9

Л. Л. Босова
А. Ю. Босова

ИНФОРМАТИКА И ИКТ

1

ИЗДАТЕЛЬСТВО БИНОМ

Л. Л. Босова, А. Ю. Босова

ИНФОРМАТИКА И ИКТ

Учебник для 9 класса

Часть 1

Рекомендовано
Министерством образования и науки
Российской Федерации
к использованию в образовательном процессе
в имеющих государственную аккредитацию
и реализующих образовательные программы
общего образования образовательных учреждениях

Москва
БИНОМ. Лаборатория знаний
2012

Введение

Уважаемые девятиклассники!

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, умений организации собственной учебной деятельности, ориентации на деятельностную жизненную позицию. Формирование таких качеств личности невозможно без фундаментального базового образования.

В курсе информатики 9 класса большое внимание уделяется фундаментальным (теоретическим) вопросам информатики. Вы будете изучать математические основы информатики, познакомитесь с моделированием, научитесь разрабатывать алгоритмы и программы, расширите свои представления об информационных и коммуникационных технологиях.

Очень важно, чтобы вы были готовы к продолжению обучения с использованием средств и методов информатики и ИКТ.

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

- целеполагание — постановка задачи на основе соотнесения того, что уже известно, и того, что требуется установить;
- планирование — определение последовательности промежуточных целей с учетом конечного результата, разбиение задачи на подзадачи, разработка последовательности и структуры действий, необходимых для достижения цели при помощи 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 корректив в план действий в случае обнаружения ошибки;
- оценка — осознание человеком того, насколько качественно им решена задача.

Как и ранее, в этом учебнике кроме основной информации содержатся многочисленные ссылки на образовательные ресурсы сети Интернет, в том числе на такие порталы, как:

1) Единая коллекция цифровых образовательных ресурсов (http://school-collection.edu.ru/);
2) Федеральный центр информационных образовательных ресурсов (http://fcior.edu.ru/).

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.

Изучая теоретический материал, работая с дополнительными материалами, отвечая на вопросы, решая задачи и выполняя практические задания на компьютере, вы сможете полностью подготовиться к сдаче выпускного экзамена по курсу информатики и ИКТ в новой форме, требования к которому размещены на сайте http://fipi.ru/.

В работе с учебником вам помогут навигационные значки:

 — важное утверждение или определение;

 — ссылка на ресурс в Интернете;

 — пример решения задачи;

 — дополнительная интересная информация;

 — задания для подготовки к итоговой аттестации;

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

 — информация, полезная для решения практических задач;

 — задания для практических работ.

Желаем успехов в изучении информатики и ИКТ!

Глава 1
МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

§ 1.1
Системы счисления

Ключевые слова:

- система счисления
- цифра
- алфавит
- позиционная система счисления
- основание
- развёрнутая форма записи числа
- свёрнутая форма записи числа
- двоичная система счисления
- восьмеричная система счисления
- шестнадцатеричная система счисления

1.1.1. Общие сведения о системах счисления

Система счисления — это знаковая система, в которой приняты определённые правила записи чисел. Знаки, при помощи которых записываются числа (рис. 1.1), называются **цифрами**, а их совокупность — **алфавитом** системы счисления.

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

Пример 1. У вавилонян узловыми являлись числа 1, 10, 60; в римской системе счисления узловыми являются числа 1, 5, 10, 50, 100, 500 и 1000, обозначаемые соответственно I, V, X, L, C, D, M.

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

1) унарные системы;

2) непозиционные системы;

3) позиционные системы.

Простейшая и самая древняя система — так называемая **унарная система счисления**. В ней для записи любых чисел используется всего один символ — палочка, узелок, зарубка, камушек. Длина записи числа при таком кодировании прямо связана с его величиной, что роднит этот способ с геометрическим представлением чисел в виде отрезков. Именно унарная система лежит в фундаменте арифметики, и именно она до сих пор вводит первоклассников в мир счёта. Унарные системы ещё называют системами бирок.

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

В непозиционных системах счисления числа образуются путём сложения узловых чисел.

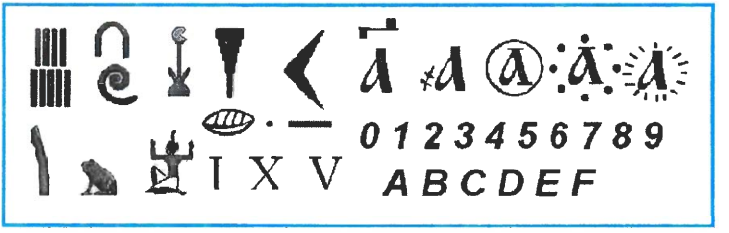


Рис. 1.1. Знаки, используемые для записи чисел в различных системах счисления

Пример 2. В древнеегипетской системе счисления числа 1, 2, 3, 4, 10, 13, 40 обозначались соответственно следующим образом:

|, ||, |||, ||||, ∩, ∩|||, ∩∩∩

Те же числа в римской системе счисления обозначаются так: I, II, III, IV, X, XIII, XL. Здесь алгоритмические числа получаются путём сложения и вычитания узловых чисел с учётом следующего правила: каждый меньший знак, поставленный справа от большего, прибавляется к его значению, а каждый меньший знак, поставленный слева от большего, вычитается из него.

Система счисления называется **позиционной**, если количественный эквивалент цифры в числе зависит от её положения в записи числа.
Основание позиционной системы счисления равно количеству цифр, составляющих её алфавит.

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. В ней алгоритмические числа образуются следующим образом: значения цифр умножаются на «веса» соответствующих разрядов и все полученные значения складываются. Это отчётливо прослеживается в числительных русского языка, например: «три-ста пять-десят семь».

Основанием позиционной системы счисления может служить любое натуральное число $q > 1$.

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

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

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

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

Здесь:

A — число;

q — основание системы счисления;

a_i — цифры, принадлежащие алфавиту данной системы счисления;

n — количество целых разрядов числа;

m — количество дробных 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 $\pm\, a_{n-1}a_{n-2}...a_1a_0,a_{-1}...a_{-m}$.[1]

[1] Далее будут рассматриваться только положительные целые числа.

Пример 3. Рассмотрим десятичное число 14351,1. 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:

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

1.1.2. Двоичная система 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.

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

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

Например:

$$10011_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 2^4 + 2^1 + 2^0 = 19_{10}.$$

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

Получим из формулы (1′) правило перевода целых десятичных чисел в двоичную систему счисления.

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

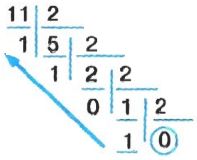
Полученное частное опять разделим на 2, остаток от деления будет равен a_1.

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

$$a_0, a_1, a_2, ..., a_{n-1},$$

которые входят в двоичное представление исходного числа и совпадают с остатками при его последовательном делении на 2. При записи исходного числа в двоичной системе счисления следует учитывать, что остатки от деления на 2 нами получены в порядке, обратном порядку расположения соответствующих цифр в двоичном представлении исходного числа.

Пример 4. Переведём десятичное число 11 в двоичную систему счисления. Рассмотренную выше последовательность действий (алгоритм перевода) можно изобразить так:



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

Пример 5. Если десятичное число достаточно большое, то более удобен следующий способ записи рассмотренного выше алгоритма:

363	181	90	45	22	11	5	2	1
1	1	0	1	0	1	1	0	1

$363_{10} = 101101011_2$

1.1.3. 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:

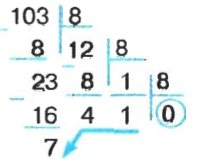
$$a_{n-1}a_{n-2}...a_1a_0 = a_{n-1} \cdot 8^{n-1} + a_{n-2} \cdot 8^{n-2} + ... + a_0 \cdot 8^0. \qquad (1'')$$

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

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 следует последовательно выполнять деление данного числа и получаемых целых частных на 8 до тех пор, пока не получим частное, равное нулю. Исходное число в новой системе счисления составляется последовательной записью полученных остатков, начиная с последнего.

Пример 6. Переведём десятичное число 103 в восьмеричную систему счисления.



$103_{10} = 147_8$

1.1.4. Шестнадцатеричная система счисления

Основание: $q = 16$.

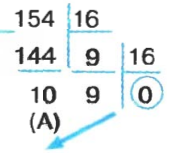
Алфавит: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Здесь только десять цифр из шестнадцати имеют общепринятое обозначение 0, ..., 9. Для записи цифр с десятичными количественными эквивалентами 10, 11, 12, 13, 14, 15 обычно используются первые пять букв латинского алфавита.

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

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

Пример 7. Переведём десятичное число 154 в шестнадцатеричную систему счисления.



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

1.1.5. Правило перевода целых десятичных чисел в систему счисления с основанием q

Для перевода целого десятичного числа в систему счисления с основанием q следует:

1) последовательно выполнять деление данного числа и получаемых целых частных на основание новой системы счисления до тех пор, пока не получим частное, равное нулю;
2) полученные остатки, являющиеся цифрами числа в новой системе счисления, привести в соответствие с алфавитом новой системы счисления;

3) составить число в новой системе счисления, записывая его, начиная с последнего полученного остатка.

Составим таблицу соответствия десятичных, двоичных, восьмеричных и шестнадцатеричных чисел от 0 до 20.

Десятичная система	Двоичная система	Восьмеричная система	Шестнадцатеричная система
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

В Единой коллекции цифровых образовательных ресурсов (http://school-collection.edu.ru/) размещена интерактивная анимация «Преобразование десятичного числа в другую систему счисления». С её помощью можно понаблюдать за переводом произвольного целого числа от 0 до 512 в позиционную систему счисления, основание которой не превышает 16.

В размещённой там же виртуальной лаборатории «Цифровые весы» вы сможете освоить ещё один способ перевода целых десятичных чисел в другие системы счисления — метод разностей.

1.1.6. 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

Пример 8. Таблица двоичного сложения предельно проста. Так как 1 + 1 = 10, то 0 остаётся в данном разряде, а 1 переносится в следующий разряд.

	1	0	0	1
+	1	0	1	0
1	0	0	1	1

	1	1	1	1
+				1
1	0	0	0	0

Пример 9. Операция умножения выполняется по обычной схеме, применяемой в десятичной системе счисления, с последовательным умножением множимого на очередную цифру множителя.

		1	0	1	1
	×		1	0	1
		1	0	1	1
	+				
1	0	1	1		
1	1	0	1	1	1

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

1.1.7. «Компьютерные» системы счисления

В компьютерной технике используется двоичная система счисления, обеспечивающая ряд преимуществ перед другими системами:

- 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 информацией 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 величины в восьмеричной или шестнадцатеричной системах счисления. В результате длина исходного слова сокращается в три, четыре раза соответственно. Это делает информацию более удобной для рассмотрения и анализа.

С помощью ресурса «Интерактивный задачник, раздел "Системы счисления"» (http://school-collection.edu.ru/) вы сможете проверить, насколько прочно вы усвоили изученный в этом параграфе материал.

www

САМОЕ ГЛАВНОЕ

Система счисления — это знаковая система, в которой приняты определённые правила записи чисел. Знаки, при помощи которых записываются числа, называются **цифрами**, а их совокупность — **алфавитом** системы счисления.

Система счисления называется **позиционной**, если количественный эквивалент цифры в числе зависит от её положения в записи числа. **Основание** позиционной системы счисления равно количеству цифр, составляющих её алфавит.

Основанием позиционной системы счисления может служить любое натуральное число $q > 1$.

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

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

Здесь:

A — число;

q — основание системы счисления;

a_i — цифры, принадлежащие алфавиту данной системы счисления;

n — количество целых разрядов числа;

m — количество дробных разрядов числа;

q^i — «вес» i-го разряда.

Вопросы и задания

1. Чем различаются унарные, позиционные и непозиционные системы счисления?
2. Цифры каких систем счисления приведены на рис. 1.1?
3. Объясните, почему позиционные системы счисления с основаниями 5, 10, 12 и 20 называют системами счисления анатомического происхождения.
4. Как от свёрнутой формы записи десятичного числа перейти к его развёрнутой форме?
5. Запишите в развёрнутом виде числа:
   а) $143{,}511_{10}$;
   б) 143511_8;
   в) 143511_{16};
   г) $1435{,}11_5$.
6. Запишите десятичные эквиваленты следующих чисел:
   а) 172_8;
   б) $2EA_{16}$;
   в) 101010_2;
   г) $10{,}1_2$;
   д) 243_6.
7. Укажите, какое из чисел 110011_2, 111_4, 35_8 и $1B_{16}$ является:
   а) наибольшим;
   б) наименьшим.

8. Какое минимальное основание имеет система счисления, если в ней записаны числа 123, 222, 111, 241? Определите десятичный эквивалент данных чисел в найденной системе счисления.
9. Верны ли следующие равенства?
   а) $33_4 = 21_7$;
   б) $33_8 = 21_4$.
10. Найдите основание x системы счисления, если:
   а) $14_x = 9_{10}$;
   б) $2002_x = 130_{10}$.
11. Переведите целые числа из десятичной системы счисления в двоичную:
   а) 89;
   б) 600;
   в) 2010.
12. Переведите целые числа из десятичной системы счисления в восьмеричную:
   а) 513;
   б) 600;
   в) 2010.
13. Переведите целые числа из десятичной системы счисления в шестнадцатеричную:
   а) 513;
   б) 600;
   в) 2010.
14. Заполните 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

15. Выполните операцию сложения над двоичными числами:

а) $101010 + 1101$;

б) $1010 + 1010$;

в) $10101 + 111$.

16. Выполните операцию умножения над двоичными числами:

а) $1010 \cdot 11$;

б) $111 \cdot 101$;

в) $1010 \cdot 111$.

17. Расставьте знаки арифметических операций так, чтобы были верны следующие равенства в двоичной системе:

а) $1100\ ?\ 11\ ?\ 100 = 100000$;

б) $1100\ ?\ 10\ ?\ 10 = 100$;

в) $1100\ ?\ 11\ ?\ 100 = 0$.

18. 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.

19. Какими преимуществами и недостатками обладает двоичная система счисления по сравнению с десятичной?

20. Разработайте таблицы сложения и умножения для восьмеричной системы счисления.

21. Постройте граф, отражающий разновидности систем счисления.

22. Подготовьте небольшое сообщение об одной из систем счисления (когда и где применялась, какие символы использовались и т. д.).

§ 1.2
Представление информации в компьютере

Ключевые слова:

- разряд
- беззнаковое представление целых чисел
- представление целых чисел со знаком
- представление вещественных чисел
- формат с плавающей запятой

1.2.1. Представление целых чисел

Память компьютера состоит из ячеек, каждая из которых представляет собой физическую систему, состоящую из некоторого числа 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 — разрядов двоичного числа. Именно поэтому каждый элемент ячейки называют **битом** или **разрядом** (рис. 1.2).

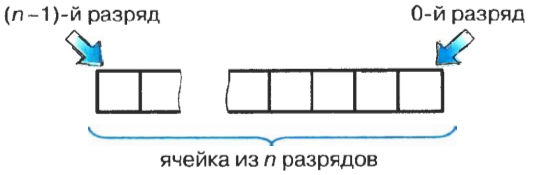


Рис. 1.2. Ячейка памяти

Для компьютерного представления целых чисел используется несколько различных способов представления, отличающихся друг от друга количеством разрядов (под целые числа обычно отводится 8,

16, 32 или 64 разрядов) и наличием или отсутствием знакового разряда. Беззнаковое представление можно использовать только для неотрицательных целых чисел, отрицательные числа представляются только в знаковом виде.

Широкое распространение в вычислительной технике получили беззнаковые данные. К ним относятся такие объекты, 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.

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

Ниже приведены максимальные значения для беззнаковых целых n-разрядных чисел:

Количество битов	Минимальное значение	Максимальное значение
8	0	255 ($2^8 - 1$)
16	0	65 535 ($2^{16} - 1$)
32	0	4 294 967 295 ($2^{32} - 1$)
64	0	18 446 744 073 709 551 615 ($2^{64} - 1$)

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

Пример 1. 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

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 — 1. Такое представление чисел называется **прямым кодом**. В компьютере прямые коды используются для хранения положительных чисел в запоминающих устройствах, для выполнения операций с положительными числами.

На сайте Федерального центра информационно-образовательных ресурсов (http://fcior.edu.ru/) размещён информационный модуль «Число и его компьютерный код». С помощью этого ресурса вы можете получить дополнительную информацию по изучаемой теме.

Для выполнения операций с отрицательными числами используется **дополнительный код**, позволяющий заменить операцию вычитания сложением. Узнать алгоритм образования дополнительного кода вы можете с помощью информационного модуля «Дополнительный код», размещённого на сайте Федерального центра информационно-образовательных ресурсов (http://fcior.edu.ru/).

1.2.2. Представление вещественных чисел

Любое вещественное число A может быть записано в **нормальной** (научной, экспоненциальной) форме:

$A = \pm m \cdot q^p$, где:

m — мантисса числа;

q — основание системы счисления;

p — порядок числа.

Например, число 472 000 000 может быть представлено так: $47{,}2 \cdot 10^7$, $472 \cdot 10^6$, $4720 \cdot 10^5$ и т. д.

С нормальной формой записи чисел вы могли встречаться при выполнении вычислений с помощью калькулятора, когда в качестве ответа получали записи следующего вида: 4.72E+8.

Здесь знак «E» обозначает основание десятичной системы счисления и читается как «умножить на десять в степени».

Из приведённого выше примера видно, что положение запятой в записи числа может изменяться. Поэтому представление в компьютере вещественных чисел в нормальной форме называется представлением в **формате с плавающей запятой**.

Для единообразия мантиссу обычно записывают как правильную дробь, имеющую после запятой цифру, отличную от нуля. В этом случае число 472 000 000 будет представлено как $0{,}472 \cdot 10^9$.

Число в формате с плавающей запятой может занимать в памяти компьютера 32 или 64 разряда. При этом выделяются разряды для хранения знака мантиссы, знака порядка, порядка и мантиссы.

Пример:

0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Знак и порядок								Знак и мантисса																					

Диапазон представления вещественных чисел определяется количеством разрядов, отведённых для хранения порядка числа, а точность определяется количеством разрядов, отведённых для хранения мантиссы.

Максимальное значение порядка числа, как видно из приведённого выше примера, составляет $1111111_2 = 127_{10}$, и, следовательно, максимальное значение числа:

$$0{,}11111111111111111111111 \cdot 10^{1111111}$$

Попытайтесь самостоятельно выяснить, каков десятичный эквивалент этой величины.

Широкий диапазон представления чисел в формате с плавающей запятой важен для решения научных и инженерных задач. Вместе с тем следует понимать, что алгоритмы обработки чисел в формате с плавающей запятой более трудоёмки по сравнению с алгоритмами обработки целых чисел.

САМОЕ ГЛАВНОЕ

Для компьютерного представления целых чисел используются несколько различных 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 стандартной 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. Положительные числа хранятся в компьютере в прямом коде, отрицательные — в дополнительном.

Вещественные числа в компьютере хранятся в формате с плавающей запятой. При этом любое число записывается так:

$A = \pm m \cdot q^p$, где:

m — мантисса числа;

q — основание системы счисления;

p — порядок числа.

Вопросы и задания

1. Как в памяти компьютера представляются целые положительные и отрицательные числа?
2. Любое целое число можно рассматривать как вещественное, но с нулевой дробной частью. Обоснуйте целесообразность наличия особых способов компьютерного представления целых чисел.
3. Представьте число 63_{10} в беззнаковом 8-разрядном формате.
4. Найдите десятичные эквиваленты чисел по их прямым кодам, записанным в 8-разрядном формате со знаком:

   а) 01001100;

   б) 00010101.
5. Какие из чисел 443_8, 101010_2, 256_{10} можно сохранить в 8-разрядном формате?
6. Запишите следующие числа в естественной форме:

   а) $0{,}3800456 \cdot 10^2$;

   б) $0{,}245 \cdot 10^{-3}$;

   а) 1,256900E+5;

   а) 9,569120E–3.
7. Запишите число $2010{,}0102_{10}$ пятью различными способами в нормальной форме.
8. Запишите следующие числа в нормальной форме с нормализованной мантиссой — правильной дробью, имеющей после запятой цифру, отличную от нуля:

   а) $217{,}934_{10}$;

   б) 75321_{10};

   в) $0{,}00101_{10}$.
9. Изобразите схему, связывающую основные понятия, рассмотренные в данном параграфе.

§ 1.3
Элементы алгебры логики

Ключевые слова:

- алгебра логики
- высказывание
- логическая операция
- конъюнкция
- дизъюнкция
- отрицание
- логическое выражение
- таблица истинности
- законы логики

1.3.1. Высказывание

Алгебра в широком смысле этого слова — 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*» мож-

но 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. Например, предложение «$X < 12$» становится высказыванием только при замене переменной каким-либо конкретным значением: «$5 < 12$» — истинное высказывание; «$12 < 12$» — ложное высказывание.

Обоснование истинности или ложности высказываний решается теми науками, к сфере которых они относятся. Алгебра логики отвлекается от смысловой содержательности высказываний. Её интересует только то, 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, вычисления значений, упрощения и преобразования высказываний.

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 информации. С применением элементов алгебры логики вы будете встречаться и во многих других разделах информатики.

1.3.2. Логические операции

Высказывания бывают простые и сложные. Высказывание называется **простым**, если никакая его часть сама не является высказыванием. **Сложные** (составные) высказывания строятся из простых с помощью логических операций.

Рассмотрим основные логические операции, определённые над высказываниями. Все они соответствуют связкам, употребляемым в естественном языке.

Название логической операции	Логическая связка
Конъюнкция	«и»; «а»; «но»; «хотя»
Дизъюнкция	«или»
Инверсия	«не»; «неверно, что»

Конъюнкция

Рассмотрим два высказывания: A = «*Основоположником алгебры логики является Джордж Буль*», B = «*Исследования Клода Шеннона позволили применить алгебру логики в вычислительной технике*». Очевидно, новое высказывание «*Основоположником алгебры логики является Джордж Буль, и исследования Клода Шеннона позволили применить алгебру логики в вычислительной технике*» истинно только в том случае, когда одновременно истинны оба исходных высказывания.

Самостоятельно установите истинность или ложность трёх рассмотренных высказываний.

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

Для записи конъюнкции используются следующие знаки: $\wedge$, $\cdot$, И, &. Например: $A \wedge B$, $A \cdot B$, A И B, $A\&B$.

Конъюнкцию можно описать в виде таблицы, которую называют **таблицей истинности:**

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

В таблице истинности перечисляются все возможные значения исходных высказываний (столбцы A и B), причём соответствующие им двоичные числа, как правило, располагают в порядке возрастания: 00, 01, 10, 11. В последнем столбце записан результат выполнения логической операции для соответствующих операндов.

Иначе конъюнкцию называют логическим умножением.
Подумайте почему.

Дизъюнкция

Рассмотрим два высказывания: A = «*Идея использования в логике математической символики принадлежит Готфриду Вильгельму Лейбницу*», B = «*Лейбниц является основоположником бинарной арифметики*». Очевидно, новое высказывание «*Идея использования в логике математической символики принадлежит Готфриду Вильгельму Лейбницу или Лейбниц является основоположником бинарной арифметики*» ложно только в том случае, когда одновременно ложны оба исходных высказывания.

Самостоятельно установите истинность или ложность трёх рассмотренных высказываний.

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

Для записи дизъюнкции используются следующие знаки: $\vee$, $|$, ИЛИ, $+$. Например: $A \vee B$, $A|B$, A ИЛИ B, $A+B$.

Дизъюнкция определяется следующей таблицей истинности:

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

Иначе дизъюнкцию называют логическим сложением. Подумайте почему.

Инверсия

Инверсия — логическая операция, которая каждому высказыванию ставит в соответствие новое высказывание, значение которого противоположно исходному.

Для записи инверсии используются следующие знаки: НЕ, $\neg$, $\bar{}$. Например: НЕ , $\neg$, $\bar{}$.

Инверсия определяется следующей таблицей истинности:

A	$\overline{A}$
0	1
1	0

Инверсию иначе называют логическим отрицанием.

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 одно и то же, «*Я знаю китайский язык*». Отрицанием высказывания «*Все юноши 9-х классов — отличники*» является высказывание «Неверно, *что все юноши 9-х классов — отличники*», другими словами, «*Не все юноши 9-х классов — отличники*».

Таким образом, при построении отрицания к простому высказыванию либо используется речевой оборот «неверно, что ...», либо отрицание строится к сказуемому, тогда к соответствующему глаголу добавляется частица «не».

Любое сложное высказывание можно записать в виде **логического выражения** — выражения, содержащего логические переменные, знаки логических операций и скобки. Логические операции в логическом выражении выполняются в следующей очерёдности: инверсия, конъюнкция, дизъюнкция. Изменить порядок выполнения операций можно с помощью расстановки скобок.

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

Пример 1. Пусть A = «*На Web-странице встречается слово "крейсер"*», B = «*На Web-странице встречается слово "линкор"*». Рассматривается некоторый сегмент сети Интернет, содержащий 5 000 000 Web-страниц. В нём высказывание A истинно для 4800 страниц, высказывание B — для 4500 страниц, а высказывание $A \vee B$ — для 7000 страниц. Для какого количества Web-страниц в этом случае будут истинны следующие выражения и высказывание?

а) НЕ (A ИЛИ B);

б) A & B;

в) *На Web-странице встречается слово "крейсер" И не встречается слово "линкор"*.

Решение. Изобразим множество всех Web-страниц рассматриваемого сектора сети Интернет кругом, внутри которого разместим два круга: одному из них соответствует множество Web-страниц, где истинно высказывание A, второму — где истинно высказывание B (рис. 1.3).

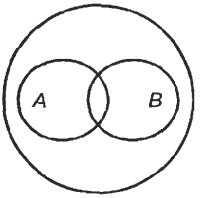


Рис. 1.3. Графическое изображение множеств Web-страниц

Изобразим графически множества Web-страниц, для которых истинны выражения и высказывание а) – в) (рис. 1.4)

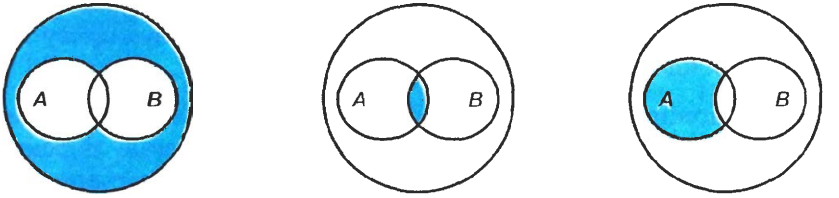


Рис. 1.4. Графическое изображение множеств Web-страниц, для которых истинны выражения и высказывание а) – в)

Построенные схемы помогут нам ответить на вопросы, содержащиеся в задании.

Выражение A ИЛИ B истинно для 7000 Web-страниц, а всего страниц 5 000 000. Следовательно, выражение A ИЛИ B ложно для 4 993 000 Web-страниц. Иначе говоря, для 4 993 000 Web-страниц истинно выражение НЕ (A ИЛИ B).

Выражение $A \vee B$ истинно для тех Web-страниц, где истинно A (4800), а также тех Web-страниц, где истинно B (4500). Если бы все Web-страницы были различны, то выражение $A \vee B$ было бы истинно для 9300 (4800 + 4500) Web-страниц. Но, согласно условию, таких Web-страниц всего 7000. Это значит, что на 2300 (9300 – 7000)

Web-страницах встречаются оба слова одновременно. Следовательно, выражение A & B истинно для 2300 Web-страниц.

Чтобы выяснить, для скольких Web-страниц истинно высказывание A и одновременно ложно высказывание B, следует из 4800 вычесть 2300. Таким образом, высказывание *«На Web-странице встречается слово "крейсер" И не встречается слово "линкор"»* истинно на 2500 Web-страницах.

Самостоятельно запишите логическое выражение, соответствующее рассмотренному высказыванию.

На сайте Федерального центра информационно-образовательных ресурсов (http://fcoir.edu.ru/) размещён информационный модуль «Высказывание. Простые и сложные высказывания. Основные логические операции». Знакомство с этим ресурсом позволит вам расширить представления по изучаемой теме.

1.3.3. Построение таблиц истинности для логических выражений

Для логического выражения можно построить таблицу истинности, показывающую, 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) подсчитать общее число логических операций в выражении;
3) установить последовательность выполнения логических операций с учётом скобок и приоритетов;
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. 3;
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 целый ряд n–разрядных двоичных чисел от 0 до $2^n - 1$;
8) провести заполнение таблицы по столбцам, выполняя логические операции в соответствии с установленной последовательностью.

Построим таблицу истинности для логического выражения $A \vee A \,\&\, B$. В нём две переменные, две операции, причём сначала выполняется конъюнкция, а затем — дизъюнкция. Всего в таблице будет четыре столбца:

A	B	$A\&B$	$A \vee A\&B$

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

Заполненная таблица истинности имеет вид:

A	B	$A\&B$	$A \vee A\&B$
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 таком случае говорят, что логическое выражение $A \vee A \,\&\, B$ **равносильно** логическому выражению A.

1.3.4. Свойства логических операций

Рассмотрим основные свойства **(законы)** алгебры логики.

1. **Переместительный (коммутативный) закон**
   - для логического умножения:
   $$A \,\&\, B = B \,\&\, A;$$
   - для логического сложения:
   $$A \vee B = B \vee A.$$
2. **Сочетательный (ассоциативный) закон**
   - для логического умножения:
   $$(A \,\&\, B) \,\&\, C = A \,\&\, (B \,\&\, C);$$
   - для логического сложения:
   $$(A \vee B) \vee C = A \vee (B \vee C).$$

При одинаковых знаках операций скобки можно ставить произвольно или вообще опускать.

3. **Распределительный (дистрибутивный) закон**
   - для логического умножения:
   $$A \,\&\, (B \vee C) = (A \,\&\, B) \vee (A \,\&\, C);$$

- для логического сложения:

$$A \vee (B \,\&\, C) = (A \vee B) \,\&\, (A \vee C).$$

4. Закон двойного отрицания

$$\overline{\overline{A}} = A.$$

Двойное отрицание исключает отрицание.

5. Закон исключения третьего

- для логического умножения:

$$A \,\&\, \overline{A} = 0;$$

- для логического сложения:

$$A \vee \overline{A} = 1.$$

Из двух противоречивых высказываний об одном и том же предмете одно всегда истинно, а второе — ложно, третьего не дано.

6. Закон повторения

- для логического умножения:

$$A \,\&\, A = A;$$

- для логического сложения:

$$A \vee A = A.$$

7. Законы операций с 0 и 1

- для логического умножения:

$$A \,\&\, 0 = 0;\ A \,\&\, 1 = A;$$

- для логического сложения:

$$A \vee 0 = A;\ A \vee 1 = 1.$$

8. Законы общей инверсии

- для логического умножения:

$$\overline{A \,\&\, B} = \overline{A} \vee \overline{B};$$

- для логического сложения:

$$\overline{A \vee B} = \overline{A} \,\&\, \overline{B}.$$

Законы алгебры логики могут быть доказаны с помощью таблиц истинности.

Докажем распределительный закон для логического сложения:

$$A \vee (B \,\&\, C) = (A \vee B) \,\&\, (A \vee C).$$

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

A	B	C	$B \& C$	$A \vee (B \& C)$	$A \vee B$	$A \vee C$	$(A \vee B) \& (A \vee C)$
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

Пример 2. Найдём значение логического выражения $(X < 3) \& \overline{(X < 2)}$ для числа $X = 0$.

Решение. При $X = 0$ получаем следующее логическое выражение: $(0 < 3) \& \overline{(0 < 2)}$. Так как логические выражения $0 < 3$, $0 < 2$ истинны, то, подставив их значения в логическое выражение, получаем: $1 \& \overline{1} = 1 \& 0 = 0$.

1.3.5. Решение логических задач

Рассмотрим несколько способов решения логических задач.

Задача 1. 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*». Составим таблицу истинности, с которой представим высказывания каждого мальчика[1].

[1] С учётом того, что ваза разбита одним внуком, можно было составлять не всю таблицу, а только её фрагмент, содержащий следующие наборы входных 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	0	1	1	1	0	1	0	
0	0	1	0	1	0	0	1	1	✓
0	1	0	1	0	1	0	1	0	
0	1	1	0	0	0	0	1	1	
1	0	0	1	1	1	1	0	0	
1	0	1	0	1	0	1	0	1	
1	1	0	1	0	1	1	0	0	✓
1	1	1	0	0	0	1	0	1	

Исходя из того, что знает о внуках бабушка, следует искать в таблице строки, содержащие в каком-либо порядке три комбинации значений: 00, 11, 01 (или 10). Таких строк в таблице оказалось две (они отмечены галочками). Согласно второй из них, вазу разбили Коля и Вася, что противоречит условию. Согласно первой из найденных строк, вазу разбил Серёжа, он же оказался хитрецом. Шутником оказался Вася. Имя правдивого внука — Коля.

Задача 2. 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 места. 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.

Познакомиться с другими способами решения логических задач, а также принять участие в Интернет-олимпиадах и конкурсах по их решению вы сможете на сайте «Математика для школьников» (http://www.kenqyry.com/).

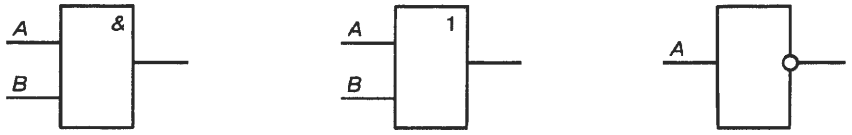
На сайте http://www.kaser.com/ вы сможете скачать демонстрационную версию очень полезной, развивающей логику и умение рассуждать логической головоломки Шерлок.

1.3.6. 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**.

