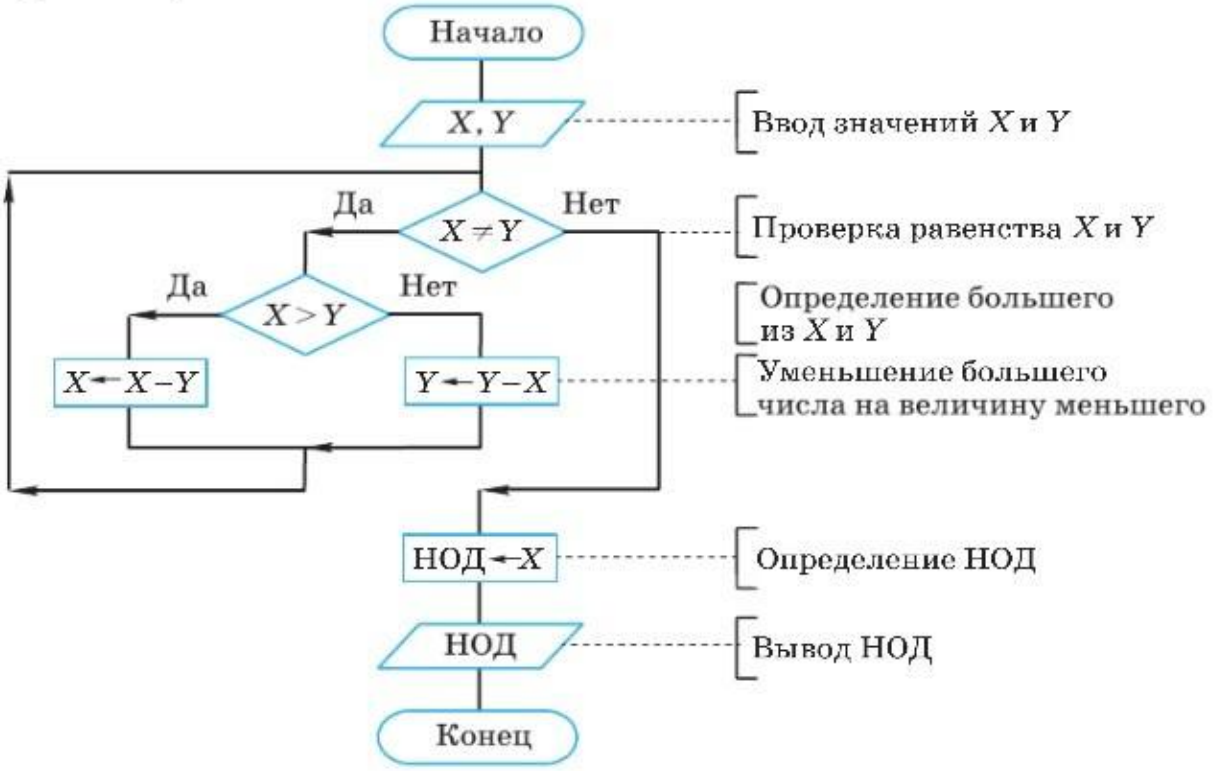


Рис. 3.3. Запись алгоритма Евклида с помощью блок-схемы

Создание детальной блок-схемы сложного алгоритма — трудоёмкая задача. Кроме того, блок-схема, не умещающаяся на одном стандартном листе, теряет своё основное преимущество — наглядность. При разработке сложных алгоритмов блок-схемы удобно использовать в качестве средства для наглядного представления решения задачи в общем виде.

3.2.3. Языки программирования

Языки программирования — формальные языки, предназначенные для записи компьютерных программ, т. е. алгоритмов, исполнителем которых является компьютер. Каждый из них характеризуется:

- *алфавитом* — набором используемых символов;
- *синтаксисом* — системой правил, по которым из символов алфавита образуются правильные конструкции языка;
- *семантикой* — системой правил, строго определяющей смысл и способ употребления конструкций языка.

Языков программирования очень много. На уроках информатики в 8–9 классах вы по своему выбору сможете познакомиться с одним из языков программирования Pascal (Паскаль) или Python (Питон); их основы изложены в двух следующих главах учебника. Кроме того, желающие смогут попробовать свои силы в программировании на Школьном (учебном) алгоритмическом языке, который также называют русским алгоритмическим языком или алгоритмическим языком КуМир.

Школьный алгоритмический язык был введён в употребление академиком А. П. Ершовым в 1985 году.

Андрей Петрович Ершов (1931–1988) — выдающийся советский учёный, инициатор введения курса информатики в школы нашей страны. Его работы оказали огромное влияние на формирование и развитие вычислительной техники во всём мире.

Для записи алгоритмов на Школьном алгоритмическом языке используется некоторое ограниченное множество слов, смысл и способ употребления которых заданы раз и навсегда. Это так называемые служебные слова: **алг** (алгоритм), **нач** (начало), **кон** (конец) и др. При записи алгоритмов в книгах служебные слова выделяются жирным шрифтом, в тетради и на доске — подчёркиванием.

В общем виде программу на Школьном алгоритмическом языке можно представить так:

```
алг <название алгоритма>
нач
  <последовательность команд>
кон
```

С основными конструкциями Школьного алгоритмического языка вы познакомитесь, работая с Роботом, Черепахой, Чертёжником и другими исполнителями, встроенными в систему программирования КуМир.

Эти же конструкции мы будем использовать при записи алгоритмов на *псевдокоде* — смеси русского языка и Школьного алгоритмического языка.

Пример 4

Алгоритм на псевдокоде, позволяющий из полного сосуда ёмкостью 12 л отлить половину, пользуясь двумя пустыми сосудами ёмкостью 8 и 5 л:

```
алг переливания
нач
  наполнить сосуд ёмкостью 8 л из сосуда ёмкостью 12 л
  наполнить сосуд ёмкостью 5 л из сосуда ёмкостью 8 л
  вылить всё из сосуда ёмкостью 5 л в сосуд ёмкостью 12 л
  вылить всё из сосуда ёмкостью 8 л в сосуд ёмкостью 5 л
  наполнить сосуд ёмкостью 8 л из сосуда ёмкостью 12 л
  долить из сосуда ёмкостью 8 л в сосуд ёмкостью 5 л
  вылить всё из сосуда ёмкостью 5 л в сосуд ёмкостью 12 л
кон
```

Такая форма записи, ориентированная на исполнителя-человека, помогает понять или изложить сущность того или иного алгоритма. Запись на псевдокоде более формализована, чем другие словесные способы записи алгоритмов. Это позволяет стандартизировать, придать единую форму записи всем алгоритмам, с которыми вы будете иметь дело.

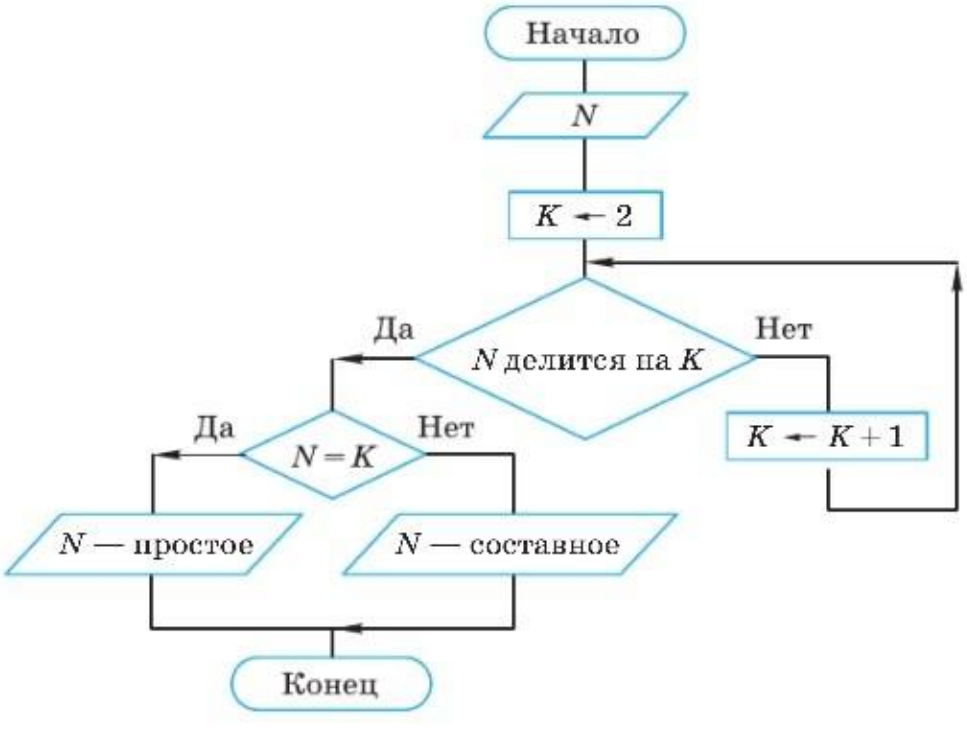
САМОЕ ГЛАВНОЕ

Существуют различные способы записи алгоритмов: словесное описание, построчная запись, блок-схемы, языки программирования и др. Каждый из этих способов обладает своими достоинствами и недостатками.

Вопросы и задания

1. Каковы основные способы записи алгоритмов?
2. Чем вызвано существование многих способов записи алгоритмов?
3. Приведите словесное описание алгоритма нахождения наименьшего общего кратного (НОК) двух целых чисел:
   а) с помощью разложения чисел на простые множители;
   б) через НОД.
4. Представьте в виде построчной записи алгоритм решения следующей задачи: «Имеются четыре арбуза различной массы. Как, пользуясь чашечными весами без гирь, путём не более пяти взвешиваний расположить арбузы по возрастанию веса?»

5. Представьте с помощью блок-схемы алгоритм решения следующей задачи: «Из трёх монет одинакового достоинства одна фальшивая (более лёгкая). Как её найти с помощью одного взвешивания на чашечных весах без гирь?» Постройте блок-схему с помощью доступных вам инструментов векторной графики (например, встроенных в текстовый процессор) или же с помощью одного из онлайн-сервисов, имеющихся в сети Интернет.
6. Запишите с помощью псевдокода алгоритм построения окружности заданного радиуса r, проходящей через заданные точки A и B.
7. В системе программирования КуМир запишите и выполните алгоритм переливаний из примера 4 для исполнителя Водолей.
8. «Прочитайте» блок-схему:



Сформулируйте словесное описание этого же алгоритма.

9. Сформулируйте основное отличие словесного описания алгоритма от описания на формальном языке.
10. Подготовьте краткую биографическую справку об А. А. Маркове (младшем).

§ 3.3 Объекты алгоритмов

Ключевые слова:

- величина
- константа
- переменная
- тип
- имя
- выражение
- присваивание
- таблица

3.3.1. Величины

Алгоритм описывает последовательность действий, производимых над некоторыми объектами, определёнными условием задачи. Например, при решении задачи о начислении зарплаты сотрудникам предприятия такими объектами могут быть табельный номер сотрудника, его фамилия, имя, отчество, оклад, отработанное время и т. д.

В информатике отдельный информационный объект (число, символ, строка, таблица и др.) называется **величиной**.

Величины делятся на постоянные (константы) и переменные. **Постоянной (константой)** называется величина, значение которой указывается в тексте алгоритма и не меняется в процессе его исполнения. **Переменной** называется величина, значение которой может изменяться в процессе исполнения алгоритма. При исполнении алгоритма в каждый момент времени переменная обычно имеет значение, называемое текущим значением.

Пример 1

Величины, выражающие количество дней в неделе, ускорение свободного падения, количество дней в первой декаде месяца, являются константами. Величины, выражающие количество дней в месяце, пульс человека, количество дней в третьей декаде месяца, являются переменными.

В алгоритмах над величинами выполняются некоторые операции. Например:

- арифметические операции +, –, * (умножение), / (деление);
- операции отношения <, >, <=, >=, =;
- логические операции И, ИЛИ, НЕ.

Объекты, над которыми выполняются операции, называются **операндами**. Не всякий объект может быть операндом для выполнения любой операции. Например, текст не может быть объектом для выполнения арифметических операций; отрицательное число не может быть операндом для извлечения квадратного корня и т. д.

Множество величин, объединённых множеством допустимых значений и определённой совокупностью допустимых операций, называют величинами определённого **типа**. При составлении алгоритмов используют величины числового (целого и вещественного), символьного, литерного и логического типов.

В математике и физике оперируют числовыми величинами — натуральными, целыми, действительными числами. При составлении алгоритмов чаще всего используют числовые величины целого и вещественного[1] типов, которые в Школьном алгоритмическом языке обозначаются **цел** и **вещ** соответственно.

В задачах, возникающих в повседневной жизни, встречаются и нечисловые величины, значениями которых являются символы, слова, тексты и др. При составлении алгоритмов обработки текстовой информации используют величины символьного (**сим**) и литерного (**лит**) типов. Значением символьной величины является один символ: русская или латинская буква, цифра, знак препинания или другой символ. Значением литерной величины является последовательность символов. Иногда эту последовательность называют строкой или цепочкой. Литерные значения в алгоритме записывают в кавычках, например: `'2022'`, `'алгоритм'`, `'литерная величина'`.

Величины логического (**лог**) типа могут принимать всего два значения: ДА (ИСТИНА, TRUE, 1); НЕТ (ЛОЖЬ, FALSE, 0).

Для ссылок на величины используют их **имена** (идентификаторы). Имя величины может состоять из одной или нескольких латинских букв, из латинских букв и цифр: `A1`, `M`, `AP`. Рекомендуется выбирать мнемонические имена, т. е. имена, отражающие суть объектов решаемой задачи, например: `SUMMA`, `PLAN`, `TEMP`, `NUM`, `DEL` и т. д.

Если величину представить как ящик, содержимым которого является некоторое значение, то имя величины — это ярлык, повешенный на ящик.

[1] Термин «вещественный» принято использовать наряду с термином «действительный».

3.3.2. Выражения

Выражение — языковая конструкция для вычисления значения с использованием одного или нескольких операндов.

Выражения состоят из операндов (констант, переменных, функций), объединённых знаками операций. Выражения записываются в виде линейных последовательностей символов (без подстрочных и надстрочных символов, обыкновенных дробей и т. д.); знаки операций пропускать нельзя. Порядок выполнения операций определяется скобками и приоритетом (старшинством) операций; операции одинакового приоритета выполняются слева направо.

Различают арифметические, логические и строковые выражения.

Арифметические выражения служат для определения числового значения. Например, `2 * x + 3` — арифметическое выражение, значение которого при `x = 1` равно `5`, а при `x = -1` — единице. Выражение `sqrt(x)` служит для обозначения операции извлечения квадратного корня из `x`: $\sqrt{x}$.

Логические выражения описывают некоторые условия, которые могут выполняться или не выполняться. Логическое выражение может принимать одно из двух значений — ИСТИНА или ЛОЖЬ. Например, логическое выражение `(x > 5)` **и** `(x < 10)` определяет принадлежность точки `x` интервалу `(5; 10)`:

При `x = 6` значение этого выражения — ИСТИНА, а при `x = 12` — ЛОЖЬ.

Строковые выражения состоят из величин (констант, переменных) символьного и литерного типов, соответствующих функций и операций сцепления (присоединения). Операция сцепления обозначается знаком «+» и позволяет соединить в одну последовательность несколько последовательностей символов. Значениями строковых выражений являются последовательности символов. Например, если `A = 'том'`, то значение строкового выражения `'а' + A` есть `'атом'`.

3.3.3. Команда присваивания

Задать конкретное значение величины можно с помощью операции присваивания, которая записывается так:

```
<имя переменной> := <выражение>
```

Знак := читается «присвоить». Например, запись `A := B + 5` читается так: «переменной `A` присвоить значение выражения `B` плюс `5`».

Знаки присваивания := и равенства = — разные знаки:

- знак = означает равенство двух величин, записанных по обе стороны от этого знака;
- знак := предписывает выполнение операции присваивания.

Например, запись `A := A + 1` выражает не равенство значений `A` и `A + 1`, а указание увеличить значение переменной `A` на единицу.

При выполнении команды присваивания сначала вычисляется значение выражения, стоящего справа от знака :=, затем результат присваивается переменной, стоящей слева от знака :=. При этом тип выражения должен быть совместим с типом соответствующей переменной.

Свойства присваивания:

1) пока переменной не присвоено значение, она остаётся неопределённой;
2) значение, присвоенное переменной, сохраняется в ней вплоть до выполнения следующего присваивания этой переменной нового значения;
3) если мы присваиваем некоторой переменной очередное значение, то предыдущее её значение теряется безвозвратно.

Пример 2

Составим алгоритм, в результате которого переменные `A` и `B` литерного типа обменяются своими значениями.

Решение вида

```
A := B
B := A
```

неверно, так как после выполнения первой команды присваивания первоначальное значение переменной `A` будет безвозвратно утеряно. Вторая команда присвоит переменной `B` текущее значение переменной `A`. В результате обе переменные получат одно и то же значение.

Для поиска правильного решения воспользуемся аналогией. Если требуется перелить жидкость из сосуда 1 в сосуд 2, а из сосуда 2 — в сосуд 1, то без дополнительного сосуда 3 здесь не обойтись. Алгоритм переливаний представлен на рис. 3.4.

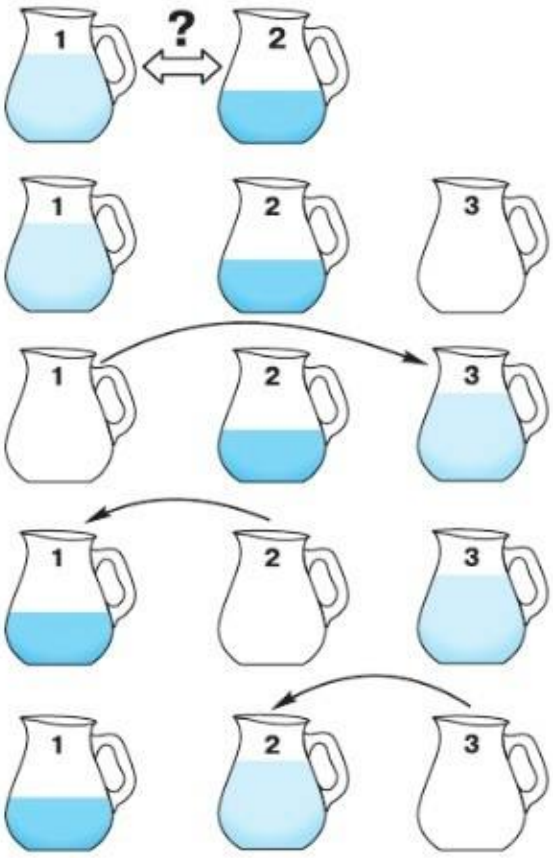


Рис. 3.4. Алгоритм переливаний жидкостей

Для решения исходной задачи введём промежуточную переменную M. Алгоритм обмена значениями переменных A и B запишем так:

```
нач
  M := A
  A := B
  B := M
кон
```

Если A и B — числовые величины, то обмен их значений можно организовать и без промежуточной переменной, например так:

```
A := A + B
B := A - B
A := A - B
```

3.3.4. Табличные величины

В практической деятельности человек часто использует всевозможные таблицы. Это, например, список учащихся в классном журнале, табель успеваемости, таблица результатов спортивных соревнований и т. д. Чаще всего встречаются линейные и прямоугольные таблицы.

Линейная таблица (одномерный массив) представляет собой набор однотипных данных, записанных в одну строку или один столбец. Элементы строки (столбца) всегда нумеруются. Например, с помощью линейной таблицы могут быть представлены дни недели (рис. 3.5, *а*) или количество уроков, пропущенных учеником в течение 5-дневной учебной недели (рис. 3.5, *б*).

1	Понедельник
2	Вторник
3	Среда
4	Четверг
5	Пятница
6	Суббота
7	Воскресенье

а

	1	2	3	4	5
Васечкин	6	6	1	0	0

б

Рис. 3.5. Примеры линейных таблиц

Прямоугольная таблица (двумерный массив) — это упорядоченный некоторым образом набор строк (столбцов), содержащих одинаковое количество элементов. Строки прямоугольных таблиц имеют свою нумерацию, столбцы — свою. Например, с помощью прямоугольной таблицы можно представить количество уроков, пропущенных всеми учениками 8 класса в течение 5-дневной учебной недели (рис. 3.6).

	1	2	3	4	5
1. Васечкин	6	6	1	0	0
2. Ионов	0	0	0	0	6
3. Радугина	0	0	1	0	0
...	...	...	...	...	...
19. Чабанюк	0	0	0	0	0

Рис. 3.6. Пример прямоугольной таблицы

Всей совокупности элементов табличной величины даётся одно имя. Элементы различают по их номерам, называемым **индексами**. Индексы записываются в квадратных скобках сразу за именем таблицы.

Если первую из рассмотренных нами таблиц (см. рис. 3.5, *а*) назвать `WEEK`, то `WEEK[1] = 'Понедельник'`, `WEEK[6] = 'Суббота'`.

Назовём третью из рассмотренных таблиц `LES`. В этой таблице положение элемента задаётся двумя индексами: номером строки и номером столбца. Например: `LES[1, 1] = 6`, `LES[2, 5] = 6`, `LES[3, 4] = 0`.

Линейные и прямоугольные таблицы наглядно можно представить как аналоги привычных нам объектов — рядов шкафчиков в раздевалках, почтовых ящиков, ячеек для хранения сумок в супермаркетах (рис. 3.7).

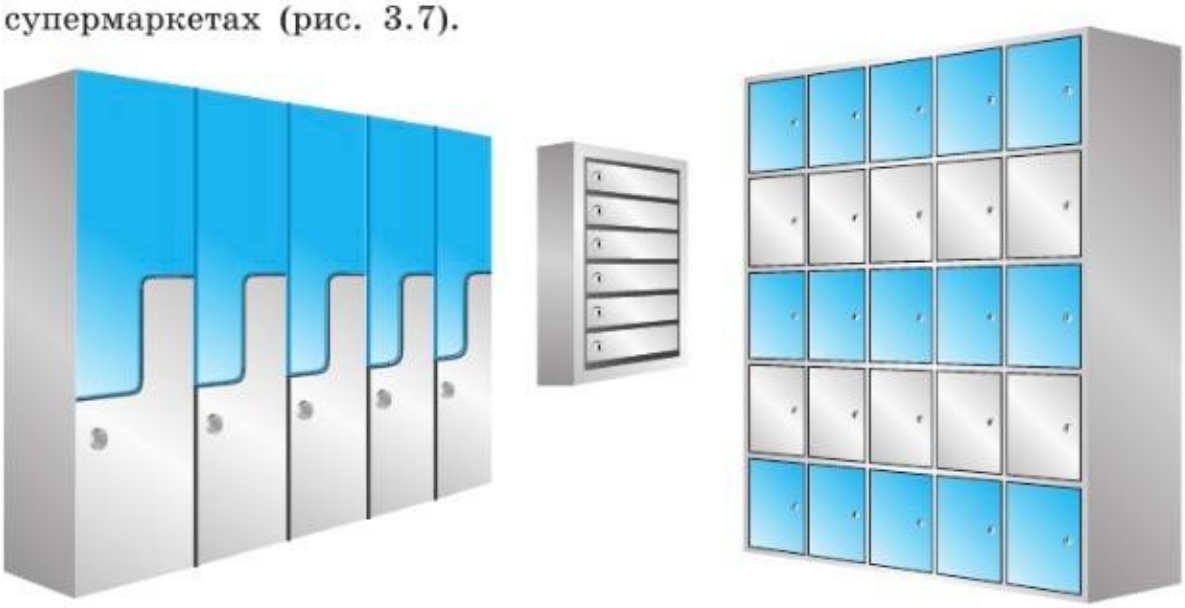


Рис. 3.7. Наглядное представление линейной и прямоугольной таблиц

САМОЕ ГЛАВНОЕ

В информатике отдельный информационный объект (число, символ, строка, таблица и др.) называется величиной.

Величины делятся на постоянные (их значения указываются в тексте алгоритма и не меняются в процессе его исполнения) и переменные (их значения могут изменяться в процессе исполнения алгоритма). При составлении алгоритмов используют величины целого, вещественного, логического, символьного и литерного типов.

Для ссылок на величины используют их имена (идентификаторы). Имя величины может состоять из одной или нескольких латинских букв, из латинских букв и цифр.

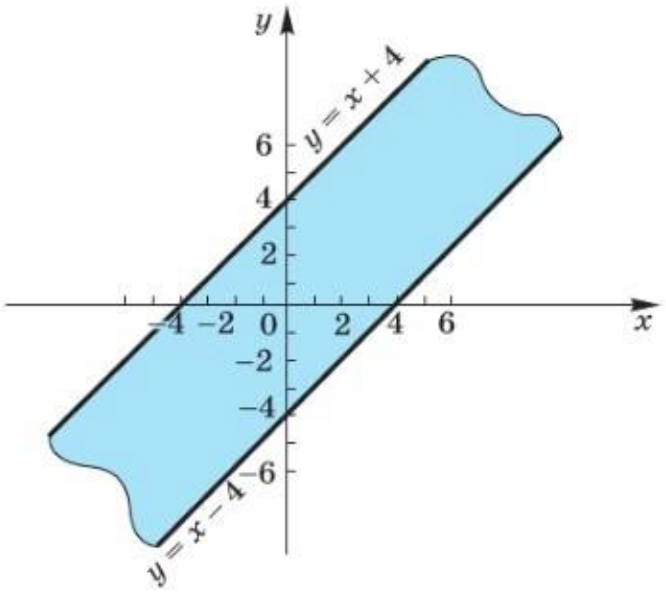
Таблица (массив) — набор некоторого числа однотипных элементов, которым присвоено одно имя. Положение элемента в таблице однозначно определяется его индексом (индексами).

Вопросы и задания

1. Что такое величина? Чем различаются постоянные и переменные величины?
2. Величины каких типов используются при записи алгоритмов?
3. Укажите тип величины, если её значение равно: `2022`; `14.48`; `'ДА'`; `FALSE`; `-125`; `'142'`; $1{,}4 \cdot 10^5$; `.123E-2`; `'пять'`.
4. Определите типы следующих величин:
   а) вес человека;
   б) марка автомобиля;
   в) год вашего рождения;
   г) площадь фигуры;
   д) название месяца года;
   е) количество мест в самолёте.
5. Работая в группе, приведите примеры допустимых и недопустимых значений для каждой из величин:
   а) температура человека;
   б) скорость автомашины;
   в) площадь страны;
   г) название дня недели.
6. Для чего предназначена команда присваивания? Каковы её основные свойства?
7. Какая команда присваивания составлена правильно?
   а) `A := B`
   б) `A = B`
   в) `A = B + 1`
   г) `A + 1 := A`
8. Придумайте свой алгоритм обмена значениями числовых переменных `A` и `B`.
9. Имеются числовые переменные `A`, `B` и `C`. Сколько промежуточных переменных потребуется для того, чтобы переменной `A` было присвоено значение переменной `B`, переменной `B` — значение переменной `C`, а переменной `C` — значение переменной `A`? Запишите соответствующий алгоритм.

10. После выполнения команды присваивания `x := x + y` значение переменной `x` равно `3`, а значение переменной `y` равно `5`. Чему были равны значения переменных `x` и `y` до выполнения указанной команды присваивания?
11. Что называют выражением? Каковы основные правила записи выражений?
12. Переведите выражение из линейной записи на Школьном алгоритмическом языке в запись на языке математики:
    а) `a * b / c`
    б) `a / b * c`
    в) `a + b / c`
    г) `(a + b) / c`
    д) `a + b / c + d`
    е) `(a + b) / (c + d)`
13. Запишите на Школьном алгоритмическом языке:
    а) $ax^2 + bx + c$;
    б) $v_0 + \frac{at^2}{2}$;
    в) $\frac{1}{2}(a+b)h$;
    г) $\frac{1+x_1x_2}{b^2c}$;
    д) $\sqrt{a^2+b^2}$.
14. Запишите логическое выражение, истинное при выполнении указанного условия и ложное в противном случае:
    а) x принадлежит отрезку [0, 1];
    б) x лежит вне отрезка [0, 1];
    в) каждое из чисел x, y положительно;
    г) хотя бы одно из чисел x, y положительно;
    д) ни одно из чисел x, y не является положительным;
    е) только одно из чисел x, y положительно.
15. Изобразите в декартовой прямоугольной системе координат область, в которой и только в которой истинно следующее логическое выражение:
    а) `(x >= -1)` **и** `(x <= 1)` **и** `(y >= -1)` **и** `(y <= 1)`
    б) `(y >= x)` **и** `(y >= -x)` **и** `(y <= 1)`

16. Запишите логическое выражение, принимающее значение TRUE, когда точка с координатами (x, y) принадлежит закрашенной области.



17. Запишите команду присваивания, в результате выполнения которой логическая переменная t получает значение TRUE, если выполняется указанное условие, и значение FALSE в противном случае:

а) x — положительное число;

б) хотя бы одно из чисел x, y, z равно нулю;

в) числа x, y, z равны между собой.

18. Какие из приведённых ниже величин целесообразно представлять с помощью таблиц? Обсудите этот вопрос в группе.

Величины: список учеников класса, рост учеников класса, средний рост учеников класса, оценка ученика по физике, средний балл ученика по физике, оценки учеников за контрольную работу по информатике, длины сторон треугольника, длины сторон нескольких треугольников, названия дней недели, имя человека, площадь фигуры, периметры нескольких прямоугольников, самая низкая температура воздуха в январе, количество девочек в классе, самая дождливая декада июня.

§ 3.4 Алгоритмическая конструкция «следование». Линейные алгоритмы

Ключевые слова:

- следование
- линейный алгоритм
- синтаксическая ошибка
- логическая ошибка

Человеку в жизни приходится решать множество различных задач. Решение каждой из них описывается своим алгоритмом, и разнообразие этих алгоритмов очень велико. Вместе с тем для записи любого алгоритма достаточно трёх основных алгоритмических конструкций (структур): следования, ветвления, повторения. Это положение выдвинул и доказал Э. Дейкстра в 70-х гг. прошлого века.

Эдсгер Вибе Дейкстра (1930–2002) — выдающийся нидерландский учёный, идеи которого оказали огромное влияние на развитие компьютерной индустрии.

3.4.1. Следование

Следование — алгоритмическая конструкция, отображающая естественный, последовательный порядок действий. Алгоритмы, в которых используется только конструкция «следование», называются **линейными алгоритмами**.

Простейший пример конструкции (структуры) «следование» — это программа передач любого телевизионного канала: телепередачи транслируют одну за другой, и следующая начинается только после того, как закончилась предыдущая.

Графическое представление алгоритмической конструкции «следование» приведено на рис. 3.8.

Рис. 3.8. Алгоритмическая конструкция «следование»

Пример 1

Линейный алгоритм приготовления отвара шиповника:

Обратите внимание, что многие из предписаний этого алгоритма могут потребовать детализации — представления в виде некоторой совокупности более мелких предписаний.

Пример 2

Дан фрагмент линейного алгоритма:

```
x := 2
y := x * x
y := y * y
x := y * x
s := x + y
```

Выясним, какое значение получит переменная s после выполнения этого фрагмента алгоритма.

Для этого составим таблицу значений переменных, задействованных в алгоритме:

Шаг алгоритма	Переменные		
	x	y	s
1	2	–	–
2	2	4	–
3	2	16	–
4	32	16	–
5	32	16	48

Составленная нами таблица значений переменных моделирует работу исполнителя этого алгоритма.

Пример 3

Некоторый исполнитель может выполнять над целыми числами, кроме операций сложения, вычитания, умножения и деления, ещё две операции: с помощью операции `div` вычисляется неполное частное, с помощью операции `mod` — остаток.

Например:

```
5 div 2 = 2; 5 mod 2 = 1; 2 div 5 = 0; 2 mod 5 = 2.
```

Покажем, как с помощью этих операций можно реализовать алгоритм работы кассира, выдающего покупателю сдачу (s) наименьшим количеством банкнот по 1000 (k1000), 500 (k500), 100 (k100) и 50 (k50).

```
k1000 := s div 1000
s := s mod 1000
k500 := s div 500
s := s mod 500
k100 := s div 100
s := s mod 100
k50 := s div 50
```

Выполните алгоритм для s = 745 и s = 1864. Составьте соответствующие таблицы значений переменных.

Пример 4

У исполнителя Альфа две команды, которым присвоены номера:

1. прибавь 2
2. умножь на a (a — неизвестное натуральное число)

Выполняя первую из них, Альфа увеличивает число на экране на 2, а выполняя вторую, умножает это число на a. Программа для исполнителя Альфа — это последовательность номеров команд. Известно, что программа 11211 переводит число 4 в число 100.

Необходимо определить значение a.

Запишем последовательность действий по преобразованию исходного числа 4 в соответствии с заданным алгоритмом:

$(4 + 2 + 2) \cdot a + 2 + 2 = 100$.

Выполним преобразовние левой части выражения:

$8 \cdot a + 4 = 100$.

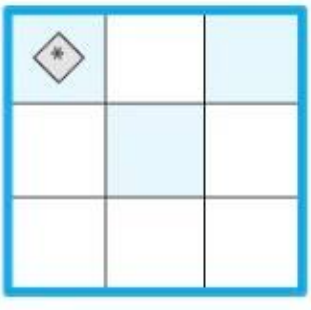
Решим линейное уравнение:

$8 \cdot a = 96$, $a = 12$.