На рис. 1.5 приведены условные обозначения (схемы) логических элементов, реализующих логическое умножение, логическое сложение и инверсию.



а *б* *в*

Рис 1.5. Логические элементы

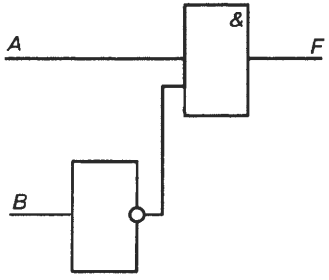
Логический элемент **И (конъюнктор)** реализует операцию логического умножения (рис. 1.5, *а*). Единица на выходе этого элемента появится только тогда, когда на всех входах будут единицы.

Логический элемент **ИЛИ (дизъюнктор)** реализует операцию логического сложения (рис. 1.5, *б*). Если хотя бы на одном входе будет единица, то на выходе элемента также будет единица.

Логический элемент **НЕ (инвертор)** реализует операцию отрицания (рис. 1.5, *в*). Если на входе элемента 0, то на выходе 1 и наоборот.

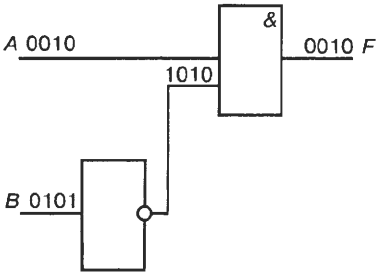
Компьютерные устройства, производящие операции над двоичными числами, и ячейки, хранящие данные, представляют собой электронные схемы, состоящие из отдельных логических элементов. Более подробно эти вопросы будут раскрыты в курсе информатики 10–11 классов.

Пример 3. Проанализируем электронную схему, т. е. выясним, какой сигнал должен быть на выходе при каждом возможном наборе сигналов на входах.



Решение. Все возможные комбинации сигналов на входах A и B 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 электронную схему.



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

Таблицу истинности можно построить и по логическому выражению, соответствующему электронной 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, $F = A \,\&\, \overline{B}$.

www Составить более полное представление о логических элементах и электронных схемах вам поможет работа с тренажёром «Логика» (http://kpolyakov.narod.ru/prog/logic.htm).

САМОЕ ГЛАВНОЕ

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

Основные **логические операции**, определённые над высказываниями: **инверсия, конъюнкция, дизъюнкция.**

Название логической операции	Логическая связка	Обозначение
Инверсия	«не», «неверно, что»	¬, —
Конъюнкция	«и», «а», «но», «хотя»	&
Дизъюнкция	«или»	∨

Таблицы истинности для основных логических операций:

A	$\overline{A}$
0	1
1	0

A	B	$A \,\&\, B$	$A \vee B$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

При вычислении логических выражений сначала выполняются действия в скобках. Приоритет выполнения логических операций: $\neg$, &, $\vee$.

Вопросы и задания

1. Объясните, почему следующие предложения не являются высказываниями.
   1) Какого цвета этот дом?
   2) Число X не превосходит единицы.
   3) $4X + 3$.
   4) Посмотрите в окно.
   5) Пейте томатный сок!
   6) Эта тема скучна.
   7) Рикки Мартин — самый популярный певец.
   8) Вы были в театре?
2. Приведите по одному примеру истинных и ложных высказываний из биологии, географии, информатики, истории, математики, литературы.
3. В следующих высказываниях выделите простые высказывания, обозначив каждое из них буквой; запишите с помощью букв и знаков логических операций каждое составное высказывание.
   1) Число 376 чётное и трёхзначное.
   2) Зимой дети катаются на коньках или на лыжах.
   3) Новый год мы встретим на даче или на Красной площади.
   4) Неверно, что Солнце движется вокруг Земли.
   5) Земля имеет форму шара, который из космоса кажется голубым.

6) На уроке математики старшеклассники отвечали на вопросы учителя, а также писали самостоятельную работу.

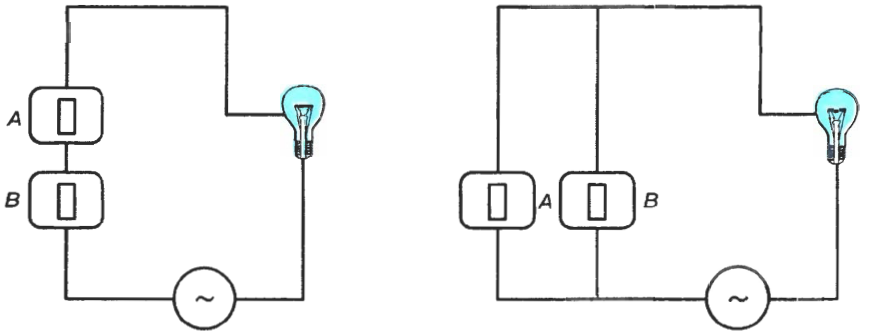
4. Постройте отрицания следующих высказываний.

1) Сегодня в театре идёт опера «Евгений Онегин».
2) Каждый охотник желает знать, где сидит фазан.
3) Число 1 есть простое число.
4) Натуральные числа, оканчивающиеся цифрой 0, не являются простыми числами.
5) Неверно, что число 3 не является делителем числа 198.
6) Коля решил все задания контрольной работы.
7) Во всякой школе некоторые ученики интересуются спортом.
8) Некоторые млекопитающие не живут на суше.

5. Пусть A = *«Ане нравятся уроки математики»*, а B = *«Ане нравятся уроки химии»*. Выразите следующие формулы на обычном языке:

1) $A \,\&\, B$;
2) $\overline{A} \,\&\, \overline{B}$;
3) $A \,\&\, \overline{B}$;
4) $A \vee B$;
5) $\overline{A} \vee \overline{B}$;
6) $\overline{A} \vee \overline{B}$;
7) $\overline{(A \,\&\, B)}$;
8) $\overline{(A \vee B)}$;
9) $(A \,\&\, \overline{B})$.

6. Рассмотрите представленные на рисунке электрические схемы:



На них изображены известные вам из курса физики параллельное и последовательное соединения переключателей. В первом случае, чтобы лампочка загорелась, должны быть включены оба переключателя. Во втором случае достаточно, чтобы был включён один из переключателей. Попытайтесь самостоятельно провести аналогию между элементами электрических схем и объектами и операциями алгебры логики:

Электрическая схема	Алгебра логики
Переключатель	
Переключатель включён	
Переключатель выключен	
Последовательное соединение переключателей	
Параллельное соединение переключателей	

7. Некоторый сегмент сети Интернет состоит из 1000 сайтов. Поисковый сервер в автоматическом режиме составил таблицу ключевых слов для сайтов этого сегмента. Вот её фрагмент:

Ключевое слово	Количество сайтов, для которых данное слово является ключевым
сомики	250
меченосцы	200
гуппи	500

По запросу сомики & гуппи было найдено 0 сайтов, по запросу сомики & меченосцы — 20 сайтов, а по запросу меченосцы & гуппи — 10 сайтов.

Сколько сайтов будет найдено по запросу сомики | меченосцы | гуппи?

Для скольких сайтов рассматриваемого сегмента ложно высказывание «*Сомики — ключевое слово сайта ИЛИ меченосцы — ключевое слово сайта ИЛИ гуппи — ключевое слово сайта*»?

8. Постройте таблицы истинности для следующих логических выражений:

1) $B \,\&\, (A \vee B)$;

2) $A \,\&\, (B \vee \overline{B})$;

3) $A \,\&\, (A \vee B \vee C)$;

4) $\overline{A \vee B \vee \overline{C}}$.

9. Проведите доказательство рассмотренных в параграфе логических законов с помощью таблиц истинности.

10. Даны три числа в десятичной системе счисления: $A = 23$, $B = 19$, $C = 26$. Переведите A, B и C в двоичную систему счисления и выполните поразрядно логические операции $(A \vee B) \,\&\, C$. Ответ дайте в десятичной системе счисления.

11. Найдите значения выражений:

1) $(1 \vee 1) \vee (1 \vee 0)$;

2) $((1 \vee 0) \vee 1) \vee 1)$;

3) $(0 \,\&\, 1) \,\&\, 1$;

4) $1 \,\&\, (1 \,\&\, 1) \,\&\, 1$;

5) $((1 \vee 0) \,\&\, (1 \,\&\, 1)) \,\&\, (0 \vee 1)$;

6) $((1 \,\&\, 1) \vee 0) \,\&\, (0 \vee 1)$;

7) $((0 \,\&\, 0) \vee 0) \,\&\, (1 \vee 1)$;

8) $(A \vee 1) \vee (B \vee 0)$;

9) $((1 \,\&\, A) \vee (B \,\&\, 0)) \vee 1$;

10) $1 \vee A \,\&\, 0$.

12. Найдите значение логического выражения $\overline{(X < 3)} \,\&\, \overline{(X < 2)}$ для указанных значений числа X:

1) 1 2) 2 3) 3 4) 4

13. Пусть A = «*Первая буква имени — гласная*», B = «*Четвёртая буква имени согласная*». Найдите значение логического выражения $\overline{A} \vee B$ для следующих имён:

1) ЕЛЕНА 2) ВАДИМ 3) АНТОН 4) ФЁДОР

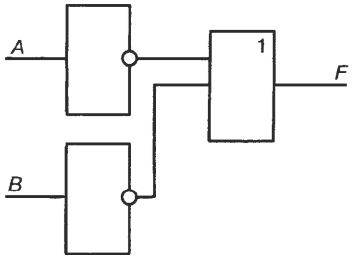
14. Разбирается дело Джона, Брауна и Смита. Известно, что один из них нашёл и утаил клад. На следствии каждый из подозреваемых сделал два заявления:

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?

15. Алёша, Боря и Гриша нашли в земле старинный сосуд. Рассматривая удивительную находку, каждый высказал по два предположения:

1) Алеша: «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?

16. Выясните, какой сигнал должен быть на выходе электронной схемы при каждом возможном наборе сигналов на входах. Составьте таблицу работы схемы. Каким логическим выражением описывается схема?



Тестовые задания для самоконтроля

1. Совокупность знаков, при помощи 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. Число 301011 может существовать 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. Чему равен результат сложения чисел 110_2 и 12_8?

а) 6_{10}
б) 10_{10}
в) 10000_2
г) 17_8

8. Ячейка памяти компьютера состоит из однородных элементов, называемых:

а) кодами
б) разрядами
в) цифрами
г) коэффициентами

9. Количество разрядов, занимаемых двухбайтовым числом, равно:

а) 8
б) 16
в) 32
г) 64

10. В знаковый разряд ячейки для отрицательных чисел заносится:

а) +
б) –
в) 0
г) 1

11. Вещественные числа представляются в компьютере в:

а) естественной форме
б) развёрнутой форме
в) нормальной форме с нормализованной мантиссой
г) виде обыкновенной дроби

12. Какое предложение не является высказыванием?

а) Никакая причина не извиняет невежливость

б) Обязательно стань отличником

в) Рукописи не горят

г) $1011_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

13. Какое высказывание является ложным?

а) Знаком $\vee$ обозначается логическая операция ИЛИ

б) Логическую операцию ИЛИ иначе называют логическим сложением

в) Дизъюнкцию иначе называют логическим сложением

г) Знаком $\vee$ обозначается логическая операция конъюнкция

14. 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

15. Для какого символьного выражения верно высказывание: «*НЕ (Первая буква согласная) И НЕ (Вторая буква гласная)*»?

а) *abcde*

б) *bcade*

в) *babas*

г) *cabab*

16. Некоторый сегмент сети Интернет состоит из 1000 сайтов. Поисковый сервер в автоматическом режиме составил таблицу ключевых слов для сайтов этого сегмента. Вот её фрагмент:

Ключевое слово	Количество сайтов, для которых данное слово является ключевым
сканер	200
принтер	250
монитор	450

Сколько сайтов будет найдено по запросу принтер | сканер | монитор, если по запросу принтер | сканер было найдено 450 сайтов, по запросу принтер & монитор — 40, а по запросу сканер & монитор — 50?

а) 900
б) 540
в) 460
г) 810

17. Какому логическому выражению соответствует следующая таблица истинности?

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

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

18. Когда сломался компьютер, его хозяин сказал: «Оперативная память не могла выйти из строя». Сын хозяина компьютера предположил, что сгорел процессор, а жёсткий диск исправен. Пришедший специалист по обслуживанию сказал, что, скорее всего, с процессором всё в порядке, а оперативная память неисправна. В результате оказалось, что двое из них сказали всё верно, а третий — всё неверно. Что же сломалось?

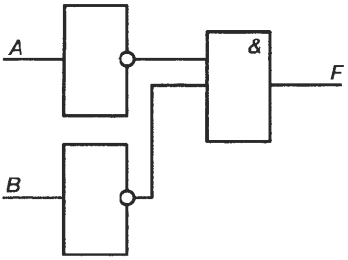
а) оперативная память
б) процессор
в) винчестер
г) процессор и оперативная память

19. 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 маши-

ны 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

20. Какое логическое выражение соответствует следующей схеме?



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

Глава 2
МОДЕЛИРОВАНИЕ И ФОРМАЛИЗАЦИЯ

§ 2.1
Моделирование как метод познания

Ключевые слова:
- модель
- моделирование
- цель моделирования
- натурная (материальная) модель
- информационная модель
- формализация
- классификация информационных моделей

2.1.1. Модели и моделирование

Человек стремится познать объекты (предметы, процессы, явления) окружающего мира, т. е. понять, как устроен конкретный объект, каковы его структура, основные свойства, законы развития и взаимодействия с другими объектами. Для решения многих практических задач важно знать:
- как изменятся признаки объекта при определённом воздействии на него со стороны других объектов («Что будет, если...?»);
- какое надо произвести воздействие на объект, чтобы изменить его признаки в соответствии с новыми требованиями («Как сделать, чтобы...?»);
- какое сочетание свойств объекта является наилучшим в заданных условиях («Как сделать лучше?»).

Одним из методов познания объектов окружающего мира является **моделирование**, состоящее в создании и исследовании упрощённых заменителей реальных объектов. Объект-заменитель принято назы-

вать **моделью**, а исходный объект — прототипом или оригиналом. Примеры моделей приведены на рис. 2.1.

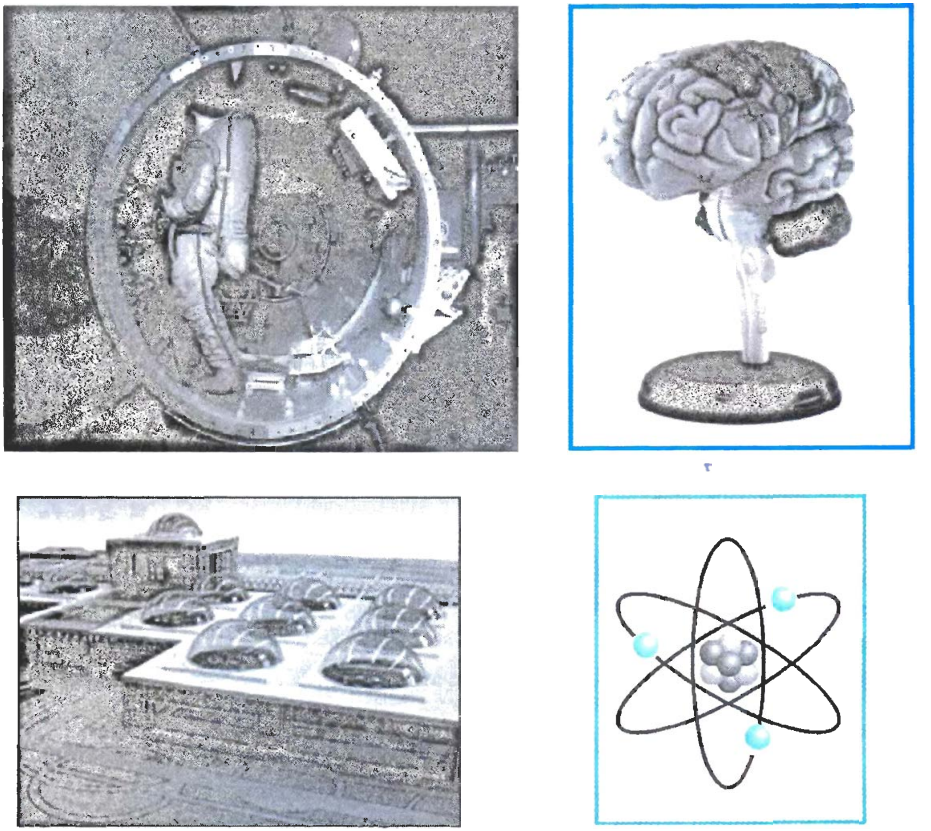


Рис. 2.1. Примеры моделей

К созданию моделей прибегают, когда исследуемый объект слишком велик (Солнечная система) или слишком мал (атом), когда процесс протекает очень быстро (переработка топлива в двигателе внутреннего сгорания) или очень медленно (геологические процессы), когда исследование объекта может оказаться опасным для окружающих (атомный взрыв), привести к разрушению его самого (проверка сейсмических свойств высотного здания) или когда создание реального объекта очень дорого (новое архитектурное решение) и т. д.

Модель не является точной копией объекта-оригинала: она отражает только часть его свойств, отношений и особенностей поведения.

Чем больше признаков объекта отражает модель, тем она полнее. Однако отразить в модели все признаки объекта-оригинала невозможно, а чаще всего и не нужно. Признаки объекта-оригинала, которые должны быть воспроизведены в модели, определяются **целью моделирования** — назначением будущей модели. Эти признаки называются существенными для данной модели с точки зрения цели моделирования.

Подумайте, какие признаки объекта «театр» будут существенными при создании его модели с точки зрения: 1) строительной компании, занимающейся возведением здания театра; 2) режиссёра, готовящего постановку нового спектакля; 3) кассира, продающего билеты; 4) зрителя, собирающегося посетить представление.

Модель — это новый объект, который отражает существенные с точки зрения цели моделирования признаки изучаемого предмета, процесса или явления.

Моделирование — метод познания, заключающийся в создании и исследовании моделей.

Поскольку любая модель всегда отражает только часть признаков оригинала, можно создавать и использовать разные модели одного и того же объекта. Например: мяч может воспроизвести только одно свойство Земли — её форму, обычный глобус отражает ещё расположение материков, а глобус, входящий в состав действующей модели Солнечной системы, — ещё и траекторию движения Земли вокруг Солнца.

Отразить в модели признаки оригинала можно разными способами.

Во-первых, признаки можно скопировать, воспроизвести. Такую модель называют **натурной (материальной)**. Примерами натурных моделей являются муляжи и макеты — уменьшенные или увеличенные копии, воспроизводящие внешний вид моделируемого объекта (глобус), его структуру (модель Солнечной системы) или поведение (радиоуправляемая модель автомобиля).

Во-вторых, признаки оригинала можно описать на одном из языков кодирования информации — дать словесное описание, привести формулу, схему или чертеж и т. д. Такую модель называют **информационной**. В дальнейшем мы будем рассматривать именно информационные модели.

Информационная модель — описание объекта-оригинала на одном из языков кодирования информации.

2.1.2. Этапы построения информационной модели

Любая модель строится для решения некоторой задачи. Построение информационной модели начинается с анализа условия этой задачи, выраженного на естественном языке (рис. 2.2).

В результате анализа условия задачи определяется объект моделирования и цель моделирования.

После определения цели моделирования в объекте моделирования выделяются свойства, основные части и связи между ними, существенные с точки зрения именно этой цели. При этом должно быть чётко определено, что дано (какие исходные данные известны, какие данные допустимы) и что требуется найти в решаемой задаче. Также должны быть указаны связи между исходными данными и результатами.

Следующим этапом построения информационной модели является формализация — представление выявленных связей и выделенных существенных признаков объекта моделирования в некоторой форме (словесное описание, таблица, рисунок, схема, чертёж, формула, алгоритм, компьютерная программа и т. д.).

Формализация — это замена реального объекта его формальным описанием, т. е. его информационной моделью.

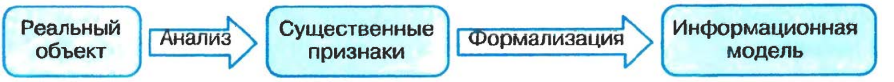


Рис. 2.2. Этапы создания информационной модели

Пример. Ученик 9 класса к уроку литературы должен выучить наизусть три первые строфы первой главы романа А. С. Пушкина «Евгений Онегин», содержащие 42 строки. Сколько ему потребуется времени на выполнение этого задания, если первую строку он может запомнить за 5 секунд, а на запоминание каждой следующей строки ему требуется времени на 10 секунд больше, чем на запоминание предыдущей строки?

В данном случае объектом моделирования является процесс запоминания стихотворения учеником; цель моделирования состоит в том, чтобы получить формулу для расчёта времени, необходимого ученику для заучивания стихотворения.

С точки зрения цели моделирования, существенной является следующая информация: время запоминания первой строки (5 секунд);

разница во времени запоминания очередной и предыдущей строк (10 секунд); количество строк, подлежащих запоминанию (42 строки). Это исходные данные. Результатом является время, необходимое для заучивания всех 42 строк фрагмента романа.

Так как время для заучивания каждой строки, начиная со второй, получается добавлением ко времени, требуемому для заучивания пре- дыдущей строки, постоянного числа, то можно говорить об арифметической прогрессии:

5, 15, 25, 35, ...

Первым членом этой прогрессии является $a_1 = 5$, разность прогрессии $d = 10$, число членов прогрессии $n = 42$.

Из курса алгебры известна формула для вычисления суммы n первых членов арифметической прогрессии:

$$S_n = \frac{2a_1 + d(n-1)}{2} n.$$

Эта формула и является искомой информационной моделью. С её помощью самостоятельно вычислите время, необходимое ученику для заучивания стихотворения.

Информационные модели существуют отдельно от объектов моделирования и могут подвергаться обработке независимо от них. Построив информационную модель, человек использует её вместо объекта-оригинала для исследования этого объекта, решения поставленной задачи.

По адресу http://earth.google.com/intl/ru/ размещено приложение «Google Планета Земля», предоставляющее возможность путешествовать по нашей планете, не вставая с кресла. Это трёхмерная модель планеты, перемещаясь по которой вы можете: просматривать спутниковые фотографии земной поверхности; осматривать города, отдельные здания и всемирно известные достопримечательности в трёхмерном изображении; исследовать отдалённые галактики, созвездия и планеты; совершать путешествия в прошлое и т. д.

2.1.3. Классификация информационных моделей

Существует множество вариантов классификации информационных моделей. Рассмотрим некоторые из них.

Если взять за основу классификации предметную область, то можно выделить физические, экологические, экономические, социологические и другие модели.

В зависимости от учёта фактора времени выделяют динамические (изменяющиеся с течением времени) и статические (не изменяющиеся с течением времени) модели.

В зависимости от формы представления информации об объекте моделирования различают знаковые, образные и смешанные (образно-знаковые) виды информационных моделей.

Знаковые информационные модели строятся с использованием различных естественных и формальных языков (знаковых систем). Знаковая информационная модель может быть представлена в форме текста на естественном языке или программы на языке программирования, в виде формулы и т. д.

Образные информационные модели (рисунки, фотографии и др.) представляют собой зрительные образы объектов, зафиксированные на каком-либо носителе информации.

В смешанных информационных моделях сочетаются образные и знаковые элементы. Примерами смешанных информационных моделей могут служить географические карты, графики, диаграммы и пр. Во всех этих моделях используются одновременно и графические элементы, и знаки.

САМОЕ ГЛАВНОЕ

Модель — это новый объект, который отражает существенные с точки зрения цели моделирования признаки изучаемого предмета, процесса или явления.

Моделирование — метод познания, заключающийся в создании и исследовании моделей.

Цель моделирования (назначение будущей модели) определяет признаки объекта-оригинала, которые должны быть воспроизведены в модели.

Различают натурные и информационные модели. **Натурные модели** — реальные предметы, в уменьшенном или увеличенном виде воспроизводящие внешний вид, структуру или поведение моделируемого объекта. **Информационные модели** — описания объекта-оригинала на одном из языков кодирования информации.

Формализация — процесс замены реального объекта его формальным описанием, т. е. его информационной моделью.

По форме представления различают образные, знаковые и смешанные (образно-знаковые) информационные модели.

Вопросы и задания

1. Что такое модель? В каких случаях используется моделирование?
2. Подтвердите на примерах справедливость следующих высказываний:

   а) одному объекту может соответствовать несколько моделей;
   б) одна модель может соответствовать нескольким объектам.
3. Приведите примеры натурных и информационных моделей.
4. В приведённом перечне моделей укажите те, которые могут использоваться для:

   а) представления объектов окружающего мира;
   б) объяснения известных фактов;
   в) проверки гипотез и получения новых знаний об исследуемых объектах;
   г) прогнозирования;
   д) управления.

   Модели: макет застройки жилого района; фотоснимки движения воздушных масс; расписание движения поездов; модель полёта самолёта новой конструкции в аэродинамической трубе; схема строения внутренних органов человека.
5. Приведите пример информационной модели:

   а) ученика вашего класса;
   б) игрока баскетбольной команды;
   в) пациента ветеринарной лечебницы;
   г) квартиры жилого дома;
   д) книги в библиотеке;
   е) кассеты (диска) со звукозаписью (видеозаписью);
   ж) города.
6. Опишите этапы построения информационной модели. В чём суть этапа формализации?
7. Перечислите виды информационных моделей в зависимости от формы представления информации об объекте моделирования. Приведите примеры информационных моделей каждого вида.
8. Ознакомьтесь с 3D-моделями, размещёнными в Единой коллекции цифровых образовательных ресурсов (www.school-collection.edu.ru/). К какому классу моделей их можно отнести?

§ 2.2
Знаковые модели

Ключевые слова:
- словесные модели
- математические модели
- компьютерные модели

2.2.1. Словесные модели

Словесные модели — это описания предметов, явлений, событий, процессов на естественных языках.

Например, гелиоцентрическая модель мира, которую предложил Коперник, словесно описывалась следующим образом:
- Земля вращается вокруг своей оси и вокруг Солнца;
- орбиты всех планет проходят вокруг Солнца.

Множество словесных моделей содержится в ваших школьных учебниках: в учебнике истории представлены модели исторических событий, в учебнике географии — модели географических объектов и природных процессов, в учебнике биологии — модели объектов животного и растительного мира.

Произведения художественной литературы — это тоже модели, так как они фиксируют внимание читателя на определённых сторонах человеческой жизни. Анализируя литературное произведение, вы выделяете в нём объекты и их свойства, отношения между героями, связи между событиями, проводите параллели с другими произведениями и т. п. Самое непосредственное отношение к понятию модели имеет такой литературный жанр, как басня. Смысл этого жанра состоит в переносе отношений между людьми на отношения между вымышленными персонажами, например животными.

Такие особенности естественного языка, как многозначность, использование слов в прямом и переносном значении, синонимия, омонимия и т. п., придают человеческому общению выразительность, эмоциональность, красочность. Вместе с тем наличие этих особенностей делает естественный язык непригодным для создания информационных моделей во многих сферах профессиональной деятельности (например, в системах «человек — компьютер»).

2.2.2. Математические модели

Основным языком информационного моделирования в науке является язык математики.

Информационные модели, построенные с использованием математических понятий и формул, называются **математическими моделями**.

Язык математики представляет собой совокупность множества формальных языков; с некоторыми из них (алгебраическим, геометрическим) вы познакомились в школе, другие сможете узнать при дальнейшем обучении.

Пример 1. На рис. 2.3 приведена геометрическая модель доказательства теоремы Пифагора. Она столь проста, что доказательство равенства $c^2 = a^2 + b^2$ становится очевидным.

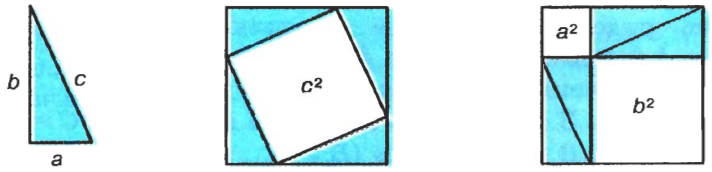


Рис. 2.3. Геометрическая модель доказательства теоремы Пифагора

Действительно, на рисунке представлены два одинаковых квадрата со стороной $a + b$. Если из площадей этих квадратов вычесть площади четырёх одинаковых прямоугольных треугольников, то оставшиеся площади должны быть равны. Но в первом случае оставшаяся площадь равна c^2, а во втором $a^2 + b^2$, что и доказывает теорему Пифагора.

Язык алгебры позволяет формализовать функциональные зависимости между величинами, записав соотношения между количественными характеристиками объекта моделирования. В школьном курсе

физики рассматривается много функциональных зависимостей, которые представляют собой математические модели изучаемых явлений или процессов.

Пример 2. Зависимость координаты тела от времени при прямолинейном равномерном движении имеет вид:

$x = x_0 + v_x t$.

Изменение координаты тела x при прямолинейном равноускоренном движении в любой момент времени t выражается формулой:

$$x = x_0 + v_{0x} t + \frac{a_x t^2}{2}.$$

С помощью языка алгебры логики строятся **логические модели** — формализуются (записываются в виде логических выражений) простые и сложные высказывания, выраженные на естественном языке. Путём построения логических моделей удаётся решать логические задачи, создавать логические модели устройств и т. д.

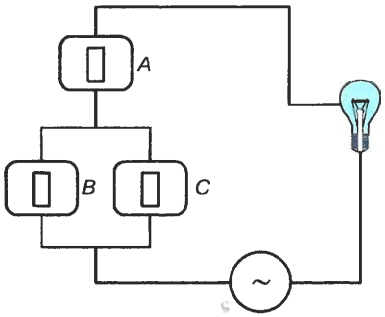
Пример 3. Требуется спроектировать электрическую цепь, показывающую итог тайного голосования комиссии в составе председателя и двух рядовых членов. При голосовании «за» каждый член комиссии нажимает кнопку. Предложение считается принятым, если члены комиссии проголосуют за него единогласно либо если свои голоса «за» отдадут председатель и один из рядовых членов комиссии. В этих случаях загорается лампочка.

Решение. Пусть голосу председателя соответствует переключатель A, голосам рядовых членов — переключатели B и C. Тогда $F(A, B, C) = A \,\&\, B \,\&\, C \vee A \,\&\, B \vee A \,\&\, C$.

Упростим полученное логическое выражение:

$$F(A, B, C) = A \,\&\, B \,\&\, (C \vee 1) \vee A \,\&\, C = A \,\&\, B \,\&\, 1 \vee A \,\&\, C =$$
$$= A \,\&\, B \vee A \,\&\, C = A \,\&\, (B \vee C).$$

Мы получили логическую модель, позволяющую построить следующую схему проектируемой электрической цепи:



2.2.3. Компьютерные математические модели

Многие процессы, происходящие в окружающем нас мире, описываются очень сложными математическими соотношениями (уравнениями, неравенствами, системами уравнений и неравенств). До появления компьютеров, обладающих высокой скоростью вычислений, у человека не было возможности проводить соответствующие вычисления, на счёт «вручную» уходило очень много времени.

В настоящее время самые сложные математические модели могут быть реализованы[1] на компьютере. При этом используются такие средства, как:

- системы программирования;
- электронные таблицы;
- специализированные математические пакеты и программные средства для моделирования.

Математические модели, реализованные с помощью систем программирования, электронных таблиц, специализированных математических пакетов и программных средств для моделирования, называются **компьютерными математическими моделями**.

Средства компьютерной графики позволяют визуализировать результаты расчётов, получаемых в процессе работы с компьютерными моделями.

С помощью ресурса «Математическая модель» (http://school-collection.edu.ru/) вы сможете смоделировать полёт снаряда, выпущенного из пушки при различных исходных данных.

Особый интерес для компьютерного математического моделирования представляют сложные системы, элементы которых могут вести себя случайным образом. Примерами таких систем являются многочисленные *системы массового обслуживания*: билетные кассы, торговые предприятия, ремонтные мастерские, служба скорой помощи, транспортные потоки на городских дорогах и многие другие модели. Многим знакома ситуация, когда, придя в кассу, магазин, парикмахерскую, мы застаём там очередь. Приходится либо вставать в очередь и какое-то время ждать, либо уходить, т. е. покидать систему необслуженным. Возможны случаи, когда заявок на обслуживание в системе

[1] Реализация математической модели — это расчёт состояния (выходных параметров) моделируемой системы по формулам, связывающим её входные параметры.

мало или совсем нет; в этом случае она работает с недогрузкой или простаивает. В системах массового обслуживания количество заявок на обслуживание, время ожидания и точное время выполнения заявки заранее предсказать нельзя — это случайные величины.

Имитационные модели воспроизводят поведение сложных систем, элементы которых могут вести себя случайным образом.

Имитационное моделирование — это искусственный эксперимент, при котором вместо проведения натурных испытаний с реальным оборудованием проводят опыты с помощью компьютерных моделей. Для получения необходимой информации осуществляется многократный «прогон» моделей со случайными исходными данными, генерируемыми компьютером. В результате образуется такой же набор данных, который можно было бы получить при проведении опытов на реальном оборудовании или в реальной системе. Однако имитационное моделирование на компьютере осуществляется гораздо быстрее и обходится значительно дешевле, чем натурные эксперименты.

С помощью ресурса «Имитационное моделирование» (http://school-collection.edu.ru/) вы сможете смоделировать ситуацию в системе массового обслуживания — магазине.

САМОЕ ГЛАВНОЕ

Словесные модели — это описания предметов, явлений, событий, процессов на естественных языках.

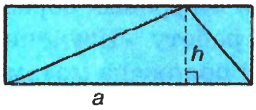
Информационные модели, построенные с использованием математических понятий и формул, называются **математическими моделями.**

Математические модели, реализованные с помощью систем программирования, электронных таблиц, специализированных математических пакетов и программных средств для моделирования, называются **компьютерными математическими моделями.**

Имитационные модели воспроизводят поведение сложных систем, элементы которых могут вести себя случайным образом.

Вопросы и задания

1. Приведите 2–3 собственных примера словесных моделей, рассматриваемых на уроках истории, географии, биологии.
2. Вспомните басни И. А. Крылова: «Волк и ягнёнок», «Ворона и лисица», «Демьянова уха», «Квартет», «Лебедь, Щука и Рак», «Лисица и виноград», «Слон и Моська», «Стрекоза и Муравей», «Тришкин кафтан» и др. Какие черты характера людей и отношения между людьми смоделировал в них автор?
3. На основании следующей геометрической модели докажите справедливость формулы $S_{тр} = 1/2 \cdot a \cdot h$.



4. Решите, составив математическую модель, следующую задачу.

   Пароход прошёл 4 км против течения реки, а затем прошёл ещё 33 км по течению, затратив на весь путь один час. Найдите собственную скорость парохода, если скорость течения реки равна 6,5 км/ч.
5. Требуется спроектировать электрическую цепь, показывающую итог тайного голосования комиссии в составе трёх членов. При голосовании «за» член комиссии нажимает кнопку. Предложение считается принятым, если оно собирает большинство голосов. В этом случае загорается лампочка.
6. Решите, составив логическую модель, следующую задачу.

   На международных соревнованиях по прыжкам в воду первые пять мест заняли спортсмены из Германии, Италии, Китая, России и Украины. Ещё до начала соревнований эксперты высказали свои предположения об их итогах:

   1) Первое место займёт спортсмен из Китая, а спортсмен из Украины будет третьим.
   2) Украина будет на последнем месте, а Германия — на предпоследнем.
   3) Германия точно будет четвёртой, а первое место займёт Китай.
   4) Россия будет первой, а Италия — на втором месте.
   5) Нет, Италия будет пятой, а победит Германия.

По окончании соревнований выяснилось, что каждый эксперт был прав только в одном утверждении. Какие места в соревновании заняли участники?

7. В середине прошлого века экономисты оценили ежегодный объём вычислений, необходимых для эффективного управления народным хозяйством страны. Он составил 10^{17} операций. Можно ли справиться с таким объёмом вычислений за год, если привлечь к работе миллион вычислителей, каждый из которых способен выполнять одну операцию в секунду?
8. Приведите примеры использования компьютерных моделей. Найдите соответствующую информацию в сети Интернет.
9. В Единой коллекции цифровых образовательных ресурсов найдите лабораторную работу «Изучение закона сохранения импульса». В её основу положена математическая модель, описывающая движение тела, брошенного под углом к горизонту, с последующим делением тела на два осколка. Экспериментально проверьте закон сохранения импульса, выполнив работу согласно имеющемуся в ней описанию.
10. В Единой коллекции цифровых образовательных ресурсов (http://school-collection.edu.ru/) найдите игру «Равноплечий рычаг». Изучите правила игры. Вспомните физическую закономерность, положенную в её основу. Попытайтесь «победить» компьютер и сформулировать выигрышную стратегию.

§ 2.3
Графические информационные модели

Ключевые слова:

- схема
- карта
- чертёж
- график
- диаграмма
- граф
- сеть
- дерево

2.3.1. Многообразие графических информационных моделей

В графических информационных моделях для наглядного отображения объектов используются условные графические изображения (образные элементы), зачастую дополняемые числами, символами и текстами (знаковыми элементами). Примерами графических моделей могут служить всевозможные схемы, карты, чертежи, графики и диаграммы.

Схема — это представление некоторого объекта в общих, главных чертах с помощью условных обозначений. С помощью схем может быть представлен и внешний вид объекта, и его структура. Схема как информационная модель не претендует на полноту предоставления информации об объекте. С помощью особых приёмов и графических обозначений на ней более рельефно выделяется один или несколько признаков рассматриваемого объекта. Примеры схем приведены на рис. 2.4.

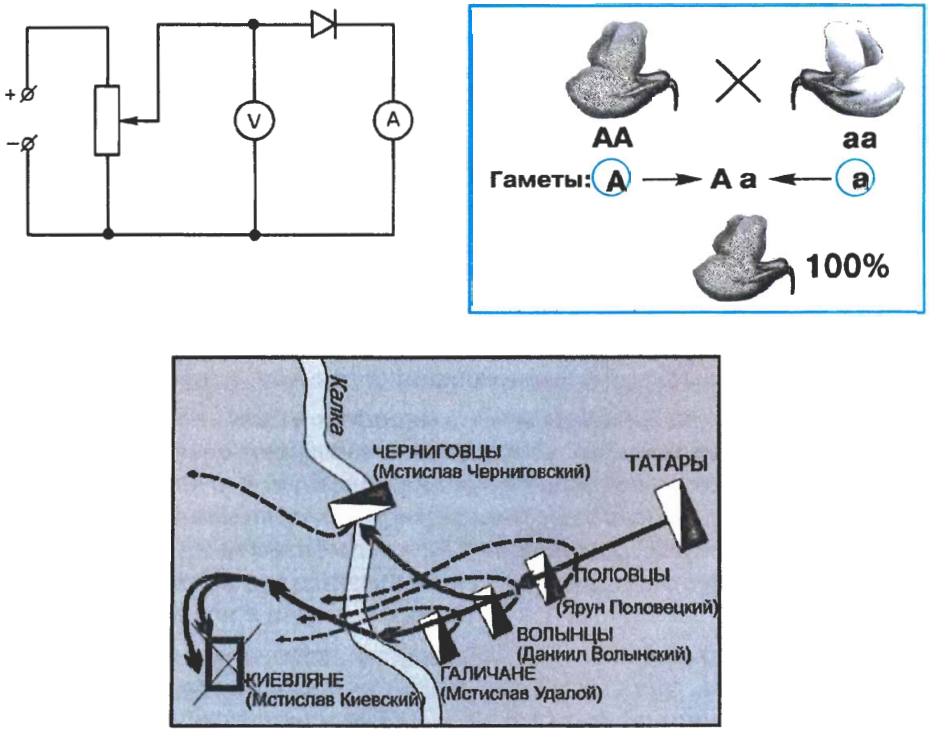


Рис. 2.4. Примеры схем, используемых на уроках физики, биологии, истории

Уменьшенное обобщённое изображение поверхности Земли на плоскости в той или иной системе условных обозначений даёт нам географическая **карта**.

Чертёж — условное графическое изображение предмета с точным соотношением его размеров, получаемое методом проецирования. Чертёж содержит изображения, размерные числа, текст. Изображения дают представления о геометрической форме объекта, числа — о величине объекта и его частей, надписи — о названии, масштабе, в котором выполнены изображения.

График — линия, дающая наглядное представление о характере зависимости одной величины (например, пути) от другой (например, времени). График позволяет отслеживать динамику изменения данных.

Диаграмма — графическое изображение, дающее наглядное представление о соотношении каких-либо величин или нескольких значений одной величины, об изменении их значений. Более подробно

типы диаграмм и способы их построения будут рассмотрены при изучении электронных таблиц.

2.3.2. Графы

Если объекты некоторой системы изобразить вершинами, а связи между ними — линиями, то мы получим информационную модель рассматриваемой системы в форме графа. **Граф** состоит из **вершин**, связанных линиями — **рёбрами**. Вершины графа могут изображаться кругами, овалами, точками, прямоугольниками и т. д.

Граф называется **взвешенным**, если его вершины или рёбра характеризуются некоторой дополнительной информацией — **весами** вершин или рёбер.

На рис. 2.5 с помощью взвешенного графа изображены дороги между пятью населёнными пунктами *A*, *B*, *C*, *D*, *E*; веса рёбер — протяжённость дорог в километрах.

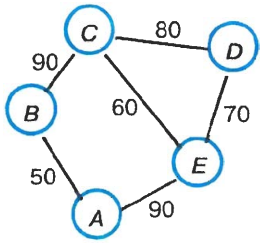


Рис. 2.5. Взвешенный граф

Путь по вершинам и рёбрам графа, в который любое ребро графа входит не более одного раза, называется **цепью**. Цепь, начальная и конечная вершины которой совпадают, называется **циклом**.

Граф с циклом называется **сетью**. Если героев некоторого литературного произведения представить вершинами графа, а существующие между ними связи изобразить рёбрами, то мы получим граф, называемый **семантической сетью**.

Графы как информационные модели находят широкое применение во многих сферах нашей жизни. Например, можно существующие или вновь проектируемые дома, сооружения, кварталы изображать вершинами, а соединяющие их дороги, инженерные сети, линии электропередач и т. п. — рёбрами графа. По таким графам можно планировать оптимальные транспортные маршруты, кратчайшие объездные пути, расположение торговых точек и других объектов.

Дерево — это граф, в котором нет циклов, т. е. в нём нельзя из некоторой вершины пройти по нескольким различным рёбрам и вернуться в ту же вершину. Отличительной особенностью дерева является то, что между любыми двумя его вершинами существует единственный путь.

Всякая иерархическая система может быть представлена с помощью дерева. У дерева выделяется одна главная вершина, называемая его **корнем**. Каждая вершина дерева (кроме корня) имеет только одного предка, обозначенный им объект входит в один класс[1] высшего уровня. Любая вершина дерева может порождать несколько **потомков** — вершин, соответствующих классам нижнего уровня. Такой принцип связи называется «один-ко-многим». Вершины, не имеющие порождённых вершин, называются **листьями**.

Родственные связи между членами семьи удобно изображать с помощью графа, называемого **генеалогическим** или **родословным деревом**.

www Ресурс «Живая Родословная» (http://school-collection.edu.ru/) — инструмент для формирования и анализа генеалогических деревьев, содержащий примеры родословных. С его помощью вы можете изучить генеалогические деревья многих известных семей и построить генеалогическое дерево своей семьи.

2.3.3. Использование графов при решении задач

Графы удобно использовать при решении некоторых классов задач.

Пример 1. Для того чтобы записать все трёхзначные числа, состоящие из цифр 1 и 2, можно воспользоваться графом (деревом) на рис. 2.6.

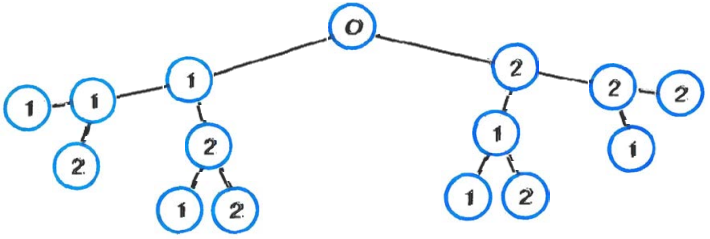


Рис. 2.6. Дерево для решения задачи о записи трёхзначных чисел

[1] Класс — множество объектов, обладающих общими признаками.

Дерево можно не строить, если не требуется выписывать все возможные варианты, а нужно просто указать их количество. В этом случае рассуждать нужно так: в разряде сотен может быть любая из цифр 1 и 2, в разряде десятков — те же два варианта, в разряде единиц — те же два варианта. Следовательно, число различных вариантов: $2 \cdot 2 \cdot 2 = 8$.

В общем случае, если известно количество возможных вариантов выбора на каждом шаге построения графа, то для вычисления общего количества вариантов нужно все эти числа *перемножить*.

Пример 2. Рассмотрим несколько видоизменённую классическую задачу о переправе.

На берегу реки стоит крестьянин (*К*) с лодкой, а рядом с ним — собака (*С*), лиса (*Л*) и гусь (*Г*). Крестьянин должен переправиться сам и перевезти собаку, лису и гуся на другой берег. Однако в лодку кроме крестьянина помещается либо только собака, либо только лиса, либо только гусь. Оставлять же собаку с лисой или лису с гусем без присмотра нельзя — собака представляет опасность для лисы, а лиса — для гуся. Как крестьянин должен организовать переправу?

Для решения этой задачи составим граф, вершинами которого будут исходное размещение персонажей на берегу реки, а также всевозможные промежуточные состояния, достигаемые из предыдущих за один шаг переправы. Каждую вершину-состояние переправы обозначим овалом и свяжем рёбрами с состояниями, образованными из неё (рис. 2.7).

Недопустимые по условию задачи состояния выделены пунктирной линией; они исключаются из дальнейшего рассмотрения. Начальное и конечное состояния переправы выделены жирной линией.

На графе видно, что существует два решения этой задачи. Приведём соответствующий одному из них план переправы:

1) крестьянин перевозит лису;

2) крестьянин возвращается;

3) крестьянин перевозит собаку;

4) крестьянин возвращается с лисой;

5) крестьянин перевозит гуся;

6) крестьянин возвращается;

7) крестьянин перевозит лису.

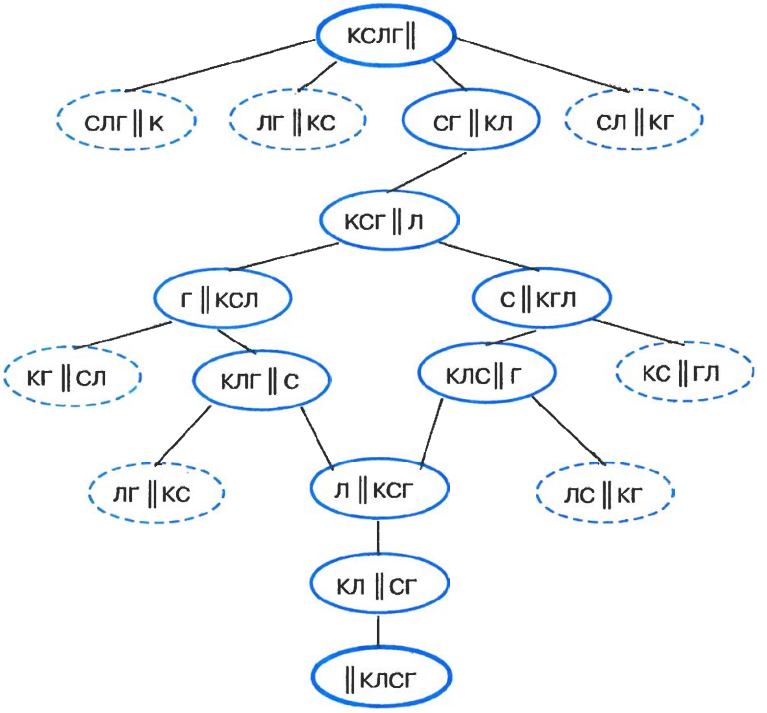


Рис. 2.7. Граф переправы

Пример 3. Рассмотрим следующую игру: сначала в кучке лежит 5 спичек; два игрока убирают спички по очереди, причём за 1 ход можно убрать 1 или 2 спички; выигрывает тот, кто оставит в кучке 1 спичку. Выясним, кто выигрывает при правильной игре — первый (I) или второй (II) игрок.

Игрок I может убрать одну спичку (в этом случае их останется 4) или сразу 2 (в этом случае их останется 3).

Если игрок I оставил 4 спички, игрок II может своим ходом оставить 3 или 2 спички. Если же после хода первого игрока осталось 3 спички, второй игрок может выиграть, взяв две спички и оставив одну.

Если после игрока II осталось 3 или 2 спички, то игрок I в каждой из этих ситуаций имеет шанс на выигрыш.

Таким образом, при правильной стратегии игры всегда выиграет первый игрок. Для этого своим первым ходом он должен взять одну спичку.

На рис. 2.8 представлен граф, называемый **деревом игры**; на нём отражены все возможные варианты, в том числе ошибочные (проигрышные) ходы игроков.

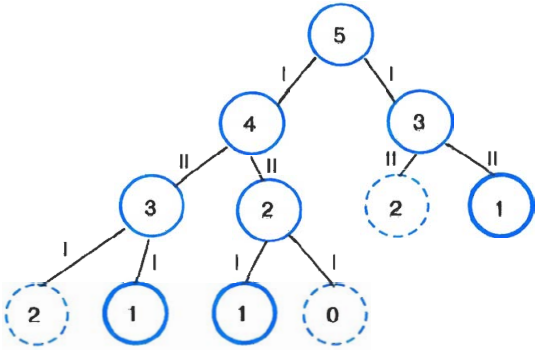


Рис. 2.8. Дерево игры

САМОЕ ГЛАВНОЕ

В **графических информационных моделях** для наглядного отображения объектов используются условные графические изображения (образные элементы), зачастую дополняемые числами, символами и текстами (знаковыми элементами). Примерами графических моделей могут служить всевозможные схемы, карты, чертежи, графики и диаграммы, графы.

Граф состоит из вершин, связанных линиями — **рёбрами**. Граф называется **взвешенным**, если его вершины или рёбра характеризуются некоторой дополнительной информацией — весами вершин (рёбер).

Путь по вершинам и рёбрам графа, в который любое ребро графа входит не более одного раза, называется **цепью**. Цепь, начальная и конечная вершины которой совпадают, называется **циклом**. Граф с циклом называется **сетью**.

Граф иерархической системы называется **деревом**. Отличительной особенностью дерева является то, что между любыми двумя его вершинами существует единственный путь.

Вопросы и задания

1. Какие информационные модели относят к графическим?
2. Приведите примеры графических информационных моделей, с которыми вы имеете дело:
   а) при изучении других предметов;
   б) в повседневной жизни.

3. Что такое граф? Что является вершинами и рёбрами графа на рис. 2.5? Приведите примеры цепей и циклов, имеющихся в этом графе. Определите, какие два пункта наиболее удалены друг от друга (два пункта считаются самыми удалёнными, если длина кратчайшего пути между ними больше, чем длина кратчайшего пути между любыми другими двумя пунктами). Укажите длину кратчайшего пути между этими пунктами.
4. Приведите пример системы, модель которой можно представить в форме графа. Изобразите соответствующий граф.
5. Грунтовая дорога проходит последовательно через населённые пункты A, B, C и D. При этом длина грунтовой дороги между A и B равна 40 км, между B и C — 25 км, и между C и D — 10 км. Между A и D дороги нет. Между A и C построили новое асфальтовое шоссе длиной 30 км. Оцените минимально возможное время движения велосипедиста из пункта A в пункт B, если его скорость по грунтовой дороге — 20 км/ч, по шоссе — 30 км/ч.
6. Составьте семантическую сеть по русской народной сказке «Колобок».
7. Что такое дерево? Моделями каких систем могут служить деревья? Приведите пример такой системы.
8. Сколько трёхзначных чисел можно записать с помощью цифр 2, 4, 6 и 8 при условии, что в записи числа не должно быть одинаковых цифр?
9. Сколько существует трёхзначных чисел, все цифры которых различны?
10. Для составления цепочек используются бусины, помеченные буквами: A, B, C, D, E. На первом месте в цепочке стоит одна из бусин A, C, E. На втором — любая гласная, если первая буква гласная, и любая согласная, если первая согласная. На третьем месте — одна из бусин C, D, E, не стоящая в цепочке на первом месте. Сколько цепочек можно создать по этому правилу?
11. Два игрока играют в следующую игру. Перед ними лежит куча из 6 камней. Игроки берут камни по очереди. За один ход можно взять 1, 2 или 3 камня. Проигрывает тот, кто забирает последний камень. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

§ 2.4
Табличные информационные модели

Ключевые слова:
- таблица
- таблица «объект—свойство»
- таблица «объект—объект»

В табличных информационных моделях информация об объектах представляется в виде прямоугольной таблицы, состоящей из столбцов и строк.

Вам хорошо известно табличное представление расписания уроков, в табличной форме представляются расписания движения автобусов, самолётов, поездов и многое другое.

Представленная в таблице информация наглядна, компактна и легко обозрима.

2.4.1. Представление данных в табличной форме

В качестве информационных моделей объектов, обладающих одинаковыми наборами свойств, как правило, используются таблицы типа **«объект—свойство»**.

Например, информацию о регионах нашей страны можно представить с помощью таблицы, фрагмент которой приведён ниже (табл. 2.1).

В этой таблице каждая строка содержит информацию об одном объекте — регионе; столбцы — отдельные характеристики (свойства) рассматриваемых объектов: название, дата образования, площадь и т. д. Такие таблицы могут содержать числовую, текстовую и графическую информацию.

Таблица 2.1

Регионы Российской Федерации

Название	Дата образования (ДД.ММ.ГГ)	Площадь (тыс. км^2)	Население (тыс. чел.)
Астраханская область	27.12.1943	44,1	1006,3
Архангельская область	23.09.1937	587,4	1336,5
Белгородская область	06.01.1954	27,1	1511,6
Владимирская область	14.08.1944	29,0	1524,0
Вологодская область	23.09.1937	145,7	1269,6
Воронежская область	13.06.1934	52,4	2378,8
Калужская область	05.07.1947	29,9	1041,6

В пустую строку таблицы вы можете записать информацию о своём регионе.

В таблицах типа **«объект—объект»** отражается взаимосвязь между объектами одного или нескольких классов. Например, в школьных журналах есть таблица «Сведения о количестве уроков, пропущенных обучающимися»; её фрагмент представлен в табл. 2.2:

Таблица 2.2

Сведения о пропусках уроков

№	Список	Месяц: январь							
		Число							
		10	11	12	13	14	17	18	19
1	Акуленко Иван								
2	Баранов Владимир	6	6	6					
3	Варнаков Олег								
4	Егорова Виктория				5	6	1		
5	Машкова Карина						6	6	6

В этой таблице отражена связь «количество пропущенных уроков» между объектами класса «обучающиеся» и класса «число».

В таблице «Расстояния между городами» (табл. 2.3) представлены расстояния между парами объектов, принадлежащих одному классу «город». В свободные строку и столбец можете добавить информацию о своём населённом пункте.

Таблица 2.3

Расстояния между городами (км)

Город	Город				
	Москва	Петрозаводск	Самара	Казань	
Москва		1076	1069	815	
Петрозаводск	1076		2145	1891	
Самара	1069	2145		631	
Казань	815	1891	631		

В форме таблицы «объект–объект» можно представить информацию о наличии границ (сухопутной, морской, озёрной, речной) России с другими странами; её фрагмент представлен ниже (табл. 2.4)

Таблица 2.4

Граница Российской Федерации

Страна	Граница			
	сухопутная	речная	озёрная	морская
Норвегия	1	1	0	1
Финляндия	1	1	1	1
Латвия	1	1	1	0
Корея	0	1	0	1
Япония	0	0	0	1

Если граница соответствующего вида есть, то в нужную ячейку ставится 1, а если нет — 0.

Важная особенность этой таблицы состоит в том, что в ней фиксируются не количественные (сколько?), а качественные свойства (наличие/отсутствие связи между объектами).

2.4.2. Использование таблиц при решении задач

Рассмотрим несколько примеров задач, которые удобно решать с помощью табличных информационных моделей.

Пример 1. Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 3 камня, а во второй — 2 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 3 раза число камней в какой-то куче, или добавляет 1 камень в какую-то кучу. Выигрывает игрок, после хода которого общее число камней в двух кучах становится не менее 16. Кто выигрывает при безошибочной игре — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Ранее мы рассмотрели способ записи решения подобных задач с помощью дерева. Сейчас оформим решение в виде таблицы (табл. 2.5):

Таблица 2.5

Таблица игры

Исходное положение	1-й игрок — 1-й ход	2-й игрок — 1-й ход	1-й игрок — 2-й ход	2-й игрок — 2-й ход
1	2	3	4	5
3,2,5	9,2,11	27,2,29 ✓		
	3,6,9	3,18,21✓		
	4,2,6	12,2,14	36,2,38✓	
		4,6,10	12,6,18✓	
		5,2,7	15,2,17✓	
		4,3,7	12,3,15	36,3,39✓
			4,9,13	12,9,21✓
			5,3,8	15,3,18✓
			4,4,8	12,4,16✓
	3,3,6	9,3,12	27,3,12✓	
		4,3,7[1]		

[1] Вариант (как повторный) исключается из дальнейшего рассмотрения.

Три числа в каждой ячейке таблицы обозначают соответственно количество камней в кучках и их сумму. В первом столбце зафиксировано распределение камней перед игрой (исходное положение).

Во втором столбце рассмотрены все возможные варианты ходов первого игрока; победить с первого хода он не может.

В третьем столбце рассмотрены имеющиеся выигрышные варианты ходов второго игрока (отмечены «галочкой»). При безошибочной игре первого игрока такие ситуации возникнуть не должны. Поэтому рассматриваем все возможные ходы второго игрока в случаях, когда у него нет выигрышного хода. Если получены одинаковые варианты, то все из них, кроме одного, исключаем из дальнейшего рассмотрения.

В четвёртом столбце отмечены имеющиеся выигрышные варианты второго хода первого игрока. При безошибочной игре второго игрока такие ситуации возникнуть не должны. Поэтому рассматриваем все возможные ходы первого игрока в случае, когда у него нет выигрышного хода.

В пятом столбце отмечены выигрышные ходы второго игрока, имеющиеся при всех вариантах хода первого игрока.

Таким образом, при безошибочной игре соперников побеждает второй игрок. Его первый ход должен быть таким, чтобы в кучках стало 4 и 3 камня.

Пример 2. С помощью взвешенного графа на рис. 2.5 представлена схема дорог, соединяющих населённые пункты A, B, C, D, E. Построим таблицу, соответствующую этому графу.

	A	*B*	*C*	*D*	*E*
A	×	50			90
B	50	×	90		
C		90	×	80	60
D			80	×	70
E	90		60	70	×

Если между парой населённых пунктов существует дорога, то в ячейку на пересечении соответствующих строки и столбца записывается число, равное её длине. Имеющиеся в таблице пустые клетки означают, что дорог между соответствующими населёнными пунктами нет. Построенная таким образом таблица называется **весовой матрицей**.

Для решения некоторых задач бывает удобно по имеющейся таблице строить граф. При этом одной и той же таблице могут соответствовать графы, внешне не похожие друг на друга. Например, рассмотренной выше таблице кроме графа на рис. 2.5 соответствует граф на рис. 2.9.

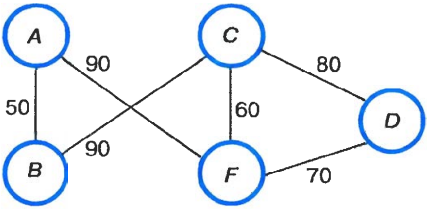


Рис. 2.9. Вариант графа, представляющего схему дорог

Пример 3. Таблицы типа «объект—объект» удобно использовать для решения логических задач, в которых требуется установить взаимно однозначное соответствие между объектами нескольких классов. Рассмотрим задачу, в которой объекты связаны тремя парами отношений.

Три подружки — Аня, Света и Настя — купили различные молочные коктейли в белом, голубом и зелёном стаканчиках. Ане достался не белый стаканчик, а Свете — не голубой. В белом стаканчике не банановый коктейль. В голубой стаканчик налит ванильный коктейль. Света не любит клубничный коктейль.

Требуется выяснить, какой коктейль и в каком стаканчике купила каждая из девочек.

Создадим три следующие таблицы:

Стаканчик	Девочка		
	Аня	Света	Настя
Белый			
Голубой			
Зелёный			

Стаканчик	Коктейль		
	банановый	ванильный	клубничный
Белый			
Голубой			
Зелёный			

Коктейль	Девочка		
	Аня	Света	Настя
Банановый			
Ванильный			
Клубничный			

Отметим в таблицах информацию, содержащуюся в условии задачи:

Стаканчик	Девочка		
	Аня	Света	Настя
Белый	0		
Голубой		0	
Зелёный			

Стаканчик	Коктейль		
	банановый	ванильный	клубничный
Белый	0		
Голубой		1	
Зелёный			

Коктейль	Девочка		
	Аня	Света	Настя
Банановый			
Ванильный			
Клубничный		0	

Имеющейся во второй таблице информации достаточно для того, чтобы заполнить всю эту таблицу:

Стаканчик	Коктейль		
	банановый	ванильный	клубничный
Белый	0	0	1
Голубой	0	1	0
Зелёный	1	0	0

Используя факты, что Света купила не клубничный коктейль и что этот коктейль был налит в белый стаканчик, заполняем всю первую таблицу:

Стаканчик	Девочка		
	Аня	Света	Настя
Белый	0	0	1
Голубой	1	0	0
Зелёный	0	1	0

На основании информации в первой и второй таблицах можем заполнить всю третью таблицу:

Коктейль	Девочка		
	Аня	Света	Настя
Банановый	0	1	0
Ванильный	1	0	0
Клубничный	0	0	1

Ответ: Аня купила ванильный коктейль в голубом стаканчике, Света — банановый коктейль в зелёном стаканчике, Настя — клубничный коктейль в белом стаканчике.

САМОЕ ГЛАВНОЕ

В табличных информационных моделях информация об объекте или процессе представляется в виде прямоугольной таблицы, состоящей из столбцов и строк. Представленная в таблице информация наглядна, компактна и легко обозрима.

Таблица типа **«объект—свойство»** — это таблица, содержащая информацию о свойствах отдельных объектах, принадлежащих одному классу.

Таблица типа **«объект—объект»** — это таблица, содержащая информацию о некотором одном свойстве пар объектов, чаще всего принадлежащих разным классам.

Вопросы и задания

1. Какие преимущества обеспечивают табличные информационные модели по сравнению со словесными описаниями? Приведите пример.
2. Приведите примеры табличных информационных моделей, с которыми вы имеете дело:
   а) на уроках в школе;
   б) в повседневной жизни.
3. К какому типу относится таблица «Табель успеваемости», расположенная в конце вашего дневника?
4. Узнайте, в каких случаях в ячейку таблицы ставится знак «×». Почему мы использовали этот знак в таблице (пример 2)?
5. Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 1 камень, а во второй — 2 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 3 раза число камней в какой-то куче, или добавляет 2 камня в какую-то кучу. Выигрывает игрок, после хода которого общее число камней в двух кучах становится не менее 17. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.
6. Таблица стоимости перевозок устроена следующим образом: числа, стоящие на пересечениях строк и столбцов таблиц, означают стоимость проезда между соответствующими соседними станциями. Если пересечение строки и столбца пусто, то станции не являются соседними. Стоимость проезда по маршруту складывается из стоимостей проезда между соседними станциями. Перевозки между населёнными пунктами A, B, C, D, F осуществляют три компании, представившие стоимость своих услуг в табличной форме. Какая компания обеспечивает минимальную стоимость проезда из A в B?

1)

	A	B	C	D	E
A	×		3	1	
B		×	4		2
C	3	4	×		2
D	1			×	
E		2	2		×

2)

	A	*B*	*C*	*D*	*E*
A	×		3	1	1
B		×	4		
C	3	4	×		2
D	1			×	
E	1		2		×

3)

	A	*B*	*C*	*D*	*E*
A	×		3	1	4
B		×	4		2
C	3	4	×		2
D	1			×	
E	4	2	2		×

7. Соревнования по плаванию были в самом разгаре, когда стало ясно, что первые четыре места займут мальчики из пятёрки лидеров. Их имена: Валерий, Николай, Михаил, Игорь, Эдуард, фамилии: Симаков, Чигрин, Зимин, Копылов, Блинов (имена и фамилии названы в произвольном порядке). Нашлись знатоки, которые предсказали, что первое место займёт Копылов, второе — Валерий, третье — Чигрин, четвёртое — Эдуард. Но ни один из ребят не занял того места, какое ему предсказывали. На самом деле первое место завоевал Михаил, второе — Симаков, третье — Николай, четвёртое — Блинов, а Чигрин не попал в четвёрку сильнейших. Назовите имя и фамилию каждого из лидеров.

8. В Норильске, Москве, Ростове и Пятигорске живут четыре супружеские пары (в каждом городе — одна пара). Имена этих супругов: Антон, Борис, Давид, Григорий, Ольга, Мария, Светлана, Екатерина. Антон живёт в Норильске, Борис и Ольга — супруги, Григорий и Светлана не живут в одном городе, Мария живёт в Москве, Светлана — в Ростове. В каком городе проживает каждая из супружеских пар?

9. Постройте граф, отражающий разновидности информационных моделей.

§ 2.5
База данных как модель предметной области

Ключевые слова:
- информационная система
- база данных
- иерархическая база данных
- сетевая база данных
- реляционная база данных
- запись
- поле
- ключ

2.5.1. Информационные системы и базы данных

Современный человек в своей практической деятельности всё чаще и чаще использует различные **информационные системы**, обеспечивающие хранение, поиск и выдачу информации по его запросам. Примерами информационных систем являются:
- справочная адресная служба большого города;
- транспортная информационная система, обеспечивающая не только возможность получения справочной информации о расписании поездов и самолётов, но и покупку железнодорожных и авиабилетов;
- информационно-поисковая система, содержащая информацию правового характера.

Центральной частью любой информационной системы является база данных.

База данных (БД) — совокупность данных, организованных по определённым правилам, отражающая состояние объектов и их отношений в некоторой предметной области (транспорт, медицина, образование, право и т. д.), предназначенная для хранения во внешней памяти компьютера и для постоянного применения.

Базу данных можно рассматривать как информационную модель предметной области.

Основными способами организации данных в базах данных являются иерархический, сетевой и реляционный (рис. 2.9).

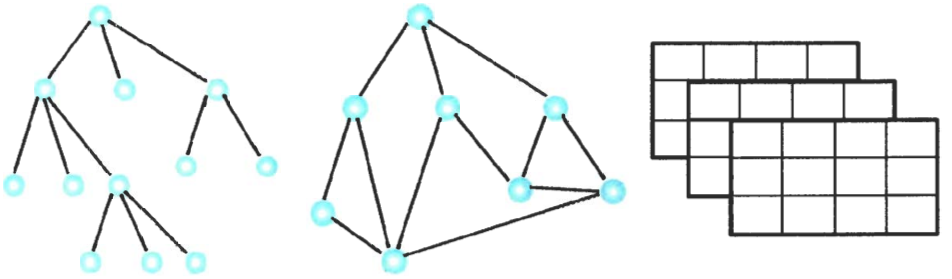


Рис. 2.10. Способы организации данных в БД: *а*) иерархический, *б*) сетевой, *в*) реляционный

В **иерархической базе данных** существует упорядоченность объектов по уровням. Между объектами существуют связи: каждый объект может быть связан с объектами более низкого уровня. Говорят, что такие объекты находятся в отношении предка к потомку. Иерархический способ организации данных реализован в системе папок операционной системы Windows. Верхний уровень занимает папка *Рабочий стол*. Папки второго уровня *Мой компьютер*, *Корзина* и *Сетевое окружение* являются её потомками. Папка *Мой компьютер* является предком для папок *Диск A*, *Диск C* и т. д. Поиск какого-либо объекта в такой базе данных может оказаться довольно трудоёмким из-за необходимости последовательно проходить несколько предшествующих иерархических уровней.