3.4.2. Ограниченность линейных алгоритмов

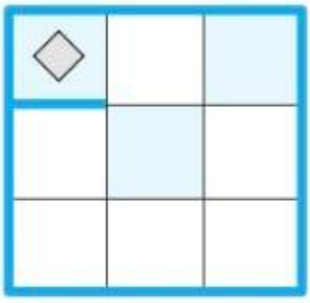
Пример 5

У исполнителя Робот есть четыре команды перемещения (вверх, вниз, влево и вправо), при выполнении каждой из них Робот перемещается на одну клетку в соответствующем направлении. По команде закрасить Робот закрашивает клетку, в которой он находится. Запишем линейный алгоритм, исполняя который Робот нарисует на клетчатом поле следующий узор и вернётся в исходное положение, обозначенное звёздочкой:




```
алг узор
нач
  закрасить
  вправо
  вправо
  закрасить
  вниз
  влево
  закрасить
  влево
  вверх
кон
```

А теперь подумайте, что произойдёт, если Робот приступит к выполнению этой же программы в обстановке, в которой между двумя клетками есть стена:



Очевидно, что Робот закрасит все клетки, но не сможет выполнить последнюю команду: пытаясь подняться вверх, Робот столкнётся со стеной, что приведёт к его разрушению.

Рассмотренный пример демонстрирует особенность линейных алгоритмов, состоящую в том, что их исполнитель не может уклониться от выполнения ни одной из последовательно идущих команд, даже если очевидно, что очередное действие бессмысленно. Однако исполнитель всё равно попытается его выполнить, потому что «не умеет» различать такие ситуации.

Практически любая задача, возникающая в жизни, предполагает ситуации, когда выбор действия зависит от тех или иных условий и не может быть жёстко предопределён заранее. По этой причине исключительно линейные алгоритмы находят весьма ограниченное применение.

3.4.3. Ошибки в алгоритмах

Алгоритм, рассмотренный в примере 5, вы можете выполнить в среде КуМир. Для этого его необходимо дополнить первой строкой: `использовать Робот`.

Если вы всё введёте безошибочно, то получите ожидаемый результат. Если же, например, вместо команды `влево` вы введёте `влева` или `в лево`, то на экране увидите сообщение об ошибке (рис 3.9). Такие ошибки называются **синтаксическими** и могут быть легко устранены: достаточно внимательно отслеживать все сообщения, появляющиеся на экране, и вносить исправления в запись той или иной команды.

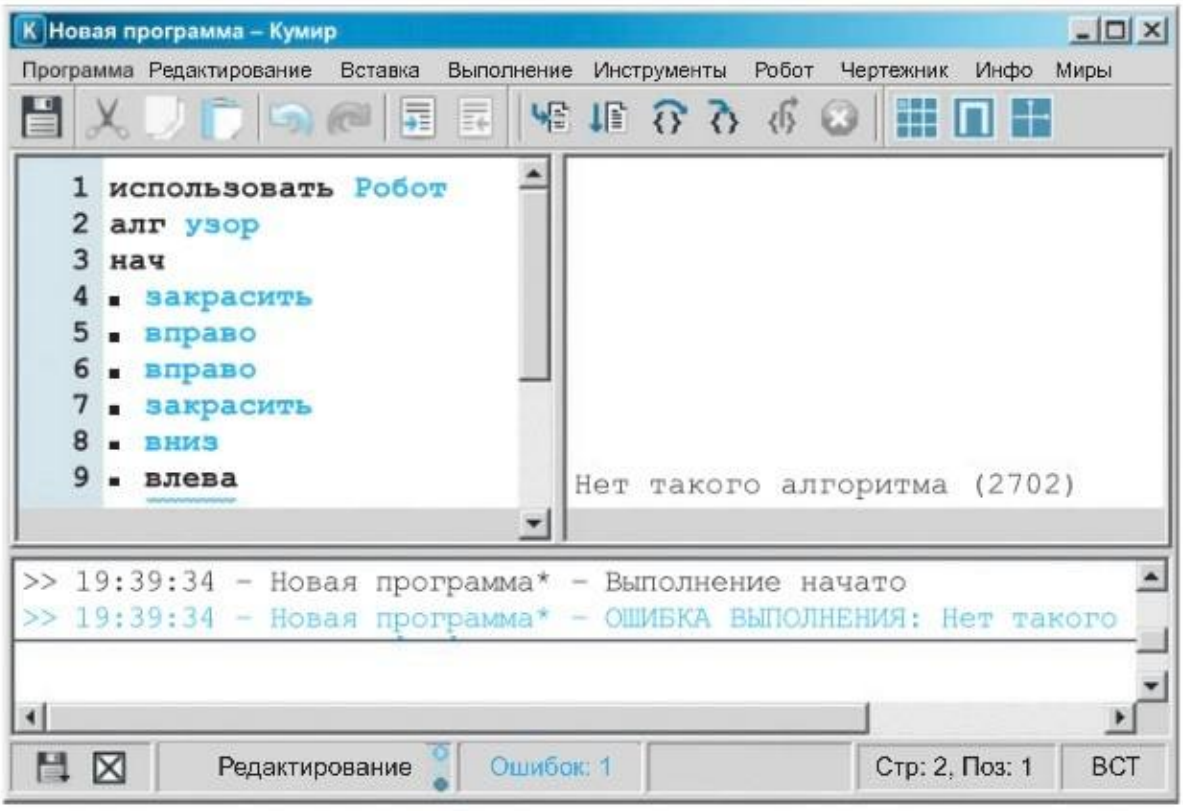
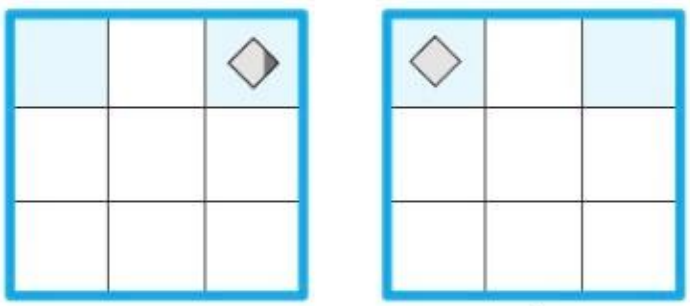


Рис. 3.9. Синтаксическая ошибка в алгоритме

Другую разновидность ошибок называют **логическими**. В случае логической ошибки программа выполняется, но не приводит к желаемому результату:

1) при выполнении алгоритма может возникнуть отказ, если исполнитель не может выполнить некоторую команду (рис. 3.10, *а*);
2) алгоритм может быть завершён, но результат его работы не будет соответствовать поставленной задаче (рис. 3.10, *б*).



а *б*

Рис. 3.10. Логические ошибки в алгоритме

Все ошибки исправляются в процессе отладки программы.

САМОЕ ГЛАВНОЕ

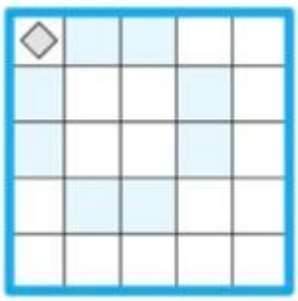
Для записи любого алгоритма достаточно трёх основных алгоритмических конструкций (структур): следования, ветвления, повторения.

Следование — алгоритмическая конструкция, отображающая естественный, последовательный порядок действий. Алгоритмы, в которых используется только структура «следование», называются линейными.

Практически любая задача, возникающая в жизни, предполагает ситуации, когда выбор действия зависит от тех или иных условий и не может быть жёстко предопределён заранее. По этой причине исключительно линейные алгоритмы находят ограниченное применение.

Вопросы и задания

1. Какие алгоритмы называются линейными? Зависит ли в линейном алгоритме последовательность выполняемых действий от исходных данных?
2. Приведите пример линейного алгоритма:
   а) из повседневной жизни;
   б) из литературного произведения;
   в) из любой предметной области, изучаемой в школе.
3. Запишите линейный алгоритм, исполняя который Робот нарисует на клетчатом поле следующий узор и вернётся в исходное положение.



4. По алгоритму восстановите формулу.

```
a1 := 1 / x
a2 := a1 / x
a3 := a2 / x
a4 := a3 / x
y := a1 + a2
y := y + a3
y := y + a4
```

5. Какое значение получит переменная y после выполнения следующего алгоритма?

```
x := 1
y := 2 * x
y := y + 3
y := y * x
y := y + 4
y := y * x
y := y + 5
```

Восстановите формулу вычисления y для произвольного значения x.

6. Для заданного количества суток (tfh) требуется определить количество часов (h), минут (m) и секунд (c). Составьте соответствующий линейный алгоритм.
7. Известно, что 1 миля = 1,5 версты, 1 верста = 500 саженей, 1 сажень = 3 аршина, 1 аршин = 28 дюймов, 1 дюйм = 25,4 мм. Пользуясь этой информацией, составьте линейный алгоритм перевода расстояния X миль в километры.
8. Исходное данное — целое трёхзначное число x. Выполните для x = 125 следующий алгоритм:

```
a := x div 100
b := x mod 100 div 10
c := x mod 10
s := a + b + c
```

Какой смысл имеет результат s этого алгоритма?

9. Определите значения целочисленных переменных x и y после выполнения алгоритма.

```
x := 336
y := 8
x := x div y
y := x mod y
```

10. У исполнителя Альфа две команды, которым присвоены номера:

1. прибавь 2
2. умножь на a (a — неизвестное натуральное число)

Выполняя первую из них, Альфа увеличивает число на экране на 2, а выполняя вторую, умножает это число на a. Программа для исполнителя Альфа — это последовательность номеров команд. Известно, что программа 12211 переводит число 4 в число 100. Определите значение a.

§ 3.5
Алгоритмическая конструкция «ветвление». Разветвляющиеся алгоритмы

Ключевые слова:

- ветвление
- полное ветвление
- неполное ветвление
- разветвляющиеся алгоритмы

3.5.1. Полное и неполное ветвление

Ветвление — алгоритмическая конструкция, в которой в зависимости от результата проверки условия («Да» или «Нет») предусмотрен выбор одной из двух последовательностей действий (ветвей). Алгоритмы, в основе которых лежит структура «ветвление», называют **разветвляющимися**.

Блок-схема ветвления представлена на рис. 3.11. Каждая ветвь может быть любой степени сложности (рис. 3.11, *а*); одна из ветвей может вообще не содержать предписаний (рис. 3.11, *б*).

Рис. 3.11. Структура «ветвление»: *а* — полная форма ветвления; *б* — неполная форма ветвления

На Школьном алгоритмическом языке команда ветвления записывается так.

Полная форма ветвления:

```
если <условие>
  то <действия 1>
  иначе <действия 2>
все
```

Неполная форма ветвления:

```
если <условие>
  то <действия 1>
все
```

Пример 1

```
алг правописание приставок НЕ, НИ
нач
если приставка под ударением
  то писать НЕ
  иначе писать НИ
все
```

Пример 2

```
алг сборы на прогулку
нач
если на улице дождь
  то взять зонтик
все
```

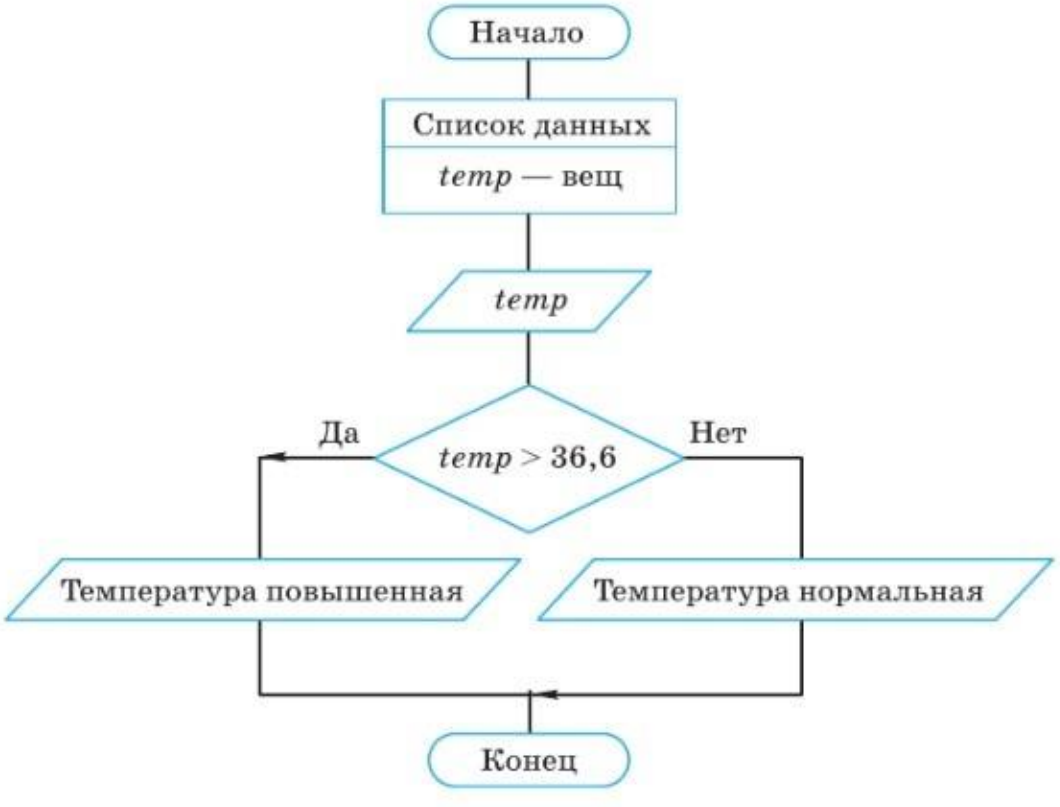
Для записи условий, в зависимости от результатов проверки которых выбирается та или иная последовательность действий, используются **операции сравнения:**

`A < B` — A меньше B;
`A <= B` — A меньше или равно B;
`A = B` — A равно B;
`A > B` — A больше B;
`A >= B` — A больше или равно B;
`A <> B` — A не равно B.

Здесь буквы `A` и `B` можно заменять на любые переменные, числа и арифметические выражения. Приведённые операции сравнения допускаются и для символьных переменных.

Пример 3

Алгоритм, встроенный в «умный» термометр, может иметь вид:

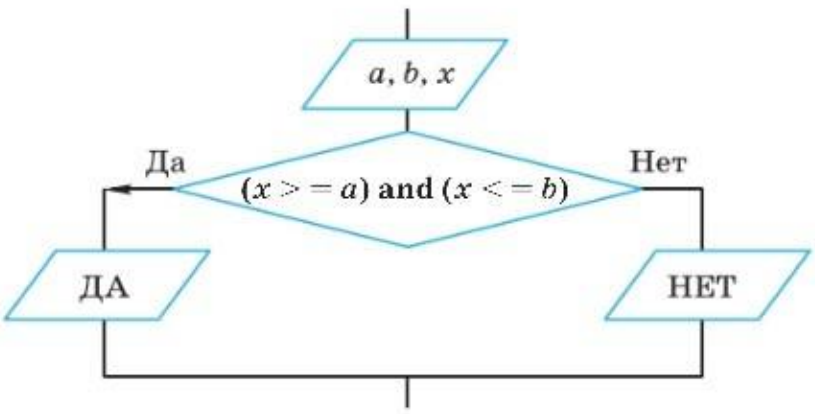


Обратите внимание на второй блок этой блок-схемы. В нём представлены имя и тип величины, обрабатываемой алгоритмом.

Условия, состоящие из одной операции сравнения, называются **простыми**. В качестве условий при организации ветвлений можно использовать и составные условия. **Составные условия** получаются из простых с помощью логических связок **and (и)**, **or (или)**, **not (не)**: **and** означает одновременное выполнение всех условий, **or** — выполнение хотя бы одного условия, **not** означает отрицание условия, записанного после слова **not**.

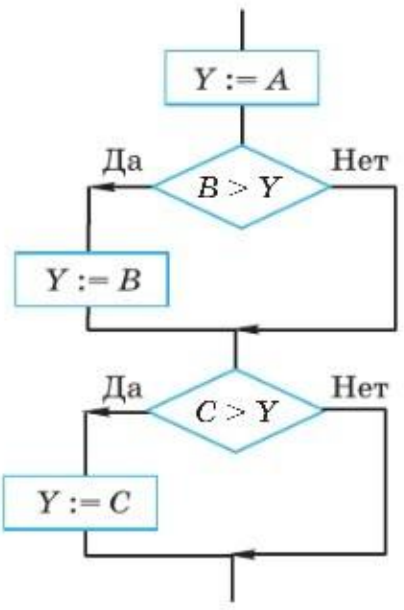
Пример 4

Алгоритм определения принадлежности точки x отрезку [a, b]. Если точка x принадлежит данному отрезку, то выводится сообщение 'ДА', в противном случае — 'НЕТ'.



3.5.2. Комбинация нескольких ветвлений

Существует достаточно много ситуаций, в которых приходится выбирать не из двух, а из трёх и более вариантов. Есть разные способы построения соответствующих алгоритмов. Один из них — составить комбинацию из нескольких ветвлений.



Пример 5

Алгоритм, в котором переменной Y присваивается значение наибольшей из трёх величин A, B и C.

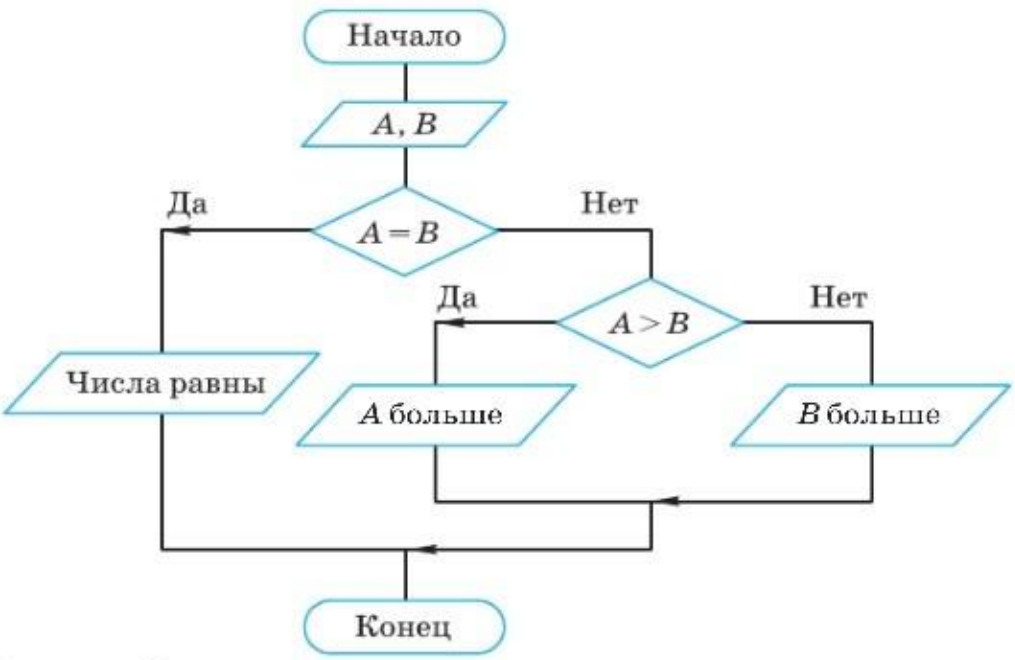
Пусть A = 10, B = 30 и C = 20. Тогда процесс выполнения алгоритма можно представить следующей таблицей:

Шаг алгоритма	Константы			Переменная	Условие
	A	B	C	Y	
	10	30	20		
1				10	
2					30 > 10 (Да)
3				30	
4					20 > 30 (Нет)

Пример 6

Разработаем алгоритм для исполнителя, работающего с парой чисел и сравнивающего первое число со вторым. Результатом выполнения алгоритма должно быть одно из трёх сообщений:

1) числа равны;
2) первое число больше;
3) второе число больше.



Пример 7

Исполнитель Робот может выполнять ту или иную последовательность действий в зависимости от выполнения следующих простых условий:

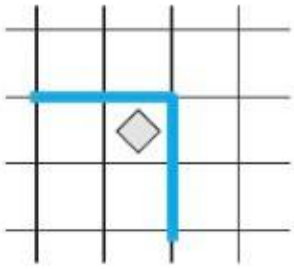
справа свободно	справа стена
слева свободно	слева стена
сверху свободно	сверху стена
снизу свободно	снизу стена
клетка чистая	клетка закрашена

Также Робот может действовать в зависимости от выполнения составных условий.

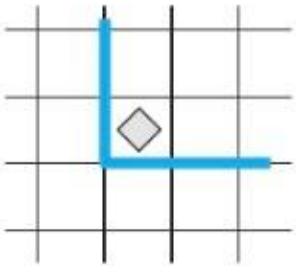
Подумайте, в какую клетку переместится Робот из клетки, обозначенной ромбиком, при выполнении следующего фрагмента алгоритма.

```
если справа свободно или снизу свободно
  то закрасить
все
если справа стена
  то влево
все
если слева стена
  то вправо
все
```

а)



б)



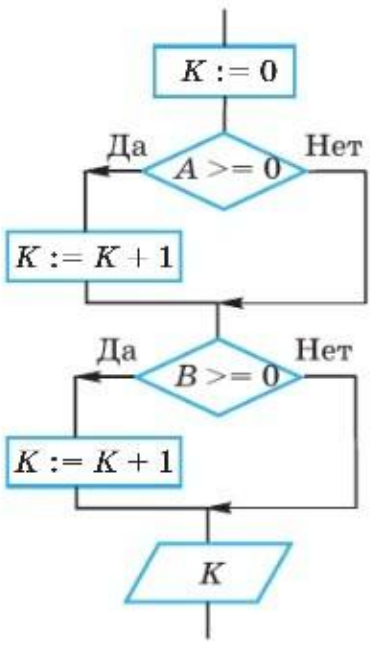
САМОЕ ГЛАВНОЕ

Ветвление — алгоритмическая конструкция, в которой в зависимости от результата проверки условия («Да» или «Нет») предусмотрен выбор одной из двух последовательностей действий (ветвей). Алгоритмы, в основе которых лежит структура «ветвление», называют разветвляющимися.

Вопросы и задания

1. Какие алгоритмы называют разветвляющимися? Согласны ли вы с утверждением, что в разветвляющемся алгоритме при любых исходных данных выполняются все действия, предусмотренные алгоритмом?

2. Приведите пример разветвляющегося алгоритма:
   1) из повседневной жизни;
   2) из литературного произведения;
   3) из любой предметной области, изучаемой в школе.
3. Дополните алгоритм из примера 5 так, чтобы с его помощью можно было найти наибольшую из четырёх величин A, B, C и D.
4. Составьте алгоритм, с помощью которого можно определить, существует ли треугольник с длинами сторон a, b, c.
5. Составьте алгоритм, с помощью которого можно определить, является ли треугольник с заданными длинами сторон a, b, c равносторонним.
6. Составьте алгоритм возведения чётного числа в квадрат, а нечётного — в куб.
7. Какая задача решается с помощью следующего алгоритма?

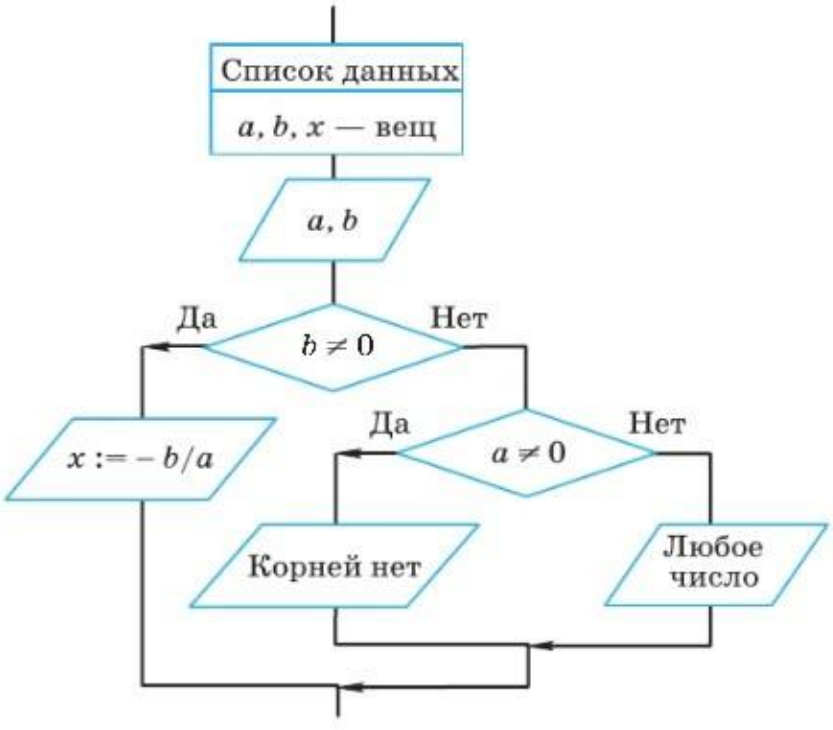


8. Составьте блок-схему алгоритма определения количества чётных чисел среди заданных целых чисел A, B и C.
9. Составьте блок-схему алгоритма определения принадлежности точки x отрезку [a, b] (пример 4) с использованием комбинации из двух ветвлений.
10. Составьте блок-схему алгоритма правописания приставок, оканчивающихся на букву «з».

11. Известно, что 31 января 2022 года было понедельником. Какие значения должны быть присвоены литерной переменной y в алгоритме, определяющем день недели для произвольного числа (chislo) января 2022 года?

```
chislo := chislo mod 7
если chislo = 3 то y: = '_'
если chislo = 4 то y: = '_'
если chislo = 5 то y: = '_'
если chislo = 6 то y: = '_'
если chislo = 0 то y: = '_'
если chislo = 1 то y: = '_'
если chislo = 2 то y: = '_'
```

12. Даны две точки на плоскости. Составьте алгоритм для определения, какая из них находится ближе к началу координат.
13. Составьте алгоритм для определения, есть ли среди цифр заданного целого трёхзначного числа одинаковые.
14. Ученик 8 класса, познакомившийся с разветвляющимися алгоритмами, решил применить свои зания на уроках математики и разработал алгоритм решения линейного уравнения $ax + b = 0$. Он очень торопился и поэтому допустил в блок-схеме алгоритма ошибки.

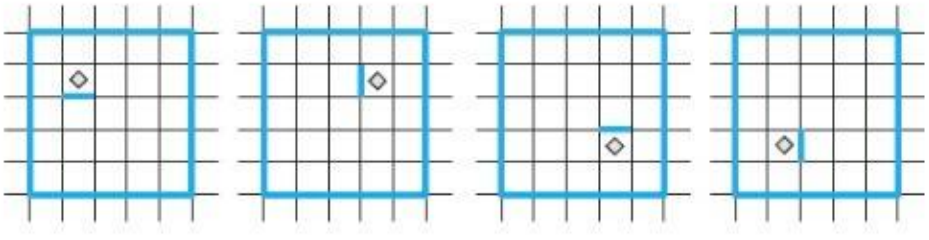


Исправьте ошибки. Проверьте правильность работы алгоритма, решив с его помощью следующие уравнения:

1) $5 \cdot x - 10 = 0$;
2) $12 \cdot x = 0$;
3) $0 \cdot x + 10 = 0$;
4) $0 \cdot x + 0 = 0$.

15. Робот находится внутри поля размером 5×5 клеток. Рядом с Роботом есть стена длиной в одну клетку. Составьте в среде (системе программирования) КуМир один из следующих алгоритмов:
   а) Робот закрашивает две клетки: клетку, в которой он находится в стартовой позиции, и клетку с другой стороны от стены;
   б) Робот закрашивает клетку с другой стороны стены и возвращается назад;
   в) Робот закрашивает клетку, в которой он стоит, и «прячется» за стену.

   Протестируйте программу, устанавливая начальное положение Робота в клетках, отмеченных ромбом.



§ 3.6 Алгоритмическая конструкция «повторение». Циклические алгоритмы

Ключевые слова:

- повторение
- циклический алгоритм
- тело цикла
- цикл с заданным условием продолжения работы
- цикл с заданным условием окончания работы
- цикл с заданным числом повторений
- цикл с переменной

Повторение — алгоритмическая конструкция, представляющая собой последовательность действий, выполняемых многократно. Алгоритмы, содержащие конструкцию повторения, называют **циклическими** или **циклами**. Последовательность действий, многократно повторяющаяся в процессе выполнения цикла, называется **телом цикла**.

В зависимости от способа организации повторения можно выделить следующие типы циклов:

1) цикл с заданным условием продолжения работы;
2) цикл с заданным условием окончания работы;
3) цикл с заданным числом повторений;
4) цикл с переменной.

3.6.1. Цикл с заданным условием продолжения работы

Цикл с заданным условием продолжения работы иначе называют: **цикл-ПОКА, цикл с предусловием.**

Логика работы этой конструкции описывается схемой, показанной на рис. 3.12.

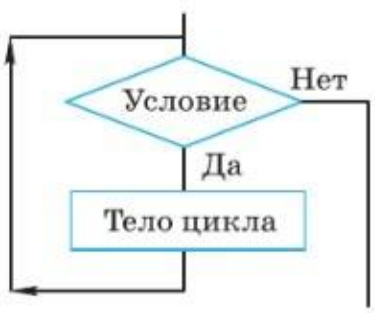


Рис. 3.12. Цикл с предусловием

На Школьном алгоритмическом языке эта конструкция записывается так:

```
нц пока <условие>
  <тело цикла (последовательность действий)>
кц
```

Выполняется цикл-ПОКА следующим образом:

1) проверяется условие (вычисляется значение логического выражения);
2) если условие выполняется («Да»), то выполняется тело цикла и снова осуществляется переход к проверке условия; если же условие не выполняется («Нет»), то выполнение цикла заканчивается.

Возможны случаи, когда тело цикла не будет выполнено ни разу.

Пример 1

Алгоритм, по которому из всех имеющихся кирпичей отбираются целые кирпичи и складываются в машину.

```
алг отбор
нач
  нц пока есть кипичи
    взять один кирпич
    если кирпич целый
      то положить кирпич в машину
      иначе отложить кирпич в сторону
    все
  кц
кон
```

Пример 2

Правее Робота (клетка с ромбиком) расположен коридор неизвестной длины. Необходимо, чтобы Робот закрасил все клетки этого коридора.

Пока будет выполняться условие `справа свободно`, Роботу следует выполнять команды:

```
вправо
закрась
```

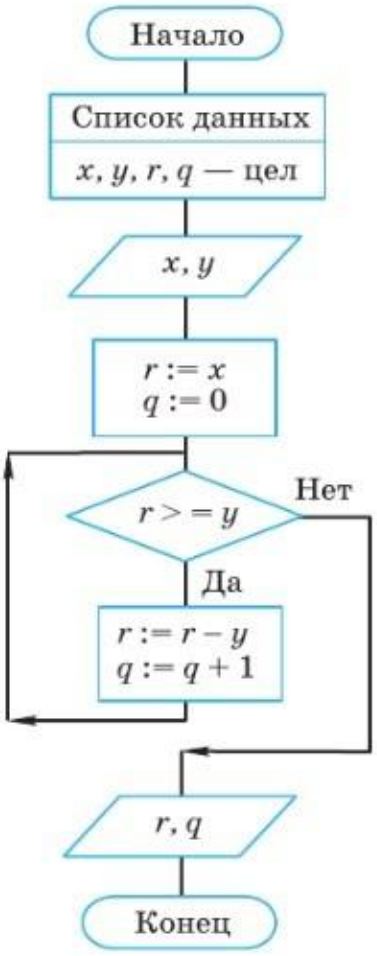
Соответствующий алгоритм для Робота будет иметь вид:

```
нц пока справа свободно
  вправо
  закрась
кц
```

Пример 3

Требуется, не пользуясь операцией деления, получить неполное частное q и остаток r от деления натурального числа x на натуральное число y.

Представим операцию деления как последовательные вычитания делителя из делимого. Причём вычитать будем до тех пор, пока результат вычитания не станет меньше вычитаемого (делителя). В этом случае количество вычитаний будет равно неполному частному от деления q, а последняя разность — остатку от деления r.



Исполним этот алгоритм для x = 23 и y = 5.

Шаг алгоритма	Операция	Переменная				Условие r >= y
		x	y	r	q	
1	ввод x	23	–	–	–	
2	ввод y		5	–	–	
3	r := x			23	–	
4	q := 0				0	
5	r >= y					23 >= 5 (Да)
6	r := r - y			18		
7	q := q + 1				1	
8	r >= y					18 >= 5 (Да)
9	r := r - y			13		
10	q := q + 1				2	
11	r >= y					13 >= 5 (Да)
12	r := r - y			8		
13	q := q + 1				3	
14	r >= y					8 >= 5 (Да)
15	r := r - y			3		
16	q := q + 1				4	
17	r >= y					3 >= 5 (Нет)
18	вывод r			3		
19	вывод q				4	

3.6.2. Цикл с заданным условием окончания работы

Цикл с заданным условием окончания работы иначе называют: **цикл-ДО, цикл с постусловием.**

Логика работы этой конструкции описывается схемой, показанной на рис. 3.13.

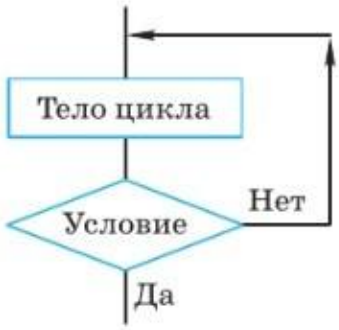


Рис. 3.13. Цикл с постусловием

На Школьном алгоритмическом языке эта конструкция записывается так:

```
нц
  <тело цикла (последоватльность действий)>
кц при <условие>
```

Выполняется цикл-ДО следующим образом:

1) выполняется тело цикла;
2) проверяется условие (вычисляется значение логического выражения); если условие не выполняется («Нет»), то снова выполняется тело цикла и осуществляется переход к проверке условия; если же условие выполняется, то выполнение цикла заканчивается.

В любом случае тело цикла будет выполнено хотя бы один раз.

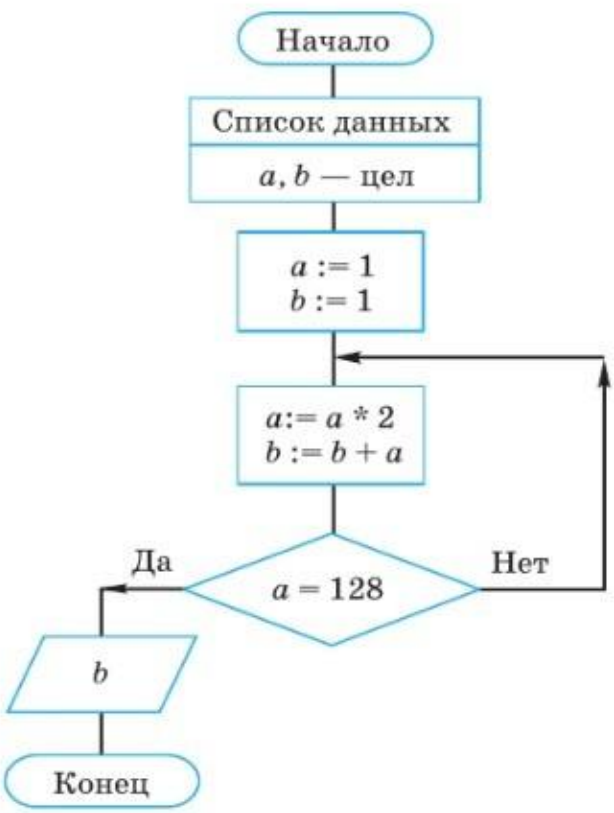
Пример 4

Алгоритм по выучиванию наизусть четверостишия:

```
алг четверостишие
нач
  нц
    прочитать четверостишие по книге 1 раз
    рассказать четверостишие
  кц при рассказано без ошибок
кон
```

Пример 5

Требуется вычислить значение переменной b согласно следующему алгоритму:



Составим таблицу значений переменных, задействованных в алгоритме:

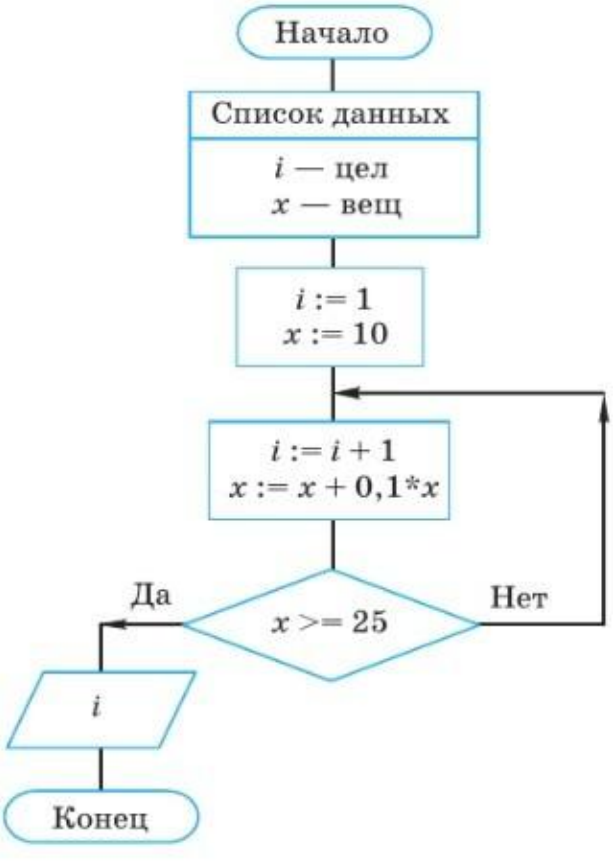
Шаг алгоритма	Операция	Переменные		Условие
		a	b	a = 128
1	a := 1	1	–	
2	b := 1		1	
3	a := a * 2	2		
4	b := b + a		3	
5	a = 128			2 = 128 (Нет)
6	a := a * 2	4		
7	b := b + a		7	
8	a = 128			4 = 128 (Нет)
9	a := a * 2	8		
10	b := b + a		15	
11	a = 128			8 = 128 (Нет)
12	a := a * 2	16		
13	b := b + a		31	
14	a = 128			16 = 128 (Нет)
15	a := a * 2	32		
16	b := b + a		63	
17	a = 128			32 = 128 (Нет)
18	a := a * 2	64		
19	b := b + a		127	
20	a = 128			64 = 128 (Нет)
21	a := a * 2	128		
22	b := b + a		255	
23	a = 128			128 = 128 (Да)
24	вывод b		255	

Ответ: b = 255.

Пример 6

Спортсмен приступает к тренировкам по следующему графику: в первый день он должен пробежать 10 км; каждый следующий день требуется увеличивать дистанцию на 10% от нормы предыдущего дня. Как только дневная норма достигнет или превысит 25 км, необходимо прекратить её увеличение и далее пробегать ежедневно ровно 25 км. Начиная с какого дня спортсмен будет пробегать 25 км?

Пусть x — количество километров, которое спортсмен пробежит в некоторый i-й день. Тогда в следующий $(i + 1)$-й день он пробежит $x + 0{,}1x$ километров ($0{,}1x$ — это 10% от x).



3.6.3. Цикл с заданным числом повторений

Для исполнителей в среде КуМир цикл с заданным числом повторений реализуется с помощью следующей конструкции:

```
нц <число повторений> раз
  <тело цикла>
кц
```

Пример 7

Если правее Робота не встретится препятствий, то, выполнив приведённый ниже алгоритм, он переместится на пять клеток вправо и закрасит эти клетки.

```
алг
нач
  нц 5 раз
    вправо; закрасить
  кц
кон
```

Пример 8

Исполнитель Чертёжник предназначен для построения рисунков на координатной плоскости. Исходное положение исполнителя — начало координат. Система команд Чертёжника:

Команда	Действие
поднять перо	Чертёжник поднимает перо
опустить перо	Чертёжник опускает перо
сместиться в точку (a, b)	Чертёжник из текущей точки с координатами (x, y) перемещается в точку с координатами (a, b)
сместиться на вектор (a, b)	Чертёжник из текущей точки с координатами (x, y) перемещается в точку с координатами (x + a, y + b). Если число a или b положительное, то значение соответствующей координаты увеличивается; если отрицательное, то уменьшается

Пусть Чертёжник находится в точке с координатами (x0, y0). Выясним, в какой точке он окажется после выполнения программы:

```
нц k раз
  сместиться на вектор (a, b)
кц
```

На основании данных о начальном положении Чертёжника и алгоритма его перемещений можно записать:

`x := x0` `y := y0`	Исходное положение

нц `k` **раз**
`x := x + a`
`y := y + b`
кц
	Преобразование координат в теле цикла

Работу этого алгоритма можно представить так:

k	x	y
–	x0	y0
1	x0 + a = x0 + 1 * a	y0 + b = y0 + 1 * b
2	x0 + a + a = x0 + 2 * a	y0 + b + b = y0 + 2 * b
3	x0 + a + a + a = x0 + 3 * a	y0 + b + b + b = y0 + 3 * b
	...	...
k	x0 + a + a + ... + a = = x0 + k * a	y0 + b + b + ... + b = = y0 + k * b

Из таблицы видно, что вместо имеющегося циклического алгоритма можно использовать линейный алгоритм, приводящий к такому же результату:

```
x := x0 + k * a
y := y0 + k * b
```

Таким образом, Чертёжник, находившийся в точке с координатами (x0, y0), после выполнения программы

```
нц k раз
  сместиться на вектор (a, b)
кц
```

окажется в точке с координатами (x0 + k * a, y0 + k * b).

Пусть Чертёжник находится в точке с координатами (1, 2). Выясним, в какой точке он окажется после выполнения программы

```
нц 5 раз
  сместиться на вектор (3, 3)
  сместиться на вектор (1, -1)
кц
```

С учётом исходных данных и алгоритма перемещений исполнителя запишем:

`x := 1` `y := 2`	Исходное положение

нц 5 **раз**
`  x := x + 3; y := y + 3`
`  x := x + 1; y := y - 1`
кц
	Преобразование координат в теле цикла

В теле цикла изменение координат точки происходит дважды. Выполнив соответствующие вычисления, тело цикла перепишем так:

```
нц 5 раз
   x := x + 4; y := y + 2
кц
```

Заменим циклический алгоритм линейным:

```
x := 1 + 5 * 4
y := 2 + 5 * 2
```

Таким образом, Чертёжник после выполнения программы окажется в точке с координатами (21, 12).

Подумайте, какой командой можно дополнить исходную программу, чтобы после выполнения программы Чертёжник:
а) оказался в начале координат;
б) вернулся в исходную точку.

Пример 9

Алгоритм изображения квадрата со стороной a для исполнителя Черепаха имеет следующий вид:

```
алг квадрат (цел a)
нач
  нц 4 раз
    вперед (a); вправо (90)
  кц
кон
```

Подумайте, как модифицировать представленный выше алгоритм в алгоритм рисования универсального правильного многоугольника (правильного треугольника, пятиугольника, шестиугольника и т. д.). Проверьте свои предположения в среде КуМир.

3.6.4. Цикл с переменной

Цикл с переменной иначе называют: **цикл-ДЛЯ, цикл с параметром.**

Логика работы этой конструкции описывается схемой, показанной на рис. 3.14.

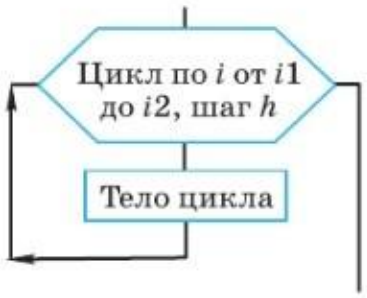


Рис. 3.14. Цикл с переменной

На Школьном алгоритмическом языке эта конструкция записывается так:

```
нц для i от i1 до i2 шаг h
  <тело цикла (последовательность действий)>
кц
```

В цикле-ДЛЯ всегда есть **переменная (параметр) цикла** — величина целого типа, изменяющаяся в ходе выполнения цикла от своего начального значения `i1` до конечного значения `i2` с шагом `h`.

Выполняется цикл-ДЛЯ следующим образом:

1) параметру цикла присваивается начальное значение;
2) параметр цикла сравнивается с конечным значением; если он не превышает конечного значения, то выполняется тело цикла, увеличивается значение параметра цикла на величину шага и снова осуществляется проверка параметра цикла; если же параметр цикла превышает конечное значение, то выполнение цикла заканчивается.

Если величина шага в цикле с параметром равна единице, то шаг не указывают. Мы ограничимся рассмотрением именно таких циклов.

Пример 10

Рассмотрим алгоритм переправы через реку воинского отряда из пяти человек. Солдаты могут воспользоваться помощью двух мальчиков — хозяев небольшой лодки, в которой может переправиться или один солдат, или один либо два мальчика.

```
алг переправа
нач
  нц для i от 1 до 5
    два мальчика переправляются на противоположный
    берег
    один мальчик высаживается на берег, другой плывёт
    обратно
    солдат переправляется через реку
    мальчик возвращается на исходную позицию
  кц
кон
```

Пример 11

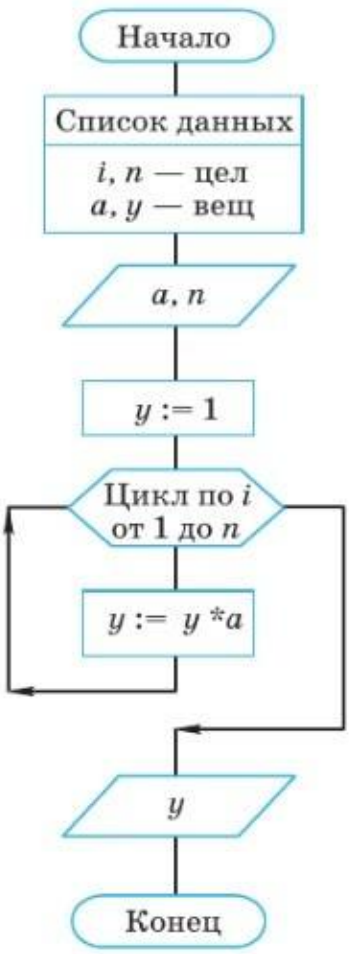
Составим алгоритм вычисления степени с натуральным показателем n для любого вещественного числа a.

По определению:

$$a^1 = a,\ a^n = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n \text{ сомножителей}},\ a \in R,\ n \in N,\ n \geqslant 2.$$

При составлении алгоритма воспользуемся единой формулой, в которой число умножений равно показателю степени:

$$a^n = \underbrace{1 \cdot a \cdot a \cdot \ldots \cdot a}_{n \text{ умножений}}$$



Исполним этот алгоритм для `a = 4` и `n = 3`.

Шаг алгоритма	Операция	Переменные				Условие
		a	n	y	i	
1	`ввод a, n`	4	3	–	–	
2	`y := 1`			1	–	
3	`i := 1`				1	
4	`i <= n`					1 <= 3 (Да)
5	`y := y * a`			4		
6	`i := i + 1`				2	
7	`i <= n`					2 <= 3 (Да)
8	`y := y * a`			16		
9	`i := i + 1`				3	
10	`i <= n`					3 <= 3 (Да)
11	`y := y * a`			64		
12	`i := i + 1`				4	
13	`i <= n`					4 <= 3 (Нет)
14	`вывод y`			64		

Как и в цикле с заданным числом повторений, цикл с переменной имеет строго фиксированное число повторений, что позволяет избежать зацикливания, т. е. ситуации, когда тело цикла выполняется бесконечно.

При каких условиях работа цикла с переменной будет аналогична работе цикла с заданным числом повторений?

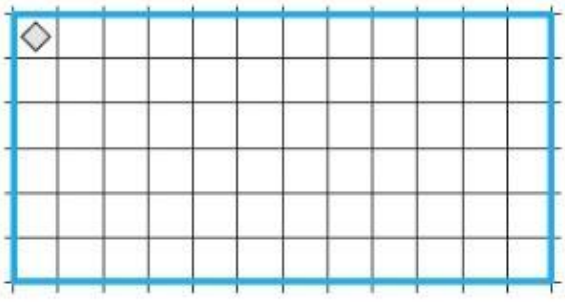
САМОЕ ГЛАВНОЕ

Повторение — алгоритмическая конструкция, представляющая собой последовательность действий, выполняемых многократно. Алгоритмы, содержащие конструкцию «повторение», называют циклическими или циклами. Последовательность действий, многократно повторяющаяся в процессе выполнения цикла, называется телом цикла. В зависимости от способа организации повторений различают четыре типа циклов:

1) цикл с заданным условием продолжения работы;
2) цикл с заданным условием окончания работы;
3) цикл с заданным числом повторений;
4) цикл с переменной.

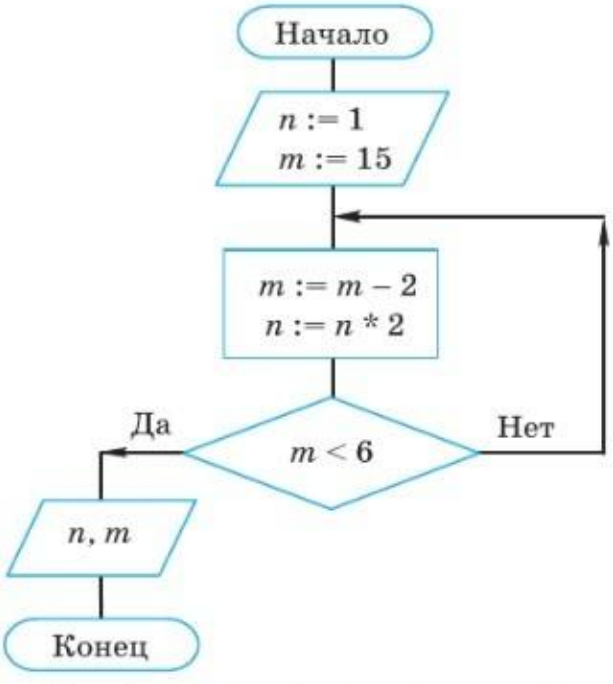
Вопросы и задания

1. Приведите пример циклического алгоритма:
   а) из повседневной жизни;
   б) из литературного произведения;
   в) из любой предметной области, изучаемой в школе.
2. Составьте в среде КуМир алгоритм, под управлением которого Робот обойдёт прямоугольную область, обнесённую стеной, по периметру и закрасит угловые клетки. Размеры области неизвестны.



3. Запас рыбы в пруду оценён в A тонн. Ежегодный прирост рыбы составляет 15%. Ежегодный план отлова — B тонн. Наименьший запас рыбы составляет C тонн. (Запас ниже C тонн уже не восстанавливается.) Составьте блок-схему алгоритма для подсчёта количества лет, в течение которых можно выдерживать заданный план.
4. Дана последовательность 5, 9, 13, 17, ... Составьте блок-схему алгоритма для определения числа слагаемых (первых членов последовательности), сумма которых равна 324.
5. Составьте алгоритм для определения количества цифр в записи произвольного натурального числа.
6. Сумма 10 000 рублей положена в банк, при этом прирост составляет 5% годовых. Составьте алгоритм, определяющий, через какой промежуток времени первоначальная сумма увеличится в два раза.
7. Одноклеточная амёба каждые три часа делится на 2 клетки. Составьте алгоритм вычисления времени, через которое будет X амёб.

8. Определите значения переменных n и m после выполнения алгоритма.



9. Исполнитель Чертёжник находится в произвольной точке координатной плоскости.

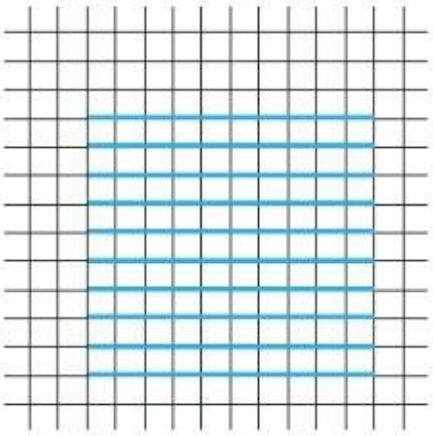
а) Где окажется Чертёжник после выполнения алгоритма?

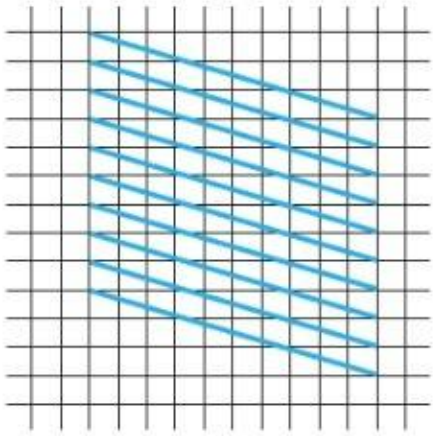
```
алг
нач
  нц 6 раз
    сместиться на вектор (4, 6)
    сместиться на вектор (-2, -4)
  кц
    сместиться на вектор (-12, -12)
кон
```

б) После выполнения алгоритма Чертёжник вернулся в исходную точку. Какие числа надо записать вместо a и b?

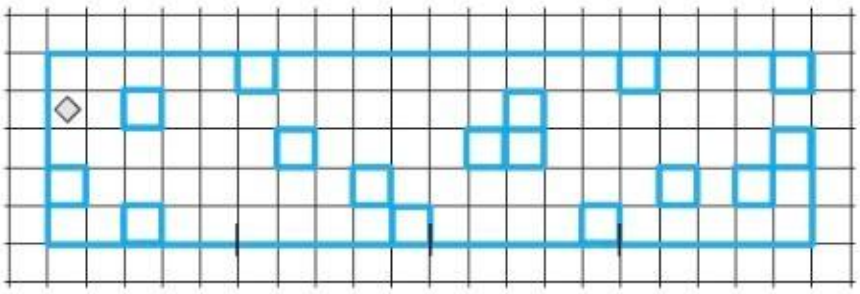
```
алг
нач
  нц 7 раз
    сместиться на вектор (0, 2)
    сместиться на вектор (a, 0)
    сместиться на вектор (0, b)
    сместиться на вектор (-1, 0)
  кц
  сместиться на вектор (-7, 7)
кон
```

10. Используя конструкцию **нц-раз-кц**, команду `сместиться в точку` и переменные, составьте для исполнителя Чертёжник в среде КуМир алгоритмы рисования следующих фигур:

а) 

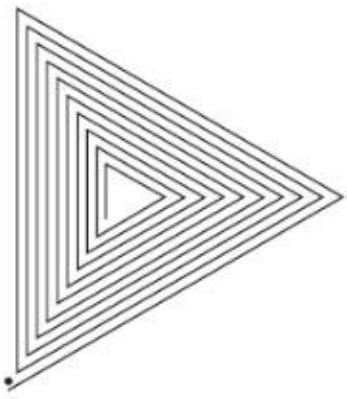
б) 

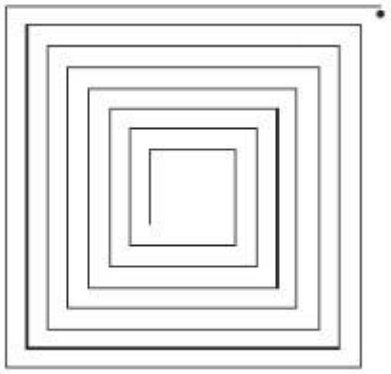
11. Исходное положение Робота в лабиринте обозначено ромбом:



Составьте циклическую программу, следуя которой Робот выйдет из лабиринта (выход справа). Используйте конструкцию **нц-4 раз-кц**; в теле цикла должно быть 7 команд.

12. Составьте для Черепахи программу рисования одного из следующих лабиринтов:

а) 

б) 

13. В следующем алгоритме для Черепахи использованы вложенные циклы.

```
алг
нач
  нц 10 раз
    нц 4 раз
      вперед (50); вправо (90)
    кц
    вправо (36)
  кц
кон
```

Исследуйте приведённый алгоритм в среде КуМир. Предложите варианты его модификации.

14. Составьте алгоритм нахождения произведения z двух натуральных чисел x и y без использования операции умножения.

15. Население города Н увеличивается на 5% ежегодно. В текущем году оно насчитывает 40 000 человек. Составьте блок-схему алгоритма вычисления предполагаемой численности населения города через 3 года. Составьте таблицу значений переменных, задействованных в алгоритме.

16. Каждая бактерия делится на две в течение 1 минуты. В начальный момент имеется одна бактерия. Составьте блок-схему алгоритма вычисления количества бактерий через 10 минут. Исполните алгоритм, фиксируя каждый его шаг в таблице значений переменных.

17. Согласны ли вы со следующими утверждениями?

а) Короткие алгоритмы могут описывать длинные последовательности действий.

б) Краткость алгоритма и скорость его выполнения совпадают.

Обсудите эти вопросы в группе. Приведите примеры, подтверждающие вашу точку зрения.

Тестовые задания для самоконтроля

1. Алгоритмом можно считать:
   а) описание процесса решения квадратного уравнения
   б) расписание уроков в школе
   в) технический паспорт автомобиля
   г) список класса в журнале
2. Как называется свойство алгоритма, означающее, что данный алгоритм применим к решению целого класса задач?
   а) Понятность
   б) Определённость
   в) Результативность
   г) Массовость
3. Как называется свойство алгоритма, означающее, что он всегда приводит к результату через конечное, возможно очень большое, число шагов?
   а) Дискретность
   б) Понятность
   в) Результативность
   г) Массовость
4. Как называется свойство алгоритма, означающее, что он задан с помощью таких предписаний, которые исполнитель может воспринимать и по которым может выполнять требуемые действия?
   а) Дискретность
   б) Понятность
   в) Определённость
   г) Массовость
5. Как называется свойство алгоритма, означающее, что путь решения задачи разделён на отдельные шаги?
   а) Дискретность
   б) Определённость
   в) Результативность
   г) Массовость

6. Как называется свойство алгоритма, означающее, что путь решения задачи определён вполне однозначно, на любом шаге не допускаются никакие двусмысленности и недомолвки?
   а) Дискретность
   б) Понятность
   в) Определённость
   г) Результативность

7. Исполнителю Черепаха был дан для исполнения следующий алгоритм:

   ```
   Повтори 10 [Вперед 10 Направо 72]
   ```

   Какая фигура появится на экране?
   а) Незамкнутая ломаная линия
   б) Правильный десятиугольник
   в) Фигура, внутренние углы которой равны 72°
   г) Правильный пятиугольник

8. Исполнитель Робот передвигается по клетчатому полю, выполняя команды, которым присвоены номера: 1 — на клетку вверх, 2 — на клетку вниз, 3 — на клетку вправо, 4 — на клетку влево. Между соседними клетками поля могут стоять стены. Если при выполнении очередного шага Робот сталкивается со стеной, то он разрушается. В результате выполнения программы 3242332411 Робот успешно прошёл из точки *А* в точку *Б*. Какую программу необходимо выполнить, чтобы Робот вернулся из точки *Б* в точку *А* по кратчайшему пути и не подвергнулся риску разрушения?
   а) 41
   б) 4131441322
   в) 2231441314
   г) 241314
   д) 14

9. Система команд исполнителя Вычислитель состоит из двух команд, которым присвоены номера:

   1. вычти 2
   2. умножь на 3

   Первая из них уменьшает число на 2, вторая увеличивает число в 3 раза. При записи алгоритмов для краткости указываются лишь номера команд. Запишите алгоритм, содержащий не более пяти команд, с помощью которого из числа 11 будет получено число 13.

10. Некоторый алгоритм строит цепочки символов следующим образом:
    1) первая цепочка состоит из одного символа — цифры 1;
    2) в начало каждой из последующих цепочек записывается число — номер строки по порядку, далее дважды подряд записывается предыдущая строка.

    Вот первые 3 строки, созданные по этому правилу:
    (1) 1
    (2) 211
    (3) 3211211

    Сколько символов будет в седьмой цепочке, созданной по этому алгоритму?

11. К четырёхзначному натуральному числу применяется следующий алгоритм:
    1) вычислить сумму первых двух цифр;
    2) вычислить сумму последних двух цифр;
    3) записать полученные два числа друг за другом в порядке убывания (невозрастания).

    Укажите число, которое может получиться в результате работы этого алгоритма:
    а) 1918
    б) 218
    в) 1212
    г) 1218

12. Наибольшей наглядностью обладает следующая форма записи алгоритмов:
    а) словесная
    б) рекурсивная
    в) графическая
    г) построчная

13. Величины, значения которых могут изменяться в процессе исполнения алгоритма, называются:
    а) постоянными
    б) константами
    в) переменными
    г) табличными

14. Величиной целого типа является:

    а) количество мест в зрительном зале

    б) рост человека

    в) марка автомобиля

    г) площадь государства

15. Укажите логическое выражение, определяющее принадлежность точки x отрезку [–10, 10].

    а) `(x > 10)` **и** `(x <-10)`

    б) `(x > 10)` **или** `(x <-10)`

    в) `(x < 10)` **или** `(x >= -10)`

    г) `(x >= -10)` **и** `(x <-10)`

16. Укажите правильный вариант записи условия «x — двузначное число».

    а) `x div 10 <= 9`

    б) `(x >= 10)` **и** `(x < 100)`

    в) `x div 100 = 0`

    г) `x mod 100 = 99`

17. Какая команда присваивания должна следовать за командами `A := A + B` и `B := A - B`, чтобы последовательное выполнение всех трёх команд вело к обмену значениями переменных A и B?

    а) `A := A + B`

    б) `A := A - B`

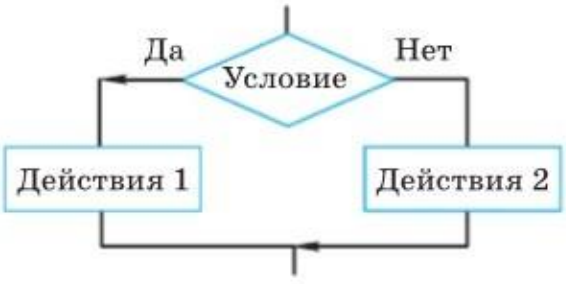
    в) `B := A + B`

    г) `B := B - A`

18. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?

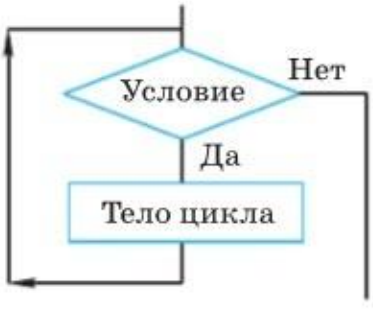
а) Линейный
б) Разветвляющийся
в) Циклический
г) Вспомогательный

19. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?



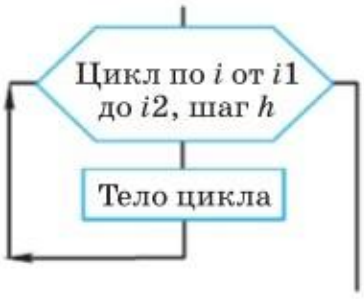
а) Линейный
б) Разветвляющийся с неполным ветвлением
в) Разветвляющийся с полным ветвлением
г) Циклический

20. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?



а) Цикл с параметром
б) Цикл с заданным условием продолжения работы
в) Цикл с заданным условием окончания работы
г) Цикл с заданным числом повторений

21. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?



а) Цикл с заданным условием продолжения работы
б) Цикл с заданным условием окончания работы
в) Цикл с постусловием
г) Цикл с переменной

22. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?

а) Цикл с заданным условием продолжения работы
б) Цикл с заданным условием окончания работы
в) Цикл с заданным числом повторений
г) Цикл с предусловием

23. Сергей, Антон, Таня и Надя, гуляя по лесу, наткнулись на овраг, который можно перейти по шаткому мосту. Сергей может перейти его за минуту, Антон — за две, Таня — за три, Надя — за четыре. Фонарик у группы только один, и он обязательно нужен для перехода по мосту, который выдерживает только двоих человек. Когда два человека вместе идут по мосту, то идут они со скоростью более медлительного из них. Ребята смогли разработать алгоритм перехода на другую сторону оврага за минимально возможное время. Какое время они затратили на его выполнение?

а) 10 минут
б) 11 минут
в) 12 минут
г) 13 минут

24. Дан фрагмент линейного алгоритма:

```
a := 8
b := 6 + 3 * a
a := b / 3 * a
```

Чему равно значение переменной a после его выполнения?

25. Выполните следующий фрагмент линейного алгоритм для a = x, b = y.

```
a := a + b
b := b - a
a := a + b
b := -b
```

Какие значения присвоены переменным a и b?

а) y, x
б) x + y, x - y
в) x, y
г) -y, x

26. Определите значения целочисленных переменных x и y после выполнения фрагмента алгоритма:

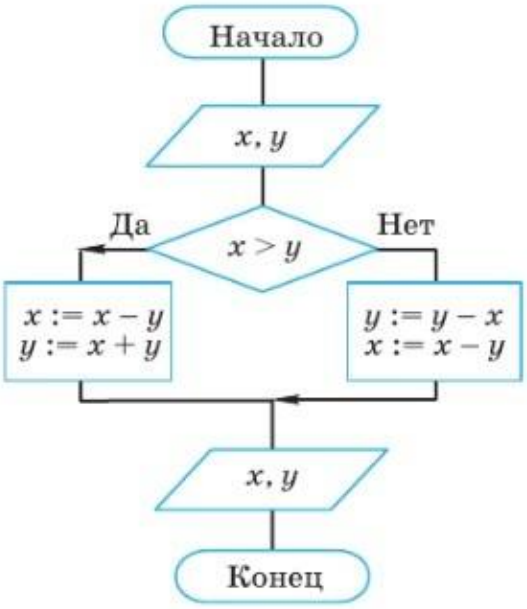
```
x := 11
y := 5
t := y
y := x mod y
x := t
y := y + 2 * t
```

а) x = 11, y = 5
б) x = 5, y = 11
в) x = 10, y = 5
г) x = 5, y = 10

27. Среди четырёх монет есть одна фальшивая. Неизвестно, легче она или тяжелее настоящей. Какое минимальное количество взвешиваний необходимо сделать на весах с двумя чашками без гирь, чтобы определить фальшивую монету?

а) 2 б) 3 в) 4 г) 5

28. Выполните алгоритм при `x = 10` и `y = 15`.



Укажите значения, полученные в результате его выполнения.

а) `-5, 10`

б) `5, 20`

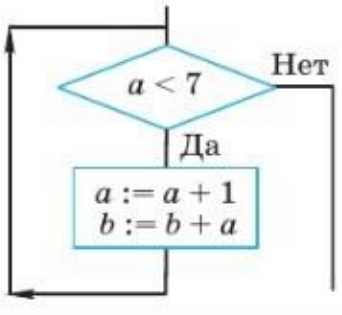
в) `10, 15`

г) `5, 5`

д) `-5, 5`

29. Выполните фрагмент алгоритма при `a = 2` и `b = 0`.

Определите значение переменной `b` после выполнения фрагмента алгоритма:



30. Определите значение переменной f после выполнения фрагмента алгоритма:

```
f := 1
нц для i от 1 до 5
  f := f * i
кц
```

31. Определите значение переменной s после выполнения фрагмента алгоритма:

```
s := 0
нц для i от 1 до 5
  s := s + i * i
кц
```

32. Исполнитель Чертёжник выполнил следующий алгоритм:

```
нц 5 раз
  сместиться на вектор (0, 2)
  сместиться на вектор (4, 0)
кц
```

На какую одну команду можно заменить этот алгоритм, чтобы Чертёжник оказался в той же точке, что и после его выполнения?

а) `сместиться на вектор (-4, 0)`

б) `сместиться на вектор (0, -2)`

в) `сместиться на вектор (20, 10)`

г) `сместиться на вектор (-20, -10)`

Глава 4
НАЧАЛА ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ ПАСКАЛЬ

§ 4.1
Общие сведения о языке программирования Паскаль

Ключевые слова:

- язык программирования
- программа
- система программирования
- алфавит
- служебные слова
- типы данных
- структура программы
- оператор присваивания

Языки программирования — это формальные языки, предназначенные для записи алгоритмов, исполнителем которых будет компьютер. Записи алгоритмов на языках программирования называются **программами**.