В **сетевой базе данных** не накладывается никаких ограничений на связи между объектами: в ней могут быть объекты, имеющие более одного предка. Сетевой способ организации данных реализован во Всемирной паутине глобальной компьютерной сети Интернет.

Наибольшее распространение получили **реляционные базы данных.** Их мы рассмотрим более подробно.

2.5.2. Реляционные базы данных

В **реляционной базе данных** (РБД) используется реляционная модель данных, основанная на представлении данных в виде таблиц.

Реляционная БД может состоять из одной или нескольких взаимосвязанных прямоугольных таблиц.

Строка таблицы РБД называется **записью**, столбец — **полем** (рис. 2.11).

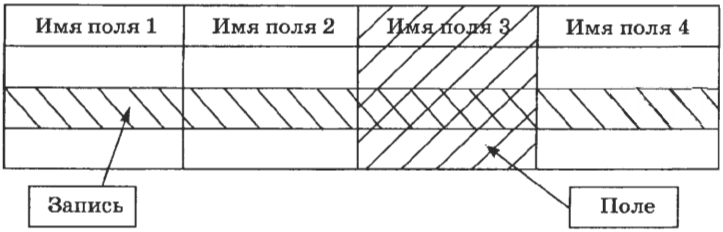


Рис. 2.11. Структура таблицы реляционной БД

Запись содержит всю информацию об одном объекте, описываемом в базе данных: об одном товаре, продаваемом в магазине; об одной книге, имеющейся в библиотеке; об одном сотруднике, работающем на предприятии, и т. п.

Поле — это одна из характеристик (атрибутов, свойств) объекта: название товара, стоимость товара, количество имеющихся в наличии товаров; название книги, автор книги, год издания; фамилия, имя, отчество сотрудника, дата рождения, специальность и т. п. Значения полей в одном столбце относятся к одной характеристике объекта.

Поле базы данных имеет имя, тип и длину.

Все имена полей таблицы должны быть разными.

Тип поля определяется типом данных, которые поле содержит. Основные типы полей:

- числовой — для полей, содержащих числовую информацию;
- текстовый — для полей, содержащих всевозможные последовательности символов;
- логический — для полей, которые могут принимать всего два значения: ДА (ИСТИНА, TRUE, 1) и НЕТ (ЛОЖЬ, FALSE, 0);
- дата — для полей, содержащих календарные даты (в нашей стране принято писать день, а потом месяц и год).

Длина поля — это максимальное количество символов, которые могут содержаться в поле.

Для записи структуры таблицы можно применять следующую форму:

ИМЯ_ТАБЛИЦЫ (ИМЯ ПОЛЯ 1, ИМЯ ПОЛЯ 2, ...)

Например, описать однотабличную базу данных «Календарь погоды» можно так:

КАЛЕНДАРЬ_ПОГОДЫ (ДЕНЬ, ТЕМПЕРАТУРА, ВЛАЖНОСТЬ, ДАВЛЕНИЕ, НАПРАВЛЕНИЕ ВЕТРА, СКОРОСТЬ ВЕТРА).

Здесь поле ДЕНЬ будет иметь тип «дата», поля ТЕМПЕРАТУРА, ВЛАЖНОСТЬ, ДАВЛЕНИЕ, СКОРОСТЬ ВЕТРА — числовой тип; поле НАПРАВЛЕНИЕ ВЕТРА — текстовый тип.

В таблице не должно быть совпадающих записей. Иначе говоря, должны быть поле или совокупность полей, значения которых для каждой записи всегда различны.

Например, значения поля ДЕНЬ базы данных «Календарь погоды» всегда будут разными в разных записях.

В базе данных УЧЕНИК (ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО, ДАТА РОЖДЕНИЯ, СЕРИЯ СВИДЕТЕЛЬСТВА О РОЖДЕНИИ, НОМЕР СВИДЕТЕЛЬСТВА О РОЖДЕНИИ, КЛАСС) наверняка не будут совпадать только значения совокупности таких полей, как СЕРИЯ СВИДЕТЕЛЬСТВА О РОЖДЕНИИ и НОМЕР СВИДЕТЕЛЬСТВА О РОЖДЕНИИ.

Поле или совокупность полей, значения которых в записях не повторяются (являются уникальными), называют **ключом** таблицы базы данных.

САМОЕ ГЛАВНОЕ

База данных (БД) — совокупность данных, организованных по определённым правилам, отражающая состояние объектов и их отношений в некоторой предметной области (транспорт, медицина, образование, право и т. д.), предназначенная для хранения во внешней памяти компьютера и постоянного применения. Базу данных можно рассматривать как информационную модель предметной области.

Основными способами организации данных в базах данных являются иерархический, сетевой и реляционный. В **реляционных базах данных** (РБД) используется реляционная модель данных, основанная на представлении данных в виде таблиц.

Строка таблицы РБД называется **записью**, столбец — **полем**. Поле или совокупность полей, значения которых в разных записях не по-

вторяются (являются уникальными), называют **ключом** таблицы базы данных.

Вопросы и задания

1. Что такое информационная система? Приведите пример информационной системы.
2. Что такое база данных?
3. Назовите основные способы организации данных в базах данных.
4. Какие базы данных называются реляционными?
5. Что такое запись? Какую информацию она содержит?
6. Что такое поле? Какую информацию содержит поле?
7. Перечислите основные типы полей РБД.
8. Для полей однотабличной базы данных КОЛЛЕКЦИЯ (КОД, НАЗВАНИЕ ЭКСПОНАТА, АВТОР, МЕСТО ИЗГОТОВЛЕНИЯ, ГОД ИЗГОТОВЛЕНИЯ, ФИО ПРЕДЫДУЩЕГО ВЛАДЕЛЬЦА, ДАТА ПРИОБРЕТЕНИЯ, СТОИМОСТЬ ЭКСПОНАТА, УПОМИНАНИЕ В КАТАЛОГАХ (да/нет)) укажите тип каждого поля.
9. Что такое ключ таблицы базы данных? Что может служить ключом в базе данных КОЛЛЕКЦИЯ?
10. Продумайте состав, типы полей и ключ однотабличной базы данных:

    а) ТУРАГЕНТСТВО;

    б) ВИДЕОТЕКА;

    в) АВТОСАЛОН;

    г) РЕГИОНЫ РФ.

§ 2.6
Система управления базами данных

Ключевые слова:
- СУБД
- таблица
- форма
- запрос
- условие выбора
- отчёт

2.6.1. Что такое СУБД

Программное обеспечение для создания баз данных, хранения и поиска в них необходимой информации называется **системой управления базами данных (СУБД)**.

С помощью СУБД пользователь может:
- создавать структуру базы данных;
- заполнять базу данных информацией;
- редактировать (исправлять, дополнять) структуру и содержание базы данных;
- выполнять сортировку (упорядочивание) информации;
- осуществлять поиск информации в базе данных;
- выводить нужную информацию на экран монитора, в файл и на бумажный носитель;
- устанавливать защиту базы данных.

Именно наличие СУБД превращает огромный объём хранимых в компьютерной памяти сведений в мощную справочную систему, способную быстро производить поиск и отбор необходимой нам информации.

2.6.2. Интерфейс СУБД

Существуют СУБД, с помощью которых создаются крупные промышленные информационные системы. Для работы с этими системами нужны специальные знания, в том числе владение специализированными языками программирования.

Для ведения личных баз данных, а также баз данных небольших организаций используются более простые СУБД, работать с которыми могут обычные пользователи. Наиболее распространёнными СУБД такого типа являются Microsoft Access и OpenOffice.org Base. При запуске любой из них на экран выводится окно, имеющее строку заголовка, строку меню, панели инструментов, рабочую область и строку состояния (рис. 2.12).

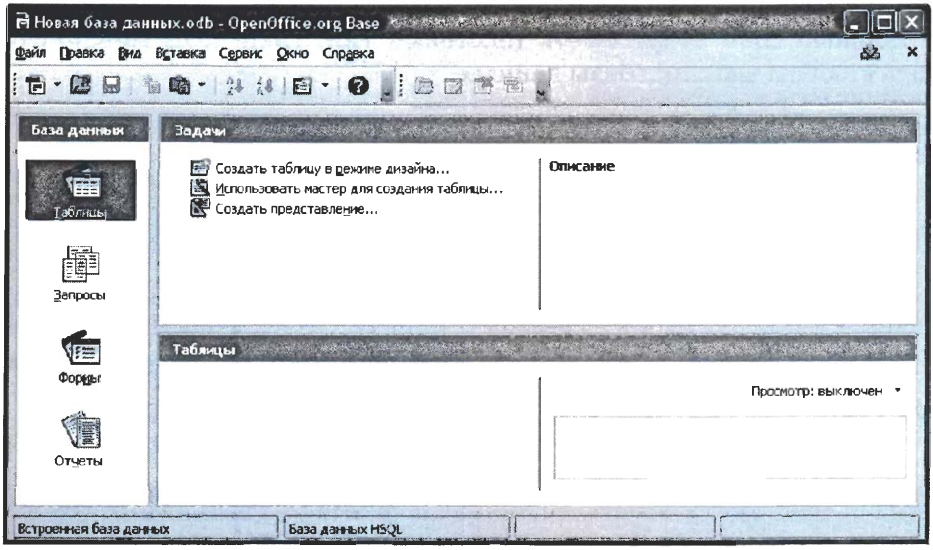


Рис. 2.12. Среда OpenOffice.org Base

Основными объектами СУБД являются таблицы, формы, запросы, отчёты.

Таблицы — это главный тип объектов. С ними вы уже знакомы. В таблицах хранятся данные. Реляционная база данных может состоять из множества взаимосвязанных таблиц.

Формы — это вспомогательные объекты. Они создаются для того, чтобы сделать более удобной работу пользователя при вводе, просмотре и редактировании данных в таблицах.

Запросы — это команды и результаты обращения пользователя к СУБД для поиска данных, сортировки, добавления, удаления и обновления записей.

Отчёты — это документы, сформированные на основе таблиц и запросов и предназначенные для вывода на печать.

2.6.3. Создание базы данных

В качестве примера рассмотрим процесс создания базы данных «Наш класс». Она будет состоять из одной таблицы, имеющей следующую структуру:

СПИСОК (КОД, ФАМИЛИЯ, ИМЯ, ДАТА РОЖДЕНИЯ, ПОЛ, РОСТ, АДРЕС, УВЛЕЧЕНИЕ, НАЛИЧИЕ ПК).

Поля КОД и РОСТ будут числовыми; поле ДАТА РОЖДЕНИЯ будет иметь тип дата; поле НАЛИЧИЕ ПК будет логическим; все остальные поля будут иметь текстовый тип. Поле КОД можно считать ключом таблицы базы данных.

Имя поля	Тип поля
Код	Числовой
Фамилия	Текстовый
Имя	Текстовый
Дата рождения	Дата
Пол	Текстовый
Рост	Числовой
Адрес	Текстовый
Увлечение	Текстовый
Наличие ПК	Логический

Создание базы данных начинается с открытия файла, в котором она будет храниться. Для этого нужно после запуска программы OpenOffice.org Base следовать указаниям мастера баз данных:

1) создать новую базу данных;

2) зарегистрировать базу данных (указать путь и имя файла).

Далее следует описать структуру таблицы (указать имена и типы всех полей) и ввести данные в таблицу.

Данные можно вводить непосредственно в **таблицу** (рис. 2.13), а можно создать для этого специальный шаблон — **форму** (рис. 2.14).

Рис. 2.13. Таблица для ввода данных

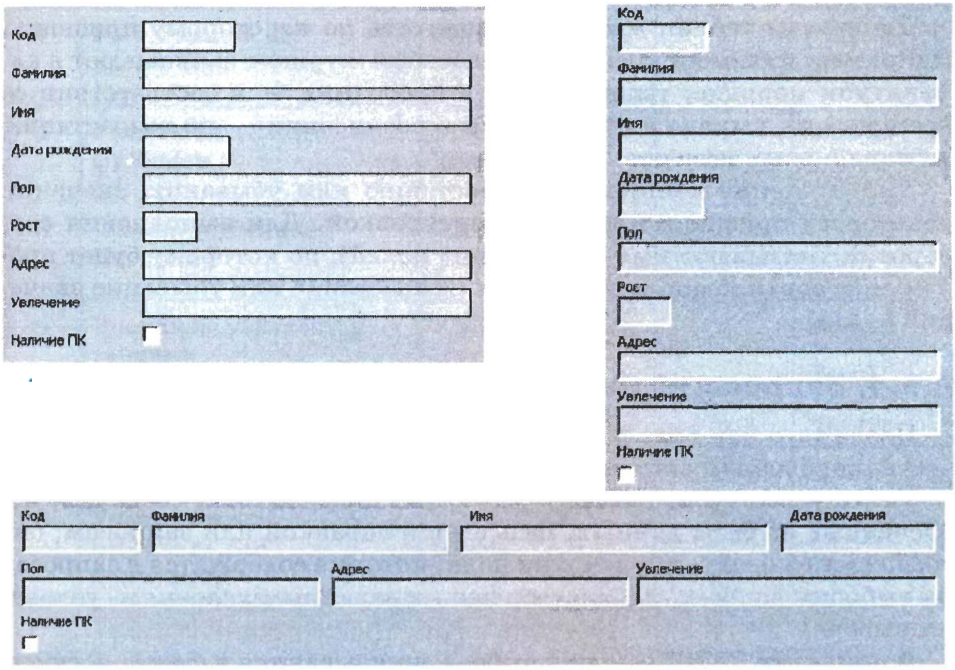


Рис. 2.14. Формы для ввода данных

После выполнения всех перечисленных выше действий будет получен следующий результат (рис. 2.15):

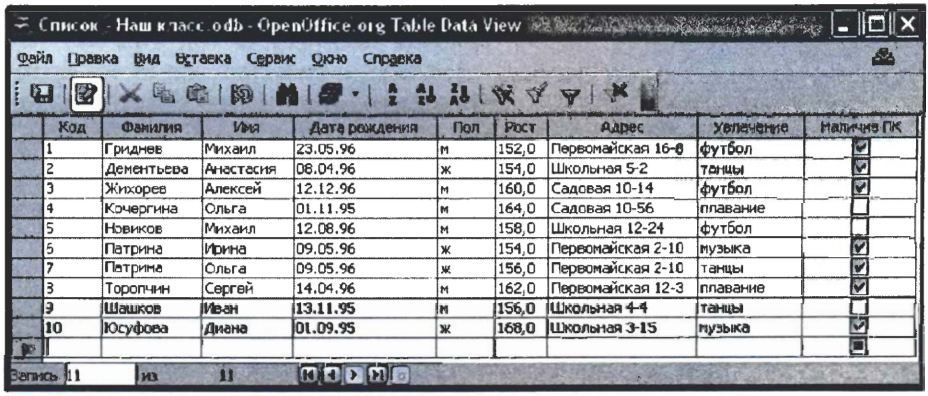


Рис. 2.15. Таблица «Список» базы данных «Наш класс»

Созданная и сохранённая база данных в дальнейшем может быть открыта для добавления новых записей, исправления и удаления существующих, изменения содержимого отдельных полей и структуры всей таблицы.

Данные из таблиц можно упорядочить по некоторому признаку. Например, фамилии учеников в классном журнале записывают в алфавитном порядке; телепередачи в программе — в соответствии со временем их выхода в эфир; уроки в расписании — по возрастанию их порядковых номеров.

Упорядочение данных по возрастанию или убыванию значений некоторого признака называют **сортировкой**. Для выполнения сортировки указывают имя поля (имена полей), по которому будет произведена сортировка, и её порядок (возрастание или убывание значений поля).

2.6.4. Запросы на выборку данных

После того как база данных создана, её можно использовать в качестве справочной системы.

Таблица, содержащая интересующие пользователя сведения, извлечённые из базы данных, называется справкой или **запросом**; она содержит только те записи и их поля, которые содержатся в запросах на выборку данных, удовлетворяющих заданным условиям (условиям выбора).

В командах СУБД условия выбора записываются в форме логических выражений — формализованных высказываний, сформулированных на естественном языке (табл. 2.6).

В логических выражениях имена полей базы данных связываются с определёнными значениями этих полей операциями отношений:

= равно;

<> не равно;

< меньше;

> больше;

<= меньше или равно (не больше);

>= больше или равно (не меньше).

На уроках математики вы применяете эти операции, составляя и решая числовые равенства, неравенства и их системы.

Операции отношений применимы и к текстовым полям. Их сравнение построено на лексикографическом принципе: из двух слов меньшим считается то слово, первая буква которого идёт по алфавиту раньше; если первые несколько букв двух слов одинаковы, то сравнение производится по первой различающейся букве.

Таблица 2.6

№	Высказывание	Логическое выражение	Номер записи	Значение
1	Рост ученика не превышает 160 см	РОСТ<=160	1	Истина
			4	Ложь
2	Ученик увлекается футболом	УВЛЕЧЕНИЕ='футбол'	1	Истина
			2	Ложь
3	Фамилия ученика — Патрина	ФАМИЛИЯ='Патрина'	6	Истина
			1	Ложь
4	Ученик не увлекается танцами	УВЛЕЧЕНИЕ<>'танцы'	2	Истина
			1	Ложь
5	Ученик родился в 1996 году	ДАТА>#31.12.95#	8	Истина
			10	Ложь
6	Ученик имеет персональный компьютер	НАЛИЧИЕ ПК=1	7	Истина
			9	Ложь

Значение поля текстового типа и некоторая текстовая величина равны, если они содержат одинаковое количество символов и все их символы, стоящие в позициях с одинаковыми номерами, совпадают.

При сравнении текстовых величин следует иметь в виду, что пробел — это тоже символ, хотя он и «меньше» любой буквы.

Сравнение дат построено иначе — одна дата считается меньше другой, если она относится к более раннему времени. Например, истинными будут следующие отношения:

01.11.95<02.11.95;
29.11.95<02.12.95;
29.11.95<01.11.96.

Условия выбора могут задаваться не только простыми, но и сложными логическими выражениями, содержащими логические операции. С основными логическими операциями И, ИЛИ, НЕ вы познакомились в главе 1.

Таблица 2.7

№	Высказывание	Логическое выражение	Номер записи	Значение
1	Рост ученика больше 160 см, и ученик увлекается плаванием	РОСТ>160 И УВЛЕЧЕНИЕ='плавание'	4	Истина
			10	Ложь
2	Рост ученика больше 160 см или ученик увлекается плаванием	РОСТ>160 ИЛИ УВЛЕЧЕНИЕ='плавание'	10	Истина
			1	Ложь
3	День рождения Ольги не 09.05.96	ИМЯ='Ольга' И ДАТА#09.05.96#	4	Истина
			7	Ложь

С помощью запросов пользователь может быстро найти в базе данных и вывести на экран компьютера интересующую его информацию. Но для решения большинства практических задач найденную информацию необходимо представить в определённой форме и подготовить к выводу на печать. Этот этап работы называется подготовкой **отчёта**.

САМОЕ ГЛАВНОЕ

Программное обеспечение для создания баз данных, хранения и поиска в них необходимой информации называется **системой управления базами данных (СУБД)**.

Основными объектами СУБД являются **таблицы, формы, запросы, отчёты.**

С помощью запросов на выборку данных, удовлетворяющих заданным условиям (условиям выбора), пользователь получает из базы данных только те записи и их поля, которые ему нужны. В командах СУБД условия выбора записываются в форме логических выражений.

Вопросы и задания

1. Что такое СУБД?
2. Какая СУБД установлена на компьютерах в вашем классе?
3. С чего начинается создание БД?

4. Перечислите основные объекты СУБД. Какие функции они выполняют?
5. Ниже в табличной форме представлены характеристики ноутбуков, имеющихся в продаже в компьютерном салоне:

№	Название	Жёсткий диск (ГБ)	Оперативная память (МБ)
4	Sony Vaio AW2X	500	4096
3	Lenovo S10e	250	3072
5	Asus F70SL	250	2048
1	Acer E525	160	2048
2	Samsung NC20	160	1024
6	Roverbook V212	120	1024

а) Какую строку будет занимать запись, содержащая сведения о ноутбуке Asus F70SL, после сортировки по возрастанию значений поля НАЗВАНИЕ?

б) Какую строку будет занимать запись, содержащая сведения о ноутбуке Asus F70SL, после сортировки по убыванию значений поля ЖЁСТКИЙ ДИСК?

в) Какую строку будет занимать запись, содержащая сведения о ноутбуке Asus F70SL, после сортировки сначала по убыванию значений поля ОПЕРАТИВНАЯ ПАМЯТЬ, затем по возрастанию значений поля ЖЁСТКИЙ ДИСК?

6. Как будет выглядеть список (фамилия, имя) учеников после сортировки по возрастанию значений поля ДАТА РОЖДЕНИЯ базы данных «Наш класс» (рис. 2.14)?
7. Укажите все записи базы данных «Наш класс» (рис. 2.14), для которых будут истинными простые логические выражения 1–6 (табл. 2.6).
8. Укажите все записи базы данных «Наш класс» (рис. 2.14), для которых будут истинными сложные логические выражения 1–3 (табл. 2.7).
9. Какова цель запроса на выборку?

10. Ниже в табличной форме представлен фрагмент базы данных с годовыми оценками учащихся:

Фамилия	Пол	Алгебра	Геометрия	Информатика	Физика
Алексеева	Ж	3	3	4	3
Воронин	М	4	4	4	3
Ильин	М	4	3	3	4
Костин	М	5	4	5	4
Сизова	Ж	5	5	5	4
Школина	Ж	5	5	5	5

Сколько записей в данном фрагменте удовлетворяет следующему условию?

а) АЛГЕБРА>3 И ИНФОРМАТИКА>4 И ПОЛ='М'

б) (АЛГЕБРА>4 ИЛИ ИНФОРМАТИКА>4) И ПОЛ='Ж'

в) ФИЗИКА=3 ИЛИ АЛГЕБРА=3 ИЛИ ГЕОМЕТРИЯ=3 ИЛИ ИНФОРМАТИКА=3

г) (ФИЗИКА=3 ИЛИ АЛГЕБРА=3) И (ГЕОМЕТРИЯ=3 ИЛИ ИНФОРМАТИКА=3)

11. Ниже в табличной форме представлен фрагмент базы данных с результатами олимпиады по информатике:

Фамилия	Пол	Задача1	Задача2	Задача3	Сумма
Жариков	М	15	20	25	60
Костин	М	10	10	10	30
Кузнецов	М	20	25	30	75
Михайлова	Ж	25	20	10	55
Сизова	Ж	30	30	30	90
Старовойтова	Ж	20	25	25	70
Школина	Ж	30	25	25	80

Сколько записей в данном фрагменте удовлетворяет следующему условию?

а) ПОЛ='М' И СУММА>55

б) (ЗАДАЧА1<ЗАДАЧА2) И (ЗАДАЧА2<ЗАДАЧА3)

в) ЗАДАЧА1=30 ИЛИ ЗАДАЧА2=30 ИЛИ ЗАДАЧА3=30

г) ЗАДАЧА1=30 И ЗАДАЧА2=30 И ЗАДАЧА3=30

Тестовые задания для самоконтроля

1. Выберите верное утверждение:
   а) Один объект может иметь только одну модель
   б) Разные объекты не могут описываться одной моделью
   в) Электрическая схема — это модель электрической цепи
   г) Модель полностью повторяет изучаемый объект
2. Выберите неверное утверждение:
   а) Натурные модели — реальные объекты, в уменьшенном или увеличенном виде воспроизводящие внешний вид, структуру или поведение моделируемого объекта
   б) Информационные модели описывают объект-оригинал на одном из языков кодирования информации
   в) Динамические модели отражают процессы изменения и развития объектов во времени
   г) За основу классификации моделей может быть взята только предметная область, к которой они относятся
3. Какие признаки объекта должны быть отражены в информационной модели ученика, позволяющей получать следующие сведения: возраст учеников, увлекающихся плаванием; количество девочек, занимающихся танцами; фамилии и имена учеников старше 14 лет?
   а) имя, фамилия, увлечение
   б) имя, фамилия, пол, пение, плавание, возраст
   в) имя, увлечение, пол, возраст
   г) имя, фамилия, пол, увлечение, возраст

4. Выберите элемент информационной модели учащегося, существенный для выставления ему оценки за контрольную работу по информатике:
   а) наличие домашнего компьютера
   б) количество правильно выполненных заданий
   в) время, затраченное на выполнение контрольной работы
   г) средний балл за предшествующие уроки информатики
5. Замена реального объекта его формальным описанием — это:
   а) анализ
   б) моделирование
   в) формализация
   г) алгоритмизация
6. Выберите знаковую модель:
   а) рисунок
   б) схема
   в) таблица
   г) формула
7. Выберите образную модель:
   а) фотография
   б) схема
   в) текст
   г) формула
8. Выберите смешанную модель:
   а) фотография
   б) схема
   в) текст
   г) формула
9. Описания предметов, ситуаций, событий, процессов на естественных языках — это:
   а) словесные модели
   б) логические модели
   в) геометрические модели
   г) алгебраические модели

10. Модели, реализованные с помощью систем программирования, электронных таблиц, специализированных математических пакетов и программных средств для моделирования, называются:

а) математическими моделями

б) компьютерными моделями

в) имитационными моделями

г) экономическими моделями

11. Файловая система персонального компьютера наиболее адекватно может быть описана в виде:

а) математической модели

б) табличной модели

в) натурной модели

г) иерархической модели

12. Графической моделью иерархической системы является:

а) цепь

б) сеть

в) генеалогическое дерево

г) дерево

13. Расписание движения электропоездов может рассматриваться как пример:

а) табличной модели

б) графической модели

в) имитационной модели

г) натурной модели

14. Какая тройка понятий находится в отношении «объект – натурная модель — информационная модель»?

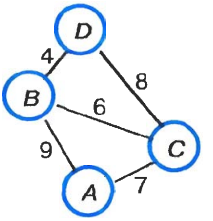
а) человек — анатомический скелет — манекен

б) человек — медицинская карта — фотография

в) автомобиль — рекламный буклет с техническими характеристиками автомобиля — атлас автомобильных дорог

г) автомобиль — игрушечный автомобиль — техническое описание автомобиля

15. На схеме изображены дороги между населёнными пунктами A, B, C, D и указаны протяжённости этих дорог.



Определите, какие два пункта наиболее удалены друг от друга. Укажите длину кратчайшего пути между ними.

а) 17

б) 15

в) 13

г) 9

16. Населённые пункты A, B, C, D соединены дорогами. Время проезда на автомобиле из города в город по соответствующим дорогам указано в таблице:

	A	B	C	D
A	×	2	4	4
B	2	×	5	3
C	4	5	×	1
D	4	3	1	×

Турист, выезжающий из пункта A, хочет посетить все города за кратчайшее время. Укажите соответствующий маршрут.

а) $ABCD$

б) $ACBD$

в) $ADCB$

г) $ABDC$

17. В школе учатся четыре ученика — Андреев, Иванов, Петров, Сидоров, имеющие разные увлечения. Один из них увлекается теннисом, другой — бальными танцами, третий — живописью, четвёртый — пением. О них известно:

- Иванов и Сидоров присутствовали на концерте хора, когда пел их товарищ;
- Петров и теннисист позировали художнику;

- теннисист дружит с Андреевым и хочет познакомиться с Ивановым.

Чем увлекается Андреев?

а) теннисом

б) живописью

в) танцами

г) пением

18. Два игрока играют в следующую игру. Перед ними лежат три кучки камней, в первой из которых 2 камня, во второй — 3 камня, в третьей — 4 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или удваивает число камней в какой-то куче, или добавляет по два камня в каждую из куч. Выигрывает игрок, после хода которого либо в одной из куч становится не менее 15 камней, либо общее число камней во всех трёх кучах становится не менее 25. Кто выигрывает при безошибочной игре обоих игроков?

а) игрок, делающий первый ход

б) игрок, делающий второй ход

в) каждый игрок имеет одинаковый шанс на победу

г) для этой игры нет выигрышной стратегии

19. База данных — это:

а) набор данных, собранных на одной дискете

б) таблица, позволяющая хранить и обрабатывать данные и формулы

в) прикладная программа для обработки информации пользователя

г) совокупность данных, организованных по определённым правилам, предназначенная для хранения во внешней памяти компьютера и постоянного применения

20. Какая база данных основана на табличном представлении информации об объектах?

а) иерархическая

б) сетевая

в) распределённая

г) реляционная

21. Строка таблицы, содержащая информацию об одном конкретном объекте, — это:

а) поле

б) запись

в) отчёт

г) форма

22. Столбец таблицы, содержащий определённую характеристику объекта, — это:

а) поле

б) запись

в) отчёт

г) ключ

23. Системы управления базами данных используются для:

а) создания баз данных, хранения и поиска в них необходимой информации

б) сортировки данных

в) организации доступа к информации в компьютерной сети

г) создания баз данных

24. Какое из слов НЕ является названием базы данных?

а) Microsoft Access

б) OpenOffice.org Base

в) OpenOffice.org Writer

г) FoxPro

25. Ниже в табличной форме представлен фрагмент базы данных:

№	Наименование товара	Цена	Количество
1	Монитор	7654	20
2	Клавиатура	1340	26
3	Мышь	235	10
4	Принтер	3770	8
5	Колонки акустические	480	16
6	Сканер планшетный	2880	10

На какой позиции окажется товар «Сканер планшетный», если произвести сортировку данной таблицы по возрастанию столбца КОЛИЧЕСТВО?

а) 5

б) 2

в) 3

г) 6

26. Ниже в табличной форме представлен фрагмент базы данных «Продажа канцелярских товаров»:

Наименование	Цена	Продано
Карандаш	5	60
Линейка	18	7
Папка	20	32
Ручка	25	40
Тетрадь	15	500

Сколько записей в данном фрагменте удовлетворяет условию ЦЕНА>20 ИЛИ ПРОДАНО<50?

а) 1

б) 2

в) 3

г) 4

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.

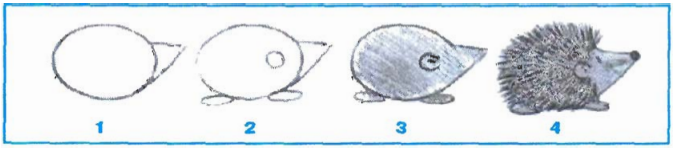
Пример 1. Задача «Найти среднее арифметическое двух чисел» решается в три шага:

1) задумать два числа;
2) сложить два задуманных числа;
3) полученную сумму разделить на 2.

Пример 2. Задача «Внести деньги на счёт телефона» подразделяется на следующие шаги:

1) подойти к терминалу по оплате платежей;
2) выбрать оператора связи;
3) ввести номер телефона;
4) проверить правильность введённого номера;
5) вставить денежную купюру в купюроприёмник;
6) дождаться сообщения о зачислении денег на счет;
7) получить чек.

Пример 3. Этапы решения задачи «Нарисовать весёлого ёжика» представлены графически:



1) 2) 3) 4)

Нахождение среднего арифметического, внесение денег на телефонный счёт и рисование ежа — на первый взгляд совершенно разные процессы. Но у них есть общая черта: каждый из этих процессов описывается последовательностями кратких указаний, точное следование которым позволяет получить требуемый результат. Последовательности указаний, приведённые в примерах 1–3, являются алгоритмами решения соответствующих задач. Исполнитель этих алгоритмов — человек.

Алгоритм может представлять собой описание некоторой последовательности вычислений (пример 1) или шагов нематематического характера (примеры 2–3). Но в любом случае перед его разработкой должны быть чётко определены начальные условия (исходные дан-

ные) 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, грамматические правила, правила геометрических построений и т. д.

Анимации «Работа с алгоритмом», «Наибольший общий делитель», «Наименьшее общее кратное» (http://school-collection.edu.ru/) помогут вам вспомнить некоторые алгоритмы, изученные на уроках русского языка и математики.

Пример 4. Некоторый алгоритм приводит к тому, что из одной цепочки символов получается новая цепочка следующим образом:

1. Вычисляется длина (в символах) исходной цепочки символов.
2. Если длина исходной цепочки нечётна, то к исходной цепочке справа приписывается цифра 1, иначе цепочка не изменяется.
3. Символы попарно меняются местами (первый — со вторым, третий — с четвёртым, пятый — с шестым и т. д).
4. Справа к полученной цепочке приписывается цифра 2.

Получившаяся таким образом цепочка является результатом работы алгоритма.

Так, если исходной была цепочка A#B, то результатом работы алгоритма будет цепочка #A1B2, а если исходной цепочкой была АБВ@, то результатом работы алгоритма будет цепочка БА@В2.

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 одну и ту же команду 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 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.

Система команд исполнителя. Предписание исполнителю о выполнении отдельного законченного действия называется **командой.** 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 управления* и *программного управления*. В первом случае исполнитель ожидает команд от человека и каждую поступившую команду немедленно выполняет. Во втором случае исполнителю сначала задаётся полная последовательность команд (программа), а затем он выполняет все эти команды в автоматическом режиме. Ряд исполнителей работает только в одном из названных режимов.

Рассмотрим примеры исполнителей.

Пример 5. Исполнитель **Черепашка** перемещается на экране компьютера, оставляя след в виде линии. Система команд Черепашки состоит из двух команд:

`Вперёд n` (где `n` — целое число) — вызывает передвижение Черепашки 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 Черепашки на `m` градусов по часовой стрелке.

Запись `Повтори k [<Команда1> <Команда2> ... <Командаn>]` означает, что последовательность команд в скобках повторится *k* раз.

Подумайте, какая фигура появится на экране после выполнения Черепашкой следующего алгоритма.

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

Пример 6. Система команд исполнителя **Вычислитель** состоит из двух команд, которым присвоены номера:

1 — `вычти 1`
2 — `умножь на 3`

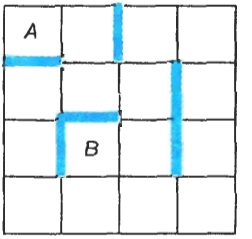
Первая из них уменьшает число на 1, вторая увеличивает число в 3 раза. При записи алгоритмов для краткости указываются лишь номера команд. Например, алгоритм `21212` означает следующую последовательность команд:

```
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 7. 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, 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 от встречи со стенами?

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 — модель деятельности исполнителя алгоритмов.

3.1.3. Свойства алгоритма

Не любая инструкция, последовательность предписаний или план действий может считаться алгоритмом. Каждый алгоритм обязательно обладает следующими свойствами: дискретность, понятность, определённость, результативность и массовость.

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 на следующем шаге.

Свойство результативности означает, что алгоритм должен обеспечивать возможность получения результата после конечного, возможно, очень большого, числа шагов. При этом результатом считается не только обусловленный постановкой задачи ответ, но и вывод о невозможности продолжения по какой-либо причине 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 8. Рассмотрим один из методов нахождения всех простых чисел, не превышающих n. Этот метод называется «решето Эратосфена», по имени предложившего его древнегреческого учёного Эратосфена.

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

1) выписать подряд все целые 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.

Более наглядное представление о методе нахождения простых чисел вы сможете получить с помощью анимации «Решето Эратосфена» (http://school-collection.edu.ru/).

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

- дискретности — процесс нахождения простых чисел разбит на шаги;
- понятности — каждая команда понятна ученику 9 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 9. Из кучки, содержащей любое, большее трёх, 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 кучке кратно 3, то уступить ход противнику, иначе начинать игру.
2. Своим очередным ходом каждый раз дополнять число предметов, взятых соперником, до 3 (число оставшихся предметов должно быть кратно 3).

Исполнитель может не вникать в смысл того, что он делает, и не рассуждать, почему он поступает так, а не иначе, то есть 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) создается машина (автоматическое устройство), способная выполнять эти операции в последовательности, заданной в алгоритме;
3) человек освобождается от рутинной деятельности, выполнение алгоритма поручается автоматическому устройству.

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. Подберите синонимы к слову «предписание».
3. Приведите примеры алгоритмов, изучаемых вами в школе.
4. Кто может быть исполнителем алгоритма?
5. Приведите пример формального исполнителя. Приведите пример, когда человек выступает в роли формального исполнителя.
6. Какие команды должны быть у робота, выполняющего функции: а) кассира в магазине; б) дворника; в) охранника?
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?
10. Перечислите основные свойства алгоритма.
11. К чему может привести отсутствие какого-либо свойства у алгоритма? Приведите примеры.
12. В чём важность возможности формального исполнения алгоритма?
13. Последовательность чисел строится по следующему алгоритму: первые два числа последовательности принимаются равными 1; каждое следующее число последовательности принимается рав-

ным сумме двух предыдущих чисел. Запишите 10 первых членов этой последовательности.

14. Некоторый алгоритм получает из одной цепочки символов новую цепочку следующим образом. Сначала записывается исходная цепочка символов, после нее 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?

15. Найдите в сети Интернет анимацию шагов алгоритма Эратосфена. С помощью алгоритма Эратосфена найдите все простые числа, не превышающие 50.

16. Что будет результатом исполнения Черепашкой (см. пример 5) алгоритма?

```
Повтори 8 [Направо 45 Вперёд 45]
```

17. Запишите алгоритм для исполнителя Вычислитель (пример 6), содержащий не более 5 команд:

а) получения из числа 3 числа 16;

б) получения из числа 1 числа 25.

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

19. В какой клетке должен находиться исполнитель Робот (пример 7), чтобы после выполнения алгоритма `3241` в неё же и вернуться?

§ 3.2
Способы записи алгоритмов

Ключевые слова:
- словесное описание
- построчная запись
- блок-схема
- школьный алгоритмический язык

Существуют различные способы записи алгоритмов. Основными среди них являются:
- словесные;
- графические;
- на алгоритмических языках.

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. Словес-

ное описание имеет минимум ограничений и является наименее формализованным. Однако все разговорные языки обладают неоднозначностью, поэтому могут возникнуть различные толкования текста алгоритма, заданного таким образом. Алгоритм в словесной форме может оказаться очень объёмным и трудным для восприятия.

Пример 1. Словесное описание алгоритма нахождения наибольшего общего делителя (НОД) пары целых 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 — 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

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

Блок-схема представляет собой графический документ, дающий представление о порядке работы алгоритма. Здесь предписания изображаются с помощью различных геометрических фигур, а последовательность выполнения шагов указывается с помощью линий, соединяющих эти фигуры. Направления линий связи слева направо и сверху вниз считаются стандартными, и линии связи изображаются без стрелок, в противоположном случае — со стрелками.

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 схемы. В блок данных входит одна линия связи, и из блока исходит 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 исходит одна линия связи.

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

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

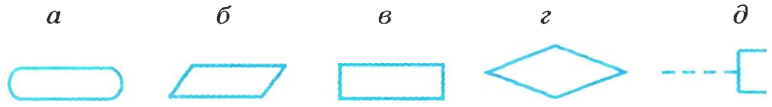


Рис. 3.2. Обозначения на блок-схемах

Пример 3. Запись алгоритма Евклида с помощью блок-схемы (рис. 3.3).

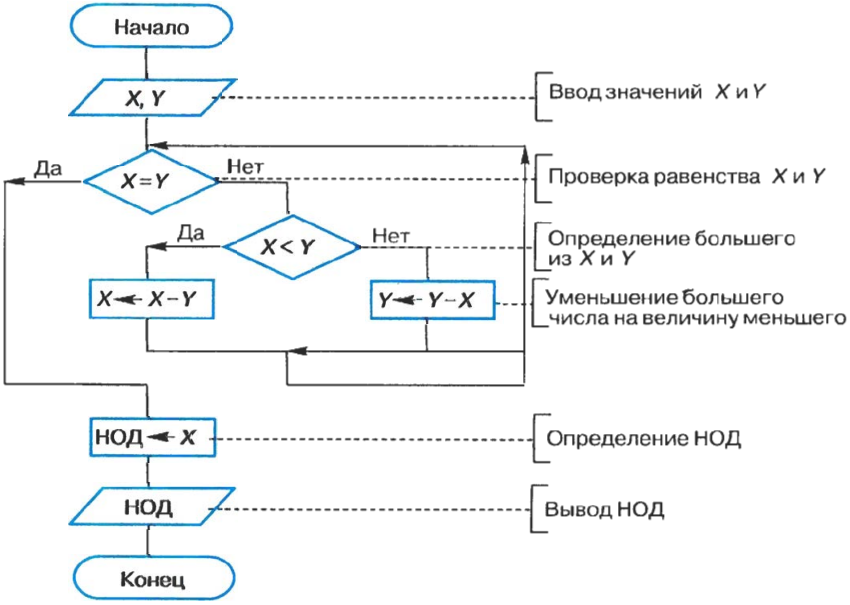


Рис. 3.3. Запись алгоритма Евклида с помощью блок-схемы

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 представлениям решения задачи в общем виде.

3.2.3. Алгоритмические языки

Алгоритмические языки — формальные языки, предназначенные для записи алгоритмов. 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.

Класс алгоритмических языков очень широк. При изучении курса информатики в школах используются различные версии школьного (учебного) алгоритмического языка.

Школьный алгоритмический язык. Для записи алгоритмов на школьном алгоритмическом языке используется некоторое ограни-

ченное число слов, смысл и способ употребления которых заданы раз и навсегда. Это так называемые служебные слова: **алг** (алгоритм), **дано**, **надо**, **нач** (начало), **кон** (конец), **арг** (аргумент), **рез** (результат) 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
```

Пример 4. Алгоритм, 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 сосуд ёмкостью 5 л
  вылить всё из сосуда ёмкостью 5 л в сосуд ёмкостью 12 л
кон
```

По ссылке http://www.niisi.ru/kumir/ вы можете скачать систему КуМир (Комплект учебных миров), в которой используется школьный алгоритмический язык, со встроенными исполнителями Робот, Чертёжник, Водолей и другими. Кумир работает в операционных системах Windows и Linux.

Далее, говоря об алгоритмическом языке, мы будем иметь в виду именно школьный алгоритмический язык.

САМОЕ ГЛАВНОЕ

Существуют различные способы записи алгоритмов: словесное описание, построчная запись, блок-схемы, школьный алгоритмический язык 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. Дайте словесное описание алгоритма сложения двух обыкновенных дробей a/b и c/d.
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 их 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?».
6. Запишите на алгоритмическом языке алгоритм построения окружности заданного радиуса r, проходящей через заданные точки A и B.
7. В среде КуМир запишите и выполните алгоритм переливаний (пример 4) для исполнителя Водолей.

§ 3.3
Объекты алгоритмов

Ключевые слова:
- величина
- константа
- переменная
- тип
- имя
- присваивание
- выражение
- таблица

3.3.1. Величины

Алгоритмы описывают 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 меняется 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 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 алгоритмическом языке обозначаются **цел** и **вещ** соответственно.

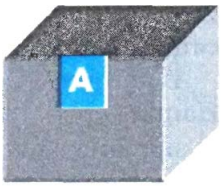
В задачах, возникающих в повседневной жизни, встречаются и нечисловые величины, значениями которых являются символы, слова, тексты и др. При составлении алгоритмов обработки текстовой информации используют величины символьного (**сим**) и литерного (**лит**) типов. Значением символьной величины является 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: 'алгоритм', 'литерная величина', '2011'.

Величины логического (**лог**) типа могут принимать всего два значения:

- ДА (ИСТИНА, TRUE, 1);
- НЕТ (ЛОЖЬ, FALSE, 0).

[1] Слово «вещественный» принято использовать вместо слова «действительный».

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, CENA и т. д.



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

3.3.2. Выражения

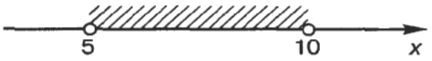
Выражение — языковая конструкция для вычисления значения с помощью одного или нескольких операндов.

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 пяти, 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 удовлетворяться или не удовлетворяться. 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.

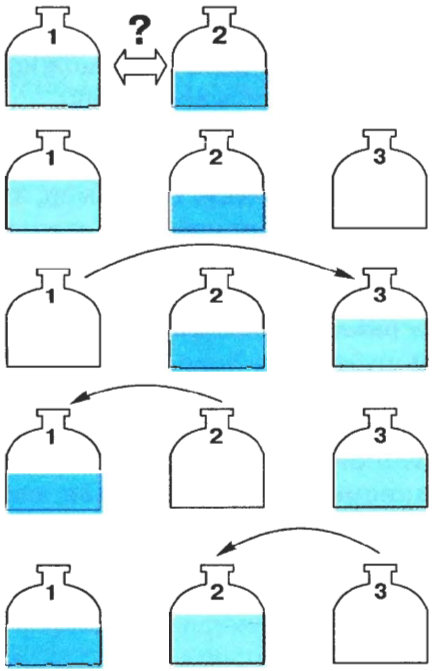


Рис. 3.4. Алгоритм переливаний жидкостей

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

```
алг обмен значениями (лит A, B)
  арг A, B
  рез A, B
нач лит M
  M:=A
  A:=B
  B:=M
кон
```

Если *A* и *B* — числовые величины, то обмен их значениями 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 человека часто используются всевозможные таблицы. Это, например, список учащихся в классном журнале, табель успеваемости, таблица результатов спортивных соревнований и т. д. При этом наиболее часто 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 9 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**. Индекс записывается 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*. Тогда *LES*[1,1] = 6, *LES*[2,5] = 6, *LES*[3,4] = 0.

Образно линейная и прямоугольная таблицы показаны на рис. 3.7.

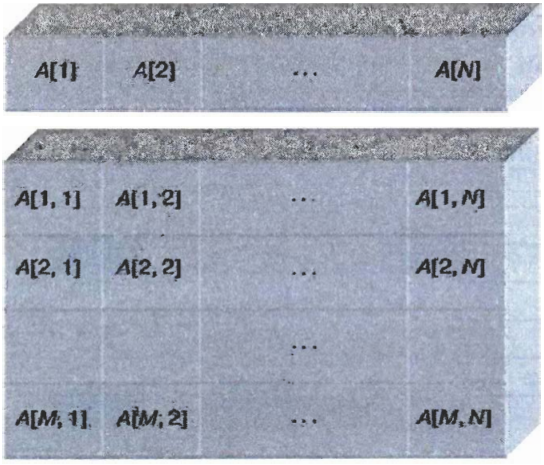


Рис. 3.7. Образное представление линейной и прямоугольной таблиц

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 меняются 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 индексами.

Вопросы и задания

1. Что такое величина? Чем отличаются постоянные и переменные величины?
2. Величины каких типов используются при записи алгоритмов?
3. Укажите тип величины, если её значение равно: 2010; 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. Приведите по одному примеру допустимых и недопустимых значений для каждой из величин:

   а) температура человека;
   б) скорость автомашины;
   в) площадь государства;
   г) название дня недели.

6. Для чего предназначена команда присваивания? Каковы её основные свойства?
7. Какие команды присваивания составлены 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. 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 на алгоритмическом языке.
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 из линейной записи в общепринятую:

а) $a * b / c$;
г) $(a + b) / c$;
б) $a / b * c$;
д) $a + b / c + d$;
в) $a + b / c$;
е) $(a + b) / (c + d)$.

13. Запишите на алгоритмическом языке:

а) $ax^2 + bx + c$;

б) $v + \frac{at^2}{2}$;

в) $\frac{1}{2}(a + b)h$;

г) $\frac{1 + x_1 x_2}{b^2 c}$;

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;

ж) точка с координатами (x, y) лежит в круге радиуса r с центром в начале координат.

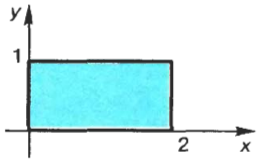
15. Изобразите в декартовой прямоугольной системе координат область, в которой и только в которой истинны следующие логические выражения:

а) `(x>=-1)` **и** `(x<=1)` **и** `(y>=-1)` **и** `(y<=1)`

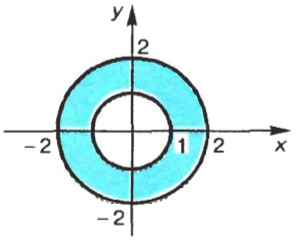
б) `(y>=x)` **и** `(y>=-x)` **и** `(y<=1)`

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

а



б



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;

г) уравнение $ax^2 + bx + c = 0$ имеет ровно один корень.

18. Какие из приведённых ниже величин целесообразно представлять с помощью таблиц?

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, названия дней месяца, имя человека, имена девочек, площадь фигуры, периметры нескольких прямоугольников, самая холодная температура воздуха в январе, количество девочек в классе, самые жаркие дни лета, самая дождливая декада июня.

§ 3.4
Основные алгоритмические конструкции

Ключевые слова:
- следование
- ветвление
- повторение
- линейные алгоритмы
- разветвляющиеся алгоритмы
- циклические алгоритмы

Человеку в жизни и практической деятельности 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.

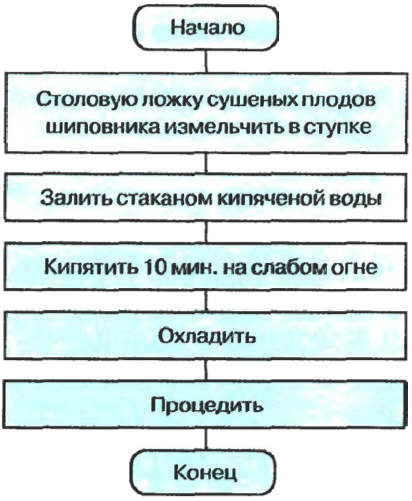
3.4.1. Следование

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

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

Рис. 3.8. Алгоритмическая конструкция «следование»

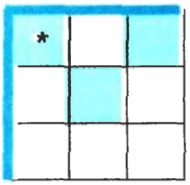
Пример 1. Линейный алгоритм приготовления отвара шиповника.



Обратите внимание, что многие из предписани.· того алгоритма могут потребовать детализации — представления виде некоторой совокупности более мелких предписаний.

Пример 2. У исполнителя Робот есть четыре к‹ анды 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:




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

Пример 3. 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 4. Некоторый исполнитель может выполнять над целыми числами кроме операций сложения, вычитания, умножения и деления ещё две операции: с помощью операции `div` вычисляется целое 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 500 (*k*500), 100 (*k*100), 50 (*k*50) и 10 (*k*10) рублей.

```
k500:=s div 500
s:=s mod 500
k100:=s div 100
s:=s mod 100
k50:=s div 50
s:=s mod 50
k10:=s div 10
```

Ознакомьтесь с модулем для коллективной работы «Линейные алгоритмы» (http://school-collection.edu.ru/). Совместно с товарищами постарайтесь составить алгоритмы для имеющихся в модуле задач. Пройдите тестирование.

3.4.2. 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. 3.9. Каждая ветвь может быть любой степени сложности (рис. 3.9, *а*), а может вообще не содержать предписаний (рис. 3.9, *б*).

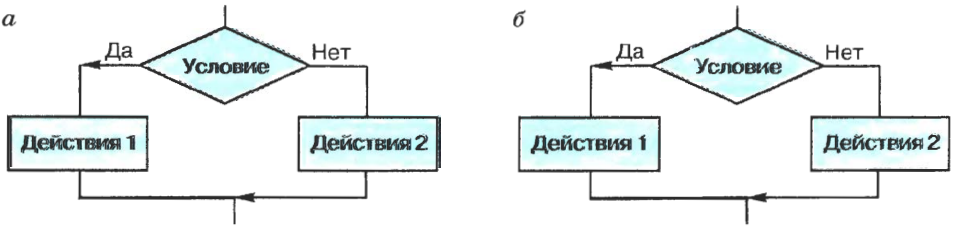


Рис. 3.9. Структура «ветвление»: *а* — полная форма ветвления; *б* — сокращённая форма ветвления

На алгоритмическом языке команда ветвления записывается так:

Полная форма ветвления:

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

Сокращённая форма ветвления:

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

Пример 5

```
алг правописание частиц НЕ, НИ
нач
  если частица под ударением
    то писать НЕ
    иначе писать НИ
  все
кон
```

Пример 6

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

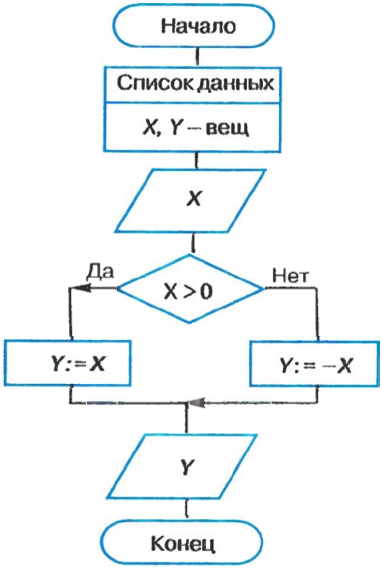
Для записи условий, по которым разветвляется алгоритм, используются **операции сравнения**:

`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.

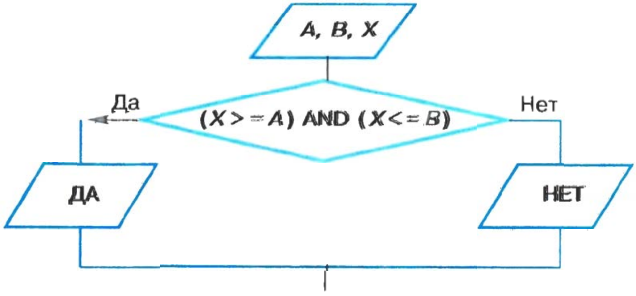
Пример 7. Алгоритм вычисления функции $f(x) = |x|$ для произвольного числа x. (Блок-схема расположена на следующей странице.)

Обратите внимание на второй блок этой блок-схемы. В нём представлены имена и типы величин (данных), обрабатываемых в алгоритме.



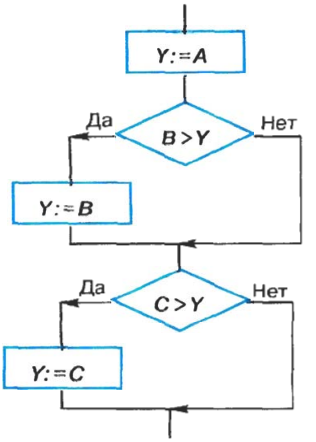
Условия, состоящие из одной операции сравнения, называются **простыми**. В качестве условий при организации ветвлений можно использовать и составные условия. **Составные условия** получаются из простых с помощью логических связок `and` **(и)**, `or` **(или)**, `not` **(не)**: `and` означает одновременное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

Пример 8. Алгоритм определения принадлежности точки X отрезку $[A; B]$. Если точка X принадлежит данному отрезку, то выводится ответ 'ДА', в противном случае — 'НЕТ'.



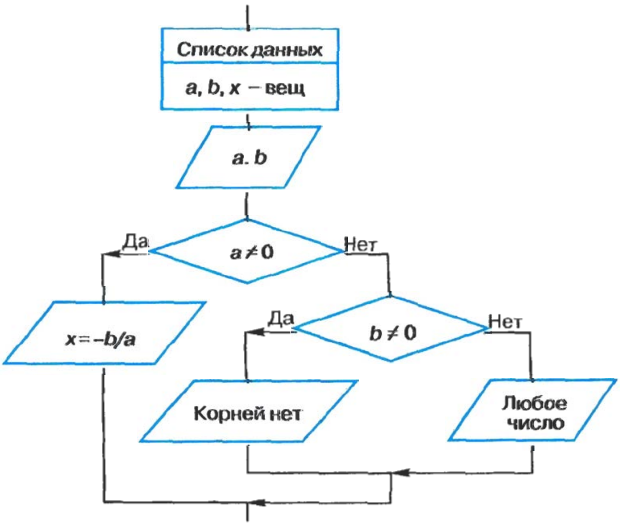
Существует достаточно много ситуаций, в которых приходится выбирать не из двух, а из трёх и более вариантов. Есть разные способы построения соответствующих алгоритмов. Один из них — составить комбинацию из нескольких ветвлений.

Пример 9. Алгоритм, в котором переменной Y присваивается значение большей из трёх величин A, B и C.



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 10. Алгоритм решения линейного уравнения $ax + b = 0$.



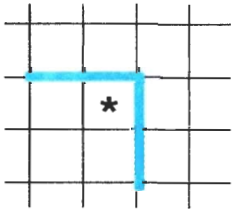
Пример 11. Исполнитель Робот может выполнять ту или иную последовательность действий в зависимости от выполнения следующих простых условий:

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

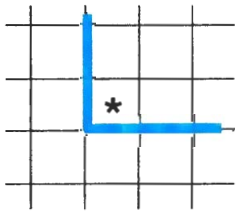
Также Робот может действовать в зависимости от выполнения составных условий. Подумайте, в какую клетку переместится Робот при выполнении следующего фрагмента алгоритма.

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

а



б



Ознакомьтесь с модулем для коллективной работы «Алгоритмы с ветвящейся структурой» (http://school-collection.edu.ru/). Совместно с товарищами постарайтесь составить алгоритмы для имеющихся в модуле задач. Пройдите тестирование.

3.4.3. Повторение

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 повторений различают три типа циклов:

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

Цикл с заданным условием продолжения работы (цикл-ПОКА, цикл с предусловием). Логика работы этой конструкции описывается схемой, показанной на рис. 3.10.

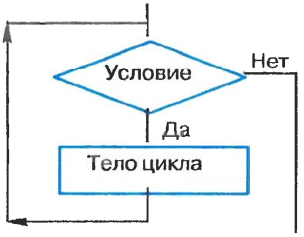


Рис. 3.10. Цикл с предусловием

На алгоритмическом языке эта конструкция записывается так:

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

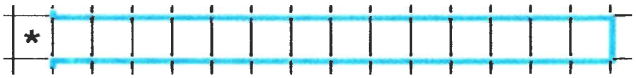
Выполняется цикл-ПОКА следующим образом: 1) проверяется условие (вычисляется значение логического выражения); 2) если условие удовлетворяется (Да), то выполняется тело цикла и снова осуществляется переход к проверке условия; если же условие не удовлетворяется, то выполнение цикла заканчивается. Возможны случаи, когда тело цикла не будет выполнено ни разу.

Пример 12. Алгоритм, по которому из всех имеющихся кирпичей отбираются целые кирпичи и складываются в машину.

```
алг отбор
нач
  нц пока есть кирпичи
     взять один кирпич
     если кирпич целый
       то положить кирпич в машину
```

```
        иначе отложить кирпич в сторону
       все
      кц
кон
```

Пример 13. Правее Робота расположен коридор неизвестной длины. Необходимо, чтобы Робот закрасил все клетки этого коридора.



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

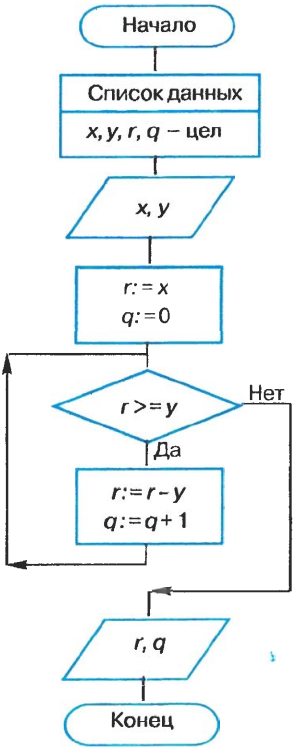
```
вправо
закрась
```

Соответствующий алгоритм для Робота будет иметь вид:

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

Пример 14. Требуется, не пользуясь операцией деления, получить частное q и остаток r от деления целого числа x на целое число y.

Представим операцию деления как последовательность вычитания делителя из делимого. Причём вычитать будем до тех пор, пока результат вычитания не станет меньше вычитаемого (делителя). В этом случае количество вычитаний будет частным от деления q, а последняя разность — остатком 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	23	5	-	-	
3	$r := x$	23	5	23	-	
4	$q := 0$	23	5	23	0	
5	$r >= y$					23 > 5 (Да)
6	$r := r - y$	23	5	18	0	
7	$q := q + 1$	23	5	18	1	
8	$r >= y$					18 > 5 (Да)
9	$r := r - y$	23	5	13	1	
10	$q := q + 1$	23	5	13	2	
11	$r >= y$					13 > 5 (Да)
12	$r := r - y$	23	5	8	2	
13	$q := q + 1$	23	5	8	3	
14	$r >= y$					8 > 5 (Да)
15	$r := r - y$	23	5	3	3	
16	$q := q + 1$	23	5	3	4	
17	$r >= y$					3 < 5 (Нет)
18	Вывод r			3		
19	Вывод q				4	

www

Ознакомьтесь с модулем для коллективной работы «Циклические алгоритмы с предусловием» (http://school-collection.edu.ru/). Совместно с товарищами постарайтесь составить алгоритмы для имеющихся в модуле задач. Пройдите тестирование.

Цикл с заданным условием окончания работы (цикл-ДО, цикл с постусловием). Логика работы этой конструкции описывается схемой, показанной на рис. 3.11.

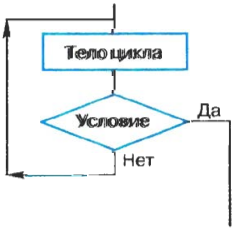


Рис. 3.11. Цикл с постусловием

На алгоритмическом языке эта конструкция записывается так:

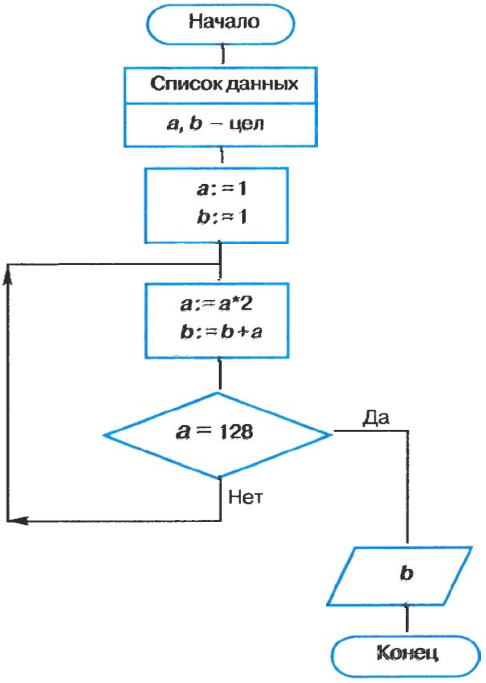
```
нц
  <тело_цикла (последовательность действий)>
кц при <условие>
```

Выполняется цикл-ДО следующим образом: 1) выполняется тело цикла; 2) проверяется условие (вычисляется значение логического выражения); если условие не удовлетворяется (Нет), то снова выполняется тело цикла и осуществляется переход к проверке условия; если же условие удовлетворяется, то выполнение цикла заканчивается. В любом случае тело цикла будет выполнено хотя бы один раз.

Пример 15. Алгоритм по выучиванию наизусть четверостишия.

```
алг четверостишие
нач
  нц
    прочитать четверостишие по книге 1 раз
    рассказать четверостишие
  кц при не сделал ошибку
кон
```

Пример 16. Вычислим значение переменной b согласно следующему алгоритму:



Составим таблицу значений переменных, задействованных в алгоритме:

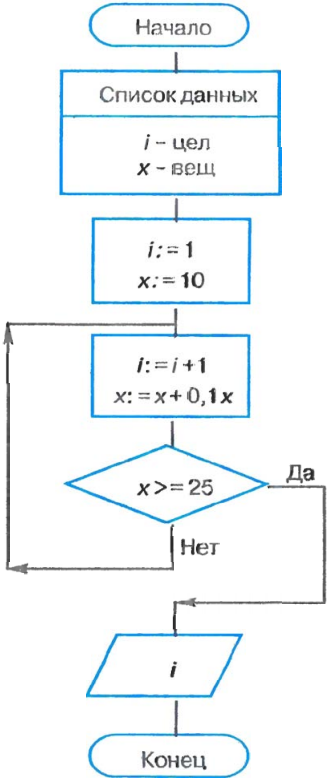
Шаг алгоритма	Операция	Переменные		Условие
		a	b	$a = 128$
1	$a := 1$	1	—	Нет
2	$b := 1$	1	1	
3	$a := a * 2$	2	1	
4	$b := b + a$	2	3	
5	$a = 128$			2 = 128 (Нет)
6	$a := a * 2$	4	3	
7	$b := b + a$	4	7	
8	$a = 128$			4 = 128 (Нет)
9	$a := a * 2$	8	7	
10	$b := b + a$	8	15	
11	$a = 128$			8 = 128 (Нет)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

Ответ: $b = 255$.

Пример 17. Спортсмен приступает к тренировкам по следующему графику: в первый день он должен пробежать 10 км; каждый следующий день следует увеличивать дистанцию на 10% от нормы преды-

дущего 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).



Ознакомьтесь с модулем для коллективной работы «Циклические алгоритмы с постусловием» (http://school-collection.edu.ru/). Совместно с товарищами постарайтесь составить алгоритмы для имеющихся в модуле задач. Пройдите тестирование.

Цикл с заданным числом повторений (цикл-ДЛЯ, цикл с параметром). Логика работы этой конструкции описывается схемой, показанной на рис. 3.12.

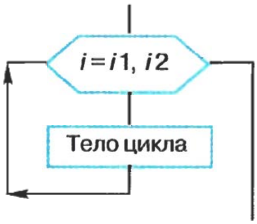


Рис. 3.12. Цикл с параметром

На алгоритмическом языке эта конструкция записывается так:

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

В цикле-ДЛЯ всегда есть параметр цикла — величина целого типа, изменяющаяся в ходе выполнения цикла от своего начального значения до конечного значения.

Выполняется цикл-ДЛЯ следующим образом: 1) параметру цикла присваивается начальное значение; 2) параметр цикла сравнивается с конечным значением; если параметр цикла не превышает конечное значение, то выполняется тело цикла, увеличивается значение параметра цикла и снова осуществляется проверка параметра цикла; если же параметр цикла превышает конечное значение, то выполнение цикла заканчивается.

В отличие от двух предыдущих конструкций (цикл-ПОКА, цикл-ДО) цикл-ДЛЯ имеет строго фиксированное число повторений, что позволяет избежать зацикливания, т. е. ситуации, когда тело цикла выполняется бесконечно.

Пример 18. 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 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
```

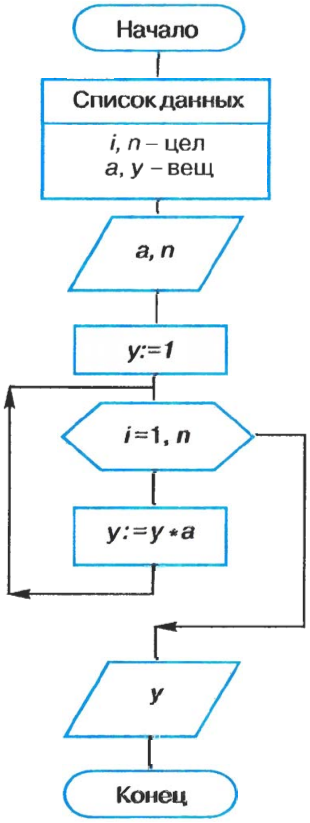
Пример 19. Составим алгоритм вычисления степени с натуральным показателем n для любого вещественного числа a.