Существует несколько тысяч языков программирования. Мы с вами познакомимся с языком программирования **Паскаль**, который был разработан в 70-х гг. прошлого века Никлаусом Виртом (Швейцария). Своё название этот язык получил в честь французского учёного Блеза Паскаля, известного не только своими достижениями в математике, физике и философии, но и созданием первой в мире механической машины, выполнявшей сложение чисел.

Язык Паскаль считается универсальным языком программирования, так как он может применяться для записи алгоритмов решения самых разных задач (вычислительных, обработки текстов, построения графических изображений, поиска информации и т. д.). Он поддерживает *процедурный стиль программирования*, в соответствии с которым программа представляет собой последовательность операторов, задающих те или иные действия[1].

[1] С другими стилями программирования вы познакомитесь при изучении курса информатики в 10–11 классах.

Никлаус Вирт (род. в 1934 г.) — швейцарский учёный, специалист в области информатики, один из известнейших теоретиков в области разработки языков программирования, профессор компьютерных наук. Разработчик языка Паскаль и ряда других языков программирования.

Записать программу на языке программирования можно ручкой на листке бумаги. Для того чтобы запустить программу на выполнение на компьютере, необходимо воспользоваться **системой программирования**, представляющей собой набор компьютерных инструментов, который включает редактор текста, транслятор, отладчик и другие составляющие.

Редактор текста — это программа для ввода, редактирования и форматирования текста программы на языке программирования.

Транслятор — инструмент, предназначенный для преобразования программ, написанных на языках программирования, в программы на машинном языке. Трансляторы делятся на два класса: *компиляторы* и *интерпретаторы*. Компилятор переводит весь исходный текст программы на машинный язык. Интерпретатор последовательно переводит на машинный язык и выполняет операторы исходного текста программы.

Отладчик — инструмент для поиска и исправления ошибок в программе. Основные функции отладчика: пошаговое выполнение программы с отображением результатов, остановка в заранее определённых точках, изменение значений переменных.

Мы будем работать с системой программирования **PascalABC.NET** (рис. 4.1).

По ссылке **http://gotourl.ru/11951** вы найдёте много полезной информации для начинающих программистов, сможете скачать систему программирования PascalABC.NET. Основные сведения по отладке программ в системе программирования PascalABC.NET приведены в приложении 1 учебника.

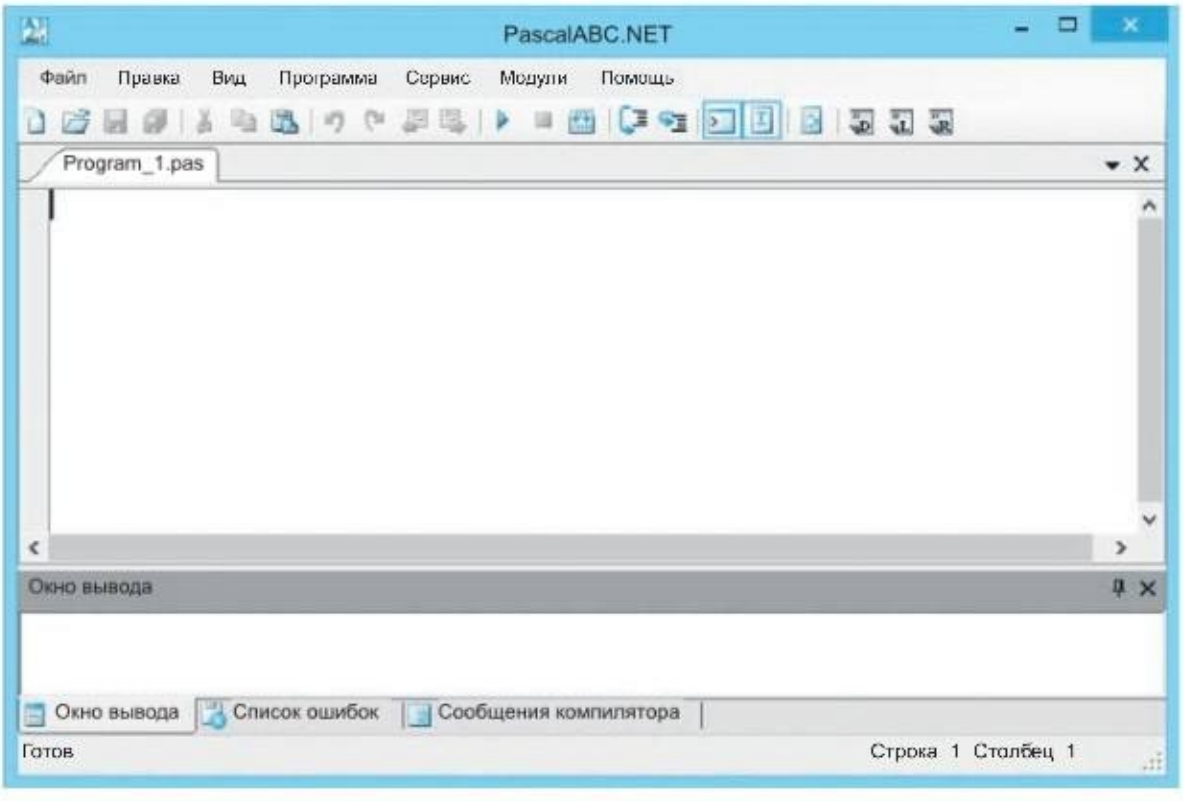


Рис. 4.1. Интерфейс системы программирования PascalABC.NET

4.1.1. Алфавит и словарь языка

Основой языка программирования Паскаль, как и любого другого языка, является **алфавит** — набор допустимых символов, которые можно использовать для записи программы. Это:

- латинские прописные буквы (A, B, C, ..., X, Y, Z);
- латинские строчные буквы (a, b, c, ..., x, y, z);
- арабские цифры (0, 1, 2, ..., 7, 8, 9);
- специальные символы (знак подчёркивания; знаки препинания; круглые, квадратные и фигурные скобки; знаки арифметических операций и др.).

В качестве неделимых элементов (составных символов) рассматриваются следующие последовательности символов:

:= (знак операции присваивания);

>= и <= (знаки $\geqslant$ и $\leqslant$);

<> (знак $\neq$);

// (начало строчного комментария).

В языке существует также некоторое количество различных цепочек символов, рассматриваемых как единые смысловые элементы с фиксированным значением. Такие цепочки симво-

лов называются **служебными словами**. В табл. 4.1 приведены основные служебные слова, которые мы будем использовать при записи программ на языке Паскаль.

Таблица 4.1

Служебные слова языка Паскаль

Служебное слово языка Паскаль	Значение служебного слова
`and`	и
`array`	массив
`begin`	начало
`do`	выполнить
`else`	иначе
`end`	конец
`for`	для
`if`	если
`of`	из
`or`	или
`procedure`	процедура
`program`	программа
`repeat`	повторять
`then`	то
`to`	до (увеличивая до)
`until`	до (до тех пор, пока)
`var`	переменная
`while`	пока

Для обозначения констант, переменных, программ и других объектов используются **имена** — любые отличные от служебных слов последовательности букв, цифр и символа подчёркивания, начинающиеся с буквы или символа подчёркивания.

Прописные и строчные буквы в именах не различаются.

Длина имени может быть любой. Для удобства мы будем пользоваться именами, длина которых не превышает 8 символов.

4.1.2. Типы данных, используемые в языке Паскаль

В языке Паскаль используются различные типы данных. Мы будем пользоваться некоторыми из так называемых **простых типов данных** (табл. 4.2).

Таблица 4.2

Некоторые типы данных в языке Паскаль

Название	Обозначение	Допустимые значения
Целочисленный	`integer`	–2 147 483 648 ... 2 147 483 647
Вещественный	`real`	$-1.8 \cdot 10^{308}$ ... $1.8 \cdot 10^{308}$
Символьный	`char`	Произвольный символ
Строковый	`string`	Последовательность символов произвольной длины
Логический	`boolean`	`True` и `False`

В вещественном числе целая часть от дробной отделяется точкой. Пробелы внутри числа недопустимы.

4.1.3. Структура программы на языке Паскаль

В программе, записанной на языке Паскаль, можно выделить:

1) заголовок программы;
2) раздел **uses**;
3) раздел описания используемых данных;
4) раздел описания действий по преобразованию данных (программный блок).

Заголовок программы состоит из служебного слова **program** и имени программы. После имени программы ставится точка с запятой.

Раздел **uses** может содержать список имён подключаемых модулей. Например, с помощью строки

```
uses GraphABC;
```

подключается модуль `GraphABC`, позволяющий создавать графические объекты.

Раздел описания данных состоит из раздела описания констант (**const**), раздела описания переменных (**var**) и некоторых других разделов[1].

[1] В 8 классе мы ограничимся рассмотрением разделов описания констант и переменных, оставив изучение других разделов для старшей школы.

Переменная в программировании — это поименованная область оперативной памяти, в которой могут храниться данные определённого типа.

В разделе описания переменных указываются имена используемых в программе переменных и их типы.

Имена переменных одного типа перечисляются через запятую, затем после двоеточия указывается их тип; описание каждого типа заканчивается точкой с запятой. Ниже приведён пример раздела описания переменных:

```
var
  i, j: integer;     — целый тип
  x: real;           — вещественный тип
  a: char;           — символьный тип
```

Программа может не иметь заголовка; в ней может отсутствовать раздел **uses** и раздел описания данных. Обязательной частью программы является программный блок. Он содержит команды, описывающие алгоритм решения задачи. Программный блок начинается со слова **begin** и заканчивается словом **end** с точкой.

Ниже приведён общий вид **программы**:

```
program <имя программы>;
uses <список модулей>;
const <список постоянных значений>;
var <описание используемых переменных>;
begin
  <оператор 1>;
  <оператор 2>;
  …
  <оператор N>
end.
```

Операторы — языковые конструкции, с помощью которых в программах записываются действия, выполняемые над данными в процессе решения задачи.

Точка с запятой служит разделителем между операторами, а не является окончанием соответствующего оператора.

Перед оператором **end** точку с запятой ставить не нужно.

4.1.4. Оператор присваивания

Основное преобразование данных, выполняемое компьютером, — присваивание переменной нового значения, что означает изменение содержимого области памяти; оно осуществляется

оператором присваивания, аналогичным команде присваивания Школьного алгоритмического языка. Общий вид оператора:

```
<имя переменной> := <выражение>
```

Операция присваивания допустима для всех приведённых в табл. 4.2 типов данных. Выражения в языке Паскаль конструируются по рассмотренным ранее правилам для Школьного алгоритмического языка.

Рассмотрим процесс выполнения операторов присваивания на следующем примере:

```
a := 10;
b := 5;
s := a + b
```

При выполнении оператора `a := 10` в ячейку оперативной памяти компьютера с именем `a` (в переменную `a`) заносится значение `10`; при выполнении оператора `b := 5` в ячейку оперативной памяти компьютера с именем `b` заносится значение `5`. При выполнении оператора `s := a + b` значения ячеек оперативной памяти с именами `a` и `b` переносятся в процессор, где над ними выполняется операция сложения. Полученный результат заносится в ячейку оперативной памяти с именем `s` (рис. 4.2).

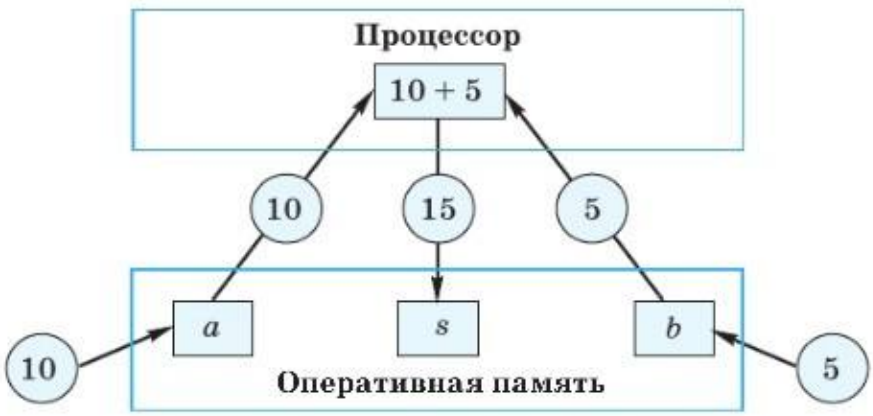


Рис. 4.2. Процесс выполнения оператора присваивания

САМОЕ ГЛАВНОЕ

Паскаль — универсальный язык программирования, получивший своё название в честь выдающегося учёного Блеза Паскаля.

В языке Паскаль используются различные типы данных: целочисленный (`integer`), вещественный (`real`), символьный (`char`), строковый (`string`), логический (`boolean`) и другие.

В программе, записанной на языке Паскаль, можно выделить:

1) заголовок программы;
2) раздел **uses**;
3) раздел описания используемых данных;
4) раздел описания действий по преобразованию данных (программный блок).

Общий вид программы:

```
program <имя программы>;
uses <список модулей>;
const <список постоянных значений>;
var <описание используемых переменных>;
begin
  <оператор 1>;
  <оператор 2>;
  …
  <оператор N>
end.
```

Вопросы и задания

1. В честь кого назван язык программирования Паскаль? Подготовьте краткую биографическую справку об этом учёном.
2. Почему язык программирования Паскаль считается универсальным?
3. Что входит в состав алфавита языка Паскаль?
4. Каких требований следует придерживаться при выборе имён для различных объектов в языке Паскаль?
5. Указывая название, обозначение и диапазон, опишите известные вам типы данных, используемые в языке Паскаль.
6. В чём разница между числами 100 и 100.0 в языке Паскаль?
7. Какую структуру имеет программа, записанная на языке Паскаль?
8. Как записывается раздел описания переменных?
9. Запишите в тетради раздел описания переменных, необходимых для вычисления:
   а) значения функции $y = x^2$;
   б) площади прямоугольника;

в) стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек;

г) стоимости покупки, состоящей из нескольких тетрадей, нескольких ручек и нескольких карандашей.

10. Опишите процесс выполнения операторов присваивания:

```
a := 3; b := 4; a := a + b
```

11. Запишите в тетради оператор для:

а) вычисления среднего арифметического переменных x1 и x2;

б) уменьшения на единицу значения переменной k;

в) увеличения на единицу значения переменной i;

г) вычисления стоимости покупки, состоящей из нескольких тетрадей, нескольких ручек и нескольких карандашей.

§ 4.2 Организация ввода и вывода данных

Ключевые слова:

- оператор вывода write
- формат вывода
- оператор ввода read

4.2.1. Вывод данных

В предыдущем параграфе мы познакомились со структурой программы на языке Паскаль, научились описывать переменные, рассмотрели оператор присваивания. Этого достаточно для того, чтобы записать программу преобразования данных. Но результат этих преобразований нам виден не будет.

Для вывода данных из оперативной памяти на экран компьютера используется оператор вывода write:

```
write (<выражение 1>, <выражение 2>, ..., <выражение N>)
```

список вывода

Здесь в круглых скобках помещается список вывода — список выражений, значения которых выводятся на экран. Это могут быть числовые, символьные и логические выражения, в том числе переменные и константы.

Произвольный набор символов, заключённый в апострофы, считается строковой константой. Строковая константа может содержать любые символы, набираемые на клавиатуре.

Пример 1

Оператор `write ('s=', s)` выполняется так:

1) на экран выводятся символы, заключённые в списке вывода в апострофы: `s=` ;

2) на экран выводится значение переменной, хранящееся в ячейке оперативной памяти с именем `s`.

Если значение переменной `s` равно `15`, то на экране появится: `s=15`.

Пример 2

При выполнении оператора вывода все элементы списка вывода выводятся непосредственно друг за другом. Так, в результате работы оператора

```
write (1, 20, 300)
```

на экран будет выведена последовательность цифр `120300`, которая будет восприниматься нами как число `120300`, а не как три отдельные числовые константы. Сделать выводимые данные более доступными для восприятия можно разными способами.

Вариант организации вывода	Оператор вывода	Результат
Добавить разделители — запятые	`write (1, ',', 20, ',', 300)`	`1,20,300`
Добавить разделители — пробелы	`write (1, ' ', 20, ' ', 300)`	`1 20 300`
Указать формат вывода	`write (1:3, 20:4, 300:5)`	`  1  20  300`

Формат вывода — это указываемое после двоеточия целое число, определяющее, сколько позиций на экране должна занимать выводимая величина. Если цифр в числе меньше, чем зарезервированных под него позиций на экране, то свободные позиции дополняются пробелами слева от числа. Если указанное в формате вывода после двоеточия число меньше, чем необходимо, то оно автоматически будет увеличено до минимально необходимого.

Пример 3

Для вывода вещественного числа в списке вывода для каждого выражения указываются два параметра:

1) общее количество позиций, отводимых под число (с учётом точки-разделителя);
2) количество позиций в дробной части числа.

Оператор вывода	Результат выполнения оператора
`write ('s=', s:2:0)`	`s=15`
`write ('s=', s:3:1)`	`s=15.0`
`write ('s=', s:4:1)`	`s=15.0`
`write ('s=', s:5:1)`	`s= 15.0`

Как можно объяснить одинаковые результаты выполнения операторов во 2-й и 3-й строках приведённой выше таблицы? Обсудите этот вопрос в группе.

При выполнении нескольких операторов `write` вывод осуществляется в одной и той же строке. Оператор `writeln` после вывода на экран всех элементов из списка вывода осуществляет переход на следующую строку (линию, *line*). Других различий между операторами `write` и `writeln` нет.

4.2.2. Первая программа на языке Паскаль

Пользуясь рассмотренными операторами, составим программу, вычисляющую длину окружности и площадь круга радиусом 5,4 см.

Исходным данным в этой задаче является радиус: `r = 5,4` см. Результатом работы программы должны быть величины: `c` — длина окружности и `s` — площадь круга; `c`, `s` и `r` — величины вещественного типа.

Исходные данные и результаты связаны соотношениями, известными из курса математики: $c = 2\pi r$, $s = \pi r^2$.

Программа, реализующая вычисления по этим формулам, будет иметь вид:

```
const
  pi = 3.14;
var
  r, c, s: real;
```

```
begin
  r := 5.4;
  c := 2*pi*r;
  s := pi*r*r;
  writeln ('c=', c:7:4);
  write ('s=', s:7:4)
end.
```

Эта программа верна и решает поставленную задачу. Запустив её на выполнение, вы должны получить следующий результат (рис. 4.3).

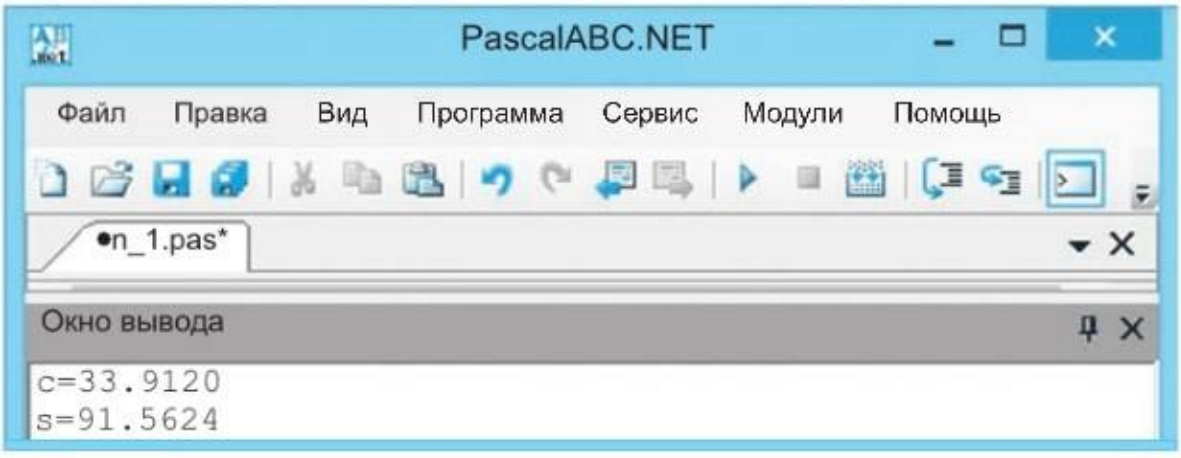


Рис. 4.3. Результат выполнения первой программы

После ввода текста программы и её запуска что-то может пойти не так, и вы окажетесь в одной из двух ситуаций:

1) одна из строк окрасилась красным цветом и появилось сообщение об ошибке — это **синтаксическая ошибка**: возможно, вы пропустили или написали не тот символ, забыли поставить знак «;» и т. д.;
2) программа выполнилась, но получен другой ответ — это **логическая ошибка**: возможно, вы ошиблись при записи арифметических выражений.

Исправьте ошибку и повторно запустите программу на выполнение. Такой процесс называется **отладкой программы**. Он будет завершён в тот момент, когда вы получите требуемый результат.

И всё-таки составленная нами программа имеет существенный недостаток: она находит длину окружности и площадь круга для единственного значения радиуса (5,4 см). Для того чтобы вычислить длину окружности и площадь круга для другого значения радиуса, потребуется вносить изменения непосредственно в текст

программы, а именно изменять оператор присваивания. Внесение изменений в существующую программу по меньшей мере не всегда удобно (например, когда программа большая и операторов присваивания много). Ниже вы познакомитесь с оператором, позволяющим вводить исходные данные в процессе работы программы, не прибегая к изменению текста программы.

4.2.3. Ввод данных с клавиатуры

Для ввода значений переменных в оперативную память используется оператор ввода `read`:

`read (<имя переменной 1>, <имя переменной 2>, ..., <имя переменной N>)`

список ввода

При выполнении оператора `read` компьютер переходит в режим ожидания данных: пользователь должен ввести данные с клавиатуры и нажать клавишу *Enter*[1]. Несколько значений переменных числовых типов могут вводиться через пробел или через запятую. При вводе символьных переменных пробел и запятая воспринимаются как символы, поэтому ставить их нельзя.

Первое введённое пользователем значение переменной помещается в ячейку памяти, имя которой расположено первым в списке ввода, и т. д. Поэтому типы вводимых значений (входного потока) должны соответствовать типам переменных, указанных в разделе описания переменных.

Пример

Пусть

```
var
  i, j: integer;
  x: real;
```

Присвоим переменным `i`, `j`, `x` значения `1`, `0` и `2.5`. Для этого воспользуемся оператором

```
read (i, j, x)
```

и организуем входной поток одним из следующих способов:

```
1 0 2.5 <Enter>
```

```
1 0 <Enter>
2.5 <Enter>
```

```
1 <Enter>
0 <Enter>
2.5 <Enter>
```

Здесь мы представили входной поток в виде одной, двух и трёх строк.

[1] Нажатием клавиши *Enter* может сопровождаться ввод каждого значения.

Для ввода данных с клавиатуры можно также использовать оператор `readln`. После выполнения `readln` осуществляется автоматический переход на новую строку входного потока, даже если в текущей строке остались невведённые символы. Таким образом, `readln` позволяет считывать лишь начальную часть введённой пользователем строки и, проигнорировав её окончание, переходить к следующей строке.

Усовершенствуем предыдущую программу, организовав в ней ввод данных с помощью оператора `read`. А чтобы пользователь знал, для чего предназначена программа, и понимал, какое именно действие ожидает от него компьютер, выведем соответствующие текстовые сообщения с помощью оператора `writeln`:

```
const
  pi = 3.14;
var
  r, c, s: real;
begin
  writeln ('Вычисление длины окружности и площади круга');
  write ('Введите r>>');
  read (r);
  c := 2*pi*r;
  s := pi*r*r;
  writeln ('c=', c:7:4);
  write ('s=', s:7:4)
end.
```

Результат выполнения усовершенствованной программы в PascalABC.NET представлен на рис. 4.4.

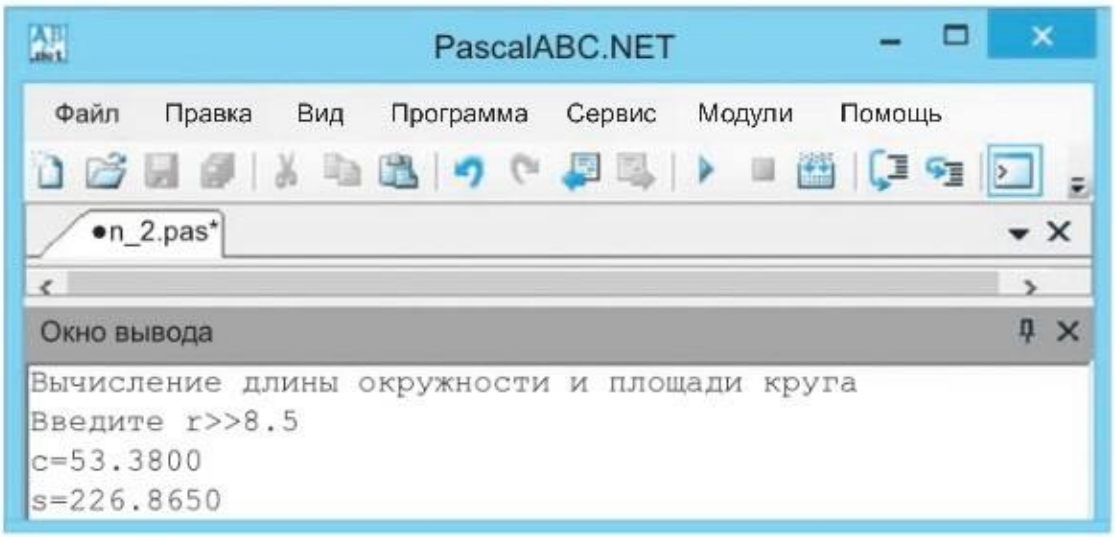


Рис. 4.4. Результат выполнения усовершенствованной программы

С помощью усовершенствованной программы можно вычислить длину окружности и площадь круга для любого значения r. Иначе говоря, программа решает не единичную задачу, а целый класс задач. Кроме того, в программе понятно и удобно организован ввод исходных данных и вывод получаемых результатов. Это обеспечивает дружественность пользовательского интерфейса.

САМОЕ ГЛАВНОЕ

Для ввода в оперативную память значений переменных используются операторы ввода read и readln.

Для вывода данных из оперативной памяти на экран монитора используются операторы вывода write и writeln.

Ввод исходных данных и вывод результатов должны быть организованы понятно и удобно; это обеспечивает дружественность пользовательского интерфейса.

Вопросы и задания

1. Целочисленным переменным i, j, k нужно присвоить соответственно значения 10, 20 и 30. Запишите оператор ввода, соответствующий входному потоку:

   а) 20 10 30

   б) 30 20 10

   в) 10 30 20

   Проверьте себя, дописав и выполнив в среде программирования следующую программу:

```
var
  i, j, k: real;
begin
  // Место для оператора ввода
  writeln ('i=', i);
  writeln ('j=', j);
  write ('k=', k)
end.
```

2. Опишите переменные, необходимые для вычисления площади треугольника по длинам его трёх сторон, и запишите оператор, обеспечивающий ввод необходимых исходных данных.

3. Пусть a — целочисленная переменная, которой присвоено значение 15. Что является результатом выполнения оператора?

   а) `write (a)`

   б) `write ('a')`

   в) `write ('a=', a)`

   Проверьте себя, дописав и выполнив в среде программирования следующую программу:

```
var
  a: real;
begin
  a := 15
  // Место для оператора вывода
end.
```

4. Каким образом можно вывести на экран вещественное число? Поэкспериментируйте с форматом вывода вещественного числа 12.5, указывая разное число позиций для его целой и дробной частей.

```
var
  a: real;
begin
  a := 12.5
  // Место для оператора вывода
end.
```

5. Запишите операторы ввода значений двух переменных и вывода их в обратном порядке.

6. Дан фрагмент программы:

```
read (a); read (b); c: = a + b; write (a); write (b);
write (c)
```

   Сделайте его запись короче, сократив количество операторов ввода и вывода.

7. Дан фрагмент программы:

```
a := 10; b := a + 1; a := b - a; write (a, b)
```

   Что будет выведено на экран компьютера?

8. Напишите программу, которая вычисляет площадь и периметр прямоугольника по длинам двух его сторон.

§ 4.3 Программирование линейных алгоритмов

Ключевые слова:

- вещественный тип данных
- целочисленный тип данных
- символьный тип данных
- строковый тип данных
- логический тип данных

Программы, реализующие линейные алгоритмы, являются простейшими. Все имеющиеся в них операторы выполняются последовательно, один за другим.

Программируя линейные алгоритмы, рассмотрим более подробно целочисленные, логические, символьные и строковые типы данных, а также познакомимся с графическими возможностями системы PascalABC.NET.

4.3.1. Числовые типы данных

Вы уже знакомы с основными числовыми типами данных `integer` и `real`. К данным этих типов применимы стандартные функции, часть из которых приведена в табл. 4.3.

Таблица 4.3

Стандартные функции языка Паскаль

Функция	Назначение	Тип аргумента	Тип результата
`abs(x)`	Модуль x	`integer, real`	Такой же, как у аргумента
`sqr(x)`	Квадрат x	`integer, real`	Такой же, как у аргумента
`sqrt(x)`	Квадратный корень из x	`integer, real`	`real`
`round(x)`	Округление x до ближайшего целого	`real`	
`int(x)`	Целая часть x	`real`	
`frac(x)`	Дробная часть x	`real`	
`random`	Случайное число от 0 до 1	–	`real`
`random(x)`	Случайное число от 0 до x	`integer`	`integer`

Пример 1

Исследуем работу функций round, int и frac, применив их к некоторому вещественному x. Программа будет иметь вид:

```
var
  x: real;
begin
  writeln ('Исследование функций round, int, frac');
  write ('Введите x>>');
  readln (x);
  writeln ('Округление: ', round (x));
  writeln ('Целая часть: ', int (x));
  writeln ('Дробная часть: ', frac (x))
end.
```

Запустите программу несколько раз для каждого $x \in \{10.2; 10.8; -10.2; -10.8\}$. Что вы можете сказать о типе результата каждой из этих функций? С чем, по вашему мнению, связан вид результата работы функции frac(x)?
Обсудите этот вопрос в группе.

4.3.2. Целочисленный тип данных

Над целыми числами в языке Паскаль выполняются следующие операции: сложение (+), вычитание (–), умножение (*), получение неполного частного (div), получение остатка от деления (mod) и деление (/). Результаты первых пяти операций — целые числа. Результатом операции деления может быть вещественное число.

Пример 2

Запишем программу нахождения суммы цифр вводимого с клавиатуры натурального трёхзначного числа.

Используем тот факт, что положительное трёхзначное число можно представить в виде следующей суммы: $x = a \cdot 100 + b \cdot 10 + c$, где a, b, c — цифры числа.

```
var
  x, a, b, c, s: integer;
begin
  writeln ('Нахождение суммы цифр трёхзначного числа');
  write ('Введите исходное число x>>');
  readln (x);
  a := x div 100;
  b := x mod 100 div 10;
```

```
    c := x mod 10;
    s := a + b + c;
    write ('s=', s)
end.
```

Чему равна сумма цифр числа 123? А числа –123? Совпадают ли ваши результаты с результатами работы программы? Как можно объяснить и исправить ошибку в программе?
Обсудите этот вопрос в группе.

4.3.3. Символьный и строковый типы данных

Значением символьной величины (тип `char`) в языке Паскаль является любой из символов, который можно получить на экране нажатием на клавиатуре одной из клавиш или комбинации клавиш.

В тексте программы константу символьного типа можно задать, заключив любой изображаемый символ в апострофы: `'F'`, `'5'`, `'*'`.

Если значение символьной переменной считывается с клавиатуры, то его следует набирать без апострофов.

Значением строковой величины (тип `string`) является произвольная последовательность символов, заключённая в апострофы.

Чтобы обратиться в программе к конкретному символу строки, надо указать имя строковой переменной и порядковый номер (индекс) символа в этой строке. Так, запись `s[1]` обозначает первый символ строки `s`.

В Паскале строки можно сцеплять и сравнивать. Сравнение строк происходит путём посимвольного сравнения кодов образующих их символов.

Рассмотрим некоторые функции, применимые к символьным и строковым величинам (табл. 4.4).

Таблица 4.4

Функции обработки символьных и строковых величин

Функция	Назначение	Тип аргумента	Тип результата
`ord(s)`	Код, соответствующий символу `s` в кодировке Unicode	`char`	`integer`
`chr(x)`	Символ, соответствующий коду `x` в кодировке Unicode	`integer`	`char`

Окончание табл. 4.4

Функция	Назначение	Тип аргумента	Тип результата
length(s)	Длина строки s	string	integer
copy(s, k, n)	Копирование n символов строки s, начиная с позиции k	s – string k – integer n – integer	string

Пример 3

Запишем программу, в которой для введённой с клавиатуры буквы на экран выводится её код. Затем на экран выводится строка, представляющая собой последовательность из трёх букв используемой кодовой таблицы: буквы, предшествующей исходной; исходной буквы; буквы, следующей за исходной.

```
var
  a: char;
  kod: integer;
  b: string;
begin
  writeln ('Код и строка');
  write ('Введите исходную букву>>');
  read (a);
  kod := ord(a);
  b := chr(kod-1) + chr(kod) + chr(kod+1);
  writeln ('Код буквы ', a, ' - ', kod);
  write ('Строка: ', b)
end.
```

Пример 4

Запишем программу, которая из двух строк формирует третью строку и определяет её длину.

```
var
  a, b, c: string;
  n: integer;
begin
  writeln ('Новая строка');
  a := 'информация';
  b := 'автоматика';
  c := copy (a, 1, 5) + copy (b, 5, 6);
  writeln (c);
  n := length(c);
  write ('n=', n)
end.
```

4.3.4. Логический тип данных

Как известно, величины логического типа принимают всего два значения; в Паскале это `false` и `true`. Эти константы определены так, что `false < true`.

Логические значения получаются в результате выполнения операций сравнения числовых, символьных, строковых и логических выражений. Поэтому в Паскале логической переменной можно присваивать результат операции сравнения.