По определению:

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

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 $n = 3$.

Шаги алгоритма	Операция	Переменные				Условие
		a	n	y	i	$i <= n$
1	Ввод a, n	4	3	–	–	
2	$y := 1$	4	3	1	–	
3	$i := 1$	4	3	1	1	
4	$i <= n$					1 <=3 (Да)
5	$y := y \cdot a$	4	3	4	1	
6	$i := i + 1$	4	3	4	2	
7	$i <= n$					2 <=3 (Да)
8	$y := y \cdot a$	4	3	16	2	
9	$i := i + 1$	4	3	16	3	
10	$i <= n$					3 <=3 (Да)
11	$y := y \cdot a$	4	3	64	3	
12	$i := i + 1$	4	3	64	4	
13	$i <= n$					4 <=3 (Да)

Пример 20. Для исполнителя Робот цикл с известным числом повторений реализуется с помощью следующей конструкции:

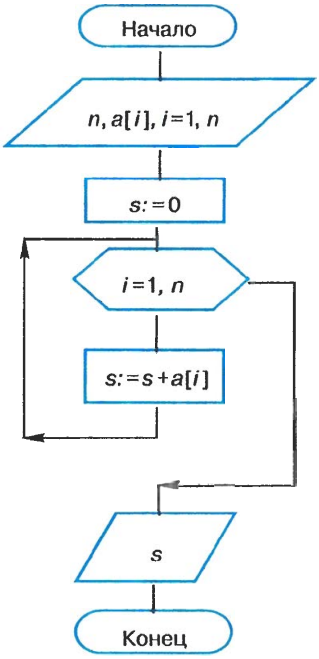
```
нц <число повторений> раз
  <тело цикла>
кц
```

Так, 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 21. В некотором населённом пункте N домов. Известно, сколько людей проживает в каждом из домов. Составим алгоритм подсчёта жителей населённого пункта.

Исходные данные (количество жильцов) представим с помощью линейной таблицы A, содержащей N элементов: $A[1]$ — количество жильцов дома 1, $A[2]$ — количество жильцов дома 2, ..., $A[N]$ — количество жильцов дома N. В общем случае $A[i]$ — количество жильцов дома i, где i принимает все значения от 1 до n ($i = \overline{1, n}$). Результат работы алгоритма обозначим через s.



Ознакомьтесь с модулем для коллективной работы «Циклические алгоритмы с параметром» (http://school-collection.edu.ru/). Совместно с товарищами постарайтесь составить алгоритмы для имеющихся в модуле задач. Пройдите тестирование.

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 ко-

торых используется только структура «следование», называются **линейными**.

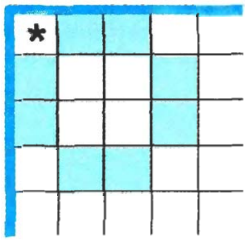
Ветвление — алгоритмическая конструкция, в которой в зависимости от результата проверки условия (да или нет) предусмотрен выбор одной из двух последовательностей действий (ветвей). Алгоритмы, в основе которых лежит структура «ветвление», называют разветвляющимися.

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 три типа циклов:

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

Вопросы и задания

1. Какие алгоритмы называются линейными?
2. Приведите пример линейного алгоритма из повседневной жизни.
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 фрагмента 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).

7. Известно, что 1 миля = 7 вёрст, 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
```

Чем является результат s этого алгоритма?

9. Определите значение целочисленных переменных x и y после выполнения фрагмента алгоритма.

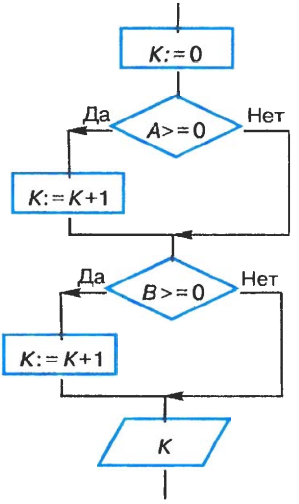
```
x:=336
y:=8
x:=x div y
y:=x mod y
```

10. Какие алгоритмы называют разветвляющимися?

11. Приведите пример разветвляющегося алгоритма из повседневной жизни.

12. Дополните алгоритм из примера 9 так, чтобы с его помощью можно было найти наибольшую из четырёх величин A, B, C и D.

13. Составьте алгоритм, с помощью которого можно определить, существует ли треугольник с длинами сторон a, b, c.
14. Составьте алгоритм, с помощью которого можно определить, является ли треугольник с заданными длинами сторон a, b, c равносторонним.
15. Составьте алгоритм возведения чётного числа в квадрат, а нечётного — в куб.
16. Какая задача решается с помощью следующего алгоритма?



17. Запишите алгоритм определения количества чётных чисел среди заданных целых чисел A, B и C.
18. Запишите алгоритм определения принадлежности точки X отрезку $[A; B]$ (пример 8) с использованием комбинации из двух ветвлений.
19. Запишите алгоритм правописания приставок, начинающихся с буквы «з» («с»).
20. Известно, что 31 января 2011 года приходится на понедельник. Какие значения должны быть присвоены литерной переменной y в алгоритме, определяющем день недели для любого числа (*chislo*) января 2011 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:='…'
```

21. Даны две точки на плоскости. Определите, какая из них находится ближе к началу координат.

22. Определите, есть ли среди цифр заданного целого трёхзначного числа одинаковые.

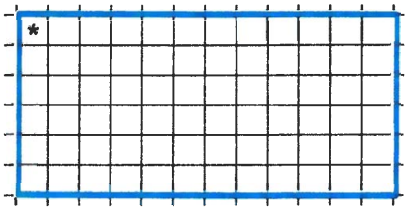
23. Приведите пример циклического алгоритма:

а) из повседневной жизни;

б) из литературного произведения;

в) из любой предметной области, изучаемой в школе.

24. Напишите алгоритм, под управлением которого Робот обойдёт прямоугольную область, обнесённую стеной, по периметру и закрасит угловые клетки. Размеры области неизвестны.



25. 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.

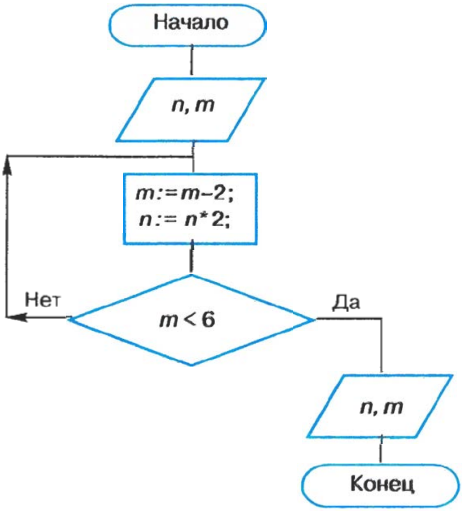
26. Дана последовательность 5, 9, 13, 17, … . Составьте блок-схему алгоритма для подсчёта числа слагаемых, сумма которых равна 324.

27. Составьте алгоритм для определения количества цифр в записи произвольного натурального числа.

28. Сумма 10 000 рублей положена в сберегательный банк, при этом прирост составляет 5% годовых. Составьте алгоритм, определяющий, через какой промежуток времени первоначальная сумма увеличится в два раза.

29. Одноклеточная амёба каждые три часа делится на 2 клетки. Составьте алгоритм вычисления времени, через которое будет X амёб.

30. Определите значения переменных n и m после выполнения фрагмента алгоритма.



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

32. Население города H увеличивается на 5% ежегодно. В текущем году оно составляет 40 000 человек. Составьте блок-схему алгоритма вычисления предполагаемой численности населения города через 3 года. Составьте таблицу значений переменных, задействованных в алгоритме.

33. 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.

34. Объявлен набор в школьную баскетбольную команду. Известен рост каждого из N учеников, желающих попасть в эту команду. Составьте алгоритм подсчёта количества претендентов, имеющих шанс попасть в команду, если рост игрока команды должен быть не менее 170 см.

§ 3.5
Конструирование алгоритмов

Ключевые слова:

- последовательное построение алгоритма
- вспомогательный алгоритм
- формальные параметры
- фактические параметры
- рекурсивный алгоритм

3.5.1. Последовательное построение алгоритма

Существуют различные методы конструирования (разработки, построения) алгоритмов. Мы познакомимся с одним из них — методом последовательного построения (уточнения) алгоритма. Иначе он называется **методом разработки «сверху вниз», нисходящим методом** или методом **пошаговой детализации.**

Процесс последовательного построения алгоритма выглядит следующим образом.

На первом шаге мы считаем, что перед нами совершенный исполнитель, который «всё знает и всё умеет». Поэтому достаточно определить исходные данные и результаты алгоритма, а сам алгоритм представить в виде единого предписания — постановки задачи (рис. 3.13).

Если исполнитель не обучен исполнять заданное предписание, то необходимо представить это предписание в виде совокупности более простых предписании (команд). Для этого:

- задачу разбивают на несколько частей, каждая из которых проще всей задачи;
- решение каждой части задачи формулируют в отдельной команде, которая также может выходить за рамки системы команд исполнителя;

Рис. 3.13. Линейный алгоритм, являющийся результатом первого этапа детализации задачи

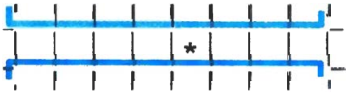
- при наличии в алгоритме предписаний, выходящих за пределы возможностей исполнителя, такие предписания вновь представляются в виде совокупности ещё более простых предписаний.

Процесс продолжается до тех пор, пока все предписания не будут понятны исполнителю.

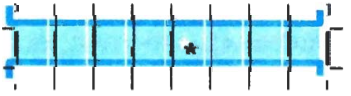
Объединяя полученные предписания в единую совокупность выполняемых в определённой последовательности команд, получаем требуемый алгоритм решения исходной задачи.

3.5.2. Разработка алгоритма методом последовательного уточнения для исполнителя Робот

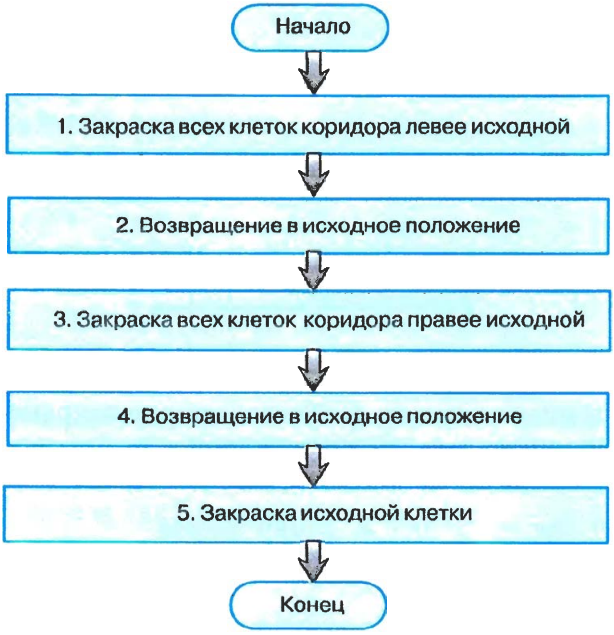
Известно, что Робот находится где-то в горизонтальном коридоре. Ни одна из клеток коридора не закрашена.



Составим алгоритм, под управлением которого Робот закрасит все клетки этого коридора и вернётся в исходное положение.



Представим план действий Робота следующими укрупнёнными шагами (модулями):

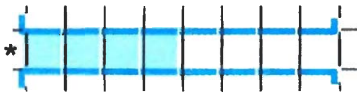


Детализируем каждый из пяти модулей.

1. Чтобы закрасить все клетки коридора, находящиеся левее Робота, прикажем Роботу шагнуть влево и выполнить цикл-ПОКА:

```
влево
нц пока сверху стена и снизу стена
  закрасить; влево
кц
```

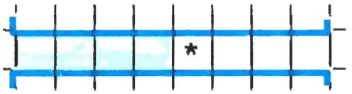
Под управлением этого алгоритма Робот закрасит все клетки коридора, находящиеся левее от него, и окажется на клетке рядом с левой границей коридора.



2. Командой вправо вернём Робота в коридор. Наша задача — вернуть Робота в исходную точку. Эта точка имеет единственный отличительный признак — она не закрашена. Поэтому пока занимаемая Роботом клетка оказывается закрашенной, будем перемещать его вправо.

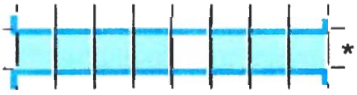
```
вправо
  нц пока клетка закрашена
    вправо
  кц
```

Под управлением этого алгоритма Робот окажется в исходной клетке.



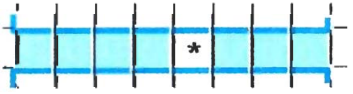
3. Выполнив команду вправо, Робот пройдёт исходную клетку и займёт клетку правее исходной. Теперь можно закрашивать клетки коридора, расположенные правее исходной.

```
вправо
  нц пока сверху стена и снизу стена
    закрасить; вправо
  кц
```



4. Так как, выполнив предыдущий алгоритм, Робот оказался правее коридора, командой влево вернём его в коридор. Возвращение в исходную точку обеспечивается алгоритмом:

```
влево
нц пока клетка закрашена
  влево
кц
```



5. По команде закрасить Робот закрашивает исходную точку.

Полностью программа управления Роботом выглядит так:

```
алг
нач
  влево
  нц пока сверху стена и снизу стена
```

```
    закрасить; влево
  кц
    вправо
  нц пока клетка закрашена
    вправо
  кц
    вправо
  нц пока сверху стена и снизу стена
    закрасить; вправо
  кц
    влево
  нц пока клетка закрашена
    влево
  кц
    закрасить
кон
```

3.5.3. Вспомогательные алгоритмы

При построении новых алгоритмов нередко возникают ситуации, когда в разных местах алгоритма необходимо выполнение одной и той же последовательности шагов обработки данных. Для такой последовательности шагов создают отдельный алгоритм, называемый вспомогательным. В качестве вспомогательных могут использоваться алгоритмы, ранее разработанные для решения других задач.

Вспомогательный алгоритм — алгоритм, целиком используемый в составе другого алгоритма.

При представлении алгоритмов с помощью блок-схем для обозначения команды вызова вспомогательного алгоритма используется блок «предопределённый процесс» (рис. 3.14), внутри которого записывается название (имя) вспомогательного алгоритма, после которого в скобках перечисляются параметры — входные данные и результаты.

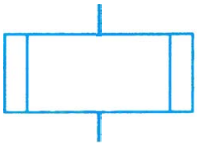


Рис. 3.14. Блок «предопределённый процесс»

Вспомогательный алгоритм делает структуру алгоритма более понятной.

Пример 1. Построим алгоритм вычисления степени $y = a^x$, где x — целое число, $a \neq 0$.

По определению степени с целым показателем:

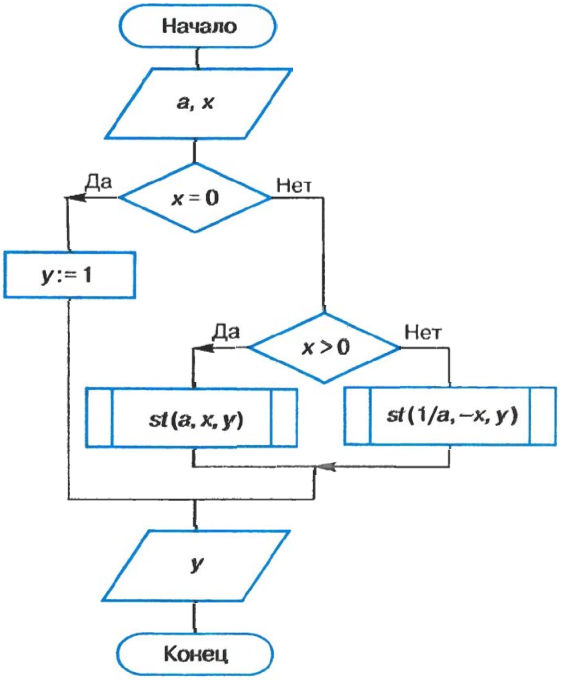
$a^0 = 1,\ a \neq 0;$

$a^{-n} = 1/a^n,\ a \neq 0,\ n \in N.$

Исходя из определения и учитывая, что $\frac{1}{a^{-x}} = \left(\frac{1}{a}\right)^{-x}$, можно записать:

$$y = \begin{cases} 1 \text{ при } x = 0, \\ a^x \text{ при } x > 0, \\ \left(\frac{1}{a}\right)^{-x} \text{ при } x < 0. \end{cases}$$

Ранее мы уже рассматривали алгоритм возведения произвольного вещественного числа в натуральную степень (§ 3.4, пример 19). Обозначим этот алгоритм $st(a, n, y)$ и воспользуемся им в качестве вспомогательного алгоритма. Решение задачи вычисления степени $y = a^x$, где x — целое число, $a \neq 0$, представим блок-схемой:



Этот алгоритм является основным по отношению к вызываемому в нем вспомогательному алгоритму.

Напомним, что параметрами используемого вспомогательного алгоритма были величины a, n, y. Это **формальные параметры**, они используются при описании алгоритма. При конкретном обращении к вспомогательному алгоритму формальные параметры заменяются **фактическими параметрами**, т. е. именно теми величинами, для которых будет исполнен вспомогательный алгоритм. Типы, количество и порядок следования формальных и фактических параметров должны совпадать.

Команда вызова вспомогательного алгоритма исполняется следующим образом (рис. 3.15):

1) формальные входные данные вспомогательного алгоритма заменяются значениями фактических входных данных, указанных в команде вызова вспомогательного алгоритма;
2) для заданных входных данных исполняются команды вспомогательного алгоритма;
3) полученные результаты присваиваются переменным с именами фактических результатов;
4) осуществляется переход к следующей команде основного алгоритма.

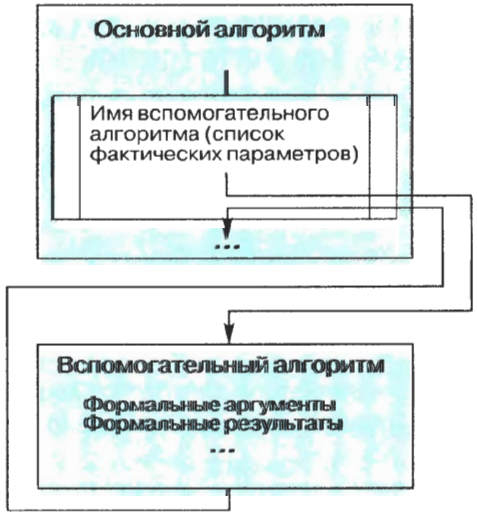
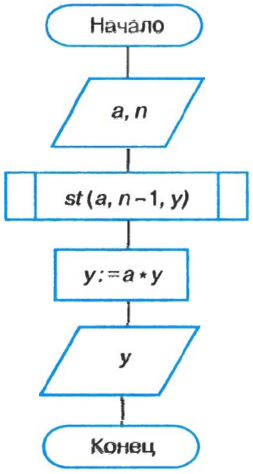


Рис. 3.15. Схема выполнения команды вызова вспомогательного алгоритма

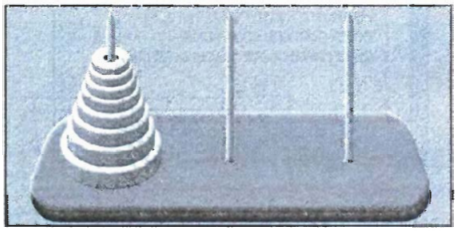
Алгоритм, в котором прямо или косвенно содержится ссылка на него же как на вспомогательный алгоритм, называют **рекурсивным**.

Рассмотрим несколько примеров рекурсивных алгоритмов.

Пример 2. Алгоритм вычисления степени с натуральным показателем n для любого вещественного числа a можно представить в виде рекурсивного.

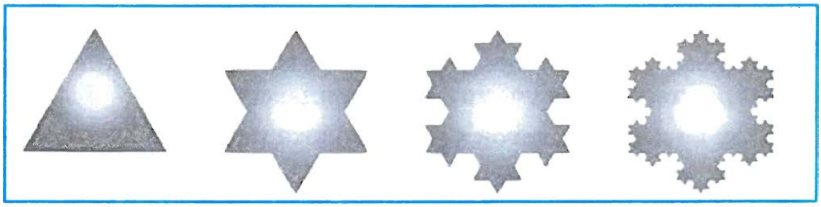


Пример 3. Рекурсивный алгоритм положен в основу эффективного решения головоломки «Ханойская башня».



Интерактивная игра «HanoiSetup» (http://files.school-collection. edu.ru/) поможет вам вспомнить условие и алгоритм решения головоломки.

Пример 4. Рассмотрим алгоритм построения геометрической фигуры, которая называется снежинкой Коха. Шаг процедуры построения состоит в замене средней трети каждого из имеющихся отрезков двумя новыми той же длины, как показано на рисунке.



Начальное состояние Первый шаг Второй шаг Третий шаг

С каждым шагом фигура становится всё причудливее. Граница снежинки Коха — положение кривой после выполнения бесконечного числа шагов.

Попробуйте подсчитать, сколько рёбер в границе снежинки Коха после четвёртого шага; после пятого шага.

САМОЕ ГЛАВНОЕ

Один из основных методов конструирования алгоритмов — **метод последовательного построения алгоритма**. Его суть состоит в том, что: исходная задача разбивается на несколько частей, каждая из которых проще всей задачи, и решение каждой части формулируется в отдельной команде; если получаются команды, выходящие за пределы возможностей исполнителя, то они представляются в виде совокупности ещё более простых предписаний. Процесс продолжается до тех пор, пока все предписания не будут понятны исполнителю.

Вспомогательный алгоритм — алгоритм, целиком используемый в составе другого алгоритма.

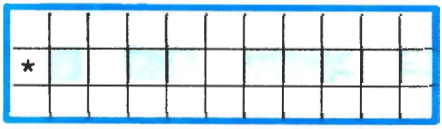
Алгоритм, в котором прямо или косвенно содержится ссылка на него же как на вспомогательный алгоритм, называют **рекурсивным**.

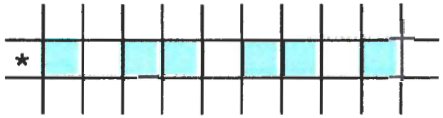
Вопросы и задания

1. Почему при решении сложной задачи затруднительно сразу конкретизировать все необходимые действия?
2. В чём заключается метод последовательного уточнения при построении алгоритма?
3. Какая связь между методом последовательного построения алгоритма и такими процессами, как написание сочинения или подготовка к многодневному туристическому походу?

4. Известен рост каждого из N учеников 9А класса и M учеников 9Б класса. Опишите укрупнёнными блоками алгоритм сравнения среднего роста учеников этих классов.

5. В ряду из десяти клеток правее Робота некоторые клетки закрашены. Последняя закрашенная клетка может примыкать к стене. Составьте алгоритм, который закрашивает клетки выше и ниже каждой закрашенной клетки. Проверьте работу алгоритма в следующих случаях:





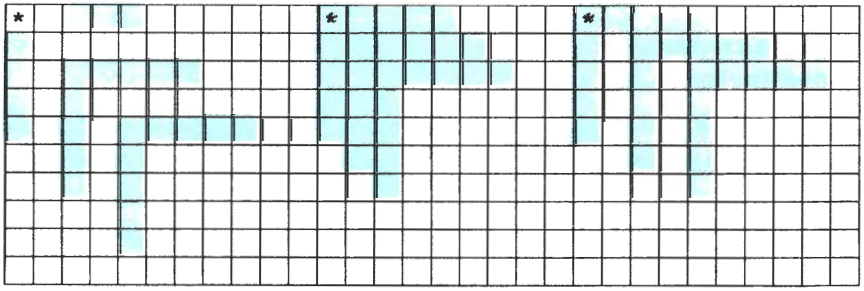
6. Для чего нужны вспомогательные алгоритмы?

7. Опишите процесс выполнения команды вызова вспомогательного алгоритма в основном алгоритме.

8. Сталкивались ли вы с идеей формальных и фактических параметров при изучении математики и физики? Приведите пример.

9. Какие алгоритмы называют рекурсивными? Приведите пример рекурсии из жизни.

10. Составьте алгоритмы, под управлением которых Робот закрасит указанные клетки.



а) *б)* *в)*

а *б* *в*

§ 3.6
Алгоритмы управления

Ключевые слова:
- управление
- алгоритм управления
- обратная связь

3.6.1. Управление

Управление — это процесс целенаправленного воздействия на объект; осуществляется для организации функционирования объекта по заданной программе.

В середине прошлого века выдающийся американский учёный Норберт Винер (1894—1964), изучавший различные технические и биологические системы, установил, что управление в них осуществляется по общей схеме. Винер считается основоположником науки об управлении — **кибернетики**.

Управляемым объектом (объектом управления) может быть техническое устройство (например, автомобиль), один человек (например, ученик, солдат) или коллектив (например, оркестр, работники предприятия).

Управляющим объектом (управляющей системой) может быть человек (например, шофёр, дирижёр оркестра, учитель, директор), коллектив (например, правительство, парламент), а может быть и техническое устройство (например, автоматический регулятор, компьютер).

Последовательность команд по управлению объектом, приводящая к заранее поставленной цели, называется **алгоритмом управления**.

Простейшие алгоритмы управления могут состоять из одной команды или представлять собой линейную последовательность команд. Более сложные алгоритмы управления содержат ветвления и циклы.

3.6.2. Обратная связь

Для управления нужна информация. Во-первых, управляющий объект должен получить информацию о том, что ему нужно, т. е. он должен знать цель своих действий. Во-вторых, управляющий объект должен знать, как можно достичь поставленной цели. Важно, что информация о цели и способах её достижения должна быть известна управляющему объекту до начала процесса управления.

Пример 1. Управление движением автомашин (объект управления) на перекрёстке с помощью светофора (управляющий объект). В этой ситуации управляющее воздействие формируется в зависимости от заложенной в управляющем объекте исходной информации. Светофор не воспринимает текущую информацию о состоянии движения на перекрёстке, он не изменяет алгоритм управления от того, что с какой-то стороны скопилось очень много машин и образовалась «пробка».

Обратная связь — это процесс передачи информации о состоянии объекта управления в управляющую систему.

Обратная связь позволяет корректировать управляющие воздействия управляющей системы на объект управления в зависимости от состояния объекта управления. Обратная связь предусмотрена в ряде бытовых приборов (например, утюг с терморегулятором, холодильник, кастрюля-скороварка), в живых организмах, в обществе.

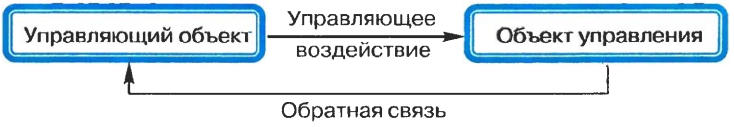


Рис. 3.16. Кибернетическая модель управления

В настоящее время очень часто роль управляющей системы отводится компьютеру, в память которого заложена программа управления, предусматривающая все варианты информации, которые могут быть получены по обратной связи.

Пример 2. Если вместо обычного светофора на дорожном перекрёстке будет установлен «интеллектуальный» светофор — высокотехнологичное устройство, оснащённое датчиками, фиксирующими скорость движения на дороге и плотность транспортных потоков, то управление движением станет более рациональным за счёт учёта информации, поступающей от объекта управления (рис. 3.16).

САМОЕ ГЛАВНОЕ

Управление — процесс целенаправленного воздействия на объект; осуществляется для организации функционирования объекта по заданной программе.

Последовательность команд по управлению объектом, приводящая к заранее поставленной цели, называется **алгоритмом управления**.

Вопросы и задания

1. Что такое управление? Приведите примеры управляющих систем и управляемых ими объектов.
2. Что такое алгоритм управления? Приведите примеры ситуаций, в которых имеют место линейные, разветвляющиеся и циклические алгоритмы.
3. Что изучает наука кибернетика?
4. Какая информация нужна для управления? Приведите пример.
5. Что такое обратная связь?
6. Опишите кибернетическую модель управления.

Тестовые задания для самоконтроля

1. Алгоритмом можно считать:
   а) описание 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 Черепашке 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 прошел из точки *А* в точку *Б*. Какую программу необходимо выполнить, чтобы вернуться из точки *Б* в точку *А* по кратчайшему пути и не подвергнуться 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.

   Ответ: _____

10. Некоторый алгоритм строит цепочки символов следующим образом:

    - первая цепочка состоит из одного символа — цифры 1;
    - 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. Наибольшей наглядностью обладают следующие формы записи алгоритмов:

    а) словесные

    б) рекурсивные

    в) графические

    г) построчные

12. Величины, значения которых меняются в процессе исполнения алгоритма, называются:

    а) постоянными

    б) константами

    в) переменными

    г) табличными

13. Величиной целого типа является

    а) количество мест в зрительном зале

б) рост человека

в) марка автомобиля

г) площадь государства

14. Какое логическое выражение истинно, если $x \in [-10, 10]$?

а) `(x > 10) И (x < -10)`

б) `(x > 10) ИЛИ (x < -10)`

в) `(x < 10) ИЛИ (x >= -10)`

г) `(x >= -10) И (x <= 10)`

15. 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`

16. 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`

17. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?

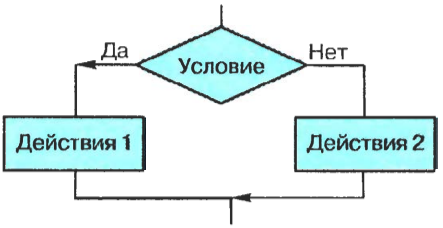
а) линейный

б) разветвляющийся

в) циклический

г) вспомогательный

18. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?



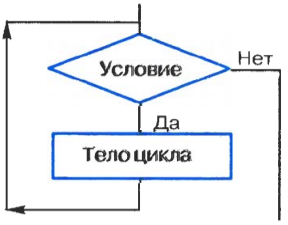
а) линейный

б) разветвляющийся с неполным ветвлением

в) разветвляющийся с полным ветвлением

г) циклический

19. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?



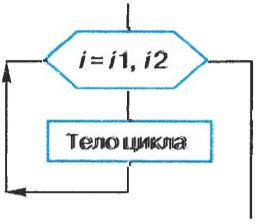
а) цикл с параметром

б) цикл с заданным условием продолжения работы

в) цикл с заданным условием окончания работы

г) цикл с заданным числом повторений

20. К какому виду алгоритмов можно отнести алгоритм, схема которого представлена ниже?



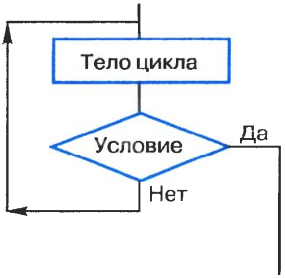
а) цикл с заданным условием продолжения работы

б) цикл с заданным условием окончания работы

в) цикл с постусловием

г) цикл с 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 заданным числом повторений

г) цикл с предусловием

22. 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 другой берег за минимально возможное время. Какое время она затратили на его исполнение?

а) 10 минут

б) 11 минут

в) 12 минут

г) 13 минут

23. Дан фрагмент линейного алгоритма.

```
a:=8
b:=6+3*a
a:=b/3*a
```

Чему равно значение переменной a после его исполнения?

Ответ: _____

24. Исполните следующий фрагмент линейного алгоритм для $a = x$ и $b = y$.

```
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

25. Определите значение целочисленных переменных x и y после выполнения фрагмента программы.

```
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) $x = 5$, $y = 10$

26. 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?

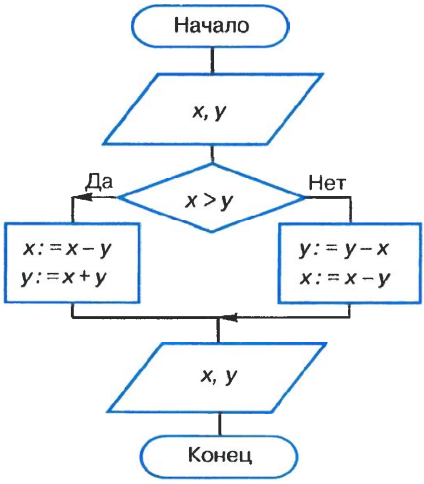
а) 2

б) 3

в) 4

г) 5

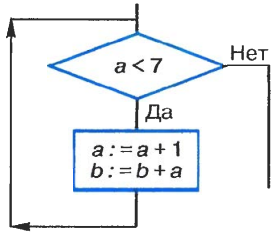
27. Исполните алгоритм при $x = 10$ и $y = 15$.



Какие значения будут получены в результате его работы?

а) –5, 10
б) 5, 20
в) 10, 15
г) 5, 5
д) –5, 5

28. Исполните фрагмент алгоритма при $a = 2$ и $b = 0$.



Определите значение переменной b после выполнения фрагмента алгоритма.

Ответ: _____

29. Определите значение переменной f после выполнения фрагмента алгоритма.

```
f:=1
нц для i от 1 до 5
  f:=f*i
кц
```

Ответ: _____

30. Определите значение переменной *s* после выполнения фрагмента алгоритма.

```
s:=0
нц для i от 1 до 5
  s:=s+i*i
кц
```

Ответ: _____

Глава 4
НАЧАЛА ПРОГРАММИРОВАНИЯ

§ 4.1
Общие сведения о языке программирования Паскаль

Ключевые слова:
- язык программирования
- программа
- 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 годах прошлого века Никлаусом Виртом (Швейцария). Своё название этот язык получил в честь французского ученого Блеза Паскаля, известного не только своими достижениями в математике, физике и философии, но и созданием первой в мире механической машины, выполнявшей сложение двух 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]

Никлаус Вирт (род. в 1934 г.) — швейцарский учёный, специалист в области информатики, один из известнейших теоретиков в области разработки языков программирования, профессор информатики (компьютерных наук). Разработчик языка Паскаль и ряда других языков программирования.