Пример 5

Напишем программу, определяющую истинность высказывания «Число `n` является чётным» для произвольного целого числа `n`.

Пусть `ans` — логическая переменная, а `n` — целая переменная. Тогда в результате выполнения оператора присваивания

```
ans := n mod 2 = 0
```

переменной `ans` будет присвоено значение `true` при любом чётном `n` и `false` — в противном случае.

```
var
  n: integer;
  ans: boolean;
begin
  writeln ('Высказывание о чётности числа');
  write ('Введите исходное число>>');
  read (n);
  ans := n mod 2 = 0;
  write ('Число ', n, ' является чётным - ', ans)
end.
```

Логическим переменным можно присваивать значения логических выражений, построенных с помощью известных вам логических операций И, ИЛИ, НЕ, которые в Паскале обозначаются соответственно **and**, **or**, **not**.

Пример 6

Напишем программу, определяющую истинность высказывания «Существует треугольник с длинами сторон `a`, `b`, `c`» для произвольных натуральных чисел `a`, `b`, `c`.

```
var
  a, b, c: integer;
  ans: boolean;
begin
  writeln ('Высказывание о существовании треугольника');
  write ('Введите значения a, b, c>>');
  readln (a, b, c);
  ans := (a < b + c) and (b < a + c) and (c < a + b);
  writeln ('Существует треугольник со сторонами ',
           a, ', ', b, ', ', c, ' - ', ans)
end.
```

4.3.5. Графические примитивы

В среде PascalABC.NET есть возможность строить изображения из графических примитивов — отрезков, прямоугольников, окружностей. Для этого используется библиотека GraphABC[1].

Положение фигур в графическом окне задается координатами — порядковыми номерами пикселей по горизонтали и по вертикали. Отсчёт значений координаты x происходит слева направо, координаты y — сверху вниз. Графическое окно по умолчанию имеет размеры 640×480 пикселей. Будьте внимательны: такая система координат отличается от той, которую вы используете на уроках математики.

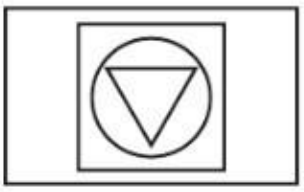
Пример 7

С помощью следующей программы строится приведённое ниже графическое изображение.

```
uses GraphABC;
begin
  Rectangle(50, 50, 550, 250);
  Line (100, 200, 150, 100);
  Line (150, 100, 200, 200);
  Line (200, 200, 100, 200);
  Circle(300, 150, 50);
  Rectangle(400, 100, 500, 200)
end.
```

[1] Некоторые возможности этой библиотеки представлены в приложении 2.

Воспроизведите программу в среде PascalABC.NET. Исследуйте операторы, изображающие графические примитивы. Видоизмените программу так, чтобы получилось изображение:



САМОЕ ГЛАВНОЕ

В языке Паскаль используются вещественный, целочисленный, символьный, строковый, логический и другие типы данных. Для них определены соответствующие операции и функции.

Вопросы и задания

1. Определите результат работы программы. Запишите математическую формулу для вычисления значения s.

```
var
  a, b, c: integer;
  p, s: real;
begin
  a := 12;
  b := 5;
  c := 13;
  p := (a + b + c) / 2;
  s := p;
  s := s * (p - a);
  s := s * (p - b);
  s := s * (p - c);
  s := sqrt(s);
  write('s = ', s)
end.
```

2. Разработайте и отладьте программу, вычисляющую y для заданного x по формуле

$$y = x^3 + 2{,}5x^2 - x + 1.$$

При этом:

а) операцию возведения в степень использовать запрещено;

б) в одном операторе присваивания можно использовать не более одной арифметической операции (сложение, умножение, вычитание);

в) в программе может быть использовано не более пяти операторов присваивания.

Подсказка: преобразуйте выражение к следующему виду:

$$y = ((x + 2{,}5)x - 1)x + 1.$$

3. Разработайте и отладьте программу, вычисляющую длину отрезка AB по заданным координатам точек A и B.

Подсказка: расстояние d между точками $A(x_a, y_a)$ и $B(x_b, y_b)$ выражается формулой

$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}.$$

Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
xa=2 ya=1 xb=10 yb=7	|AB|=10.0

4. Известны длины сторон треугольника a, b, c. Разработайте и отладьте программу, вычисляющую площадь этого треугольника. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
a=3 b=4 c=5	s=6.0

5. Известны координаты вершин A, B, C треугольника. Разработайте и отладьте программу, вычисляющую площадь этого треугольника. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
xa=2 ya=1 xb=6 yb=5 xc=10 yc=1	s=16.0

6. Если сумма налога исчисляется в рублях и копейках, то налоговая служба округляет её до ближайшего рубля (до 50 копеек — с недостатком, свыше 50 копеек (включая 50) — с избытком). Используйте компьютер, чтобы ввести точную сумму налога и вывести сумму, которую следует уплатить.

7. Исследуйте работу функции `random`, запустив многократно на выполнение программу:

```
var
  x, y, n: integer;
begin
  writeln ('Исследование функции random');
  write ('Введите x>>');
  read (x);
  write ('Введите n>>');
  read (n);
  y := random (x);
  writeln ('random(', x,')=', y);
  write ('random(', x,')+', n, '=', y + n)
end.
```

Как можно получить случайное число из интервала $(0, x)$? Как можно получить случайное число из полуинтервала $(0, x]$? Как можно получить случайное число из интервала $(n, x + n)$?

8. Одна компания выпустила три группы лотерейных билетов: для молодёжи, для взрослых и для пенсионеров. Номера билетов каждой группы лежат в пределах:
   - для молодёжи — от 1 до 100;
   - для взрослых — от 101 до 200;
   - для пенсионеров — от 201 до 250.

   С помощью компьютера выберите случайным образом лотерейный билет из каждой группы.

9. Разработайте и отладьте программу, которая для произвольного натурального двузначного числа определяет:
   а) сумму и произведение его цифр;
   б) число, образованное перестановкой цифр исходного числа.
   Тест для проверки правильности программы придумайте самостоятельно.

10. Разработайте и отладьте программу, реализующую алгоритм работы кассира, выдающего покупателю сдачу (s) наименьшим возможным количеством банкнот по 500 (k500), 200 (k200), 100 (k100) и 50 (k50) рублей. Предусмотрите вывод сообщения о том, что часть сдачи, которую невозможно выдать купюрами, будет выдана монетами. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
845	Следует сдать: банкнот по 500 руб. - 1 шт. банкнот по 200 руб. - 1 шт. банкнот по 100 руб. - 1 шт. банкнот по 50 руб. - 0 шт. монетами - 45 руб.

11. Идёт k-я секунда суток. Разработайте и отладьте программу, которая по введённой k-й секунде суток определяет, сколько целых часов h и целых минут m прошло с начала суток. Например, если $k = 13\,257 = 3 \cdot 3600 + 40 \cdot 60 + 57$, то $h = 3$ и $m = 40$. Выведите на экран фразу:
   `It is ... hours ... minutes`
   Вместо многоточий программа должна выводить значения h и m, отделяя их от слов ровно одним пробелом. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
13257	It is 3 hours 40 minutes

12. Разработайте и отладьте программу, которая вычисляет сумму кодов букв в слове БАЙТ.

13. Разработайте и отладьте программу, которая формирует и выводит на экран строку символов, коды которых равны 66, 69, 71, 73, 78.

14. Разработайте и отладьте программу, которая запрашивает три строковые величины — взаимосвязанные прилагательное, существительное и глагол, а затем выводит все варианты фраз с использованием введённых слов. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ	ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЛИСТЬЯ ЛИСТЬЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЛИСТЬЯ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЛИСТЬЯ ЗЕЛЁНЫЕ

Попробуйте доказать, что других вариантов фраз с использованием трёх данных слов не существует. Обсудите этот вопрос в группе.

15. Разработайте и отладьте программу, которая из слова ИНФОРМАТИКА получает слова ФОРМА, ФИРМА, МАК и подсчитывает общее количество символов в этих словах.

16. Есть арифметический фокус, позволяющий угадать дату рождения любого из окружающих вас людей. Для этого нужно, чтобы этот человек выполнил предварительные вычисления по следующему алгоритму: умножил число, соответствующее его дню рождения, на 2, прибавил к результату 5, полученный результат умножил на 50 и прибавил к тому, что получилось, номер месяца, в который он родился. Результат вычислений он должен сообщить вам. Для того чтобы узнать дату рождения, достаточно вычесть из результата вычислений число 250. Последние две цифры полученного числа будут соответствовать месяцу, первые две (первая одна) — числу месяца угадываемой даты рождения. Попробуйте составить программу-диалог с компьютером, в которой компьютер будет запрашивать у пользователя имя, сообщать ему алгоритм предварительных вычислений и запрашивать их результат, после чего «угадает» и сообщит пользователю день и месяц его рождения.

17. Даны значения целочисленных переменных: `a = 10`, `b = 20`. Чему будет равно значение логической переменной `rez` после выполнения операции присваивания?

а) `rez := (a = 10) or (b > 10)`

б) `rez := (a > 5) and (b > 5) and (a < 20) and (b < 30)`

в) `rez := (not(a < 15)) or (b > 20)`

18. Составьте программу, вводящую `true`, если высказывание является истинным, и `false` — в противном случае:

а) сумма цифр трёхзначного числа x является чётным числом;

б) треугольник со сторонами a, b, c является разносторонним.

§ 4.4 Программирование разветвляющихся алгоритмов

Ключевые слова:

- условный оператор
- неполный условный оператор
- вложенные ветвления

4.4.1. Условный оператор

При записи на языке Паскаль разветвляющихся алгоритмов используют **условный оператор**. Его общий вид:

```
if <условие> then
begin
  <группа операторов_1>
end
else
begin
  <группа операторов_2>
end
```

Для записи неполных ветвлений используется неполная форма условного оператора:

```
if <условие> then
begin
  <группа операторов>
end
```

Слова **if – then – else** переводятся с английского языка на русский как **если – то – иначе**.

Если в условном операторе после слов **then** и **else** следует не группа операторов, а один оператор, то слова **begin** и **end** (их называют операторными скобками) можно не использовать.

Перед **else** знак ; не ставится.

В качестве условий используются логические выражения:

- простые — записанные с помощью операций отношения;
- составные — записанные с помощью логических операций.

Пример 1

Запишем программу, определяющую, является ли введённое целое число x чётным.

```
var
  x: integer;
begin
  write ('Введите x>>');
  read (x);
  if x mod 2 = 0 then
    write (x, ' - чётное число')
  else
    write (x, ' - нечётное число')
end.
```

Пример 2

Запишем на языке Паскаль рассмотренный в п. 3.5.1 (пример 4) алгоритм определения принадлежности точки x отрезку [a, b].

```
var
  x, a, b: real;
begin
  writeln ('Принадлежность точки отрезку');
  write('Введите a, b, где  a < b>>');
  read(a, b);
  write ('Введите x>>');
  read (x);
  if (x >= a) and (x <= b) then
    write ('Точка ', x, ' принадлежит отрезку [',a,',',b,']')
  else
    write ('Точка ', x, ' не принадлежит отрезку [',a,',',b,']')
end.
```

Пример 3

Воспользуемся неполным условным оператором для записи на языке Паскаль рассмотренного в п. 3.5.2 (пример 5) алгоритма присваивания переменной y значения наибольшей из трёх величин a, b и c.

```
var
  y, a, b, c: integer;
begin
  writeln ('Нахождение наибольшей из трёх величин');
  write ('Введите a, b, c>>');
  read (a, b, c);
  y := a;
  if b >= y then
    y := b;
  if c >= y then
    y := c;
  write ('y=', y)
end.
```

Дополните эту программу так, чтобы её выполнение приводило к присваиванию переменной y значения наибольшей из четырёх величин a, b, c и d.

Пример 4

Уравнение вида $ax^2 + bx + c = 0$, где x — переменная, a, b и c — некоторые числа, причём $a \neq 0$, называется квадратным уравнением. Алгоритм решения квадратного уравнения вам хорошо известен. Запишем соответствующую программу на языке Паскаль.

```
var
  a, b, c: real;
  d: real;
  x, x1, x2: real;
begin
  writeln('Решение квадратного уравнения');
  write('Введите коэффициенты a, b, c>>');
  readln(a, b, c);
  d := b * b - 4 * a * c;
  if d < 0 then writeln('Корней нет');
  if d = 0 then
  begin
    x := -b / (2 * a);
    writeln('Корень уравнения x=', x:9:3)
  end;
```

```
    if d > 0 then
    begin
      x1 := (-b + sqrt(d)) / (2 * a);
      x2 := (-b - sqrt(d)) / (2 * a);
      writeln('Корни уравнения:');
      writeln('x1=', x1:9:3);
      writeln('x2=', x2:9:3)
    end
  end.
```

4.4.2. Многообразие способов записи ветвлений

В качестве оператора после **then** и **else** можно использовать условный оператор. Например, возможна следующая конструкция:

```
if <условие_1> then
  if <условие_2> then
    <оператор_1>
  else <оператор_2>
else <оператор_3>
```

При использовании таких сложных конструкций (их ещё называют **вложенными ветвлениями**) следует иметь в виду, что **else** всегда относится к ближайшему оператору **if**.

Пример 5

Воспользуемся вложенным ветвлением для записи на языке Паскаль ещё одного варианта решения квадратного уравнения.

```
var
  a, b, c, d, x, x1, x2: real;
begin
  writeln('Введите коэффициенты квадратного уравнения a,b,c');
  readln(a, b, c);
  d := sqr(b) - 4 * a * c;
  if d < 0
    then
      writeln('Нет корней')
    else if d = 0
         then
         begin
           x := -b / (2 * a);
           writeln('x=', x:9:3)
         end
```

```
        else
        begin
          x1 := (-b - sqrt(d)) / (2 * a);
          x2 := (-b + sqrt(d)) / (2 * a);
          writeln('x1=', x1:9:3);
          writeln('x2=', x2:9:3)
        end
end.
```

Используйте вложенные ветвления для записи программы, определяющей принадлежность точки x отрезку [a, b].

САМОЕ ГЛАВНОЕ

При записи на языке Паскаль разветвляющихся алгоритмов используют условный оператор. Его общий вид:

```
if <условие> then
begin
  <группа операторов_1>
end
else
begin
  <группа операторов_2>
end
```

Для записи неполных ветвлений используется неполная форма условного оператора:

```
if <условие> then
begin
  <группа операторов>
end
```

Вопросы и задания

1. Как на языке Паскаль записывается полное и неполное ветвление?
2. Является ли условным оператором последовательность символов?

а) `if x < y then x := 0 else read(y)`

б) `if x >= y then x := 0; y := 0 else write (z)`

в) `if x < y < z then a := a + 1`

3. Используйте операторные скобки для записи следующего фрагмента программы:

```
if a > b then c := 1;
if a > b then d := 2;
if a <= b then c := 3;
if a <= b then d := 4;
```

4. Две точки на плоскости заданы своими координатами. Разработайте, отладьте и протестируйте программу, определяющую, которая из точек находится ближе к началу координат. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
Координаты 1-й точки>>1 2 Координаты 2-й точки>>3 4	1-я точка ближе
Координаты 1-й точки>>1 2 Координаты 2-й точки>>2 1	Точки равноудалены
Координаты 1-й точки>>2 4 Координаты 2-й точки>>2 2	2-я точка ближе

5. Разработайте, отладьте и протестируйте программу, которая производит обмен значений числовых переменных x и y, если x больше y. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
x>>5 y>>6	x=5 y=6
x>>6 y>>5	x=5 y=6

6. Напишите программу для решения задачи № 11 к § 3.5 (определение дня недели).
7. Чтобы развлечься на аттракционе «Американские горки», нужно иметь рост не ниже 140 см и не выше 195 см. Разработайте и отладьте программу, которая запрашивает у посетителя его рост и делает вывод о том, можно ли его пропустить на этот аттракцион.

8. Напишите программу, предусматривающую ввод одного из чисел: 1, 2 или 3 — и построение на экране одной из трёх геометрических фигур: треугольника, если введено число 1; квадрата, если введено число 2; окружности, если введено число 3.
9. Дано натуральное трёхзначное число n. Разработайте, отладьте и протестируйте программу, определяющую:

   а) является ли данное число «перевёртышем», т. е. числом, десятичная запись которого читается одинаково слева направо и справа налево:

Входные данные	Выходные данные
122	Нет
121	Перевёртыш
222	Перевёртыш

   б) есть ли среди цифр данного числа одинаковые:

Входные данные	Выходные данные
123	Нет
121	Да
222	Да

   Протестируйте программу на приведённых входных данных.
10. Даны три натуральных числа. Разработайте, отладьте и протестируйте на приведённых данных программу, определяющую, существует ли треугольник с такими длинами сторон. Если такой треугольник существует, то определите его тип (равносторонний, равнобедренный, разносторонний). Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
a b c>>1 2 1	Не существует
a b c>>2 2 2	Равносторонний
a b c>>20 20 30	Равнобедренный
a b c>>3 4 5	Разносторонний

11. Имеются данные о количестве полных лет трёх призёров спартакиады. Разработайте и отладьте программу, определяющую и выводящую возраст самого младшего призёра.

12. Дана программа на языке Паскаль:

```
var
  s, t: integer;
begin
  read(s);
  read(t);
  if (s >= 10) or (t > 10) then
    write('Да')
  else
    write('Нет')
end.
```

Было проведено 9 запусков программы, во время которых в качестве значений переменных s и t вводились следующие пары чисел: (1, 2); (11, 2); (1, 12); (11, 12); (-11, -12); (-11, 12); (-12, 11); (10, 10); (10, 5).

Не запуская программу на выполнение, выясните, сколько было запусков, при которых программа вывела "Да". Для анализа алгоритма и исходных данных начертите в рабочей тетради и заполните следующую таблицу:

№	s	t	s >= 10	t > 10	(s >= 10) or (t > 10)	Вывод
1	1	2	0	0	0	Нет
...						
9	10	5	1	0	1	Да

13. Дан условный оператор:

```
if a < 5 then
  c := 1
else
  if a > 5 then
    c := 2
  else c := 3
```

Какое значение имеет переменная a, если в результате выполнения условного оператора переменной c присваивается значение 3?

14. Напишите программу, в которой пользователю предлагается дополнить до 100 некоторое целое число a (a — случайное число, меньшее 100). Ответ пользователя проверяется и комментируется.

15. С помощью программы сравните тройки слов и сделайте выводы о том, как происходит сравнение: KAWAI — Kawai — kawai, информатика — информатика — информатикА, 50_кг — 50_kg — 200_кг, яблоко — яблоки — яблоня.

16. Напишите программу, моделирующую работу исполнителя Автомат, который получает на вход трёхзначное натуральное число и строит новое число следующим образом:

    1) вычисляет суммы первой и второй, затем второй и третьей цифр;
    2) выводит полученные суммы в порядке неубывания.

    Например, для числа 125 будут получены суммы: 1 + 2 = 3, 2 + 5 = 7. Результат: 37.

17. Поле шахматной доски определяется парой натуральных чисел, каждое из которых не превосходит 8. Напишите программу, которая по введённым координатам двух полей (k, l) и (m, n) определяет, имеют ли эти поля один цвет. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
Координаты 1-го поля>>2 2 Координаты 2-го поля>>3 3	Поля одного цвета
Координаты 1-го поля>>2 3 Координаты 2-го поля>>3 3	Поля разного цвета
Координаты 1-го поля>>2 7 Координаты 2-го поля>>5 4	Поля одного цвета

§ 4.5 Программирование циклических алгоритмов

Ключевые слова:

- оператор цикла
- **while** (цикл-ПОКА)
- **repeat** (цикл-ДО)
- **for** (цикл-ДЛЯ)
- вложенные циклы

Оператор цикла — команда, реализующая алгоритмическую конструкцию «повторение» на языке программирования. В языке программирования Паскаль существует несколько операторов цикла.

4.5.1. Программирование циклов с заданным условием продолжения работы

Цикл с заданным условием продолжения работы (**цикл-ПОКА**) программируется в языке Паскаль с помощью оператора **while**. Общий вид оператора:

```
while <условие> do
begin
  <тело цикла>
end
```

Здесь:

<условие> — логическое выражение; пока оно истинно, выполняется тело цикла;

<тело цикла> — один или несколько операторов, описывающих последовательность действий, выполняемых многократно.

Пример 1

Следующая программа изображает в центральной части графического окна ряд окружностей радиусом 20.

```
uses GraphABC;
var
  x: integer;
begin
  x := 20;
  while x <= 620 do
```

```
  begin
    circle(x, 240, 20);
    x := x + 40
  end
end.
```

Внесите изменения в программу так, чтобы:
а) рисовались окружности радиусом 30;
б) рисовался вертикальный ряд окружностей.

Пример 2

Запишем на языке Паскаль рассмотренный в п. 3.2.2 (пример 3) алгоритм Евклида для нахождения наибольшего общего делителя двух натуральных чисел.

```
var
  x, y, nod: integer;
begin
  write('Введите x>>');
  read(x);
  write('Введите y>>');
  read(y);
  while x <> y do
  begin
    if x > y then
      x := x - y
    else
      y := y - x
  end;
  nod := x;
  write('НОД = ', nod)
end.
```

Найдите с помощью программы наибольший общий делитель для следующих пар чисел: 123 и 12; 450 и 180; 500 и 125. Как можно воспользоваться этой программой, если надо найти НОД трёх натуральных чисел, например: 450, 180 и 60?

Пример 3

Запишем на языке Паскаль рассмотренный в п. 3.6.1 (пример 3) алгоритм получения частного `q` и остатка `r` от деления натурального числа `x` на натуральное число `y` без использования операции деления.

```
var
  x, y, q, r: integer;
begin
  writeln('Частное и остаток');
  write('Введите делимое x>>');
  read(x);
  write('Введите делитель y>>');
  read(y);
  r := x;
  q := 0;
  while r >= y do
  begin
    r := r - y;
    q := q + 1
  end;
  writeln('Частное q=', q);
  write('Остаток r=', r)
end.
```

Выполните программу при x = 25 и y = 4. Каким будет результат выполнения программы при x = −10 и y = 3? Как вы можете объяснить этот результат?

Пример 4

Имеется последовательность натуральных чисел. Составим программу, получающую на вход натуральные числа, количество которых заранее неизвестно (0 — признак окончания ввода, не входит в последовательность), и подсчитывающую количество членов последовательности, оканчивающихся на 3.

```
var
  a, k: integer;
begin
  writeln ('Обработка последовательности');
  k := 0;
  writeln('Первый член последовательности>>');
  readln(a);
  while a <> 0 do
  begin
    if (a mod 10 = 3) then k := k + 1;
    writeln('Очередной член последовательности или 0>>');
    readln(a)
  end;
  writeln('k=', k)
end.
```

Самостоятельно разработайте тест для проверки работоспособности программы.

Модифицируйте программу так, чтобы в ней подсчитывалось количество членов последовательности, кратных 3.

4.5.2. Программирование циклов с заданным условием окончания работы

Цикл с заданным условием окончания работы **(цикл-ДО)** программируется в языке Паскаль с помощью оператора **repeat**. Общий вид оператора:

```
repeat
    <тело цикла>
until <условие>
```

Здесь:

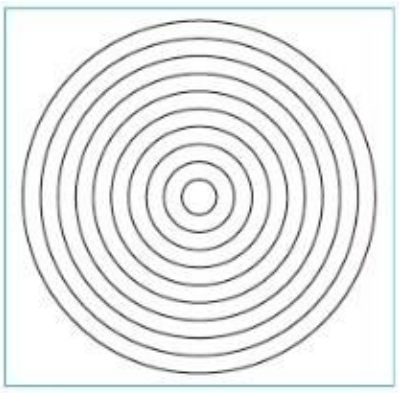
<тело цикла> — один или несколько операторов, описывающих последовательность действий, выполняемых многократно;

<условие> — логическое выражение; если оно истинно, то выполнение тела цикла прекращается.

Пример 5

Следующая программа изображает в центральной части графического окна множество окружностей с общим центром, радиус которых уменьшается от 200 до тех пор, пока не станет меньшим 10.

```
uses GraphABC;
var
  r: integer;
begin
  r := 200;
  repeat
    circle(320, 240, r);
    r := r - 5
  until r < 10
end.
```



Запустите программу на выполнение при разных значениях шага изменения радиуса. Попробуйте получить такое же изображение, использовав в программе оператор **while**.

Пример 6

Рассмотрим произвольное натуральное число n, не превосходящее 10^9. Составим программу, вычисляющую количество цифр в числе n.

Для подсчёта количества цифр в записи числа n будем действовать по следующему плану: «укорачиваем» число справа на один разряд; учитываем правую цифру как цифру в записи исходного числа. Эту группу действий повторяем до тех пор, пока число («укороченное» число) больше нуля; достижение нуля — условие прекращения вычислений.

```
var
  n, k: integer;
begin
  write('Введите число n>>');
  read (n);
  k := 0;
  repeat
    n := n div 10;
    k := k + 1
  until n = 0;
  write('k = ', k)
end.
```

Предложите решение этой задачи с использованием оператора **while**.

Попытайтесь доработать исходную программу так, чтобы в ней подсчитывалась ещё и сумма цифр исходного числа.

Пример 7

Запишем на языке Паскаль рассмотренный в п. 3.6.2 (пример 6) алгоритм решения задачи о графике тренировок спортсмена.

```
var
  i: integer;
  x: real;
begin
  writeln('График тренировок');
  i := 1;
  x := 10;
  repeat
    i := i + 1;
    x := x + 0.1 * x;
  until x >= 25;
```

```
    writeln('Начиная с  ', i, '-го дня спортсмен будет
            пробегать 25 км')
end.
```

4.5.3. Программирование циклов с фиксированным числом повторений

Цикл с фиксированным числом повторений (**цикл-ДЛЯ**, цикл с переменной) программируется в языке Паскаль с помощью оператора **for**. Его общий вид:

```
for var i := n1 to n2 do
begin
  <тело цикла>
end;
```

или

```
for var i := n2 downto n1 do
begin
  <тело цикла>
end;
```

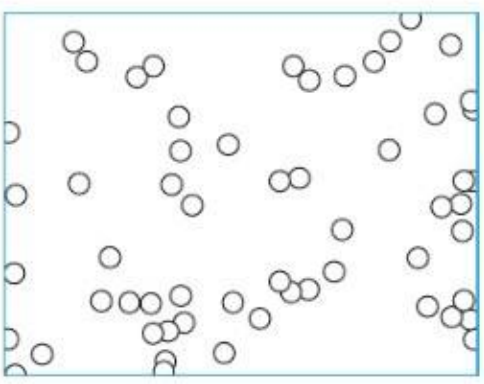
Здесь i — параметр цикла[1]; в первом случае он изменяется от меньшего значения (n1) к большему (n2) с шагом 1; во втором случае — от большего значения (n2) к меньшему (n1) с шагом −1.

Условием выхода из цикла является выход параметра за конечное значение.

Пример 8

Следующая программа изображает в графическом окне ровно 100 окружностей радиусом 10, координаты центров которых генерируются случайным образом.

```
uses GraphABC;
var
  x, y, i: integer;
begin
  for i := 1 to 100 do
  begin
    x := random(640);
    y := random(480);
    circle(x, y, 10)
  end
end.
```



[1] Мы ограничимся рассмотрением целочисленных параметров цикла.

Внесите изменения в программу так, чтобы радиусы изображаемых окружностей также генерировались случайным образом. Попробуйте получить такое же изображение, использовав в программе оператор **while**.

Пример 9

Делитель натурального числа n — это число, на которое n делится без остатка. У любого натурального числа n есть хотя бы два делителя — это 1 и n. Составим программу, которая для введённого с клавиатуры натурального числа n выводит на экран все делители этого числа.

Организуем полный перебор всех возможных вариантов делителей от 1 до n.

```
var
  i, n: integer;
begin
  write('Введите число n>>');
  read (n);
  writeln('Делители числа ', n);
  for i := 1 to n do
  begin
    if n mod i = 0 then
      writeln(i)
  end
end.
```

Модифицируйте программу так, чтобы в ней подсчитывалось количество делителей исходного числа n. Дополните программу так, чтобы на основании найденного количества делителей в ней выводилось сообщение о том, является исходное число простым или нет.

Пример 10

Запишем на языке Паскаль рассмотренный в п. 3.6.4 (пример 11) алгоритм вычисления степени с натуральным показателем n для любого вещественного числа a.

```
var
  i, n: integer;
  a, y: real;
begin
  writeln('Возведение в степень');
  write('Введите основание a>>');
  read(a);
```

```
    write('Введите показатель n>>');
    read(n);
    y := 1;
    for i := 1 to n do
      y := y * a;
    write('y=', y)
  end.
```

Пример 11

Составим программу, которая в последовательности натуральных чисел подсчитывает количество чисел, кратных 3. Программа получает на вход количество чисел в последовательности, а затем — сами числа. Программа должна вывести одно число — количество членов последовательности, кратных 3.

```
var
  i, n, a, k: integer;
begin
  writeln('Обработка последовательности');
  k := 0;
  writeln('Введите количество членов последовательности>>');
  readln(n);
  for i := 1 to n do
  begin
    writeln('Введите очередной член последовательности>>');
    readln(a);
    if a mod 3 = 0 then
      k := k + 1;
  end;
  writeln('k=', k)
end.
```

Самостоятельно разработайте тест для проверки работоспособности программы.

Модифицируйте программу так, чтобы в ней подсчитывалось количество членов последовательности, оканчивающихся на 3.

Сравните задачи и программы, рассмотренные в примерах 4 и 11. Укажите их сходство и различие.

Пример 12

Напишем программу, которая предлагает ввести строку, состоящую из строчных английских букв, и находит количество букв "s" в этой строке.

```
var
  s: string;
  i, k: integer;
begin
  write('Введите строку из строчных английских букв>>');
  read(s);
  k := 0;
  for i := 1 to length(s) do
    if s[i] = 's' then
      k := k + 1;
  write('k=', k)
end.
```

Модифицируйте программу так, чтобы в ней:
а) подсчитывалось общее количество букв "s" и "t" во введённой строке;
б) определялось, каких букв — "s" или "t" — во введённой строке больше.

4.5.4. Вложенные циклы

Цикл называется **вложенным**, если он содержится внутри (в теле) другого цикла.

Цикл, содержащий в себе другой цикл, называют *внешним*, а цикл, содержащийся в теле другого цикла, — *внутренним*. Внутренний и внешний циклы могут быть любыми из трёх видов: цикл с переменной, цикл с предусловием или цикл с постусловием.

Пример 13

Составим программу, выводящую на экран пять строк, каждая из которых состоит из десяти символов *:

```
var
  i, j: integer;
begin
  for i:=1 to 5 do
  begin
    for j:=1 to 10 do
      write('*');
    writeln
  end
end.
```

Модифицируйте программу так, чтобы в ней на экран выводилось десять строк, состоящих из пяти символов * каждая. Выясните, для чего в этой программе используется команда `writeln`.

Пример 14

Следующая программа изображает в графическом окне пять рядов окружностей радиусом 10, по 8 кругов в каждом ряду.

```
var
  i, j, x, y: integer;
begin
  y := 10;
  for i:=1 to 5 do
  begin
    x := 10;
    for j := 1 to 8 do
    begin
      circle(x, y, 10);
      x := x + 20
    end;
    y := y + 20
  end
end.
```

Внесите изменения в программу так, чтобы:
а) рисовались окружности радиусом 20;
б) ряды окружностей заполняли всё графическое окно.

САМОЕ ГЛАВНОЕ

Оператор цикла — команда, реализующая алгоритмическую конструкцию «повторение» на языке программирования. В языке программирования Паскаль существует несколько операторов цикла: **while** (цикл-ПОКА), **repeat** (цикл-ДО), **for** (цикл-ДЛЯ).

Если число повторений тела цикла известно, то лучше применить оператор **for**; в остальных случаях используются операторы **while** и **repeat**.

Вопросы и задания

1. Проанализируйте работу фрагмента программы и ответьте на вопросы.

```
x := 1;
y := 1;
while x < 5 do
begin
  y := y * 2;
  x := x + 1
end
```

а) Сколько раз выполнится тело цикла?
б) Какое значение примет переменная x после завершения программы?
в) Какое значение примет переменная y после завершения программы?
г) Сколько раз выполнится тело цикла, если заменить условие на x <= 5?
д) Сколько раз выполнится тело цикла, если заменить условие на x >= 5?
е) Сколько раз выполнится тело цикла, если заменить условие на x > 0?
ж) Что произойдёт, если из тела цикла убрать команду `x := x + 1`?
з) Сколько раз выполнится тело цикла, если заменить команду `x := x + 1` на `x := x + 2`?
и) Сколько раз выполнится тело цикла, если заменить команду `x := x + 1` на `x := x - 1`?

2. Дана последовательность операторов:

```
a := 1;
b := 2;
while a + b < 8 do
begin
  a := a + 1;
  b := b + 2
end;
s := a + b
```

Сколько раз будет повторён цикл и какими будут значения переменных a, b, s после выполнения этой последовательности операторов?

3. Требовалось написать программу вычисления факториала числа n (факториал числа n есть произведение всех целых чисел от 1 до n). Программист торопился и написал программу неправильно. Ниже приведён фрагмент его программы, в котором содержится несколько ошибок:

```
k := 1;
f := 0;
while k < n do
  f := f * k;
  k := k + 1
```

Обсудите этот фрагмент программы в группе. Найдите ошибки. Внесите необходимые исправления, допишите программу и выполните её на компьютере. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
Введите n>>5	5!=120
Введите n>>6	6!=720

4. Проанализируйте следующий цикл:

```
while a < b do
  c := a = b
```

В чём его особенность?

5. Запишите на языке Паскаль программы решения задач № 3, 4, 6, 7 из § 3.6. Используйте оператор **while**.
6. Дана последовательность операторов:

```
a := 1;
b := 1;
repeat
  a := a + 1;
  b := b * 2
until b > 8;
s := a + b
```

Сколько раз будет повторено тело цикла и какими будут значения переменных a, b, s после выполнения этой последовательности операторов? Обсудите этот вопрос в группе.

7. Напишите программу, в которой осуществляется ввод целых чисел (ввод осуществляется до тех пор, пока не будет введён ноль) и подсчёт суммы всех введённых чисел. Используйте оператор **repeat**.
8. Напишите программу, в которой осуществляется ввод целых чисел (ввод осуществляется до тех пор, пока не будет введён ноль) и определение максимального (наибольшего) из введённых чисел. Используйте оператор **repeat**.

9. Сколько раз будет выполнено тело цикла?

а)
```
for i := 0 to 15 do s := s + i;
```
б)
```
for i := 10 to 15 do s := s + i;
```
в)
```
for i := -1 to 1 do s := s + i;
```
г)
```
for i := 10 to 10 do s := s + i;
```
д)
```
k := 5;
for i := k - 1 to k + 1 do s := s + i
```

10. Напишите программу, которая 10 раз выводит на экран ваши имя и фамилию.

11. Напишите программу, выводящую на экран изображение шахматной доски, где чёрные клетки изображаются звёздочками, а белые — пробелами. Рекомендуемый вид экрана после выполнения программы:

```
*   *   *   *
  *   *   *   *
*   *   *   *
  *   *   *   *
*   *   *   *
  *   *   *   *
*   *   *   *
  *   *   *   *
```

12. Напишите программу, которая вычисляет сумму:

а) первых n натуральных чисел;
б) квадратов первых n натуральных чисел;
в) всех чётных чисел на отрезке от 1 до n;
г) всех двузначных чисел.

13. Запишите на языке Паскаль программы решения задач № 14–16 из § 3.6. Используйте оператор **for**.

14. Напишите программу, которая выводит на экран таблицу степеней двойки (от нулевой до десятой). Рекомендуемый вид экрана после выполнения программы:

```
Таблица  степеней  двойки
      0          1
      1          2
      2          4
      3          8
      4         16
      5         32
      6         64
      7        128
      8        256
      9        512
     10       1024
```

15. Напишите программу, которая выводит на экран таблицу умножения на n (n — целое число в диапазоне от 2 до 10, вводимое с клавиатуры). Протестируйте программу на следующих данных:

Входные данные	Выходные данные
Введите n>>5	5 * 2 = 10 5 * 3 = 15 5 * 4 = 20 5 * 5 = 25 5 * 6 = 30 5 * 7 = 35 5 * 8 = 40 5 * 9 = 45 5 * 10 = 50

16. После строительства дома осталось некоторое количество плиток. Их можно использовать для выкладывания прямоугольной площадки на участке рядом с домом. Если укладывать в ряд по 10 плиток, то для квадратной площадки плиток не хватит. При укладывании по 8 плиток в ряд остаётся один неполный ряд, а при укладывании по 9 плиток тоже остаётся неполный ряд, в котором на 6 плиток меньше, чем в неполном ряду при укладывании по 8.

    Напишите программу, вычисляющую, сколько всего плиток осталось после строительства дома.

17. Какой из трёх рассмотренных операторов цикла является, по вашему мнению, основным, т. е. таким, что им можно заменить два других? Обсудите этот вопрос в группе.

Тестовые задания для самоконтроля

1. Разработчиком языка Паскаль является:

   а) Блез Паскаль
   б) Никлаус Вирт
   в) Норберт Винер
   г) Эдсгер В. Дейкстра

2. Что из нижеперечисленного не входит в алфавит языка Паскаль?

   а) Латинские строчные и прописные буквы
   б) Составные символы
   в) Русские строчные и прописные буквы
   г) Знак подчёркивания

3. Какая последовательность символов не может быть именем в языке Паскаль?

   а) `_mas`
   б) `mas1`
   в) `d2`
   г) `2d`

4. Обозначение вещественного типа данных в языке Паскаль:

   а) `real`
   б) `integer`
   в) `boolean`
   г) `string`

5. В программе на языке Паскаль обязательно должен быть:

   а) заголовок программы
   б) блок описания используемых данных
   в) программный блок
   г) оператор присваивания

6. Какого раздела не существует в программе, написанной на языке Паскаль?

   а) Заголовка
   б) Примечаний
   в) Описаний
   г) Операторов

7. Языковые конструкции, с помощью которых в программах записываются действия, выполняемые в процессе решения задачи, называются:

   а) операндами
   б) операторами
   в) выражениями
   г) данными

8. Разделителями между операторами служит:

   а) точка
   б) точка с запятой
   в) пробел
   г) запятая

9. Описать переменную — это значит указать её:

   а) имя и значение
   б) имя и тип
   в) тип и значение
   г) имя, тип и значение

10. В фрагменте программы

```
program error;
begin
  SuMma := 25 - 14
end.
```

ошибкой является:

   а) некорректное имя программы
   б) не определённое имя переменной
   в) некорректное имя переменной
   г) запись арифметического выражения

11. Какая клавиша нажимается после набора последнего данного при выполнении оператора `read`?

    а) *Enter*
    б) точка с запятой
    в) пробел
    г) *Ctrl*

12. При присваивании изменяется:

    а) имя переменной
    б) тип переменной
    в) значение переменной
    г) значение константы

13. Для вывода результатов в Паскале используется оператор:

    а) `begin`
    б) `readln`
    в) `write`
    г) `print`

14. Для вычисления квадратного корня из x используется функция:

    а) `abs(x)`
    б) `sqr(x)`
    в) `sqrt(x)`
    г) `int(x)`

15. Для генерации случайного целого числа из полуинтервала [10, 20) необходимо использовать выражение:

    а) `random * 20`
    б) `random(20)`
    в) `random(10) + 10`
    г) `random(10) * 2`

16. В каком из условных операторов допущена ошибка?

    а) `if b = 0 then write('Деление невозможно');`
    б) `if a < b then min := a; else min := b;`
    в) `if a > b then max := a else max := b;`
    г) `if (a > b) and (b > 0) then c := a + b;`

17. В условном операторе и после **then**, и после **else** нельзя использовать:

    а) оператор вывода

    б) составной оператор

    в) несколько операторов

    г) условный оператор

18. Определите значение переменной c после выполнения следующего фрагмента программы:

```
a := 100;
b := 30;
a := a - b * 3;
if a > b then
  c := a + b;
else
  c := b - a;
```

    а) 20

    б) 70

    в) –20

    г) 180

19. Условный оператор

```
if a mod 2 = 0 then write('Да')else write('Нет');
```

    позволяет определить, является ли число a:

    а) целым

    б) двузначным

    в) чётным

    г) простым

20. Какого оператора цикла не существует в языке Паскаль?

    а) **for**

    б) **while**

    в) **repeat...until**

    г) **goto**

21. Цикл в фрагменте программы

```
p := 2;
repeat
  p := p * 0.1
until p < 0.1
```

будет исполнен:

а) 0 раз
б) 1 раз
в) 2 раза
г) бесконечное число раз

22. Цикл в фрагменте программы

```
a := 1;
b := 1;
while a + b < 8 do
begin
  a := a + 1;
  b := b + 2
end;
```

выполнится:

а) 0 раз
б) 2 раза
в) 3 раза
г) бесконечное число раз

23. Определите значения переменных s и i после выполнения фрагмента программы:

```
s := 0;
i := 5;
while i > 0 do
begin
  s := s + i;
  i := i - 1
end;
```

а) s = 0, i = -1
б) s = 5, i = 0
в) s = 15, i = 5
г) s = 15, i = 0

24. Выберите фрагмент программы, в котором вычисляется произведение $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$.

а) `p := 0; i := 1; while i <= 5 do i := i + 1; p := p * i;`
б) `p := 1; i := 1; while i < 6 do i := i + 1; p := p * i;`

в)
```
p := 1; i := 1; while i < 6 do
    begin p := p * i; i := i + 1 end;
```
г)
```
p := 1; i := 1; while i > 5 do
    begin i := i + 1; p := p * i end;
```

25. В данном фрагменте программы

```
s := 0;
for i := 1 to 10 do
s := s + 2 * i
```

вычисляется:

а) сумма целых чисел от 1 до 10
б) сумма чётных чисел от 1 до 10
в) удвоенная сумма целых чисел от 1 до 10
г) сумма первых десяти натуральных чётных чисел

Глава 5
НАЧАЛА ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON

§ 5.1
Общие сведения о языке программирования Python

Ключевые слова:

- язык программирования
- программа
- среда программирования
- алфавит
- служебные слова
- типы данных
- структура программы
- оператор присваивания

Языки программирования — это формальные языки, предназначенные для записи алгоритмов, исполнителем которых будет компьютер. Записи алгоритмов на языках программирования называются **программами**.

Существует несколько тысяч языков программирования. Один из самых популярных современных языков программирования называется **Python** (произносится «па́йтон», хотя в России многие называют язык просто «питон»). Его разработал в 1991 году нидерландский программист Гвидо ван Россум. Язык Python непрерывно совершенствуется, и сейчас большинство программистов используют его третью версию — Python 3. Именно с этой версией будем работать и мы.

Python — язык программирования, предназначенный для решения самого широкого круга задач. С его помощью можно обрабатывать различные данные, проводить математические вычисления, создавать изображения, работать с базами данных, разрабатывать веб-сайты.

Записать программу на языке программирования можно ручкой на листке бумаги. Для того чтобы запустить программу на выполнение на компьютере, необходимо воспользоваться **системой (средой) программирования**, представляющей собой набор компьютерных инструментов, который включает редактор текста, транслятор, отладчик и другие составляющие.

Редактор текста — это программа для ввода, редактирования и форматирования текста программы на языке программирования.

Транслятор — инструмент, предназначенный для преобразования программ, написанных на языках программирования, в программы на машинном языке. Трансляторы делятся на два класса: *компиляторы* и *интерпретаторы*. Компилятор переводит весь исходный текст программы на машинный язык. Интерпретатор последовательно переводит на машинный язык и выполняет операторы исходного текста программы.

Отладчик — инструмент для поиска и исправления ошибок в программе. Основные функции отладчика: пошаговое выполнение программы с отображением результатов, остановка в заранее определённых точках, изменение значений переменных.

Вводить, просматривать, редактировать, запускать, отлаживать программы на Python удобно с помощью интегрированной среды разработки **IDLE** (произносится «айдл»). Она встроена в установщик Python, который можно скачать по ссылке **https://gotourl.ru/15550**.

5.1.1. Алфавит и словарь языка

Основой языка программирования Python, как и любого другого языка, является алфавит — набор допустимых символов, которые можно использовать для записи программы. Это:

- латинские прописные и строчные буквы (A, B, C, …, X, Y, Z, a, b, c, …, x, y, z);
- арабские цифры (0, 1, 2, …, 7, 8, 9);
- специальные символы (знак подчёркивания; круглые, квадратные скобки; знаки арифметических операций, # — знак начала однострочного комментария и др.).

В качестве неделимых элементов (составных символов) рассматриваются последовательности символов:

>= и <= (знаки $\geqslant$ и $\leqslant$);

== (знак =);

!= (знак $\neq$);

""" или ''' (утроенные двойные или одинарные кавычки, ставящиеся в начале и в конце многострочного комментария).

В языке существует также некоторое количество различных цепочек символов, рассматриваемых как единые смысловые элементы с фиксированным значением. Такие цепочки символов

называются **служебными словами.** В табл. 5.1 приведены основные служебные слова, которые мы будем использовать при записи программ на языке Python.

Таблица 5.1

Служебные слова языка Python

Служебное слово языка Python	Значение служебного слова
`and`	и
`break`	прервать
`elif`	иначе если
`else`	иначе
`False`	ложь
`float`	вещественный (с плавающей точкой)
`for`	для
`if`	если
`input`	ввод
`int`	целый
`list`	список
`not`	не
`or`	или
`print`	печать
`str`	строковый (цепочка символов)
`True`	истина
`while`	пока

Для обозначения переменных, программ и других объектов используются **имена** (идентификаторы) — любые отличные от служебных слов последовательности букв, цифр и символа подчёркивания, начинающиеся с буквы или символа подчёркивания.

Прописные и строчные буквы в именах различаются, например: `f` и `F` — два разных имени.

Длина имени может быть любой. Для удобства рекомендуется использовать имена, передающие смысл объектов и состоящие не более чем из 15 символов.

В программах на языке Python (начиная с версии 3) есть возможность использовать в именах буквы национальных алфавитов (от русских до китайских иероглифов). Но это считается очень плохим стилем, так делать не рекомендуется. Подумайте почему.

5.1.2. Типы данных, используемые в языке Python

В языке Python используются различные типы данных (табл. 5.2).

Таблица 5.2

Некоторые типы данных в языке Python

Название	Обозначение	Допустимые значения
Целочисленный	`int (integer)`	Сколько угодно большие, размер ограничен оперативной памятью
Вещественный	`float`	Любые числа с дробной частью
Строковый	`str (string)`	Любые символы из таблицы Unicode
Логический	`bool (boolean)`	`False` и `True`

В вещественном числе целая часть от дробной отделяется точкой (например, 5.7), при этом перед точкой и после неё должно быть по крайней мере по одной цифре. Пробелы внутри числа недопустимы.

Произвольный набор символов, заключённый в одинарные или двойные кавычки, считается строковой величиной (строкой). Строка может содержать любые символы, набираемые на клавиатуре, в том числе буквы национальных алфавитов.

Переменная в программировании — это поименованная область оперативной памяти, в которой могут храниться данные определённого типа.

В отличие от многих других языков программирования, переменные в языке Python не нужно объявлять. Тип переменной определяется автоматически в тот момент, когда ей присваивается новое значение.

Тип каждой переменной может динамически изменяться по ходу выполнения программы. Определить, какой тип имеет переменная в текущий момент, можно с помощью функции (команды) `type()`.

5.1.3. Режимы работы интерпретатора Python

Интерпретатор Python может работать в двух режимах:

- в командном (интерактивном) режиме, когда каждая введённая команда сразу выполняется;
- в программном (пакетном) режиме, когда программа сначала записывается в файл (обычно имеющий расширение `.py`) и при запуске выполняется целиком.

Изучение языков программирования принято начинать с вывода на экран надписи: «Привет, мир!». На языке Python соответствующая команда будет иметь вид:

```
print("Привет, мир!")
```

Для вывода на экран последовательности символов (текста, надписи) используется встроенная функция (команда) `print`. Последовательность символов, которые должны быть выведены на экран, заключается в двойные кавычки и записывается в круглых скобках. Вместо двойных кавычек можно использовать одинарные кавычки.

В начале строки (левее команды `print`) не должно быть пробелов — таково требование языка Python.

Запустите IDLE, откроется окно IDLE Shell, в котором символы >>> означают приглашение ввести команду. После ввода строки нажмите клавишу *Enter*. В следующей строке сразу отобразится результат, а далее — приглашение для ввода новой команды (рис. 5.1). В командном режиме также можно проводить вычисления. Для этого достаточно просто записать соответствующее выражение и нажать клавишу *Enter*.

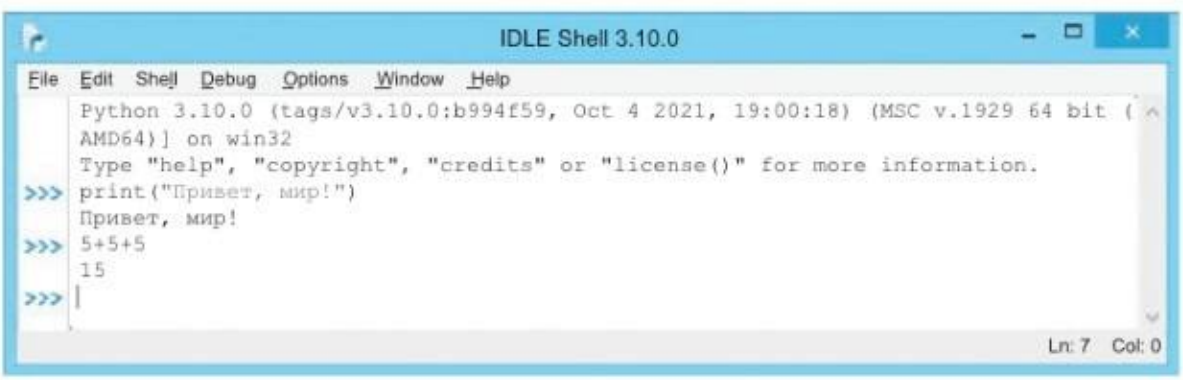


Рис. 5.1. Работа в интерактивном режиме

Для перехода в программный режим в меню **File** выбираем пункт **NewFile**. В открывшемся окне набираем текст програм-

мы, а затем сохраняем его под каким-нибудь именем (например, test.py), выбрав пункт меню **File → SaveAs**. Запустить программу на выполнение можно, выбрав пункт меню **Run → RunModule** или нажав клавишу *F5*.

В сети Интернет существуют ресурсы для запуска и отладки программ на языке Python в режиме online. Вот ссылки на некоторые из них:

http://gotourl.ru/11286
http://gotourl.ru/11287
http://gotourl.ru/11288

5.1.4. Оператор присваивания

Программа на языке программирования представляет собой последовательность операторов (инструкций, команд).

Оператор — языковая конструкция, задающая полное описание действия, которое необходимо выполнить.

Оператор присваивания выполняет запись значения в переменную.

Общий вид оператора присваивания:

```
<имя переменной> = <значение или вычисляемое выражение>
```

Операция присваивания допустима для всех приведённых в табл. 5.2 типов данных. Выражения в языке Python конструируются по рассмотренным ранее правилам для Школьного алгоритмического языка.

Пример

```
a = 10
b = 5
s = a + b
c = "Привет!"
d = 1.4 + 5.7 * a
e = a < d
f = "Мир! " + c
g = x * (a + c) / 3
```

При выполнении оператора a = 10 в ячейку оперативной памяти компьютера с именем a (в переменную a) заносится значе-

ние 10; при выполнении оператора b = 5 в ячейку оперативной памяти компьютера с именем b заносится значение 5. При выполнении оператора s = a + b значения ячеек оперативной памяти с именами a и b переносятся в процессор, где над ними выполняется операция сложения. Полученный результат заносится в ячейку оперативной памяти с именем s (рис. 5.2).

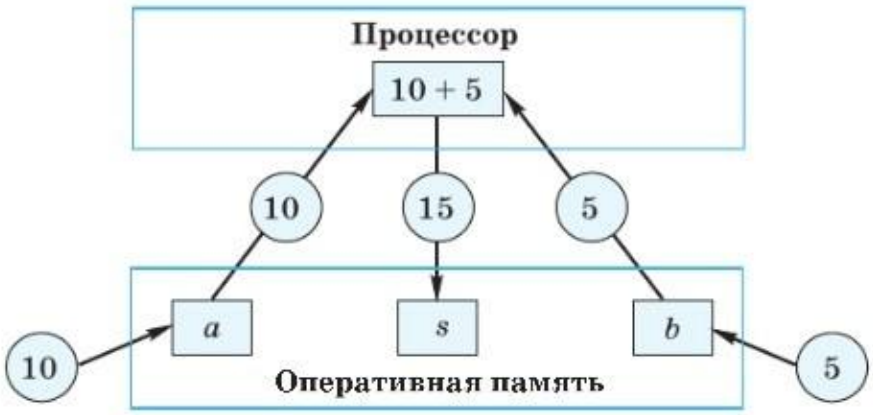


Рис. 5.2. Процесс выполнения оператора присваивания

В правой части оператора присваивания нельзя указывать переменные, которые не были заранее созданы (определены). Так, для переменных d и c из рассмотренного примера все входящие в соответствующие выражения переменные были заданы выше. Последняя строка ошибочна, так как переменная x из правой части ранее не была создана (определена).

В языке Python разрешено множественное присваивание. Запись

```
a, b = 19, 25
```

равносильна паре операторов присваивания:

```
a = 19
b = 25
```

При этом считается, что эти два действия происходят параллельно, т. е. одновременно. Если двум переменным присваивается одно и то же значение, можно применить множественное присваивание «по цепочке»:

```
a = b = 5
```

Эта запись равносильна паре операторов:

```
b = 5
a = 5
```

Для основных арифметических операций в языке Python используются те же обозначения, что и в Школьном алгоритмическом языке:

`+` — сложение;
`-` — вычитание;
`*` — умножение;
`/` — деление;
`**` — возведение в степень.

В языке Python можно использовать сокращённую запись арифметических операций.

Сокращённая запись	Полная запись
`a += b`	`a = a + b`
`a -= b`	`a = a - b`
`a *= b`	`a = a * b`
`a /= b`	`a = a / b`
`a **= 2`	`a = a ** 2`

САМОЕ ГЛАВНОЕ

Python — один из самых популярных современных языков программирования. Это язык программирования высокого уровня, предназначенный для самого широкого круга задач.

В языке Python можно работать в двух режимах:
- в командном (интерактивном) режиме, когда каждая введённая команда сразу выполняется;
- в программном режиме, когда программа сначала записывается в файл и при запуске выполняется целиком.

В языке Python используются различные типы данных: целочисленный (`int`), вещественный (`float`), строковый (`str`), логический (`bool`) и другие.

Переменные в языке Python объявлять не нужно; тип переменной автоматически определяется в тот момент, когда ей присваивается новое значение.

Для обозначения переменных, программ и других объектов используются имена (идентификаторы) — любые отличные от служебных слов последовательности букв, цифр и символа подчёркивания, начинающиеся с буквы или символа подчёркивания.

В программах на языке Python есть возможность использовать в именах буквы национальных алфавитов, но это считается очень плохим стилем, и делать так не рекомендуется.

Вопросы и задания

1. Почему язык программирования Python считается универсальным?

2. Что входит в состав алфавита языка Python?

3. Перед вами слова, которые встречаются во многих программах на языке Python. Как эти слова можно перевести на русский язык?

1) `integer`	5) `break`
2) `float`	6) `while`
3) `input`	7) `else`
4) `print`	8) `string`

4. Каких правил следует придерживаться при выборе имён для различных объектов в языке Python?

5. Отнесите каждую из последовательностей символов к одной из трёх групп:
   1 — рекомендуемые имена переменных в языке Python;
   2 — допустимые имена переменных в языке Python;
   3 — недопустимые имена переменных в языке Python.

а) `1z`	е) `ELSE`	л) `n3`
б) `_1z`	ж) `sUMMA`	м) `3n`
в) `#A`	з) `Summa`	н) `n 3`
г) `ФУ`	и) `дата`	о) `n+3`
д) `z-1`	к) `1ФУ`	п) `_1_4_5_aAb12_as555`

6. Установите соответствие между названиями типов данных и их обозначениями.

а) Целочисленный	1) `str`
б) Вещественный	2) `bool`
в) Строковый	3) `int`
г) Логический	4) `float`

7. В чём разница между числами 100 и 100.0 в языке Python?

8. Охарактеризуйте режимы работы интерпретатора Python:
   1) командный;
   2) программный.

9. В командном режиме введите последовательно строки:

```
a = 10
type(a)
a = '10 10'
type(a)
a = False
type(a)
a = 12.0
type(a)
```

Сделайте вывод о том, как изменялся тип переменной a.

10. Какая ошибка допущена в следующей программе?

```
a = 3
b = 4
s = a * b * d
print(s)
```

11. Какое значение будет присвоено переменной c в результате выполнения программы?

```
a, b = 11, 63
c = b = 55
d = b + c - a
```

12. Чему будет равно значение переменной c после выполнения программы?

а)
```
a = b = 3
a += 1
c = a + b
```

б)
```
a = b = 5
a += b
c = 2 * a - b
```

в)
```
a = b = 1
a *= 10
c = a / (2 * b)
```

г)
```
a, b = 3, 5
b += 2
c = a + b
```

д)
```
a, b = 5, 3
b += a
c = 10 * b / a
```

е)
```
b, a = 5, 2
b **= a
c = b / a * 4
```

13. Чему будут равны значения переменных a и b после выполнения программы при указанных начальных значениях? Какими будут типы переменных a и b?

```
a += 1
b += a
a *= b
b /= 5
a -= a
```

а) a = 4 и b = 0 **б)** a = 0 и b = 0

14. Запишите оператор для:

а) вычисления среднего арифметического `sred` переменных `x1` и `x2`;
б) уменьшения на единицу значения переменной `k`;
в) увеличения на единицу значения переменной `i`;
г) вычисления стоимости покупки `summa`, состоящей из нескольких тетрадей, нескольких ручек и нескольких карандашей.

§ 5.2 Организация ввода и вывода данных

Ключевые слова:

- оператор вывода `print()`
- формат вывода
- оператор ввода `input()`

5.2.1. Вывод данных

В предыдущем параграфе мы познакомились с типами данных, рассмотрели оператор присваивания. Этого достаточно для того, чтобы записать программу преобразования данных. Но результат этих преобразований нам виден не будет.

Для вывода данных из оперативной памяти на экран компьютера используется **оператор (функция) вывода `print()`:**

`print(<выражение 1>, <выражение 2>,…, <выражение N>)`

список вывода

Здесь в круглых скобках помещается список вывода — список выражений, значения которых выводятся на экран. Это могут быть числовые, символьные и логические выражения, в том числе константы и переменные.

Пример 1

Оператор `print ('s=', s)` выполняется так:
1) на экран выводятся символы, заключённые в одинарные кавычки: `s=`
2) на экран выводится значение переменной с именем `s`.

Если значение переменной `s` равно `15` и она имеет целочисленный тип, то на экране появится[1]: `s=␣15`

[1] Здесь и далее ␣ обозначает пробел.

Обратите внимание: оператор `print` вставляет между выводимыми значениями так называемый разделитель (сепаратор, от англ. *separator*). По умолчанию выводимые выражения разделяются одним пробелом, иначе говоря, разделителем между ними является пробел. Чтобы изменить разделитель, необходимо указать его новое значение после слова `sep`.

Пример 2

Разные варианты разделителей в списке вывода:

Вариант организации вывода	Оператор (функция) вывода	Результат
По умолчанию	`print(1,20,300)`	`1␣20␣300`
Убрать разделители-пробелы	`print(1,20,300,sep='')`	`120300`
Добавить разделитель-запятую	`print(1,20,300,sep=',␣')`	`1,␣20,␣300`

Вывод каждого значения с новой строки	`print(1,20,300,sep='\n')`	`1`
		`20`
		`30`

Предположим, что мы работаем с натуральными числами, каждое из которых меньше 100. Тогда на одно число на экране достаточно выделить 3 позиции: две позиции на запись самого числа и ещё одну позицию на пробел слева, разделяющий числа. Записывается это так:

```
print("{:3}{:3}{:3}".format(a, b, c))
```

Это **форматный вывод**: строка для вывода строится с помощью функции `format`. Аргументы этой функции (`a`, `b`, `c`) — это те величины, которые выводятся; они указываются в круглых скобках.

Символьная строка слева от точки — это форматная строка, которая определяет формат вывода, т. е. то, как именно величины будут представлены на экране. Фигурные скобки обозначают место для вывода очередного элемента: на первом месте выводится значение `a`, на втором — значение `b`, на третьем — `c`. Число после двоеточия — количество позиций, которые отводятся на выводимую величину. Если цифр в выводимом числе меньше, чем зарезервированных под него позиций на экране, то свободные

позиции дополняются пробелами слева от числа. Если указанное в формате вывода число меньше, чем необходимо, то оно автоматически будет увеличено до минимально необходимого.

Например, числа 12, 5 и 15 будут выведены так:

□12□□5□15

Числа 12, 5 и 1500 будут выведены следующим образом:

□12□□51500

Для вывода вещественного числа в списке вывода для каждого выражения указываются два параметра:

1) общее количество позиций, отводимых под число;
2) количество позиций в дробной части числа:

f — вещественные (float);
e — экспоненциальный формат.

Фрагмент программы	Результат выполнения оператора вывода
a = 1 / 3; b = 1 / 9 print ("{:7.3f}{:7.4f}".format(a, b))	□□0.333□0.1111
a = 1/3 print ("{:10.3e}".format(a))	□3.333e-01

При выполнении очередного оператора print() по умолчанию вывод продолжается в новой строке. Чтобы убрать переход к новой строке, используется параметр end:

```
print(a, end="")    # убран переход на новую строку
print(b)
```

5.2.2. Первая программа на языке Python

Пользуясь рассмотренными операторами, составим программу, вычисляющую длину окружности и площадь круга радиусом 5,4 см.

Исходным данным в этой задаче является радиус: r = 5,4 см. Результатом работы программы должны быть величины c и s. c — длина окружности, s — площадь круга; c, s и r — величины вещественного типа.

Исходные данные и результаты связаны соотношениями, известными из курса математики: $c = 2\pi r$, $s = \pi r^2$. Программа, реализующая вычисления по этим формулам, будет иметь вид:

```
r = 5.4
c = 2 * 3.14 * r
s = 3.14 * r ** 2
print ('c=', c)
print ('s=', s)
```

Эта программа верна и решает поставленную задачу. Запустив её на выполнение, вы получите следующий результат (рис. 5.3).

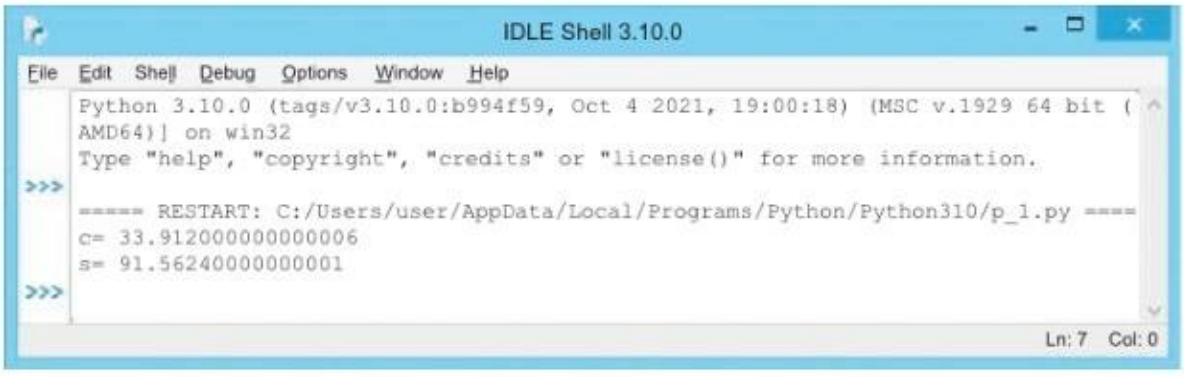


Рис. 5.3. Результат выполнения первой программы

Улучшим внешний вид результата, использовав форматный вывод (рис. 5.4).

```
print("c=", "{:7.4f}".format (c))
print("s=", "{:7.4f}".format (s))
```

Рис. 5.4. Форматный вывод результата

После ввода текста программы и её запуска что-то может пойти не так, и вы окажетесь в одной из двух ситуаций:

1) в окне IDLE появилась строка красного цвета с сообщением об ошибке — это **синтаксическая ошибка**: возможно, вы пропустили или написали не тот символ, использовали имя переменной, которая ещё не определена, и т. д.;
2) программа выполнилась, но получен другой ответ — это **логическая ошибка**: возможно, вы ошиблись при записи арифметических выражений.

Исправьте ошибку и повторно запустите программу на выполнение. Такой процесс называется **отладкой программы**. Он будет завершён в тот момент, когда вы получите требуемый результат.

И всё-таки составленная нами программа имеет существенный недостаток: она находит длину окружности и площадь круга для единственного значения радиуса (5,4 см). Для того чтобы вычислить длину окружности и площадь круга для другого значения радиуса, потребуется вносить изменения непосредственно в текст программы, а именно изменять оператор присваивания. Внесение изменений в существующую программу по меньшей мере не всегда удобно (например, когда программа большая и операторов присваивания много). Ниже вы познакомитесь с оператором, позволяющим вводить исходные данные в процессе работы программы, не прибегая к изменению текста программы.

5.2.3. Ввод данных с клавиатуры

Для ввода в оперативную память значений переменных используется **оператор (функция) ввода `input`** (от англ. *input* — ввод):

```
a = input()
```

Пара скобок говорит о том, что мы вызываем функцию. Их надо писать обязательно, даже если в скобках ничего нет.

При выполнении этой команды программа ожидает от пользователя ввода последовательности символов с клавиатуры; после того как пользователь нажмёт клавишу *Enter*, набранная им символьная строка запишется в переменную с именем `a`. Это значит, что в памяти выделяется область необходимого размера, с ней связывается имя `a` и в этой области сохраняются все полученные символы.

Если мы планируем работать не со строками, а с числами, то сразу же после считывания строки необходимо выполнить преобразование типов при помощи соответствующей функции:

`a = int(a)` — для целых чисел;
`a = float(a)` — для вещественных чисел.

Считывание строк и преобразование типов рекомендуется объединять:

`a = int(input())` — для целых чисел;
`a = float(input())` — для вещественных чисел.

Экспериментально убедитесь в истинности утверждения: «Функции `int()` и `float()` работают без ошибок, если введённая строка состоит только из цифр».

Можно совмещать вывод подсказки и ввод данных, указывая текст подсказки в скобках как аргумент функции input:

```
r = float(input('Введите радиус '))
```

Каждый оператор ввода input() захватывает только одну строку данных, причём захватывает её целиком. Для того чтобы ввести в одной строке два целых числа, разделённых пробелом (например, 10 20), используют функцию split (от англ. *split* — расщепить). Можно воспользоваться последовательностью команд:

`a, b = input().split()`	Ввод двух строковых величин, разделённых пробелом
`a, b = int(a), int(b)`	Преобразование к целому типу

Теперь рассмотрим ситуацию, когда входные данные заданы в одной строке, но разделены особыми разделителями, отличными от пробела. Типичным примером таких входных данных являются показания времени (10:33).