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 ≤ и ≥);
(* и *) (начало и конец 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. 4.1 приведены основные служебные слова, которые мы будем использовать при записи программ на языке Паскаль.

[1] С другими стилями программирования вы познакомитесь при изучении курса информатики в 10–11 классах.

Таблица 4.1

Служебные слова языка Паскаль

Служебное слово языка Паскаль	Значение служебного слова
and	и
array	массив
begin	начало
do	выполнить
else	иначе
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 подчеркивания, начинающиеся с буквы или символа подчеркивания.

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, используемых в языке Паскаль

В языке Паскаль используются различные **типы данных**. Мы будем пользоваться некоторыми из так называемых простых типов данных:

Название	Обозначение	Допустимые значения	Область памяти
Целочисленный	integer[1]	−32 768 .. 32 767	2 байта со знаком
Вещественный	real	$\pm(2{,}9 \cdot 10^{-39} .. 1.7 \cdot 10^{+38})$	6 байтов
Символьный	char	Произвольный символ алфавита	1 байт
Строковый	string	Последовательность символов длиной меньше 255	1 байт на символ
Логический	boolean	True и False	1 байт

В вещественном числе целая часть от дробной отделяется точкой, при этом перед точкой и после неё должно быть, по крайней мере, по одной цифре. Пробелы внутри числа недопустимы.

4.1.3. Структура программы на языке Паскаль

В программе, записанной на языке Паскаль, можно выделить:

1) заголовок программы;
2) блок описания используемых данных;
3) блок описания действий по преобразованию данных (программный блок).

Заголовок программы состоит из служебного слова **program** и имени программы. После имени программы ставится точка с запятой.

Блок описания данных состоит из раздела описания констант (**const**), раздела описания переменных (**var**) и некоторых других разделов[2]. В разделе описания переменных указываются имена используемых в программе переменных и их тип. 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; x: real; a: char;
```

Целый тип — Вещественный тип — Символьный тип

[1] integer — основной, но не единственный тип для работы с целочисленными данными. Дополнительную информацию по этому вопросу вы можете найти в справочниках по программированию на языке Паскаль.

[2] В 9 классе мы ограничимся рассмотрением разделов описания констант и переменных, оставив рассмотрение других разделов для старшей школы.

Программа может не иметь заголовка; в ней может отсутствовать блок 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>;
  const <список постоянных значений>;
  var <описание используемых переменных>;
begin <начало программного блока>
  <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 алгоритмического языка. Общий вид оператора:

```
<имя переменной>:=<выражение>
```

Операция присваивания допустима для всех приведённых в табл. 4.1 типов данных. Выражения в языке Паскаль конструируются по рассмотренным ранее правилам для 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* заносится значение 10; при выполнении опе-

ратора 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. 4.1).

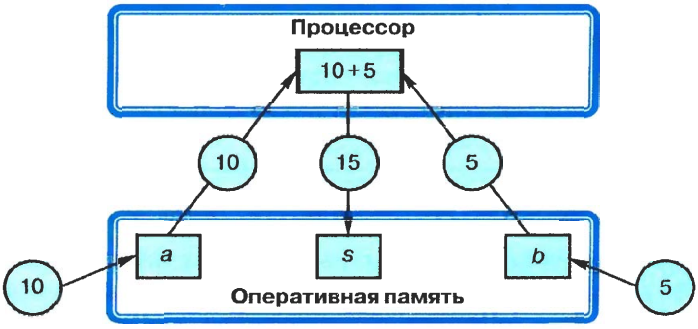


Рис. 4.1. Процесс выполнения оператора присваивания

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) описание используемых данных;
3) описание действий по преобразованию данных (программный блок).

Общий вид программы:

```
program <имя программы>;
  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?
2. Почему язык программирования Паскаль считается универсальным?
3. Что входит в состав алфавита Паскаля?
4. Каких требований следует придерживаться при выборе имён для различных объектов в языке Паскаль?
5. Указывая название, обозначение, диапазон и занимаемую область памяти, 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 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 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 `writer`
- формат вывода
- оператор ввода `read`

4.2.1. Вывод данных

В предыдущем параграфе мы познакомились со структурой программы на языке Паскаль, научились описывать данные, 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 монитора используется оператор вывода `write`:

```
write <выражение1>,<выражение2>,..., <выражениеN>
      \_____________список вывода_____________/
```

Здесь в круглых скобках помещается список вывода — список выражений, значения которых выводятся на печать. Это могут быть числовые, символьные и логические выражения, в том числе переменные и константы.

Произвольный набор символов, заключенный в апострофы, считается строковой константой. Строковая константа может содержать любые символы, набираемые на клавиатуре.

Пример. Оператор `write ('s=', s)` выполняется так:
1) на экран выводятся символы, заключенные в апострофы: `s=`
2) на экран выводится значение переменной, хранящееся в ячейке оперативной памяти с именем *s*.

Если значение переменной *s* равно 15 и она имеет целочисленный тип, то на экране появится: `s=15`.

Если значение переменной *s* равно 15, но она имеет вещественный тип, то на экране появится: `s=1.5E+01`.

При выполнении оператора вывода все элементы списка вывода печатаются 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, ' ', 2, ' ', 3)`	`1 20 300`
Указать формат вывода	`write (1:3, 20:4, 3:5)`	`  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.

Для вывода вещественного числа в формате с фиксированной запятой в списке вывода для каждого выражения указывается два параметра: 1) общее количество позиций, отводимых под число; 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:5:1);`	`s =  15.0`

При выполнение нового оператора `write` вывод продолжается в той же строке. Чтобы осуществить переход к новой строке, использу-

ется оператор `writeln`. 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 радиуса 5,4 см.

Исходными данными 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:

```
program n_1;
  const pi=3.14;
  var r, c, s: real;
begin
  r:=5.4;
  c:=2*pi*r;
  s:=pi*r*r;
  writeln ('c=', c:6:4);
  writeln ('s=', s:6:4)
end.
```

Эта программа верна и решает поставленную задачу. Запустив её на выполнение, вы получите следующий 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 в оперативную память значений переменных 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; a: char;
```

Присвоим переменным *i*, *j*, *x*, *a* значения 1, 0, 2.5 и 'A'. Для этого воспользуемся оператором

```
read (i, j, x, a)
```

и организуем входной поток одним из следующих способов:

```
1 0 2.5 A<Enter>      1,0<Enter>          1<Enter>
                      2.5,A<Enter>        0<Enter>
                                          2.5<Enter>
                                          A<Enter>
```

Здесь мы не только использовали различные разделители (пробел, запятая), но и входной поток представляли в виде одной, двух и четырёх строк.

Для ввода данных с клавиатуры можно также использовать оператор `readln`, который отличается от оператора `read` только тем, что после его выполнения курсор переходит на новую строку.

[1] Нажатием клавиши Enter может сопровождаться ввод каждого значения.

Усовершенствуем программу n_1, 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**:

```
program n_1;
 const pi=3.14;
 var r, c, s: real;
begin
  writeln('Вычисление длины окружности и площади круга');
  write('Введите  r>>');
  readln(r);
  c:=2*pi*r;
  s:=pi*r*r;
  writeln ('c=', c:6:4);
  writeln ('s=', s:6:4)
end.
```

Результат работы усовершенствованной программы:

Теперь наша программа может вычислить 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

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. Запишите оператор, обеспечивающий во время работы программы ввод значения переменной *summa*.
2. 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
3. Опишите переменные, необходимые для вычисления площади треугольника по его трём сторонам, и запишите оператор, обеспечивающий ввод необходимых исходных данных.
4. Что является результатом выполнения оператора?

   а) `write (a)`

   б) `write ('a')`

   в) `write ('a=', a)`
5. Какой тип имеет переменная *f*, если после выполнения оператора `write (f)` на экран было выведено следующее число?

   а) `125`

   б) `1.25E+2`
6. Каким образом можно вывести на экран вещественное число в формате с фиксированной запятой?
7. Запишите операторы ввода двух чисел и вывода их в обратном порядке.
8. Дан фрагмент программы:

   ```
   read (a); read (b); c:=a+b; write (a, b); write (c)
   ```

   Упростите его, сократив число операторов ввода и вывода.
9. Дан фрагмент программы:

   ```
   a:=10; b:=a+1: a:=b-a; write (a, b)
   ```

   Какие числа будут выведены на экран компьютера?
10. Напишите программу, которая вычисляет площадь и периметр прямоугольника по двум его сторонам.

§ 4.3

Программирование как этап решения задачи на компьютере

Ключевые слова:

- постановка задачи
- формализация
- алгоритмизация
- программирование
- отладка и тестирование

Компьютерные программы программисты создают для решения разнообразных задач. Программирование — важный, но не единственный этап решения задачи на компьютере. Чтобы решать задачи на компьютере, необходимо владеть языком программирования, обладать знаниями в области информационного моделирования и алгоритмизации.

4.3.1. Этапы решения задачи на компьютере

Решение задачи с использованием компьютера включает в себя этапы, показанные на рис. 4.2.

На *первом этапе* обычно строится словесная информационная модель объекта или процесса. При этом должно быть четко определено, что дано (какие исходные данные известны, какие данные допустимы) и что требуется найти в решаемой задаче. Также должны быть четко выделены наиболее существенные свойства рассматриваемого объекта, указаны связи между исходными данными и результатами.

На *втором этапе* описательная информационная модель формализуется, т. е. записывается с помощью некоторого формального языка. Для этого требуется:

- понять, к какому классу принадлежит рассматриваемая задача;

Рис. 4.2. Этапы решения задачи на компьютере

- записать известные связи между исходными данными и результатами с помощью математических соотношений;
- выбрать наиболее подходящий способ для решения задачи.

На *третьем этапе* осуществляется построение алгоритма — чёткой инструкции, задающей необходимую последовательность действий для решения задачи. Алгоритм чаще всего представляется в форме блок-схемы, ввиду её наглядности и универсальности.

На *четвёртом этапе* алгоритм записывается на одном из языков программирования. Вы учитесь записывать программы на языке Паскаль.

На *пятом этапе* осуществляется отладка и тестирование программы. Этап отладки и тестирования также называют компьютерным экспериментом.

Отладка программы — это процесс проверки работоспособности программы и исправления обнаруженных при этом ошибок. Ошибки могут быть связаны с нарушением правил записи программы на конкретном языке программирования. Их программисту помогает выявить используемая система программирования; она выдаёт на экран сообщения о выявленных ошибках.

Проверка правильности разработанной программы осуществляется с помощью тестов. Тест — это конкретный вариант значений исходных данных, для которого известен ожидаемый результат.

О правильности разработанной программы свидетельствует также соответствие полученных данных экспериментальным фактам, теоретическим положениям и т. д. При этом может возникнуть необходимость уточнить разработанную математическую модель, полнее учесть особенности изучаемого объекта или процесса. По уточнённой математической модели снова составляется программа, анализируются результаты её выполнения. Так продолжается до тех пор, пока полученные результаты не будут достаточно точно соответствовать изучаемому объекту.

4.3.2. Задача о пути торможения автомобиля

Рассмотрим последовательность прохождения этапов решения задачи на компьютере (см. рис. 4.2) на примере простой задачи.

Водитель автомобиля, движущегося с некоторой постоянной скоростью, увидев красный свет светофора, нажал на тормоз. После этого скорость автомобиля стала уменьшаться каждую секунду на 5 метров. Требуется найти расстояние, которое автомобиль пройдёт до полной остановки.

Первый этап. Дано:

v_{0x} — начальная скорость;

v_x — конечная скорость (равна нулю, так как автомобиль остановился);

a_x — ускорение (равно –5 м/с).

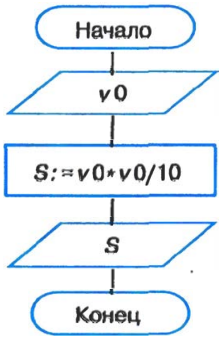
Требуется найти: s_x — расстояние, которое автомобиль пройдёт до полной остановки.

Второй этап. В данной ситуации мы имеем дело с прямолинейным равноускоренным движением тела. Формула для перемещения при этом имеет вид:

$$s_x = \frac{v_{0x}(v_x - v_{0x})}{a_x} + \frac{a_x}{2}\left(\frac{v_x - v_{0x}}{a_x}\right)^2.$$

Упростим эту формулу с учётом того, что конечная скорость равна нулю: $s_x = -\frac{v_{0x}^2}{2a_x}$. При $a_x = -5$ м/с получим: $s_x = \frac{v_{0x}^2}{10}$.

Третий этап. Представим алгоритм решения задачи в виде блок-схемы:



Четвёртый этап. Запишем данный алгоритм на языке программирования Паскаль:

```
program n_2;
  var v0, s: real;
begin
  writeln('Вычисление длины пути торможения автомобиля');
  write('Введите начальную скорость (м/с)> ');
  readln (v0);
  s:=v0*v0/10;
  writeln ('До полной остановки автомобиль пройдет ',
           s:8:4, ' м.')
end.
```

Пятый этап. Протестировать составленную программу можно, используя ту информацию, что при скорости 72 км/ч с начала торможения до полной остановки автомобиль проходит 40 метров.

Выполнив программу несколько раз при различных исходных данных, можно сделать вывод: чем больше начальная скорость автомобиля, тем большее расстояние он пройдет с начала торможения до полной остановки.

Применяя компьютер для решения задач, всегда следует помнить, что наряду с огромным быстродействием и абсолютной исполнительностью у компьютера отсутствуют интуиция и чувство здравого смысла, и он способен решать только ту задачу, программу решения которой ему подготовил человек.

САМОЕ ГЛАВНОЕ

Этапы решения задачи с использованием компьютера:
1) постановка задачи;
2) формализация;
3) алгоритмизация;
4) программирование;
5) компьютерный эксперимент.

Для решения задач на компьютере необходимо владеть языком программирования, обладать знаниями в области информационного моделирования и алгоритмизации.

Вопросы и задания

1. Перечислите основные этапы решения задачи с использованием компьютера.
2. Что происходит на этапе постановки задачи? Что является результатом этого этапа?
3. Что происходит на этапе формализации? Что является результатом этого этапа?
4. Что происходит на этапе алгоритмизации? Что является результатом этого этапа?
5. Что происходит на этапе программирования? Что является результатом этого этапа?
6. Что происходит на этапе компьютерного эксперимента? Что является результатом этого этапа?
7. Какой этап, по вашему мнению, является наиболее трудоёмким?
8. Как вы считаете, по силам ли одному специалисту реализация всех этапов решения сложной практической задачи? Обоснуйте свою точку зрения.
9. Как правило, сложные практические задачи решаются большими коллективами разработчиков. Отдельные группы в этих коллективах специализируются на выполнении одного или нескольких этапов решения задачи. Нужно ли в таком случае им иметь представление обо всех этапах решения задачи с использованием компьютера? Обоснуйте свою точку зрения.

10. Может ли пригодиться в жизни представление об этапах решения задачи с использованием компьютера? Обоснуйте свою точку зрения.

11. Уличный продавец газет получает a рублей с продажи каждой из первых 50 газет. С продажи каждой из остальных газет он получает на 20% больше.

Разработайте программу, которая вычислит заработок продавца, если он продаст за день 200 газет. Зафиксируйте свои действия на каждом из этапов решения этой задачи.

12. В аэробусе, вмещающем 160 пассажиров, три четверти мест находятся в салонах экономического класса и одна четверть мест — в салоне бизнес-класса. Стоимость билета в салоне бизнес класса составляет x рублей, что в два раза выше стоимости билета в салонах экономического класса.

Разработайте программу, которая вычислит сумму денег, полученную авиакомпанией от продажи билетов на этот рейс, если известно, что остались нераспроданными a билетов бизнес-класса и b билетов экономического класса. Выделите все этапы решения этой задачи и опишите свои действия на каждом из них.

§ 4.4
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.

4.4.1. Числовые типы данных

Вы уже знакомы с основными числовыми типами данных `integer` и `real`. К ним применимы стандартные функции, часть из которых приведена в табл. 4.2.

Таблица 4.2

Стандартные функции Паскаля

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

Функция	Назначение	Тип аргумента	Тип результата
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

Исследуем работу функций round, int и frac, применив их к некоторому вещественному x. Соответствующая программа будет иметь вид:

```
program n_3;
  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 $x \in \{10{,}2;\ 10{,}8;\ -10{,}2;\ -10{,}8\}$. Что вы можете сказать о типе результата каждой из этих функций?

4.4.2. Целочисленный тип данных

Над целыми числами в языке Паскаль выполняются следующие операции: сложение (+), вычитание (–), умножение (*), получение целого частного (div), получение целого остатка деления (mod) и деление (/). Результаты первых пяти операций — целые числа. Результатом операции деления может быть вещественное число.

Рассмотрим пример использования операций div и mod, записав на языке Паскаль программу нахождения суммы цифр вводимого с клавиатуры целого трёхзначного числа.

Используем тот факт, что трёхзначное число можно представить в виде следующей суммы: $x = a \cdot 100 + b \cdot 10 + c$, где a, b, c — цифры числа.

```
program n_4;
  var x, a, b, c, s: integer;
begin
  writeln ('Нахождение суммы цифр трёхзначного числа');
  write ('Введите исходное число>>');
  readln (x);
  a:=x div 100;
  b:=x mod 100 div 10;
  c:=x mod 10;
  s:=a+b+c;
  writeln ('s= ', s)
end.
```

4.4.3. Символьный и строковый типы данных

Значением символьной величины (тип `char`) в языке Паскаль является любой из символов, который можно получить на экране нажатием одной из клавиш или комбинации клавиш, а также некоторых других символов, в том числе и невидимых. Множество таких символов состоит из 256 элементов, каждому из которых в соответствии с используемой кодовой таблицей поставлен в соответствие код — число 0 до 255.

Символы, соответствующие первым 32 кодам, являются управляющими, а остальные — изображаемыми. К изображаемым символам относится и пробел, имеющий код 32.

Знакам препинания, знакам арифметических операций, цифрам, прописным и строчным латинским буквам соответствуют коды от 33 до 127. Буквам национального алфавита соответствуют коды с номерами 128 и далее.

В тексте программы константу символьного типа можно задать, заключив любой изображаемый символ в апострофы: `'5'`, `'B'`, `'*'`.

Если значение символьной переменной считывается с клавиатуры, то его следует набирать без апострофов.

Чтобы найти код символа, используют функцию `ord`, где в качестве параметра задают символ.

Чтобы по коду узнать символ, используют функцию `chr`, где в качестве параметра указывают код символа.

Значением строковой величины (тип `string`) является произвольная последовательность символов, заключенная в апострофы. В Паскале (как и в алгоритмическом языке) строки можно сцеплять.

Пример. Запишем на языке Паскаль 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.

```
program n_5;
  var a: char; kod: integer; b: string;
begin
  writeln ('Код и строка');
  write ('Введите исходную букву>>');
  readln (a);
  kod:=ord(a);
  b:=chr(kod-1)+a+chr(kod+1);
  writeln ('Код буквы ', a, '-', kod);
  writeln ('Строка: ', b)
end.
```

4.4.4. 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. 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.

```
program n_6;
  var n: integer; ans: boolean;
begin
  writeln ('Определение истинности высказывания
            о чётности числа');
  write ('Введите исходное число>>');
  readln (n);
```

```
  ans:=n mod 2=0;
  writeln ('Число ', n,' является четным - ', ans)
end.
```

Логическим переменным можно присваивать значения логических выражений, построенных с помощью известных вам логических функций **и**, **или**, **не**, которые в Паскале обозначаются соответственно `and`, `or`, `not`.

Пример. Напишем программу, определяющую истинность высказывания «Треугольник с длинами сторон a, b, c является равнобедренным» для произвольных целых чисел a, b, c.

```
program n_7;
  var a, b, c: integer; ans: boolean;
begin
  writeln ('Определение истинности высказывания
           о равнобедренном треугольнике');
  write ('Введите значения a, b, c>>');
  readln (a, b, c);
  ans:=(a=b) or (a=c) or (b=c);
  writeln ('Треугольник с длинами сторон ', a, ',',
           b, ',', c, ' является равнобедренным - ', ans)
end.
```

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. Для заданного x вычислите y 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: $y = ((x + 2{,}5)x - 1)x + 1$.

2. По заданным координатам точек A и B вычислите длину отрезка AB.

Пример входных данных	Пример выходных данных
xa=1 ya=2 xb=10 yb=7	|AB|=10.0

3. Известны длины сторон треугольника a, b, c. Напишите программу, вычисляющую площадь этого треугольника.

Пример входных данных	Пример выходных данных
a=3 b=4 c=5	S=6.0

4. Известны координаты вершин A, B, C треугольника. Напишите программу, вычисляющую площадь этого треугольника.

Пример входных данных	Пример выходных данных
xa=2 ya=1 xb=6 yb=5 xc=10 yc=1	S=16.0

5. 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, сколько следует уплатить.

6. Исследуйте работу функции random, запустив многократно на выполнение программу:

```
program n_8;
  var x, n: integer;
begin
  writeln ('Исследование функции random');
  randomize (*для генерации различных случайных чисел
              при каждом запуске программы  *);
  write ('Введите x>>');
  readln (x);
  write ('Введите n>>');
  readln (n);
  writeln ('random(', x, ')=', random(x));
  writeln ('random(', x, ')+', n, '=', random(x)+n)
end.
```

Как можно получить случайное число из промежутка $(0; x)$?
Как можно получить случайное число из промежутка $(0; x]$?
Как можно получить случайное число из промежутка $(n; x + n)$?

7. Одна компания выпустила лотерейные билеты трёх разрядов: для молодежи, для взрослых и для стариков. Номера билетов каждого разряда лежат в пределах:
для молодёжи — от 1 до 100;
для взрослых — от 101 до 200;
для стариков — от 201 до 250.
С помощью компьютера выберите случайным образом лотерейный билет в каждом разряде.

8. Запишите на языке Паскаль программу, которая для произвольного двузначного числа определяет:
а) сумму и произведение его цифр;
б) число, образованное перестановкой цифр исходного числа.

9. Запишите на языке Паскаль программу, реализующую алгоритм работы кассира, выдающего покупателю сдачу (s) наименьшим количеством банкнот по 500 ($k500$), 100 ($k100$), 50 ($k50$) и 10 ($k10$) рублей.

Пример входных данных	Пример выходных данных
845	Следует сдать: банкнот по 500 руб. - 1 шт. банкнот по 100 руб. - 3 шт. банкнот по 50 руб. - 0 шт. банкнот по 10 руб. - 4 шт.

10. Идёт k-я секунда суток. Разработайте 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.

Пример входных данных	Пример выходных данных
13 257	It is 3 hours 40 minutes.

11. Запишите на языке Паскаль программу, которая вычисляет сумму кодов букв в слове БАЙТ.

12. Запишите на языке Паскаль программу, которая выводит на экран строку символов, коды которых равны 66, 69, 71, 73, 78.

13. Разработайте программу, которая запрашивает три строковые величины — взаимосвязанные прилагательное, существительное и глагол, а затем печатает все варианты фраз с использованием введённых слов.

Пример входных данных	Пример выходных данных
ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ	ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЛИСТЬЯ ЛИСТЬЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЛИСТЬЯ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЛИСТЬЯ ЗЕЛЁНЫЕ

14. 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)`

15. 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.

§ 4.5

Программирование разветвляющихся алгоритмов

Ключевые слова:

- условный оператор
- сокращённая форма условного оператора
- составной оператор
- вложенные ветвления

4.5.1. Условный оператор

При записи на языке Паскаль разветвляющихся алгоритмов используют условный оператор. Его общий вид:

```
if <условие> then <оператор_1> else <оператор_2>
```

Для записи неполных ветвлений используется сокращённая форма условного оператора:

```
if <условие> then <оператор>
```

Слова **if — then — else** переводятся с английского на русский язык как **если — то — иначе**, что полностью соответствует записи ветвления на алгоритмическом языке.

Перед **else** знак «;» не ставится.

В качестве условий используются логические выражения:

- простые — записанные с помощью операций отношения;
- сложные — записанные с помощью логических операций.

Пример. Запишем на языке Паскаль рассмотренный в п. 3.4.2 (пример 8) алгоритм определения принадлежности точки x отрезку $[a; b]$.

```
program n_9;
  var x, a, b: real;
begin
  writeln ('Определение принадлежности точки отрезку');
  write ('Введите a, b>>');
  readln (a, b);
  write ('Введите x>>');
  readln (x);
  if (x>=a) and (x<=b) then
       writeln ('Точка принадлежит отрезку')
  else writeln ('Точка не принадлежит отрезку')
end.
```

Пример. Воспользуемся сокращённой формой оператора ветвления для записи на языке Паскаль рассмотренного в п. 3.4.2 (пример 9) алгоритма присваивания переменной y значения наибольшей из трёх величин a, b и c.

```
program n_10;
  var y, a, b, c: integer;
begin
  writeln ('Нахождение наибольшей из трёх величин');
  write ('Введите a, b, c>>');
  readln (a, b, c);
  y:=a;
  if (b>y) then y:=b;
  if (c>y) then y:=c;
  writeln ('y=', y)
end.
```

Дополните эту программу так, чтобы её выполнение приводило к присваиванию переменной y значения большей из четырёх величин a, b, c и d.

4.5.2. Составной оператор

В условном операторе и после **then**, и после **else** можно использовать только один оператор. Если при некотором условии требуется выполнить определённую последовательность операторов, то их объединяют в один составной оператор.

Конструкция вида

begin <последовательность операторов> **end**

называется **составным оператором**.

Пример. Алгоритм решения квадратного уравнения вам хорошо известен. Запишем соответствующую программу на языке Паскаль.

```
program n_11;
  var a, b, c: real;
  var d: real;
  var 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.
```

4.5.3. Многообразие способов записи ветвлений

В качестве оператора после **then** и **else** можно использовать условный оператор. Например, возможна следующая конструкция:

```
if <условие1> then
   if <условие2> then <оператор1>
   else <оператор2>
```

При использовании таких сложных конструкций (их ещё называют вложенными ветвлениями) следует иметь в виду, что **else** всегда относится к ближайшему оператору **if**.

Пример. Воспользуемся вложенным ветвлением для записи на языке Паскаль рассмотренного в п. 3.4.2 (пример 10) алгоритма решения линейного уравнения.

```
program n_12;
  var a, b, x: real;
begin
  writeln ('Решение линейного уравнения');
  write ('Введите коэффициенты a, b>>');
  readln (a, b);
  if a<>0 then
          begin
            x:=-b/a;
            writeln ('Корень уравнения x=', x:9:3)
          end
  else if b<>0 then writeln ('Корней нет')
       else writeln ('x - любое число');
end.
```

Как правило, для решения одной и той же задачи можно предложить несколько алгоритмов. Убедимся в этом, записав программу решения линейного уравнения, не прибегая к вложенным ветвлениям.

```
program n_13;
  var a, b, x: real;
begin
  writeln ('Решение линейного уравнения');
  write ('Введите коэффициенты a, b>>');
  readln (a, b);
  if a<>0 then
          begin
            x:=-b/a;
            writeln ('Корень уравнения x=', x:9:3)
          end;
  if (a=0) and (b<>0) then writeln ('Корней нет');
  if (a=0) and (b=0) then writeln ('x - любое число')
end.
```

Возможно, второй вариант программы покажется вам более наглядным. Но и у первого варианта есть свои преимущества: в нём делается меньше проверок.

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:

```
if <условие> then <оператор_1> else <оператор_2>
```

Для записи неполных ветвлений используется сокращённый условный оператор:

```
if <условие> then <оператор>
```

Если при некотором условии требуется выполнить определённую последовательных операторов, то их объединяют в один составной оператор, имеющий вид:

```
begin <последовательность 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. Что такое составной оператор? Для чего он используется в условном операторе?
4. Используя составной оператор, упростите следующий фрагмент 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
```

5. Дано трёхзначное число. Напишите программу, которая определяет:

   а) есть ли среди цифр заданного целого трёхзначного числа одинаковые;

Пример входных данных	Пример выходных данных
123	Нет
121	Да
222	Да

б) является ли число «перевёртышем», т. е. числом, десятичная запись которого читается одинаково слева направо и справа налево.

Пример входных данных	Пример выходных данных
122	Нет
121	Перевёртыш
222	Перевёртыш

6. Даны две точки в плоской прямоугольной системе координат. Напишите программу, определяющую, которая из точек находится ближе к началу координат.

Пример входных данных	Пример выходных данных
Координаты 1-й точки>> 1, 2 Координаты 2-й точки>> 3, 4	1-я точка ближе

7. Даны три натуральных числа. Напишите программу, определяющую, существует ли треугольник с такими длинами сторон. Если такой треугольник существует, то определите его тип (равносторонний, равнобедренный, разносторонний).

Пример входных данных	Пример выходных данных
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

8. Имеются данные о количестве полных лет трёх призёров спартакиады. Напишите программу, выбирающую и выводящую возраст самого младшего призёра.

9. Напишите программу, определяющую, лежит ли точка $A(xa, ya)$:

а) на прямой $y = kx + 1$, над ней или под ней;

Пример входных данных	Пример выходных данных
k, l>>-1 5 xa, ya >>1 2	Точка лежит под прямой.
k, l>>-1 5 xa, ya >>1 10	Точка лежит над прямой.
k, l>>-1 5 xa, ya >>1 4	Точка лежит на прямой.

б) на окружности $x^2 + y^2 = r^2$, над ней или под ней. Примеры входных данных и соответствующих им выходных данных разработайте самостоятельно.

10. Напишите программу, которая производит обмен значений переменных x и y, если x больше y.

Пример входных данных	Пример выходных данных
x>>5 y>>6	x=5 y=6
x>>6 y>>5	x=5 y=6

11. 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?

12. Напишите программу, вычисляющую значение функции:

$$y = \begin{cases} -1 \text{ при } x < 0, \\ 0 \text{ при } x = 0, \\ 1 \text{ при } x > 0. \end{cases}$$

Пример входных данных	Пример выходных данных
-5	y=-1
0	y=0
5	y=1

13. Составьте программу для решения задачи № 20 к § 3.4 (определение дня недели).

14. 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, являются ли эти поля полями одного цвета.

Пример входных данных	Пример выходных данных
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

15. Напишите программу, в которой пользователю предлагается дополнить до 100 некоторое целое число a (a — случайное число, меньшее 100). Ответ пользователя проверяется и комментируется.

§ 4.6

Программирование циклических алгоритмов

Ключевые слова:

- **while** (цикл-ПОКА)
- **repeat** (цикл-ДО)
- **for** (цикл с параметром)

4.6.1. 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** <оператор>

Здесь:

<условие> — логическое выражение; пока оно истинно, выполняется тело цикла;

<оператор> — простой или составной оператор, с помощью которого записано тело цикла.

Запишем на языке Паскаль рассмотренный в п. 3.4.3 (пример 14) алгоритм получения частного q и остатка r от деления целого числа x на целое число y без использования операции деления.

```
program n_14;
  var x, y, q, r: integer;
begin
  writeln ('Частное и остаток');
  write ('Введите делимое x>>');
  readln (x);
  write ('Введите делитель y>>');
```

```
    read (y);
    r:=x;
    q:=0;
    while r>=x do
    begin
      r:=r-y;
      q:=q+1
    end;
    writeln ('Частное q=', q);
    writeln ('Остаток r=', r)
end.
```

4.6.2. 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 <оператор1; оператор2; …; > **until** <условие>

Здесь:

<оператор1>; <оператор2>; … — операторы, образующие тело цикла;

<условие> — логическое выражение; если оно ложно, то выполняется тело цикла.

Запишем на языке Паскаль рассмотренный в п. 3.4.3 (пример 17) алгоритм решения задачи о графике тренировок спортсмена.

```
program n_15;
  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.
```

4.6.3. Программирование циклов с заданным числом повторений

Цикл с заданным числом повторений (цикл-ДЛЯ) программируется в языке Паскаль с помощью оператора **for**. Его общий вид:

```
for <параметр>:=<начальное_значение> to <конечное_значение>
    do <оператор>
```

Здесь:

<параметр> — переменная целого типа;

<начальное_значение> и <конечное_значение> — выражения того же типа, что и параметр, вычисляемые перед началом цикла;

<оператор> — простой или составной оператор — тело цикла.

При выполнении этого оператора после каждого выполнения тела цикла происходит увеличение на единицу параметра цикла; условием выхода из цикла является превышение параметром конечного значения.

Запишем на языке Паскаль рассмотренный в п. 3.4.3 (пример 19) алгоритм вычисления степени с натуральным показателем n для любого вещественного числа a.

```
program n_16;
 var i, n: integer; a, y: real;
begin
 writeln ('Возведение в степень');
 write ('Введите основание a>>');
 readln (a);
 write ('Введите показатель n>>');
 readln (n);
 y:=1;
 for i:=1 to n do y:=y*a;
 writeln ('y=', y)
end.
```

4.6.4. Различные варианты программирования циклического алгоритма

Свойством программирования является то, что для решения одной и той же задачи могут быть созданы разные программы. Вы могли убедиться в этом, программируя ветвления. Рассмотрим пример, показывающий, что и циклический алгоритм может быть запрограммирован разными способами.

Пример. Напишем программу, в которой осуществляется ввод целых чисел (ввод осуществляется до тех пор, пока не будет введён ноль) и подсчёт количества введённых положительных и отрицательных чисел.