В таких случаях надо для split() указывать конкретный символ разделителя, взятый в двойные (или одинарные) кавычки. В нашем примере:

```
hours, minutes = input().split(':')
```

Аналогично организуется считывание трёх и более переменных:

```
a, b, c = input().split()
```

Для преобразования к целому типу переменных a, b, c можно использовать конструкцию

```
a, b, c = int(a), int(b), int(c)
```

Сократить запись считывания и преобразования нескольких считанных значений в числовой тип можно с помощью функции map(), которая применяет к каждому элементу списка заданное правило:

```
a, b, c = map(int, input().split())
```

Здесь с помощью функции map() организовано применение функции int() к каждому элементу вводимого списка.

Усовершенствуем первую программу, организовав в ней ввод данных с помощью оператора input() и включив в него строку с приглашением для ввода.

```
r = float(input('Введите радиус:'))
c = 2 * 3.14 * r
s = 3.14 * r ** 2
print("c=", "{:7.4f}".format(c))
print("s=", "{:7.4f}".format(s))
```

Результат работы усовершенствованной программы представлен на рис. 5.5.

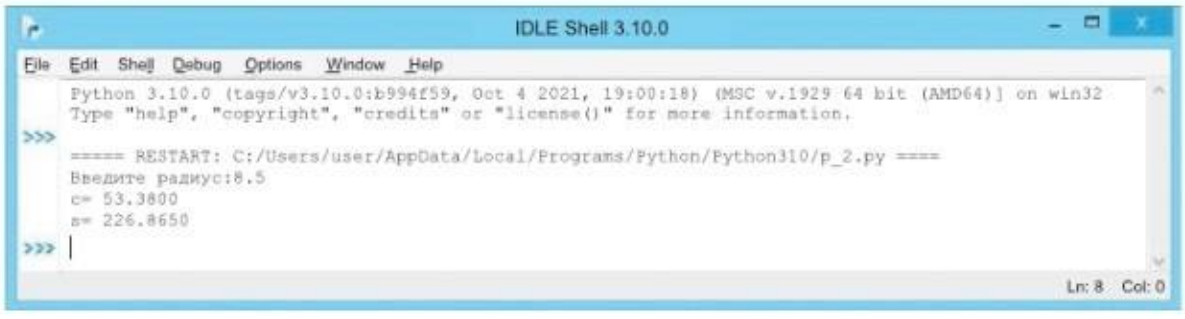


Рис. 5.5. Результат выполнения усовершенствованной программы

Теперь наша программа может вычислять длину окружности и площадь круга для любого значения `r`. Иначе говоря, она решает не единичную задачу, а целый класс задач. Кроме того, в программе понятно и удобно организован ввод исходных данных и вывод получаемых результатов. Это обеспечивает дружественность пользовательского интерфейса.

САМОЕ ГЛАВНОЕ

Оператор ввода (функция) `input()` вводит с клавиатуры символьную строку. Для преобразования строки в целое число её обрабатывают функцией `int()`, для перевода в вещественное — функцией `float()`.

Сократить запись считывания и преобразования нескольких считанных значений в числовой тип можно с помощью функции `map()`, которая применяет заданное правило к каждому вводимому элементу.

Для вывода данных из оперативной памяти на экран монитора используется оператор вывода (функция) `print()`. Элементы списка вывода разделяются запятыми. По умолчанию при выводе данные разделяются пробелами; после вывода всех данных функция `print()` переводит курсор в начало следующей строки.

Формат вывода — это указание количества знако-мест на экране, которые должна занимать выводимая величина. Форматный вывод данных выполняется с помощью функции `format()`.

Ввод исходных данных и вывод результатов должны быть организованы понятно и удобно; это обеспечивает дружественность пользовательского интерфейса.

Вопросы и задания

1. Что является результатом выполнения следующего оператора?
   а) `print(a)`
   б) `print(' a')`
   в) `print('a=', a)`

2. Напишите программу, выводящую на экран следующее забавное изображение:

```
 (\_/)
(='.'=)
(")_(")
```

3. Какой тип имеет переменная `f`, если после выполнения оператора `print(f)` на экран было выведено следующее число?
   а) `125`
   б) `125.0`

4. Дан фрагмент программы:

```
a = 10; b = a + 1; a = b - a; print(a, b)
```

   Какие числа будут выведены на экран компьютера?

5. Для каждого оператора `print` укажите соответствующий ему результат работы.

а) `print(10, 20, 30)`	1) `102030`
б) `print(10, 20, 30, sep='')`	2) `10, 20, 30`
в) `print(10, 20, 30, sep=',')`	3) `10:20:30`
г) `print(10, 20, 30, sep=':')`	4) `10 20 30`
д) `print(10, 20, 30, sep=',')`	5) `10,20,30`

6. Что будет выведено в результате работы следующей программы?

```
a = 1; b = 2; c = 3
print("{:3}".format(a))
print("{:2}{:1}{}".format(b, b, b))
print("{}{}{}{}{}".format(c, c, c, c, c))
print("{:2}{:1}{}".format(b, b, b))
print("{:3}".format(a))
```

7. Внесите изменения в программу из предыдущего задания так, чтобы в результате её выполнения выводились следующие изображения:

а)
```
  1
 2 2
3   3
 2 2
  1
```

б)
```
  1
 212
31313
 212
  1
```

в)
```
  5
 555
55555
 555
  5
```

8. Что будет выведено в результате работы следующей программы?

```
x = 143.511
print(x)
print("{:8.2f}".format(x))
print("{:.6f}".format(x))
print("{:10.3e}".format(x))
print("{:12.3e}".format(x))
```

9. Определите результат работы программы, если переменным a и b были присвоены значения 2 и 4 соответственно.

```
a = int(input())
b = int(input())
a = a * a
b **= 2
k = a * b
k *= 2
k += a + b
print(k)
```

10. Целочисленным переменным i, j, k нужно присвоить соответственно значения 10, 20 и 30. Запишите оператор ввода, соответствующий входной строке:

а) 20 10 30

б) 30 20 10

в) 10 30 20

11. Найдите ошибку в программе, которая должна вывести сумму двух введённых чисел.

```
a = input()
b = input()
summa = a + b
print(summa)
```

Проверьте правильность своего решения, выполнив программу на компьютере.

12. С клавиатуры вводятся два целых числа в строку через пробел. Выберите фрагмент программы, в котором переменным a и b будут присвоены соответствующие целочисленные значения.

1)
```
a, b = map(int(input()).split())
```
2)
```
a, b = int(input()).map(split())
```
3)
```
a = int(input())
b = int(input())
```
4)
```
a, b = map(split().int(input()))
```
5)
```
a, b = map(int(input()).int(input()))
```
6)
```
a, b = map(int, input().split())
```
7)
```
a, b = int(map(input().split()))
```
8)
```
a, b = map(int, input(),split())
```
9)
```
a, b = map(int. input().split())
```
10)
```
a, b = map(int.input(),split())
```

13. Запишите оператор, обеспечивающий ввод с клавиатуры необходимых исходных данных для вычисления дискриминанта квадратного уравнения по трём целочисленным значениям его коэффициентов.

14. Дан фрагмент программы:

```
a = input(); b = input(); d = input()
a = float(a)
b = float(b)
d = float(d)
c = a + b; print(a, b, c, end=""); print(d)
```

Упростите его, сократив число операторов.

15. Напишите программу, которая вычисляет площадь и периметр прямоугольника по длинам двух его сторон.

§ 5.3 Программирование линейных алгоритмов

Ключевые слова:

- вещественный тип данных
- целочисленный тип данных
- строковый тип данных
- логический тип данных

Программы, реализующие линейные алгоритмы, считаются наиболее простыми. Все имеющиеся в них операторы выполняются последовательно, один за другим.

Программируя линейные алгоритмы, рассмотрим более подробно вещественные, целочисленные, строковые и логические типы данных, а также познакомимся с графическими возможностями языка Python.

5.3.1. Числовые типы данных

Вы уже знакомы с числовыми типами данных `int` и `float`. К ним применимы многочисленные функции, малая часть которых приведена в табл. 5.3.

Таблица 5.3

Стандартные функции языка Python

Функция	Назначение	Тип аргумента	Тип результата
`abs(x)`	Абсолютная величина (модуль) числа x	`int, float`	Такой же, как у аргумента
`round(x)`	Округление вещественного x до заданного количества знаков после запятой (по умолчанию до нуля знаков, т. е. до ближайшего целого)	`float`	`int, float`
`sqrt(x)`	Квадратный корень из x	`int, float`	`float`
`sin(x)`	Синус угла x, заданного в радианах	`int, float`	`float`
`random()`	Случайное число от 0 до 1	–	`float`
`randint(a, b)`	Случайное целое число n, $a \leqslant n \leqslant b$	`int`	`int`

Первые три из представленных в табл. 5.3 функций встроены в язык Python; чтобы их вызвать, не надо выполнять никаких дополнительных действий.

Например, программа ввода вещественного числа и вывода его абсолютной величины может выглядеть так:

```
x = float(input())
print(abs(x))
```

Что касается функций `sqrt(x)` и `sin(x)`, то для их вызова предварительно надо подключить модуль `math`, в котором собраны математические функции; две последние из приведённых в табл. 5.3 функций требуют подключения модуля `random`.

На языке Python написано так много самых разных функций, что встраивать весь этот объём кода в сам язык нецелесообразно. Проблема доступа к дополнительным возможностям языка, обеспечиваемым этими функциями, решается с помощью модулей. Каждый модуль содержит набор функций, предназначенных для решения задач из определённой области. Так, модуль `graph` нужен для рисования геометрических фигур, модуль `random` позволяет генерировать «случайные» числа и т. д. Для доступа к функциям модуля его надо импортировать в программу. После импорта интерпретатор «знает» о существовании дополнительных функций и позволяет ими пользоваться.

Подключение модуля осуществляется командой **import**. Например, команда

```
from math import *
```

подключит к программе все функции (знак *) модуля `math`. Теперь к ним можно обращаться так же, как к встроенным функциям:

```
y = sqrt(x)
z = sin(x)
```

Для того чтобы записать в переменную `a` случайное число в диапазоне от 1 до 10, можно использовать следующие операторы:

`from random import randint`	Подключаем функцию `randint` модуля `random`
`a = randint(1, 10)`	Обращаемся к функции `randint` как к встроенной

Пример 1

Исследуем работу функций `round` и `int`, применив их к некоторому вещественному `x`. Соответствующая программа будет иметь вид

```
print('Исследование функций round, int ')
x = float(input (' Введите x>>'))
print('Округление - ', round(x))
print('Целая часть - ', int(x))
```

Запустите программу несколько раз для каждого $x = \{10{,}2;\ 10{,}8;\ -10{,}2;\ -10{,}8\}$. Что вы можете сказать о типе результата каждой из этих функций?

Попытайтесь пояснить следующие результаты работы функции `round`:

Функция с входными данными	Результат
`round(1.5)`	2
`round(2.5)`	2
`round(2.65, 1)`	2.6
`round(2.75, 1)`	2.8

Обсудите этот вопрос в группе.

5.3.2. Целочисленный тип данных

Над целыми числами в языке Python выполняются следующие операции: сложение (+); вычитание (–); умножение (*); целочисленное деление, или получение неполного частного (//); взятие остатка, или получение остатка от деления (%); деление (/); возведение в степень (**).

Результаты первых пяти операций — целые числа. Результатом операции деления может быть вещественное число.

Пример 2

Рассмотрим пример использования операций // и %, записав на языке Python программу нахождения суммы цифр вводимого с клавиатуры натурального трёхзначного числа.

Используем тот факт, что положительное трёхзначное число можно представить в виде следующей суммы: $x = a \cdot 100 + b \cdot 10 + c$, где a, b, c — цифры числа.

```
print('Нахождение суммы цифр трёхзначного числа')
x = int(input('Введите исходное число>>'))
a = x // 100
b = x % 100 // 10
c = x % 10
s = a + b + c
print('s= ', s)
```

Чему равна сумма цифр числа 123? А числа –123? Совпадают ли ваши результаты с результатами работы программы? Как можно объяснить и исправить ошибку в программе?
Обсудите этот вопрос в группе.

5.3.3. Строковый тип данных

Значением строковой величины (тип `str`) является произвольная последовательность символов, заключённая в одинарные (двойные) кавычки. Символьная строка рассматривается как единый объект.

Символом в языке Python является любой из символов, который можно получить на экране нажатием на клавиатуре одной из клавиш или комбинации клавиш, а также некоторые другие символы, в том числе и невидимые.

В тексте программы переменную строкового типа можно задать, заключив цепочку символов в одинарные или двойные кавычки:

```
d = '5'
c = 'Book'
c1 = "1*"
```

Новое значение может быть записано в строку с помощью оператора ввода с клавиатуры:

```
s = input()
```

Если значение символьной переменной считывается с клавиатуры, то его следует набирать без апострофов.

Можно проверить равенство (совпадение) строк (`d == c`) или выяснить, какая из двух строк меньше (при этом используется поочерёдное сравнение кодов пар символов, образующих слова; меньшим будет то слово, у которого код символа окажется меньше).

В Python строки можно сцеплять: `a + b` (к концу строки `a` прикрепляется, или «приписывается», строка `b`). Пусть

```
d = c1 + d + c1
```

Тогда в переменную d из нашего примера будет записана следующая строка: '1*51*'.

В результате операции a * k, где k — целое число, строка a повторяется k раз. Так, в результате выполнения команды print d * 3 будет выведена строка

```
1*51*1*51*1*51*
```

В следующем фрагменте программы запрашивается имя пользователя и выводится приветствие:

```
print('Как тебя зовут?')
name = input()
print('Привет,', name)
```

Рассмотрим некоторые функции, применимые к строковым величинам (табл. 5.4).

Таблица 5.4

Функции обработки строковых величин

Функция	Назначение	Тип аргумента	Тип результата
ord(s)	Код, соответствующий символу s в кодировке ASCII	string	integer
chr(x)	Символ, соответствующий коду n в кодировке ASCII	integer	string
len(s)	Длина строки s	string	integer
s[m:n]	Извлечение из строки s среза — последовательности символов, начиная с символа, имеющего номер m, до символа с номером n; символ с номером n в срез не входит	s — string m — integer n — integer	string

Пример 3

Запишем программу, в которой для введённого с клавиатуры символа на экран выводится его код. Затем на экран выводится строка, представляющая собой последовательность из трёх символов: символа, предшествующего исходному в кодировке ASCII; исходного символа; символа, следующего за исходным.

```
a = input()
kod = ord(a)
print(kod)
b = chr(kod - 1) + a + chr(kod + 1)
print(b)
```

Чтобы обратиться в программе к конкретному символу строки, надо указать имя строковой переменной и порядковый номер (индекс) символа в этой строке. Так, запись `s[0]` обозначает первый символ строки `s`.

Для выделения части строки используется оператор извлечения среза `s[m:k]`; здесь `s` — строка; `m` — начало среза, а `k` — окончание; символ с номером `k` в срез не входит. По умолчанию первый индекс равен `0`, а второй — длине строки.

Пример 4

Запишем программу, которая из двух строк формирует третью строку и определяет её длину.

```
a = 'информация'
b = 'автоматика'
c = a[0:5] + b[4:10]
print(c)
n = len(c)
print ('n =', n)
```

5.3.4. Логический тип данных

Как известно, величины логического типа принимают всего два значения; в Python это `False` и `True`. Эти константы определены так, что `False < True`. Логические значения получаются в результате выполнения операций сравнения числовых, строковых и логических выражений. Поэтому в Python логической переменной можно присваивать результат операции сравнения.

Пример 5

Напишем программу, определяющую истинность высказывания «Число `N` является чётным» для произвольного целого числа `N`.

Пусть `ans` — логическая переменная, а `N` — целая переменная. Тогда в результате выполнения оператора присваивания

```
ans = N % 2 == 0
```

переменной `ans` будет присвоено значение `True` при любом чётном `N` и `False` в противном случае.

```
print('Высказывание о чётности числа')
N = int(input('Введите исходное число>>'))
ans = N % 2 == 0
print('Число', N, 'является чётным -', ans)
```

Логическим переменным можно присваивать значения логических выражений, построенных с помощью известных вам логических операций И, ИЛИ, НЕ, которые в Python обозначаются соответственно **and**, **or**, **not**.

Пример 6

Напишем программу, определяющую истинность высказывания «Существует треугольник с длинами сторон a, b, c» для произвольных целых чисел a, b, c.

```
print('Высказывание о существовании треугольника')
a = int(input('Введите значение a>>'))
b = int(input('Введите значение b>>'))
c = int(input('Введите значение c>>'))
ans = (a < b + c) or (b < a + c) or (c < a + b)
print('Существует треугольник со сторонами',
      a, ', ', b, ', ', c,' - ', ans)
```

5.3.5. Графические примитивы

Чтобы строить изображения из графических примитивов (отрезков, прямоугольников, окружностей), нам потребуется модуль graph.

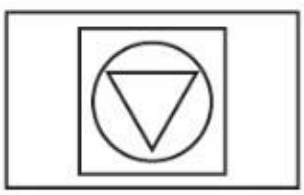
Положение фигур в графическом окне задаётся координатами — порядковыми номерами пикселей по горизонтали и по вертикали. Отсчёт значений координаты x происходит слева направо, координаты y — сверху вниз. Будьте внимательны: такая система координат отличается от той, которую вы используете на уроках математики.

Пример 7

С помощью следующей программы будет построено приведённое ниже графическое изображение.

```
from  graph import *
windowSize(640, 480)
canvasSize(640, 480)
rectangle(50, 50, 550, 250)
line (100, 200, 150, 100)
line (150, 100, 200, 200)
line (200, 200, 100, 200)
circle(300, 150, 50)
rectangle(400, 100, 500, 200)
```

Воспроизведите программу в среде IDLE. Обратите внимание: первые две команды задают размеры графического окна и «холста» для рисования. Исследуйте команды, с помощью которых построены графические примитивы. Видоизмените программу так, чтобы получилось изображение:



Более полно возможности модуля `graph` представлены в приложении 3.

САМОЕ ГЛАВНОЕ

В языке Python используются вещественный, целочисленный, строковый, логический и другие типы данных. Для них определены соответствующие операции и функции.

На языке Python написано много самых разных функций, для использования многих из них необходимо подключать специальные модули.

В языке Python реализованы операции целочисленного деления: для деления нацело используется оператор `//`, для взятия остатка от деления — оператор `%`.

Символьная строка — это последовательность символов, рассматриваемая как единый объект. Длина строки — это количество символов в строке. Знак + при работе со строками означает их сцепление в одну строку; знак * — многократное сложение строк.

Логическим переменным можно присваивать значения логических выражений, построенных с помощью логических операций **and**, **or**, **not**.

Вопросы и задания

1. Определите результат работы программы. Запишите математическую формулу для вычисления значения s.

```
from math import *
a = 12
b = 5
c = 13
p =(a + b + c) / 2
s = p
s *= p - a
s *= p - b
s *= p - c
s = sqrt(s)
print(s)
```

2. Разработайте и отладьте программу, вычисляющую y для заданного x по формуле

$$y = x^3 + 2{,}5x^2 - x + 1.$$

При этом:

а) операцию возведения в степень использовать запрещено;

б) в одном операторе присваивания можно использовать не более одной арифметической операции (сложение, умножение, вычитание);

в) в программе может быть использовано не более пяти операторов присваивания.

Подсказка: преобразуйте выражение к следующему виду: $y = ((x + 2{,}5)x - 1)x + 1$.

3. Разработайте и отладьте программу, вычисляющую длину отрезка AB по заданным координатам точек A и B.

Подсказка: расстояние d между точками $A(x_a, y_a)$ и $B(x_b, y_b)$ выражается формулой

$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}\ .$$

Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
xa=2 ya=1 xb=10 yb=7	|AB|=10.0

4. Известны длины сторон треугольника a, b, c. Разработайте и отладьте программу, вычисляющую площадь этого треугольника. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
a=3 b=4 c=5	s=6.0

5. Известны координаты вершин A, B, C треугольника. Разработайте и отладьте программу, вычисляющую площадь этого треугольника. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
xa=2 ya=1 xb=6 yb=5 xc=10 yc=1	s=16.0

6. Если сумма налога исчисляется в рублях и копейках, то налоговая служба округляет её до ближайшего рубля (до 50 копеек — с недостатком, свыше 50 копеек (включая 50) — с избытком). Напишите программу для ввода точной суммы налога и вывода суммы, которую следует уплатить.
7. В модуле `random` есть функция `randint(a, b)`, генерирующая случайное целое число из отрезка [a; b]. Соответствующая программа имеет вид:

```
print('Функция randint')
from random import randint
a = int(input ('Введите a >>'))
b = int(input ('Введите b>>'))
print('случайное целое число из отрезка
      [', a, '; ', b, ']=', randint(a, b))
```

Какие изменения следует внести в программу, чтобы получить случайное целое число из полуинтервала (a; b]? Как можно получить случайное целое число из интервала (a; b)?

8. Одна компания выпустила лотерейные билеты трёх разрядов: для молодёжи, для взрослых и для пенсионеров. Номера билетов каждого разряда лежат в пределах:
   для молодёжи — от 1 до 100;
   для взрослых — от 101 до 200;
   для пенсионеров — от 201 до 250.
   Напишите программу для выбора лотерейного билета каждого типа случайным образом.

9. Напишите на языке Python программу, которая для произвольного натурального двузначного числа определяет:
   а) сумму и произведение его цифр;
   б) число, образованное перестановкой цифр исходного числа.

10. Запишите на языке Python программу, реализующую алгоритм работы кассира, выдающего покупателю сдачу (s) наименьшим возможным количеством банкнот по 500 (k500), 200 (k200), 100 (k100) и 50 (k50) рублей. Предусмотрите вывод сообщения о том, что часть сдачи, которую невозможно выдать купюрами, будет выдана монетами. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
845	Следует сдать: банкнот по 500 руб. - 1 шт. банкнот по 200 руб. - 1 шт. банкнот по 100 руб. - 1 шт. банкнот по 50 руб. - 0 шт. монетами - 45 руб.

11. Идёт k-я секунда суток. Разработайте и отладьте программу, которая по введённой k-й секунде суток определяет, сколько целых часов h и целых минут m прошло с начала суток. Например, если k = 13 257 = 3 · 3600 + 40 · 60 + 57, то h = 3 и m = 40. Выведите на экран фразу:
`It is ... hours ... minutes`
Вместо многоточий программа должна выводить значения h и m, отделяя их от слов ровно одним пробелом. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
13257	It is 3 hours 40 minutes

12. Разработайте и отладьте программу, которая вычисляет сумму кодов букв в слове БАЙТ.

13. Разработайте и отладьте программу, которая формирует и выводит на экран строку символов, коды которых равны 66, 69, 71, 73, 78.

14. Напишите и отладьте программу, которая запрашивает три строковые величины — взаимосвязанные прилагательное, существительное и глагол, а затем выводит все варианты фраз с использованием введённых слов. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ	ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЛИСТЬЯ ЛИСТЬЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЛИСТЬЯ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЛИСТЬЯ ЗЕЛЁНЫЕ

Попробуйте доказать, что других вариантов фраз с использованием трёх данных слов не существует. Обсудите этот вопрос в группе.

15. Разработайте и отладьте программу, которая из слова ИНФОРМАТИКА получает слова ФОРМА, ФИРМА, МАК и подсчитывает общее количество символов в этих словах.

16. Есть арифметический фокус, позволяющий угадать дату рождения любого из окружающих вас людей. Для этого нужно, чтобы этот человек выполнил предварительные вычисления по следующему алгоритму: умножил число, соответствующее его дню рождения, на 2, прибавил к результату 5, полученный результат умножил на 50 и прибавил к тому, что получилось, номер месяца, в который он родился. Результат вычислений он должен сообщить вам. Для того чтобы узнать дату рождения, достаточно вычесть из результата вычислений число 250. Последние две цифры полученного числа будут соответствовать месяцу, первые две (первая одна) — числу месяца угадываемой даты рождения. Попробуйте составить программу-диалог с компьютером, в которой компьютер будет запрашивать у пользователя имя, сообщать ему алгоритм предварительных вычислений и запрашивать их результат, после чего «угадает» и сообщит пользователю день и месяц его рождения.
17. Даны значения целочисленных переменных: `a = 10`, `b = 20`. Чему будет равно значение логической переменной `rez` после выполнения операции присваивания?
    а) `rez = (a == 10) or (b > 10)`
    б) `rez = (a > 5) and (b > 5) and (a < 20) and (b < 30)`
    в) `rez = (not(a < 15)) or (b > 20)`
18. Составьте программу, вводящую `True`, если высказывание является истинным, и `False` в противном случае:
    а) сумма цифр трёхзначного числа `x` является чётным числом;
    б) треугольник со сторонами `a`, `b`, `c` является разносторонним.

§ 5.4 Программирование разветвляющихся алгоритмов

Ключевые слова:

- условный оператор
- неполный условный оператор
- каскадное ветвление

5.4.1. Условный оператор

При записи на языке Python разветвляющихся алгоритмов используют условный оператор. Его общий вид:

```
if <условие>:
    <группа операторов 1>
else:
    <группа операторов 2>
```

Слова **if** и **else** начинаются на одном уровне, а все команды внутренних блоков сдвинуты относительно этого уровня вправо на одно и то же расстояние. Начало и конец блока, который выполняется при истинности (ложности) условия, определяется именно этими сдвигами.

Обратите внимание! В языке Python сдвиги операторов относительно левой границы (отступы) влияют на работу программы. Для сдвига можно использовать пробелы (обычно не меньше двух) или символы табуляции (вставляются при нажатии на клавишу *Tab*).

Для записи неполных ветвлений используется неполная форма условного оператора:

```
if <условие>:
    <операторы>
```

В качестве условий используются логические выражения:

- простые — записанные с помощью операций отношения (<, >, >=, <=, != (не равно), == (равно));
- составные — записанные с помощью логических операций (**and**, **or**, **not**).

В языке Python разрешены двойные неравенства, например `A < B < C`.

Пример 1

Напишем программу, определяющую, является ли введённое целое число x чётным.

```
print('Определение чётности числа')
x = int(input('Введите x: '))
if x % 2 == 0:
    print  (x, ' - чётное число')
else:
    print(x, ' - нечётное число')
```

Пример 2

Запишем на языке Python рассмотренный в п. 3.5.1 (пример 4) алгоритм определения принадлежности точки x отрезку [a, b].

```
print('Определение принадлежности точки отрезку')
a = int(input('Введите a: '))
b = int(input('Введите b: '))
x = int(input('Введите x: '))
if x >= a and x <= b:
    print('Точка принадлежит отрезку')
else:
    print('Точка не принадлежит отрезку')
```

Пример 3

Воспользуемся неполным условным оператором для записи на языке Python рассмотренного в п. 3.5.2 (пример 5) алгоритма присваивания переменной y значения наибольшей из трёх величин a, b и c.

```
print ('Нахождение наибольшей из трёх величин')
a = int(input('Введите a: '))
b = int(input('Введите b: '))
c = int(input('Введите c: '))
y = a
if b > y:
    y = b
if c > y:
    y = c
print('y=', y)
```

Дополните эту программу так, чтобы её выполнение приводило к присваиванию переменной y значения наибольшей из четырёх величин a, b, c и d.

Пример 4

Уравнение вида $ax^2 + bx + c = 0$, где x — переменная, a, b и c — некоторые числа, причём $a \neq 0$, называется квадратным уравнением. Алгоритм решения квадратного уравнения вам хорошо известен. Напишем соответствующую программу на языке Python.

```
from math import *
print('Решение квадратного уравнения')
print('Введите коэффициенты a, b, c>>')
a = float(input('a='))
b = float(input('b='))
c = float(input('c='))
```

```
d = b * b - 4 * a * c
if d < 0:
    print ('Корней нет')
if d == 0:
    x = -b / (2 * a)
    print('Корень уравнения x=', "{:6.4f}". format(x))
if d > 0:
    x1 = (-b + sqrt(d)) / (2 * a)
    x2 = (-b - sqrt(d)) / (2 * a)
    print('Корни уравнения:')
    print('x1=', "{:6.4f}".format(x1))
    print('x2=', "{:6.4f}".format(x2))
```

5.4.2. Многообразие способов записи ветвлений

Внутри условного оператора могут находиться любые операторы, в том числе и другие условные операторы. Например, возможна следующая конструкция:

```
if <условие 1>:
  <группа операторов 1>
else:
    if <условие 2>:
      <группа операторов 2>
    else:
      <группа операторов 3>
```

Если после **else** сразу следует ещё один оператор **if**, можно использовать так называемое **каскадное ветвление** с ключевыми словами **elif** (сокращение от **else-if**). Если очередное условие ложно, то выполняется проверка следующего условия, и т. д.

Пример 5

Воспользуемся каскадным ветвлением для записи на языке Python ещё одного варианта решения квадратного уравнения.

```
from math import *
print('Решение квадратного уравнения')
print('Введите коэффициенты a, b, c>>')
a = float(input('a='))
b = float(input('b='))
c = float(input('c='))
d = b * b - 4 * a * c
if d < 0:
    print ('Корней нет')
```

```
elif d == 0:
    x = -b / (2 * a)
    print('Корень уравнения x=',"{:6.4f}". format(x))
else:
    x1 = (-b + sqrt(d)) / (2 *a)
    x2 = (-b - sqrt(d)) / (2 * a)
    print('Корни уравнения:')
    print('x1=', "{:6.4f}".format(x1))
    print('x2=', "{:6.4f}".format(x2))
```

Используйте вложенные ветвления для записи программы, определяющей принадлежность точки x отрезку [a, b].

САМОЕ ГЛАВНОЕ

При записи на языке Python разветвляющихся алгоритмов используют условный оператор, позволяющий выбрать один из двух вариантов действий в зависимости от выполнения некоторого условия:

```
if <условие>:
    <группа операторов 1>
else:
    <группа операторов 2>
```

Для записи неполных ветвлений используется неполная форма условного оператора:

```
if <условие>:
    <операторы>
```

В обеих частях условного оператора можно использовать любые операторы, в том числе и другие (вложенные) условные операторы.

Вопросы и задания

1. Как на языке Python записывается полное и неполное ветвление?
2. Является ли условным оператором следующая последовательность символов?

а) `if x < y: x = 0 else input(y)`

б)
```
if x >= y: x = 0; y = 0
    else: print (z)
```

в) `if x < y < z: a = a + 1`

3. Запишите следующий фрагмент программы с использованием одного условного оператора:

```
if a > b: c = 1
if a > b: d = 2
if a <= b: c = 3
if a <= b: d = 4
```

4. Две точки на плоскости заданы своими координатами. Разработайте, отладьте и протестируйте программу, определяющую, которая из точек находится ближе к началу координат. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
Координаты 1-й точки>>1 2 Координаты 2-й точки>>3 4	1-я точка ближе
Координаты 1-й точки>>1 2 Координаты 2-й точки>>2 1	Точки равноудалены
Координаты 1-й точки>>2 4 Координаты 2-й точки>>2 2	2-я точка ближе

5. Разработайте, отладьте и протестируйте программу, которая производит обмен значений числовых переменных x и y, если x больше y. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
x>>5 y>>6	x=5 y=6
x>>6 y>>5	x=5 y=6

6. Напишите программу для решения задачи № 11 к § 3.5 (определение дня недели).
7. Чтобы развлечься на аттракционе «Американские горки», нужно иметь рост не ниже 140 см и не выше 195 см. Разработайте и отладьте программу, которая запрашивает у посетителя его рост и делает вывод о том, можно ли его пропустить на этот аттракцион.

8. Напишите программу, предусматривающую ввод одного из чисел 1, 2 или 3 и построение на экране одной из трёх геометрических фигур: треугольника, если введено число 1; квадрата, если введено число 2; окружности, если введено число 3.

9. Дано натуральное трёхзначное число n. Разработайте, отладьте и протестируйте программу, определяющую:

   а) является ли данное число «перевёртышем», т. е. числом, десятичная запись которого читается одинаково слева направо и справа налево;

Входные данные	Выходные данные
122	Нет
121	Перевёртыш
222	Перевёртыш

   б) есть ли среди цифр данного числа одинаковые.

Входные данные	Выходные данные
123	Нет
121	Да
222	Да

Протестируйте программу на приведённых входных данных.

10. Даны три натуральных числа. Разработайте, отладьте и протестируйте программу, определяющую, существует ли треугольник с такими длинами сторон. Если такой треугольник существует, то определите его тип (равносторонний, равнобедренный, разносторонний). Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
a b c>>1 2 1	Не существует
a b c>>2 2 2	Равносторонний
a b c>>20 20 30	Равнобедренный
a b c>>3 4 5	Разносторонний

11. Имеются данные о количестве полных лет трёх призёров спартакиады. Разработайте и отладьте программу, определяющую и выводящую возраст самого младшего призёра.
12. Дана программа на языке Python:

```
s = int(input())
t = int(input())
if (s >= 10) or (t > 10):
    print('Да')
else:
    print('Нет')
```

Было проведено 9 запусков программы, при которых в качестве значений переменных s и t вводились следующие пары чисел: (1, 2); (11, 2); (1, 12); (11, 12); (-11, -12); (-11, 12); (-12, 11); (10, 10); (10, 5).

Не запуская программу на выполнение, выясните, сколько было запусков, при которых программа вывела "Да". Для анализа алгоритма и исходных данных начертите в рабочей тетради и заполните следующую таблицу:

№	s	t	s >= 10	t >= 10	(s >= 10) or (t > 10)	Вывод
1	1	2	0	0	0	Нет
...						
9	10	5	1	0	1	Да

13. Дан условный оператор:

```
if a < 5:
    c = 1
elif a > 5:
    c = 2
else:
    c = 3
```

Какое значение имеет переменная a, если в результате выполнения условного оператора переменной c присваивается значение 3?

14. Напишите программу, в которой пользователю предлагается дополнить до 100 некоторое целое число a (a — случайное число, меньшее 100). Ответ пользователя проверяется и комментируется.

15. С помощью программы сравните тройки слов и сделайте выводы о том, как происходит сравнение: KAWAI — Kawai — kawai, информатика — информатика — информатикА, 50_кг — 50_kg — 200_кг, яблоко — яблоки — яблоня.

16. Напишите программу, которая моделирует работу исполнителя Автомат, который получает на вход трёхзначное натуральное число и строит новое число следующим образом:
    1) вычисляет суммы первой и второй, затем второй и третьей цифр;
    2) выводит полученные суммы в порядке неубывания.

    Например, для числа 125 будут получены суммы: 1 + 2 = 3, 2 + 5 = 7. Результат: 37.

17. Поле шахматной доски определяется парой натуральных чисел, каждое из которых не превосходит 8. Напишите программу, которая по введённым координатам двух полей (k, l) и (m, n) определяет, имеют ли эти поля один цвет. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
Координаты 1-го поля>>2 2 Координаты 2-го поля>>3 3	Поля одного цвета
Координаты 1-го поля>>2 3 Координаты 2-го поля>>3 3	Поля разного цвета
Координаты 1-го поля>>2 7 Координаты 2-го поля>>5 4	Поля одного цвета

§ 5.5 Программирование циклических алгоритмов

Ключевые слова:

- оператор цикла
- **while** (цикл-ПОКА)
- **for** (цикл-ДЛЯ)
- вложенные циклы

Оператор цикла — команда, реализующая алгоритмическую конструкцию «повторение» на языке программирования. В языке программирования Python существует несколько операторов цикла.

5.5.1. Программирование циклов с заданным условием продолжения работы

Цикл с заданным условием продолжения работы (цикл-ПОКА) программируется в языке Python с помощью оператора **while**. Общий вид оператора:

```
while <условие>:
    <тело цикла>
```

Здесь:

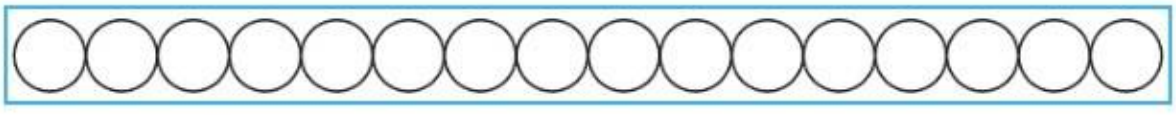
<условие> — логическое выражение; пока оно истинно, выполняется тело цикла;

<тело цикла> — один или несколько операторов, описывающих последовательность действий, выполняемых многократно.

Пример 1

Следующая программа изображает в центральной части графического окна ряд окружностей радиусом 20.

```
from graph import *
windowSize(640, 480)
canvasSize(640, 480)
x = 20
while x <= 620:
    circle(x, 240, 20)
    x = x + 40
```



Внесите изменения в программу так, чтобы:
а) рисовались окружности радиусом 30;
б) рисовался вертикальный ряд окружностей.

Пример 2

Запишем на языке Python рассмотренный в п. 3.2.2 (пример 3) алгоритм Евклида для нахождения наибольшего общего делителя двух натуральных чисел.

```
x = int(input())
y = int(input())
while x != y:
    if x > y:
        x = x - y
    else:
        y = y - x
nod = x
print('НОД = ', nod)
```

Найдите с помощью программы наибольший общий делитель для следующих пар чисел: 123 и 12; 450 и 180; 500 и 125. Как можно воспользоваться этой программой, если надо найти НОД трёх натуральных чисел, например: 450, 180 и 60?

Пример 3

Запишем на языке Python рассмотренный в п. 3.6.1 (пример 3) алгоритм получения частного `q` и остатка `r` от деления натурального числа `x` на натуральное число `y` без использования операции деления.

```
print('Частное и остаток')
x = int(input('Введите делимое x>>'))
y = int(input('Введите делитель y>>'))
r = x
q = 0
while r >= y:
    r = r - y
    q += 1
print('Частное q =', q)
print('Остаток r =', r)
```

Выполните программу при `x = 25` и `y = 4`. Каким будет результат выполнения программы при `x = -10` и `y = 3`? Как вы можете объяснить этот результат?

Пример 4

Имеется последовательность натуральных чисел. Составим программу, получающую на вход натуральные числа, количество которых заранее неизвестно (0 — признак окончания ввода,

не входит в последовательность), и подсчитывающую количество членов последовательности, оканчивающихся на 3.

```
print('Обработка последовательности')
k = 0
print('Первый член последовательности>>')
a = int(input())
while a != 0:
    if a % 10 == 3:
      k += 1
    print('Очередной член последовательности или 0>>')
    a = int(input())
print('k=', k)
```

Самостоятельно разработайте тест для проверки работоспособности программы.

Модифицируйте программу так, чтобы в ней подсчитывалось количество членов последовательности, кратных 3.

5.5.2. Программирование циклов с заданным условием окончания работы

В языке Python нет специального оператора для программирования циклов с заданным условием окончания работы, но его можно организовать с помощью цикла **while**:

```
while True:
    <операторы>
    if <условие>: break
```

Цикл

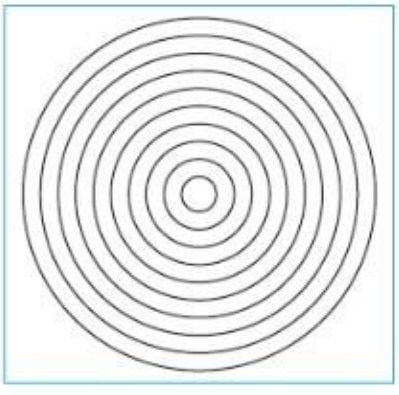
```
while True:
    <операторы>
```

будет выполняться бесконечно, потому что условие `True` всегда истинно. Выйти из такого цикла можно только с помощью специального оператора **break** (в переводе с англ. — прервать).

Пример 5

Следующая программа изображает в центральной части графического окна множество окружностей с общим центром, радиус которых уменьшается от 200 до тех пор, пока не станет меньшим 10.

```
from graph import *
windowSize(640, 480)
canvasSize(640, 480)
r = 200
while True:
    circle(320, 240, r);
    r = r - 5
    if r < 10: break
```



Запустите программу на выполнение при разных значениях шага изменения радиуса. Попробуйте получить такое же изображение, использовав в программе цикл с заданным условием продолжения работы.

Пример 6

Рассмотрим произвольное натуральное число n, не превосходящее 10^9. Составим программу, вычисляющую количество цифр в числе n.

Для подсчёта количества цифр в записи числа n будем действовать по следующему плану: «укорачиваем» число справа на один разряд; учитываем правую цифру как цифру в записи исходного числа. Эту группу действий повторяем до тех пор, пока число («укороченное» число) больше нуля; достижение нуля — условие прекращения вычислений.

```
print ('Подсчёт цифр в числе')
n = int(input('Введите число n>>'))
k = 0
while True:
    n = n // 10
    k += 1
    if n == 0: break
print ('k = ', k)
```

Предложите решение этой задачи с использованием цикла с заданным условием продолжения работы. Попытайтесь доработать исходную программу так, чтобы в ней подсчитывалась ещё и сумма цифр исходного числа.

Пример 7

Запишем на языке Python рассмотренный в п. 3.6.2 (пример 6) алгоритм решения задачи о графике тренировок спортсмена.

```
print ('График тренировок')
i = 1; x = 10
while True:
    i += 1
    x = x + 0.1 * x
    if x >= 25: break
print ('Начиная с ', i, '-го дня спортсмен будет
       пробегать 25 км', sep='')
```

5.5.3. Программирование циклов с фиксированным числом повторений

Цикл с фиксированным числом повторений (цикл с параметром) программируется в языке Python с помощью оператора **for** (в переводе с англ. — для). Его общий вид:

```
for <параметр> in range(k, n, m):
    <операторы>
```

Здесь:

`<параметр>` — переменная целого типа;

`range()` — функция, описывающая необходимое количество повторов тела цикла; в скобках может быть указано от одного до трёх чисел:

- одно число (`n`) указывает на то, что нужно последовательно перебрать все целые числа от `0` до `n`, причём само `n` не рассматривается;
- два числа (`k`, `n`) говорят о том, что нужно последовательно перебрать все целые числа, находящиеся в диапазоне от `k` (начальное значение) до `n - 1` (конечное значение);
- три числа (`k`, `n`, `m`) указывают на то, что параметр должен изменяться от `k` до `n - 1` с шагом, равным `m`;

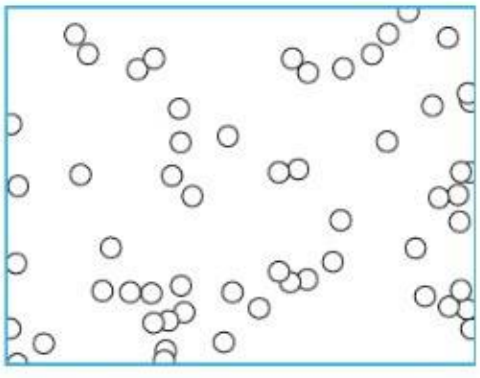
`<операторы>` — один или несколько операторов, образующих тело цикла.

При выполнении оператора **for** после каждого выполнения тела цикла происходит увеличение на шаг параметра цикла. Если `k = n` или `k > n`, цикл не выполнится ни разу.

Пример 8

Следующая программа изображает в графическом окне ровно 100 окружностей радиусом 10, координаты центров которых генерируются случайным образом.

```
from math import *
from graph import *
windowSize(640, 480)
canvasSize(640, 480)
for i in range(100):
    x = randint(1, 640)
    y = randint(1, 480)
    circle(x, y, 10)
```



Внесите изменения в программу так, чтобы радиусы изображаемых окружностей также генерировались случайным образом. Попробуйте получить такое же изображение, использовав в программе оператор **while**.

Пример 9

Делитель натурального числа n — это число, на которое n делится без остатка. У любого натурального числа n есть хотя бы два делителя — это 1 и n. Составим программу, которая для введённого с клавиатуры натурального числа n выводит на экран все делители этого числа.

Организуем полный перебор всех возможных вариантов делителей от 1 до n.

```
print('Делители числа')
n = int(input('Введите число n>>'))
for i in range(1, n + 1):
    if n % i == 0:
        print(i)
```

Модифицируйте программу так, чтобы в ней подсчитывалось количество делителей исходного числа n. Дополните программу так, чтобы на основании найденного количества делителей в ней выводилось сообщение о том, является исходное число простым или нет.

Пример 10

Запишем на языке Python рассмотренный в п. 3.6.4 (пример 11) алгоритм вычисления степени с натуральным показателем n для любого вещественного числа a.

```
print ('Возведение в степень')
a = float(input('Введите основание a>>'))
n = int(input('Введите показатель n>>'))
```

```
y = 1
for i in range(n):
    y = y * a
print('y=', y)
```

Здесь параметр цикла i будет изменяться от 0 до n - 1 включительно, следовательно, тело цикла выполнится ровно n раз.

Пример 11

Составим программу, которая в последовательности натуральных чисел подсчитывает количество чисел, кратных 3. Программа получает на вход количество чисел в последовательности, а затем — сами числа. Программа должна вывести одно число — количество членов последовательности, кратных 3.

```
print('Обработка последовательности')
k = 0
n = int(input('Введите количество членов
                последовательности>>'))
for i in range(n):
    a = int(input('Очередной член
                   последовательности>>'))
    if a % 3 == 0:
        k += 1
print('k=', k)
```

Самостоятельно разработайте тест для проверки работоспособности программы.

Модифицируйте программу так, чтобы в ней подсчитывалось количество членов последовательности, оканчивающихся на 3.

Сравните задачи и программы, рассмотренные в примерах 4 и 11. Укажите их сходство и различие.

Пример 12

Составим программу, которая предлагает ввести строку из строчных английских букв и находит количество букв "s" в этой строке.

```
s = input('Введите строку>>')
k = 0
for i in range(len(s)):
    if s[i] == 's':
        k += 1
print('k=', k)
```

Модифицируйте программу так, чтобы в ней:
а) подсчитывалось общее количество букв "s" и "t" во введённой строке;
б) определялось, каких букв — "s" или "t" — во введённой строке больше.

5.5.4. Вложенные циклы

Цикл называется **вложенным**, если он содержится внутри (в теле) другого цикла.

Цикл, содержащий в себе другой цикл, называют *внешним*, а цикл, содержащийся в теле другого цикла, — *внутренним*. Внутренний и внешний циклы могут быть любыми из трёх видов: цикл с параметром, цикл с предусловием или цикл с постусловием.

Пример 13

Составим программу, выводящую на экран пять строк, каждая из которых состоит из десяти символов *.

```
for i in range(5):
    for j in range(10):
        print('*', end='')
    print()
```

Модифицируйте программу так, чтобы в ней на экран выводилось десять строк, состоящих из пяти символов * каждая. Выясните, для чего в этой программе используется команда print().

Пример 14

Следующая программа изображает в графическом окне пять рядов окружностей радиусом 10, по 8 кругов в каждом ряду.

```
from graph import *
windowSize(640, 480)
canvasSize(640, 480)
y = 10
for i in range(5):
    x = 10
    for j in range (8):
        circle(x, y, 10)
        x += 20
    y+=20
```

Внесите изменения в программу так, чтобы:
а) рисовались окружности радиусом 20;
б) ряды окружностей заполняли всё графическое окно.

САМОЕ ГЛАВНОЕ

В языке Python имеются два вида операторов цикла: **while** (цикл с условием) и **for** (цикл с параметром).

Цикл с условием выполняется до тех пор, пока некоторое условие (условие продолжения работы цикла) не станет ложным. Если условие в заголовке цикла ложно с самого начала, тело цикла не выполнится ни разу.

Если условие в заголовке цикла всегда остаётся истинным, цикл работает бесконечно. Для досрочного выхода из цикла используют оператор **break**.

Цикл с параметром применяют тогда, когда количество повторений цикла известно заранее или может быть вычислено до начала цикла. Переменная, изменение которой определяет работу цикла, называется параметром (переменной) цикла. При вызове функции `range()` указывают от одного до трёх параметров: начальное значение, значение-ограничитель (не входящее в диапазон) и шаг изменения переменной цикла; обязательный параметр — значение-ограничитель. Если число повторений тела цикла известно, то лучше воспользоваться оператором **for**; в остальных случаях используется оператор **while**.

Вопросы и задания

1. Проанализируйте работу программы:

```
x = 1
y = 1
while x < 5:
    y *= 2
    x += 1
```

Ответьте на вопросы.
а) Сколько раз выполнится тело цикла?
б) Какое значение примет переменная `x` после завершения программы?
в) Какое значение примет переменная `y` после завершения программы?

г) Сколько раз выполнится тело цикла, если заменить условие на `x <= 5`?

д) Сколько раз выполнится тело цикла, если заменить условие на `x >= 5`?

е) Сколько раз выполнится тело цикла, если заменить условие на `x > 0`?

ж) Что произойдёт, если из тела цикла убрать команду `x += 1`?

з) Сколько раз выполнится тело цикла, если заменить команду `x += 1` на `x += 2`?

и) Сколько раз выполнится тело цикла, если заменить команду `x += 1` на `x -= 1`?

2. Дана последовательность операторов:

```
a = 1; b = 2
while a + b < 8:
    a += 1
    b += 2
s = a + b
```

Сколько раз будет повторено тело цикла и какими будут значения переменных `a`, `b`, `s` после выполнения этой последовательности операторов?

3. Требовалось написать программу вычисления факториала числа `n` (факториал числа `n` есть произведение всех целых чисел от 1 до `n`). Программист торопился и написал программу неправильно. Ниже приведён фрагмент его программы, в котором содержится несколько ошибок:

```
k = 1
f = 0
while k < n:
    f = f * k
k += 1
```

Обсудите этот фрагмент программы в группе. Найдите ошибки. Внесите необходимые исправления, допишите программу и выполните её на компьютере. Для проверки правильности программы используйте тест:

Входные данные	Выходные данные
Введите n>>5	5!=120
Введите n>>6	6!=720

4. Проанализируйте следующий цикл:

```
while a < b:
    c = a == b
```

В чём его особенность?

5. Запишите на языке Python программы решения задач № 3, 4, 6, 7 из § 3.6. Используйте оператор **while**.

6. Дана последовательность операторов:

```
a = 1
b = 1
while True:
    a += 1
    b *= 2
    if b > 8: break
s = a + b
```

Сколько раз будет повторено тело цикла и какими будут значения переменных a, b, s после выполнения этой последовательности операторов? Обсудите этот вопрос в группе.

7. Напишите программу, в которой осуществляется ввод целых чисел (ввод осуществляется до тех пор, пока не будет введён ноль) и подсчёт суммы всех введённых чисел.

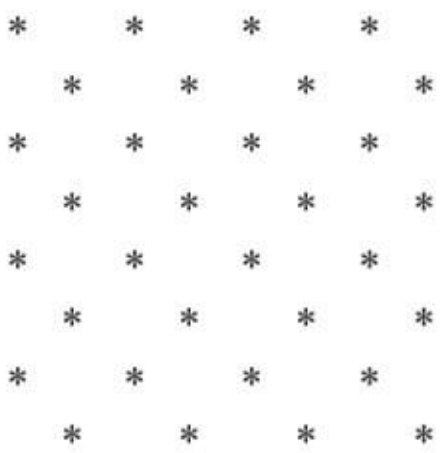
8. Напишите программу, в которой осуществляется ввод целых чисел (ввод осуществляется до тех пор, пока не будет введён ноль) и определение максимального (наибольшего) из введённых чисел.

9. Сколько раз будет выполнено тело цикла?

а)
```
for i in range(15): s += i
```
б)
```
for i in range(10, 15): s += i
```
в)
```
for i in range(-1, 1): s += i
```
г)
```
for i in range(1, 1): s += i
```
д)
```
k = 5
for i in range(k - 1, k + 1): s += i
```

10. Напишите программу, которая 10 раз выводит на экран ваши имя и фамилию.

11. Напишите программу, выводящую на экран изображение шахматной доски, где чёрные клетки изображаются звёздочками, а белые — пробелами. Рекомендуемый вид экрана после выполнения программы:



12. Напишите программу, которая вычисляет сумму:
    а) первых n натуральных чисел;
    б) квадратов первых n натуральных чисел;
    в) всех чётных чисел на отрезке от 1 до n;
    г) всех двузначных чисел.

13. Запишите на языке Python программы решения задач № 14–16 из § 3.6. Используйте оператор `for`.

14. Напишите программу, которая выводит на экран таблицу степеней двойки (от нулевой до десятой). Рекомендуемый вид экрана после выполнения программы:

```
Таблица  степеней  двойки
      0          1
      1          2
      2          4
      3          8
      4         16
      5         32
      6         64
      7        128
      8        256
      9        512
     10       1024
```

15. Напишите программу, которая выводит на экран таблицу умножения на n (n — целое число в диапазоне от 2 до 10, вводимое с клавиатуры). Протестируйте программу на следующих данных:

Входные данные	Выходные данные
Введите n>>5	5 * 2 = 10 5 * 3 = 15 5 * 4 = 20 5 * 5 = 25 5 * 6 = 30 5 * 7 = 35 5 * 8 = 40 5 * 9 = 45 5 * 10 = 50

16. После строительства дома осталось некоторое количество плиток. Их можно использовать для выкладывания прямоугольной площадки на участке рядом с домом. Если укладывать в ряд по 10 плиток, то для квадратной площадки плиток не хватит. При укладывании по 8 плиток в ряд остаётся один неполный ряд, а при укладывании по 9 плиток тоже остаётся неполный ряд, в котором на 6 плиток меньше, чем в неполном ряду при укладывании по 8.

    Напишите программу, вычисляющую, сколько всего плиток осталось после строительства дома.

17. Какой из двух рассмотренных операторов цикла является, по вашему мнению, основным, т. е. таким, что им можно заменить другой оператор? Обсудите этот вопрос в группе.

Тестовые задания для самоконтроля

1. Разработчиком языка Python является:
   а) Блез Паскаль
   б) Никлаус Вирт
   в) Норберт Винер
   г) Гвидо ван Россум
2. Что из нижеперечисленного не входит в алфавит языка Python?
   а) Латинские строчные и прописные буквы
   б) Служебные слова
   в) Русские строчные и прописные буквы
   г) Знак подчёркивания
3. Какая последовательность символов не может служить именем в языке Python?
   а) `_mas`
   б) `mas1`
   в) `d2`
   г) `2d`
4. Обозначение вещественного типа данных в языке Python:
   а) `float`
   б) `int`
   в) `bool`
   г) `str`
5. Языковые конструкции, с помощью которых в программах записываются действия, выполняемые в процессе решения задачи, называются:
   а) операндами
   б) операторами
   в) выражениями
   г) данными
6. Разделителем между операторами в одной строке служит:
   а) точка
   б) точка с запятой
   в) пробел
   г) запятая

7. При присваивании всегда изменяется:
   а) имя переменной
   б) тип переменной
   в) значение переменной
   г) значение константы
8. Для вывода результатов в Python используется оператор
   а) `while`
   б) `input`
   в) `print`
   г) `and`
9. Для вычисления квадратного корня из x используется функция:
   а) `abs(x)`
   б) `sqr(x)`
   в) `sqrt(x)`
   г) `int(x)`
10. Для генерации случайного целого числа из отрезка [10, 20] необходимо использовать выражение:
    а) `randint(2 * 10)`
    б) `randint(10 20)`
    в) `randint(10, 20)`
    г) `randint(10) * 2`
11. В каких условных операторах допущены ошибки?

```
а) if b == 0: print('Деление невозможно.')
б) if a < b: min = a; else min = b
в) if a > b : max = a
       else max = b
г) if a > b and b > 0: c = a + b
```

12. Определите значение переменной c после выполнения следующего фрагмента программы:

```
a = 100
b = 30
a = a - b * 3
if a > b:
    c = a - b
else:
    c = b - a
```

а) 20
б) 70
в) −20
г) 180

13. Условный оператор

```
if a % 2 == 0:
    print ('Да')
else:
    print ('Нет')
```

позволяет определить, является ли число a:
а) целым
б) двузначным
в) чётным
г) простым

14. Какого оператора цикла не существует в языке Python?
а) **for**
б) **while**
в) **repeat...until**

15. Цикл в фрагменте программы

```
a = 1
b = 1
while a + b < 8:
    a += 1
    b += 2
```

выполнится:
а) 0 раз
б) 2 раза
в) 3 раза
г) бесконечное число раз

16. Определите значения переменных s и i после выполнения фрагмента программы:

```
s = 0
i = 5
while i > 0:
    s += i
    i -= 1
```

а) `s = 0, i = -1`
б) `s = 5, i = 0`
в) `s = 15, i = 5`
г) `s = 15, i = 0`

17. В данном фрагменте программы

```
s = 0
for i in range (1, 11):
    s = s + 2 * i
```

вычисляется:

а) сумма целых чисел от 1 до 10
б) сумма чётных чисел от 1 до 10
в) удвоенная сумма целых чисел от 1 до 11
г) сумма первых десяти натуральных чётных чисел

КЛЮЧИ К ТЕСТОВЫМ ЗАДАНИЯМ ДЛЯ САМОКОНТРОЛЯ

Глава 1

Задание	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ответ	в	б	в	б	б	г	в	в	в	б	б	б	б	г	в

Глава 2

Задание	1	2	3	4	5	6	7	8	9	10	11
Ответ	б	г	а	а	в	б	б	в	в	в	г

Глава 3

Задание	1	2	3	4	5	6	7	8	9	10	11
Ответ	а	г	в	б	а	в	г	д	11121	127	в
Задание	12	13	14	15	16	17	18	19	20	21	22
Ответ	в	в	а	г	б	б	а	в	б	г	б
Задание	23	24	25	26	27	28	29	30	31	32	
Ответ	б	80	а	б	а	г	25	120	55	в	

Глава 4

Задание	1	2	3	4	5	6	7	8	9	10
Ответ	б	в	г	а	в	б	б	б	б	б
Задание	11	12	13	14	15	16	17	18	19	20
Ответ	а	в	в	в	в	б	в	а	в	г
Задание	21	22	23	24	25					
Ответ	в	б	г	в	в, г					

Глава 5

Задание	1	2	3	4	5	6	7	8	9	10
Ответ	г	б	г	а	б	б	в	в	в	в
Задание	11	12	13	14	15	16	17			
Ответ	б, в	а	в	в	б	г	г			

ОТВЕТЫ К ВОПРОСАМ И ЗАДАНИЯМ ДЛЯ САМОСТОЯТЕЛЬНОЙ ПОДГОТОВКИ

Глава 1

§ 1.1

7. а) 122; б) 537; в) 42; г) 99.

8. а) 110011_2; б) 111_4.

9. 5.

10. а) Да; б) нет.

11. а) 5; б) 4.

12. 25, 50, 75.

§ 1.2

2. 4096.

4. а) 1; б) 2; в) 8.

5. а) 1; б) 0; в) 7.

6. а) 55; б) 20; в) 28.

7. а) 30; б) 35; в) 70.

8. а) 16; б) 8; в) 6.

9. а) 9; б) 8; в) 11.

10. Переведите операнды в двоичную систему счисления.

§ 1.3

11. 52.

12. 61.

13. а) 10; б) 198.

§ 1.4

4. 101010_2.

5. а) DREAM; б) Moscow.

Глава 2

§ 2.1

3. а, в.

4. б, г.

7. б, в.

8. в, г.

9. б.

§ 2.2

3. а) 1; б) 1; в) 0; г) 1; д) 1; е) 1; ж) 0; з) 1; и) 1; к) 0.

4. а) 1; б) 1; в) 0; г) 1.

5. а) 0; б) 1; в) 1; г) 0; д) 0.

6. а) {2, 4, 6}; б) {∅}; в) {2, 4, 6}; г) {1, 2, 3, 4, 5, 6, 7, 8}.

7. 2300.

8. 59.

9. 58.

10. Сосуд финикийский, изготовлен в V веке.

§ 2.3

3. а) 8, (111); г) 16, для всех за исключением (0000).

4. Иван, Александр.

§ 2.4

1. а) $\overline{A} \vee \overline{B}$; б) $\overline{A} \wedge (A \vee B)$.

2. а) 10; б) 01; в) 10; г) 10.

Глава 3

§ 3.1

15. 5.

18. а) 12211; б) 22211.

19. а) 6; б) 12212.

21. 39078, 96930.

25. 17 мин.

§ 3.3

10. Не более одной переменной.

14.

г) $(x > 0)$ или $(y > 0)$;

е) $((x > 0)$ и $(y <= 0))$ или $((y > 0)$ и $(x <= 0))$.

17.

а) `t := x > 0`

в) `t := (x = y) и (x = z)`

§ 3.4

4. $y = \frac{1}{x} + \frac{1}{x^2} + \frac{1}{x^3} + \frac{1}{x^4}$.

5. $y = ((2x + 3)x + 4)x + 5$; $y = 14$ при $x = 1$.

6.
```
h := tfh * 24
m := h * 60
c := m * 60;
```

10. 4.

§ 3.5

11.
```
если chislo = 3 то y := 'понедельник'
```

15. 1)
```
использовать Робот
| использовать Робот
алг
нач
  закрасить
  если снизу стена
    то
      вправо; вниз; влево
  все
  если слева стена
    то
      вниз; влево; вверх
  все
  если сверху стена
    то
      влево; вверх; вправо
  все
  если справа стена
    то
      вверх; вправо; вниз
  все
  закрасить
кон
```

§ 3.6

8. `n = 32, m = 5`.

9. а) В исходной точке; б) `a = 2, b = -3`.

11.
```
вправо
вверх
вправо
вправо
вниз
вправо
вправо
```

16. 1024.

Глава 4

§ 4.1

9. в) Пусть `n` — количество тетрадей (обложек), `ct` — цена одной тетради (целое число), `co` — цена одной обложки (целое число), `s` — общая стоимость покупки. В этом случае раздел описания переменных будет иметь вид `var: s, n, ct, co: integer;`

11. б) `k := k - 1;`

§ 4.2

7. 111.

§ 4.3

1. 30.

17. а) `true`; б) `true`; в) `false`.

§ 4.4

2. а) Да; б) нет; в) нет.

12. 7.

13. 5.

§ 4.5

1. а) 4; б) 5; в) 16; г) 5; д) 0; е) бесконечно; ж) зацикливание; з) 2; и) бесконечно.

2. Два раза. 3, 6, 9.

6. Четыре раза. 5, 16, 21.

9. а) 16; б) 6; в) 3; г) 1; д) 3.

Глава 5

§ 5.1

11. 55.

12. а) 7; б) 15; в) 5; г) 10; д) 16; е) 50.

13. б) 0; 0.2

§ 5.2

4. 1 11.

9. 148.

§ 5.3

1. 30.

17. а) `true`; б) `true`; в) `false`.

§ 5.4

2. а) Нет; б) да; в) да.

12. 7.

13. 5.

§ 5.5

1. а) 4; б) 5; в) 16; г) 5; д) 0; е) бесконечно; ж) зацикливание; з) 2; и) бесконечно.

2. Два раза. 3, 6, 9.

6. Четыре раза. 5, 16, 21.

9. а) 15; б) 5; в) 2; г) 0; д) 2.

Приложение 1
ОТЛАДКА ПРОГРАММ В PASCALABC.NET

Команды отладчика собраны в меню **Программа** и продублированы на панели инструментов.

Команды отладки

Команда	Функция	Клавиша
Выполнить 	Запуск программы на выполнение	*F9*
Завершить 	Завершение отладки или выполнения программы	*Ctrl+F2*
Идти к курсору	Выполнение программы до строки, в которой установлен курсор	*F4*
Шаг без входа в подпрограмму 	Выполнение строки программы. При наличии в выполняемой строке вызова подпрограммы подпрограмма выполняется полностью	*F8*
Шаг с входом в подпрограмму 	Выполнение строки программы. При наличии в выполняемой строке вызова подпрограммы подпрограмма выполняется пошагово	*F7*

Некоторые строки программы можно помечать как точки останова (точки прерывания). Для этого достаточно щёлкнуть мышью в серой области левее соответствующей строки — выбранная строка на экране подкрасится красным цветом.

```
  write('Введите строку>>');
  read(s);
  k := 0;
  for i := 1 to length(s) do
    if s[i] = 's' then
      k := k + 1;
  write('k=', k)
end.
```

Когда в процессе выполнения программы выделенная строка будет достигнута, выполнение программы приостановится; строка на экране, на которой программа остановится, подкрасится жёлтым цветом.

```
  write('Введите строку>>');
  read(s);
  k := 0;
  for i := 1 to length(s) do
    if s[i] = 's' then
      k := k + 1;
  write('k=', k)
end.
```

В окне вывода (вкладка **Локальные переменные**) можно посмотреть значения переменных, задействованных в программе. Если начать пошаговое выполнение программы, то можно наблюдать за тем, как изменяются значения переменных.

Возможности отладчика помогают обнаруживать и исправлять логические ошибки в программах.

Приложение 2
НЕКОТОРЫЕ ОПЕРАТОРЫ БИБЛИОТЕКИ GRAPHABC В PASCALABC.NET

Оператор	Описание
`SetPixel(x, y, c)`	Закрашивает пиксель с координатами (`x`, `y`) цветом `c`
`MoveTo(x, y)`	Устанавливает текущую позицию рисования в точку с координатами (`x`, `y`)
`Line(x1, y1, x2, y2)`	Рисует отрезок от точки (`x1`, `y1`) до точки (`x2`, `y2`)
`LineTo(x,y)`	Рисует отрезок от текущей позиции до точки (`x`, `y`). Текущая позиция переносится в точку (`x`, `y`)
`Circle(x, y, r)`	Рисует окружность с центром (`x`, `y`) и радиусом `r`
`Ellipse(x1, y1, x2, y2)`	Рисует эллипс, ограниченный прямоугольником, заданным координатами противоположных вершин (`x1`, `y1`) и (`x2`, `y2`)
`Rectangle(x1, y1, x2, y2)`	Рисует прямоугольник, заданный координатами противоположных вершин (`x1`, `y1`) и (`x2`, `y2`)
`FloodFill(x,y,c)`	Заливает область одного цвета цветом `c`, начиная с точки (`x`, `y`)

Некоторые цветовые константы:

`clBlue`
`clGray`
`clGreen`
`clBrown`
`clWhite`
`clRed`
`clYellow`
`clBlack`

Приложение 3
НЕКОТОРЫЕ ОПЕРАТОРЫ МОДУЛЯ GRAPH В PYTHON

Оператор	Описание
`windowSize(width, height)`	Устанавливаются размеры в пикселях графического окна
`canvasSize(width, height)`	Устанавливаются размеры в пикселях области («холста») для рисования
`penSize(width)`	Установка толщины пера
`penColor(color)` `penColor(r, g, b)`	Установка цвета пера. Цвет может быть задан как символьная строка с названием цвета (`"red"`, `"green"` и т. д.), как символьная строка с HTML-кодом цвета (`"#FF00GFF"`) или значениями трёх составляющих цвета в модели RGB
`point(x, y)` `point(x, y, c)`	Нарисовать точку цвета `c` с координатами (`x`, `y`)
`moveTo(x, y)`	Устанавливает текущую позицию рисования в точку с координатами (`x`, `y`)
`line(x1, y1, x2, y2)`	Рисует отрезок от точки (`x1`, `y1`) до точки (`x2`, `y2`)
`lineTo(x, y)`	Рисует отрезок от текущей позиции до точки (`x`, `y`). Текущая позиция переносится в точку (`x`, `y`)
`circle(x, y, r)`	Рисует окружность с центром (`x`, `y`) и радиусом `r`
`rectangle(x1, y1, x2, y2)`	Рисует прямоугольник, заданный координатами противоположных вершин (`x1`, `y1`) и (`x2`, `y2`)

ОГЛАВЛЕНИЕ