Так как здесь в явном виде задано условие окончания работы, то воспользуемся оператором **repeat**.

```
program n_17;
  var n, k1, k2: integer;
begin
  k1:=0;
  k2:=0;
  repeat
    write ('Введите целое число>>');
    readln (n);
    if n>0 then k1:=k1+1;
    if n<0 then k2:=k2+1;
  until n=0;
  writeln ('Введено:');
  writeln ('положительных чисел - ', k1);
  writeln ('отрицательных чисел - ', k2)
end.
```

Имеющееся условие окончания работы можно достаточно просто преобразовать в условие продолжения работы — работа продолжается, пока $n \neq 0$. И мы можем воспользоваться оператором **while**:

```
program n_18;
  var n, k1, k2: integer;
begin
  k1:=0;
  k2:=0;
  while n<>0 do
  begin
    writeln ('Введите целое число>>');
    read (n);
    if n>0 then k1:=k1+1;
    if n<0 then k2:=k2+1;
  end;
  writeln ('Введено:');
  writeln ('положительных - ', k1);
  writeln ('отрицательных - ', k2)
end.
```

В рассмотренном примере число повторений тела цикла заранее не известно. Поэтому оператор **for** здесь применить нельзя. Если число повторений тела цикла известно, то лучше воспользоваться оператором **for**. Вместе с тем любая задача, в которой число повторений тела цикла определено заранее, может быть запрограммирована с помощью любого из трёх рассмотренных выше циклов.

САМОЕ ГЛАВНОЕ

В языке Паскаль имеются три вида операторов цикла: **while** (цикл-ПОКА), **repeat** (цикл-ДО), **for** (цикл с параметром). Если число повторений тела цикла известно, то лучше воспользоваться оператором **for**; в остальных случаях используются операторы **while** и **repeat**.

Вопросы и задания

1. 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 повторен цикл и какими будут значения переменных a, b, s после исполнения этой последовательности операторов?

2. 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 пять ошибок:

```
k:=1;
f:=0;
while k<n do
  f:=f*k;
  k:=k+1
```

Найдите ошибки. Допишите необходимые операторы и выполните программу на компьютере.

Пример входных данных	Пример выходных данных
Введите n> 5	5!=120
Введите n> 6	6!=720

3. Проанализируйте следующий цикл:

```
while a<b do
  c:=a=b;
```

В чём его особенность?

4. Запишите на языке Паскаль программы решения задач № 25–29 из § 3.4. Используйте оператор **while**.

5. Дана последовательность операторов:

```
a:=1;
b:=1;
repeat
  a:=a+1;
  b:=b*2;
until b>8;
s:=a+b
```

Сколько раз будет повторён цикл и какими будут значения переменных a, b, s после исполнения этой последовательности операторов?

6. Напишите программу, в которой осуществляется ввод целых чисел (ввод осуществляется до тех пор, пока не будет введён ноль) и подсчёт суммы и среднего арифметического введённых положительных чисел. Используйте оператор **repeat**.

7. 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**.

8. Напишите программу вычисления наибольшего общего делителя двух целых чисел:

а) используйте оператор **repeat**;

б) используйте оператор **while**.

9. Сколько раз будет выполнен цикл?

а)
```
for i:=0 to 15 do s:=s+1;
```
б)
```
for i:=10 to 15 do s:=s+1;
```
в)
```
for i:=-1 to 1 do s:=s+1;
```
г)
```
for i:=10 to 10 do s:=s+1;
```
д)
```
k:=5;
for i:=k-1 to k+1 do s:=s+1;
```

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 представлен ниже:

```
*   *   *   *
 *   *   *   *
*   *   *   *
 *   *   *   *
*   *   *   *
 *   *   *   *
*   *   *   *
 *   *   *   *
```

12. Напишите программу, которая вычисляет сумму:

а) первых n натуральных чисел;

б) квадратов первых n натуральных чисел;

в) всех чётных чисел в диапазоне от 1 до n;

г) всех двузначных чисел.

13. Напишите программу, которая генерирует 10 случайных чисел в диапазоне от 1 до 20, выводит эти числа на экран и вычисляет их среднее арифметическое.

14. Запишите на языке Паскаль программы решения задач № 32–33 из параграфа 3.4. Используйте оператор **for**.

15. Напишите программу, которая выводит на экран таблицу степеней двойки (от нулевой до десятой). Рекомендуемый вид экрана после выполнения программы представлен ниже:

```
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
```

16. Напишите программу, которая выводит на экран таблицу умножения на n (n — целое число в диапазоне от 2 до 10, вводимое с клавиатуры).

Пример входных данных	Пример выходных данных
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

17. Какой из трёх рассмотренных операторов цикла является, по вашему мнению, основным, т. е. таким, что им можно заменить два других? Обоснуйте свою точку зрения.

§ 4.7
Одномерные массивы целых чисел

Ключевые слова:
- массив
- описание массива
- заполнение массива
- вывод массива
- обработка массива
- последовательный поиск
- сортировка

До сих пор мы работали с простыми типами данных. При решении практических задач данные часто объединяются в различные **структуры данных**, например в массивы. В языках программирования массивы используются для реализации таких структур данных, как последовательности[1] и таблицы.

Массив — это поименованная совокупность однотипных элементов, упорядоченных по индексам, определяющим положение элемента в массиве.

Мы будем рассматривать одномерные массивы.

Решение разнообразных задач, связанных с обработкой массивов, базируется на решении таких типовых задач, как:
- суммирование элементов массива;
- поиск элемента с заданными свойствами;
- сортировка массива.

[1] Например, числовые последовательности в математике.

4.7.1. Описание массива

Перед использованием в программе массив должен быть описан, т. е. должно быть указано имя массива, количество элементов массива и их тип. Это необходимо для того, чтобы выделить в памяти под массив блок ячеек нужного типа. Общий вид описания массива:

```
var <имя_массива>: array [<мин_знач_индекса> ..
    <макс_знач_индекса>] of <тип_элементов>;
```

Пример

```
var a: array [1..10] of integer;
```

Здесь описан массив *a* из десяти целочисленных значений. При выполнении этого оператора в памяти компьютера будет выделено десять ячеек целого типа.

Небольшой массив с постоянными значениями может быть описан в разделе описания констант:

```
const b: array [1..5] of integer = (1, 2, 3, 5, 7);
```

В этом случае не просто выделяются последовательные ячейки памяти — в них сразу же заносятся соответствующие значения.

4.7.2. Заполнение массива

Заполнять массив можно либо вводя значение каждого элемента с клавиатуры, либо присваивая элементам некоторые значения. При этом может использоваться цикл с параметром.

Например, для ввода с клавиатуры значений элементов описанного выше массива *a* используется следующий цикл с параметром:

```
for i:=1 to 10 do read (a[i]);
```

Задавать значения элементов массива можно с помощью оператора присваивания. Например:

```
for i:=1 to 10 do a[i]:=i;
```

В следующем фрагменте программы организовано заполнение целочисленного массива *a*, состоящего из 10 элементов, случайными числами, значения которых изменяются в диапазоне от 0 до 99:

```
randomize;
for i:=1 to 10 do a[i]:=random(100);
```

4.7.3. Вывод массива

Во многих случаях бывает полезно вывести значения элементов массива на экран. Так, если значения массива генерировались случайным образом, то необходимо знать, каков исходный массив. Также нужно знать, каким стал массив после обработки.

Элементы массива можно вывести в строку, разделив их пробелом:

```
for i:=1 to 10 do write (a[i], ' ');
```

Более наглядным является следующий вариант вывода с комментариями:

```
for i:=1 to 10 do writeln ('a[', i, ']=', a[i]);
```

На основании рассмотренных примеров попытайтесь самостоятельно запишите программу, в которой осуществляется: заполнение случайным образом целочисленного массива a, состоящего из 10 элементов, значения которых изменяются в диапазоне от 0 до 99; вывод массива a на экран.

4.7.4. Вычисление суммы элементов массива

Суммирование элементов массива осуществляется по тому же принципу, что и суммирование значений простых переменных: за счёт поочерёдного добавления слагаемых:

1) определяется ячейка памяти (переменная s), в которой будет последовательно накапливаться результат суммирования;
2) переменной s присваивается начальное значение 0 — число, не влияющее на результат сложения;
3) для каждого элемента массива из переменной s считывается её текущее значение и складывается со значением элемента массива; полученный результат присваивается переменной s.

Описанный процесс наглядно можно изобразить так:

$s = 0$	$s = 0$
$s = s + a[1]$	$s = 0 + a[1]$
$s = s + a[2]$	$s = 0 + a[1] + a[2]$
$s = s + a[3]$	$s = 0 + a[1] + a[2] + a[3]$
...	...
$s = s + a[10]$	$s = 0 + a[1] + a[2] + a[3] + ... + a[10]$

Приведём основной фрагмент решения этой задачи:

```
s:=0;
for i:=1 to n do s:=s+a[i];
```

Дополните созданную в п. 4.7.3 программу формирования массива так, чтобы вычислялась сумма элементов массива и результат суммирования выводился на экран.

4.7.5. Последовательный поиск в массиве

В программировании поиск — одна из наиболее часто встречающихся задач невычислительного характера.

Можно выделить следующие типовые задачи поиска:

1) найти наибольший (наименьший) элемент массива;
2) найти элемент массива, значение которого равно заданному значению.

Компьютер не может сравнить разом весь ряд объектов. На каждом шаге он может сравнивать только два объекта. Поэтому в программе необходимо организовать последовательный просмотр элементов массива и сравнение значения очередного просматриваемого элемента с неким образцом.

Рассмотрим подробно решение задач первого типа (нахождение наибольшего (наименьшего) элемента).

Представим себе одномерный массив в виде стопки карточек, на каждой из которых написано число. Тогда идея поиска наибольшего элемента массива может быть представлена следующим образом:

1) возьмём верхнюю карточку (первый элемент массива), запомним имеющееся на карточке число (запишем его мелом на доске) как наибольшее из просмотренных; уберём карточку в сторону;

2) возьмём следующую карточку; сравним числа, записанные на карточке и на доске; если число на карточке больше, то сотрём число, записанное на доске, и запишем там то же число, что и на карточке; если же новое число не больше, то на доске оставим имеющуюся запись; уберём карточку в сторону;

3) повторим действия, описанные в п. 2, для всех оставшихся карточек в стопке.

В итоге на доске будет записано самое большое значение просмотренного массива.

Так как доступ к значению элемента массива осуществляется по его индексу, то при организации поиска наибольшего элемента в од-

номерном массиве правильнее искать его индекс. Обозначим искомый индекс *imax*. Тогда описанный выше алгоритм в сформированном нами массиве *a* на языке Паскаль можно записать так:

```
imax:=1
for i:=2 to 10 do
  if a[i]>a[imax] then imax:=i;
write ('Наибольший элемент a[', imax, ']=', a[imax])
```

Самостоятельно запишите программу, в которой осуществляется формирование целочисленного массива *a* из 10 элементов, значения которых лежат в диапазоне от 0 до 99, и поиск наибольшего элемента этого массива.

Если в массиве несколько элементов, равных максимальному значению, то данная программа найдёт первый из них (первое вхождение). Подумайте, что следует изменить в программе, чтобы в ней находился последний из максимальных элементов. Как следует преобразовать программу, чтобы с её помощью можно было найти минимальный элемент массива?

Результатом решения задачи второго типа (нахождение элемента массива, значение которого равно заданному значению) может быть:

- *n* — индекс элемента массива такой, что $a[n] = x$, где x — заданное число;
- сообщение о том, что искомого элемента в массиве не обнаружено.

Алгоритм поиска в сформированном нами массиве *a* значения, равного 50, может выглядеть так:

```
n:=0;
for i:=1 to 10 do
  if a[i]=50 then n:=i;
if n=0 then write('Нет') else write (i)
```

В этой программе последовательно просматриваются все элементы массива. Если в массиве несколько элементов, значения которых равны заданному числу, то программа найдёт последний из них.

Запишите полный текст программы и выполните её на компьютере.

Во многих случаях требуется найти первый из элементов, имеющих соответствующее значение, и дальнейший просмотр массива прекратить. Для этой цели можно использовать следующую программу:

```
i:=0;
repeat
  i:=i+1;
until (a[i]=50) or (i=10);
if a[i]=50 then write(i) else write('Нет')
```

Здесь выполнение алгоритма будет прервано в одном из двух случаев: 1) в массиве найден первый из элементов, равный заданному; 2) все элементы массива просмотрены.

Запишите полный текст программы и выполните её на компьютере.

Зачастую требуется определить количество элементов, удовлетворяющих некоторому условию. В этом случае вводится переменная, значение которой увеличивается на единицу каждый раз, когда найден нужный элемент.

Определите, количество каких элементов подсчитывается в следующем фрагменте программы.

```
k:=0;
for i:=1 to 10 do
  if a[i]>50 then k:=k+1;
write('k=', k)
```

Если требуется определить сумму значений элементов, то вводят переменную, к значению которой прибавляют значение найденного элемента массива.

Определите, какому условию удовлетворяют элементы массива, значения которых суммируются в следующем фрагменте программы.

```
s:=0;
for i:=1 to 10 do
  if (a[i]>50) and (a[i]<60)  then s:=s+a[i];
write('s=', s)
```

Запишите полные тексты двух последних программ и выполните их на компьютере.

4.7.6. Сортировка массива

Под **сортировкой (упорядочением)** массива понимают перераспределение значений его элементов в некотором определённом порядке.

Порядок, при котором в массиве первый элемент имеет самое маленькое значение, а значение каждого следующего элемента не меньше значения предыдущего элемента, называют **возрастающим**.

Порядок, при котором в массиве первый элемент имеет самое большое значение, а значение каждого следующего элемента не больше значения предыдущего элемента, называют **убывающим**.

Цель сортировки — облегчить последующий поиск элементов: искать нужный элемент в упорядоченном массиве легче.

Вы уже встречались с сортировкой при работе с базами данных. Сейчас мы рассмотрим один из возможных вариантов[1] реализации механизма этой операции — **сортировку выбором**.

Сортировка выбором (например, по убыванию) осуществляется следующим образом:

1) в массиве выбирается максимальный элемент;
2) максимальный и первый элементы меняются местами (первый элемент считается отсортированным);
3) в неотсортированной части массива снова выбирается максимальный элемент; он меняется местами с первым неотсортированным элементом массива;
4) действия, описанные в п. 3, повторяются с неотсортированными элементами массива до тех пор, пока не останется один неотсортированный элемент (его значение будет минимальным).

Рассмотрим процесс сортировки выбором на примере массива $a = \{0, 1, 9, 2, 4, 3, 6, 5\}$.

Индекс		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Значение		**0**	**1**	**9**	**2**	**4**	**3**	**6**	**5**
Шаги	**1**	**0**	**1**	**9**	**2**	**4**	**3**	**6**	**5**
	2	**9**	**1**	**0**	**2**	**4**	**3**	**6**	**5**
	3	**9**	**6**	**0**	**2**	**4**	**3**	**1**	**5**
	4	**9**	**6**	**5**	**2**	**4**	**3**	**1**	**0**
	5	**9**	**6**	**5**	**4**	**2**	**3**	**1**	**0**
	6	**9**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	7	**9**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Итог:	**9**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

[1] С **другими** способами сортировки вы познакомитесь на уроках информатики и ИКТ в 10–11 классах.

В этом массиве из восьми элементов операцию выбора максимального элемента мы проводили 7 раз. В массиве из n элементов такая операция будет проводиться $n-1$ раз. Объясните почему.

Приведём фрагмент программы, реализующий описанный алгоритм:

```
for i:=1 to n-1 do
begin
  imax:=i;
  for j:=i+1 to n do if a[j]>a[imax] then imax:=j;
  x:=a[i];
  a[i]:=a[imax];
  a[imax]:=x
end;
```

Здесь мы использовали один цикл внутри другого. Такая конструкция называется **вложенным циклом.**

Запишите полный текст программы и выполните её на компьютере для рассмотренного в примере массива a.

На сайте «Интерактивные демонстрации по программированию» (http://informatika.kspu.ru/flashprog/demos.php) вы сможете поработать с интерактивными наглядными пособиями для того, чтобы более полно представить процесс сортировки выбором и другими способами.

САМОЕ ГЛАВНОЕ

Массив — это поименованная совокупность однотипных элементов, упорядоченных по индексам, определяющим положение элементов в массиве. В языках программирования массивы используются для реализации таких **структур данных**, как последовательности и таблицы.

Перед использованием в программе массив должен быть описан. Общий вид описания одномерного массива:

```
var <имя_массива>: array [<мин_знач_индекса> ..
    <макс_знач_индекса>] of тип_элементов;
```

Заполнять массив можно либо вводя значение каждого элемента с клавиатуры, либо присваивая элементам некоторые значения. При

заполнении массива и его выводе на экран используется цикл с параметром.

Решение разнообразных задач, связанных с обработкой массивов, базируется на таких типовых задачах, как: суммирование элементов массива; поиск элемента с заданными свойствами; сортировка массива.

Вопросы и задания

1. Может ли массив одновременно содержать целые и вещественные значения?
2. Для чего необходимо описание массива?
3. Что вы можете сказать о массиве, сформированном следующим образом?

   а) `for i:=1 to 10 do a[i]:=random(101)-50;`

   б) `for i:=1 to 20 do a[i]:=i;`

   в) `for i:=1 to 5 do a[i]:=2*i-1;`
4. Запишите на языке Паскаль программу решения задачи, рассмотренной в примере 21 § 3.4. Считайте количество жильцов дома случайным числом из диапазона от 50 до 200 человек, а число домов $n = 30$.
5. Запишите на языке Паскаль программу решения задачи № 34 к § 3.4. Считайте рост претендента в команду случайным числом из диапазона от 150 до 200 см, а число претендентов $n = 50$.
6. Напишите программу, которая вычисляет среднюю за неделю температуру воздуха. Исходные данные вводятся с клавиатуры.

Пример входных данных	Пример выходных данных
Введите температуру Понедельник>> 12 Вторник>>10 Среда>>16 Четверг>>18 Пятница>>17 Суббота>>16 Воскресенье>>14	Средняя температура за неделю: 14.71

7. Дан массив из десяти целых чисел. Определите, сколько элементов этого массива имеют максимальное значение.
8. В классе 20 учеников писали диктант по русскому языку. Напишите программу, подсчитывающую количество двоек, троек, четвёрок и пятёрок, полученных за диктант.
9. В целочисленных массивах a и b содержатся длины катетов десяти прямоугольных треугольников ($a[i]$ — длина первого катета, $b[i]$ — длина второго катета i-го треугольника). Найдите треугольник с наибольшей площадью. Выведите его номер, длины катетов и площадь. Предусмотрите случай, когда таких треугольников несколько.
10. Занесите информацию о десяти европейских странах в массивы n (название страны), κ (численность населения), s (площадь страны). Выведите названия стран в порядке возрастания плотности их населения.

§ 4.8

Запись вспомогательных алгоритмов на языке Паскаль

Ключевые слова:

- подпрограмма
- процедура
- функция
- рекурсивная функция

Запись вспомогательных алгоритмов в языках программирования осуществляется с помощью подпрограмм. В Паскале подпрограмма является частью основной программы. Её описание располагается между разделом **var** и программным блоком главной программы. Если подпрограмм несколько, то их описания располагаются в произвольном порядке одно за другим.

Структура описания подпрограммы аналогична структуре главной программы. Описание подпрограммы начинается с заголовка и заканчивается оператором **end**.

В Паскале различают два вида подпрограмм: процедуры и функции.

4.8.1. Процедуры

Процедура — подпрограмма, имеющая произвольное количество входных и выходных данных.

Описание процедуры имеет вид:

```
procedure <имя процедуры> (<описание параметров-значений>;
          var: <описание параметров-переменных>);
begin
  <операторы>
end;
```

В заголовке процедуры после её имени приводится перечень формальных параметров и их типов. Входные параметры, значения которых не изменяются в программе, должны быть параметрами-значениями. Выходные (результирующие) параметры должны быть параметрами-переменными.

Для вызова процедуры достаточно указать её имя со списком фактических параметров. В качестве параметров-значений можно указывать имена переменных, константы и выражения.

Например, заголовок процедуры вычисления наибольшего общего делителя может быть описан так:

```
procedure nod (a, b: integer; var c: integer);
```

Возможны следующие варианты вызова этой процедуры:

`nod (36, 15, z)`	– в качестве параметров-значений использованы константы;
`nod (x, y, z)`	– в качестве параметров-значений использованы имена переменных;
`nod (x+y, 15, z)`	– в качестве параметров-значений использованы выражение и константа;

В любом случае между фактическими и формальными параметрами должно быть полное соответствие по количеству, порядку следования и типу.

Пример 1. Напишем процедуру для нахождения наибольшего общего делителя двух чисел с помощью алгоритма Евклида. Используем её для нахождения наибольшего общего делителя следующих шести чисел: 16, 32, 40, 64, 80 и 128.

`program n_20;`	Заголовок главной программы
`const m: array [1..6] of integer = (16, 32, 40, 64, 80, 128);`	Раздел описания констант
`var i: integer;`	Раздел описания переменных

`procedure nod (a, b: integer; var c: integer);`
`begin`
`  while a<>b do`
`  if a>b then a:=a-b else b:=b-a;`
`  c:=a`
`end;`
	Раздел описания подпрограмм

```
begin
   x:=m[1];
   for i:=2 to 6 do
   begin
     y:=c[i];
     nod (x, y, z);
     x:=z
   end;
   writeln ('НОД=', nod)
end.
```

Раздел операторов главной программы

Измените программу так, чтобы с её помощью можно было найти:
а) наибольший общий делитель следующих пяти чисел: 12, 24, 30, 48 и 30;
б) наибольший общий делитель произвольных десяти целых двузначных чисел.

4.8.2. Функции

Описание функции имеет вид:

```
function <имя_функции> (<описание входных данных>):
         <тип_функции>;
begin
   <операторы>;
   <имя_функции> := <результат>
end;
```

В заголовке функции после её имени приводится описание входных данных — указывается перечень формальных параметров и их типов. Там же указывается тип самой функции, т. е. тип результата.

Функция — подпрограмма, имеющая единственный результат, записываемый в ячейку памяти, имя которой совпадает с именем функции. Поэтому в блоке функции обязательно должен присутствовать оператор `<имя_функции>:=<результат>`.

Для вызова функции достаточно указать её имя со списком фактических параметров в любом выражении, в условиях (после слов **if**, **while**, **until**), или в операторе `write` главной программы.

Пример 2. Напишем программу нахождения максимального из четырёх целых чисел, использующую функцию поиска максимального из двух чисел:

`program n_20;`	Заголовок главной программы
`var a, b, c, d, f: integer;`	Раздел описания переменных

`function max (x, y: integer): integer;`
`begin`
`  if x>y then max:=x else max:=y;`
`end;`
	Раздел описания подпрограмм

`begin`
`  readln (a, b, c, d);`
`  f:=max(max(a, b), max(c, d));`
`  writeln ('f=', f);`
`end.`
	Раздел операторов главной программы

Измените программу так, чтобы с её помощью можно было найти:
а) максимальное из чисел a, b, c;
б) максимальное из чисел b, c, d;
в) минимальное из четырёх чисел;
г) разность максимального и минимального чисел.

Пример 3. В январе Саше подарили пару новорождённых кроликов. Через два месяца они дали первый приплод — новую пару кроликов, а затем давали приплод по паре кроликов каждый месяц. Каждая новая пара также даёт первый приплод (пару кроликов) через два месяца, а затем — по паре кроликов каждый месяц. Сколько пар кроликов будет у Саши в декабре?

Составим математическую модель этой задачи. Обозначим через $f(n)$ количество кроликов в месяце с номером n. По условию задачи, $f(1) = 1$, $f(2) = 1$, $f(3) = 2$. Из двух пар, имеющихся в марте, дать приплод в апреле сможет только одна: $f(4) = 3$. Из пар, имеющихся в апреле, дать приплод в мае смогут только пары, родившиеся в марте: $f(5) = f(4) + f(3) = 3 + 2 = 5$. В общем случае: $f(n) = f(n - 1) + f(n - 2)$, $n \geq 3$.

Числа 1, 1, 2, 3, 5, 8, ... образуют так называемую **последовательность Фибоначчи**, названную в честь итальянского математика, впервые решившего соответствующую задачу ещё в начале XIII века.

Оформим в виде функции вычисление члена последовательности Фибоначчи.

```
function f (n: integer): integer;
begin
  if (n=1) or (n=2)  then f:=1
  else f:=f(n-1)+f(n-2)
end;
```

Полученная функция считается рекурсивной — в ней реализован способ вычисления очередного значения функции через вычисление её предшествующих значений.

САМОЕ ГЛАВНОЕ

Запись вспомогательных алгоритмов в языках программирования осуществляется с помощью подпрограмм. В Паскале различают два вида подпрограмм: процедуры и функции.

Процедура — подпрограмма, имеющая произвольное количество входных и выходных данных.

Функция — подпрограмма, имеющая единственный результат, записываемый в ячейку памяти, имя которой совпадает с именем функции.

Вопросы и задания

1. Для чего используются подпрограммы?
2. В чём основное различие процедур и функций?
3. Напишите программу вычисления наименьшего общего кратного следующих четырёх чисел: 36, 54, 18 и 15. Используйте процедуру вычисления наибольшего общего делителя двух чисел.
4. Напишите программу перестановки значений переменных a, b, c в порядке возрастания, т. е. так, чтобы $a < b < c$. Используйте процедуру `swap`.

```
procedure swap (var x, y: integer);
  var m: integer;
begin
  m:=x;
  x:=y;
  y:=m
end;
```

Исходные данные вводятся с клавиатуры.

Пример входных данных	Пример выходных данных
1 2 3	1 2 3
2 1 3	1 2 3
3 1 2	1 2 3
2 3 1	1 2 3

5. Напишите программу поиска наибольшего из четырёх чисел с использованием подпрограммы поиска наибольшего из трёх чисел.
6. Видоизмените программу сортировки массива выбором так, чтобы в ней использовалась процедура выбора наибольшего элемента массива.
7. Напишите программу вычисления выражения: $s = 1! + 2! + 3! + ... + n!$ Используйте функцию вычисления факториала.
8. Напишите программу вычисления выражения: $s = x^3 + x^5 + x^n$, где x и n вводятся с клавиатуры. Используйте функцию вычисления степени.
9. Напишите функцию, вычисляющую длину отрезка по координатам его концов. С помощью этой функции напишите программу, вычисляющую периметр треугольника по координатам его вершин.
10. Напишите функцию, вычисляющую площадь треугольника по целочисленным координатам его вершин. С помощью этой функции вычислите площадь четырёхугольника по координатам его вершин.

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) служебные слова
   в) русские строчные и прописные буквы
   г) знак подчеркивания

3. Какая последовательность символов не может служить именем в языке Паскаль?

   а) `_mas`
   б) `maS1`
   в) `d2`
   г) `2d`

4. Вещественные числа имеют тип данных:

   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 данном 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 в операторе `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 промежутка [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 writeln('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) **loop**

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 ищется 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 p:=p*i; i:=i+1 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 чётных чисел

26. Имеется описание:

```
var c: array [1..20] of integer;
```

Для хранения массива c будет отведено ... последовательных ячеек памяти объёмом ... байтов.

а) 40, 20
б) 20, 320
в) 20, 40
г) 20, 20

27. Чему равна сумма элементов $a[1]$ и $a[4]$ массива, сформированного следующим образом.

```
for i:=1 to 5 do a[i]:=i*(i+1);
```

а) 30
б) 5
в) 22
г) 40

28. Массив описан следующим образом:

```
const b: array [1..5] of integer = (1, 2, 3, 5, 11);
```

Значение выражения `b[5]*p[4]-p[2]-p[3]*p[1]` равно:

а) 50
б) 15
в) −11
г) 22

29. Для записи вспомогательных алгоритмов в языке Паскаль используются:

а) массивы
б) составные операторы
в) процедуры и функции
г) операторы и операнды

30. Между формальными и фактическими параметрами следует соблюдать соответствие:

а) по типу параметров
б) по количеству параметров
в) по порядку следования параметров
г) по всему, перечисленному в п. а–в)

Ответы и решения к вопросам и заданиям для самостоятельной подготовки

Глава 1

§ 1.1. 2. Приведены знаки, используемые для записи чисел в следующих системах счисления: древнеегипетской, вавилонской, майя, римской, старославянской, двоичной, троичной, ..., шестнадцатеричной. **7.** а) 110011_2; б) 111_4. **8.** Минимальное основание 5. 38, 62, 31, 71. **9.** а) да; б) нет. **10.** а) 5; б) $2 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 2 \cdot x^0 = 130$, $2x^3 = 128$, $x^3 = 64$, $x = 4$. **17.** а) $12 \cdot 3 - 4 = 32$; б) $12 : 2 - 2 = 4$; в) $12 : 3 - 4 = 0$. **18.** а) 10; б) 198.

§ 1.2. 3. 00111111. **4.** а) +76. **5.** 101010_2. **9.** а) $0{,}217934 \cdot 10^3$; б) $0{,}75321 \cdot 10^5$; в) $0{,}101 \cdot 10^{-2}$.

§ 1.3. 6.

Электрическая схема	Алгебра логики
Переключатель	Высказывание
Переключатель включён	1
Переключатель выключен	0
Последовательное соединение переключателей	Конъюнкция
Параллельное соединение переключателей	Дизъюнкция

7. 920, 80.
10. 18.
12. 1) 0; 2) 0; 3) 1; 4) 1.
13. 1) 1; 2) 1; 3) 0; 4) 1.
14. Смит и Джон.
15. Финикийский сосуд, изготовлен в V веке.
16. $F = \overline{A} \wedge B$.

Глава 2

§ 2.2. 4. $4/(v - 6{,}5) + 33/(v + 6{,}5) = 1$, $v = 32{,}5$ км/ч. **5.** $F(A, B, C) = A\&B + A\&C + B\&C$. **6.** 1 — Россия, 2 — Китай, 3 — Украина, 4 — Германия, 5 — Италия.

§ 2.3. 5. Для решения задачи постройте взвешенный граф, каждому ребру которого будет соответствовать время, затрачиваемое велосипедистом, на преодоление соответствующего расстояния. **8.** 24. **9.** 648. **10.** 16. **11.** Выигрывает второй игрок. Своим первым ходом ему нужно получить одну из ситуаций: (3,4) или (1,18) — здесь числа в скобках обозначают количество камней в первой и второй кучах соответственно.

§ 2.4. 5. Выигрывает первый игрок. Своим первым ходом он должен добавить по два камня в каждую из кучек. **6.** Третья компания. **7.** I — Михаил Зимин, II — Эдуард Симаков, III — Николай Копылов, IV — Валерий Блинов, V — Игорь Чигрин. **8.** Антон — Екатерина — Норильск, Борис — Ольга — Пятигорск, Давид — Светлана — Ростов, Григорий — Мария — Москва.

§ 2.6. 5. а) 2; б) 3; в) **4. 10.** а) 2; б) 2; в) 3; г) **1. 11.** а) 1; б) 2; в) 3; г) 1.

Глава 3

§ 3.1. 1. 1, 1, 2, 3, 5, 8, 13, ... **18.** 6, 12212.

§ 3.3. 9. Не более одной переменной. **14.** г) $(x > 0)$ или $(y > 0)$; е) $((x > 0)$ и $(y <= 0))$ или $(y > 0)$ и $(x <= 0))$; ж) $x*x + y*y <= r*r$. **17.** а) `t:=x>0`; в) `t:=(x=y) и (x=z)`.

§ 3.4. 5. $y = ((2x + 3)x + 4)x + 5$ при $x = 1$ $y = 14$. **6.** $h := tfh*24$. **20. если** *chislo* = 3 **то** $y :=$ понедельник.

§ 3.5. Пример 4. Пусть k_n — количество ребер в границе снежинки Коха на шаге n; тогда $k_n = k_{n-1}*4$. Отсюда: $k_1=12$, $k_2=48$, $k_3=192$, $k_4=768$, $k_5=3012$.

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. 9. 1 11.

§ 4.3. 11. `s:=a*50+a*1.2*150;`

§ 4.4. 14. а) `true`; б) `true`; в) `false`.

§ 4.5. 2. а) да; б) нет; в) нет. **11.** 5.

§ 4.6. 1. Два раза. 3, 6, 9. **5.** Четыре раза. 5, 16, 21.

9. а) 16; б) 6; в) 3; г) 1; д) 3.

§ 4.7. 3.

в)

i	1	2	3	4	5
a[*i*]	1	3	5	7	9

Ключи к тестовым заданиям для самоконтроля

Глава 1

Задание	1	2	3	4	5	6	7	8	9	10
Ответ	в	б	в	б	б	г	в	б	б	г
Задание	11	1	13	14	15	16	17	18	19	20
Ответ	в	б	г	а	а	г	в	б	б	г

Глава 2

Задание	1	2	3	4	5	6	7	8	9	10
Ответ	в	г	г	б	в	г	а	б	а	б
Задание	11	12	13	14	15	16	17	18	19	20
Ответ	г	г	а	г	в	г	б	а	г	в
Задание	21	22	23	24	25	26				
Ответ	б	а	а	в	в	в				

Глава 3

Задание	1	2	3	4	5	6	7	8	9	10
Ответ	а	г	в	б	а	в	г	д	11121	127
Задание	11	12	13	14	15	16	17	18	19	20
Ответ	в	в	а	г	б	б	а	в	б	г
Задание	21	22	23	24	25	26	27	28	29	30
Ответ	б	б	80	а	б	а	г	25	120	55

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	26	27	28	29	30
Ответ	в	б	г	б	г	в	в	а	в	г

Оглавление

Учебное издание

Босова Людмила Леонидовна
Босова Анна Юрьевна

ИНФОРМАТИКА И ИКТ
Учебник для 9 класса

В двух частях
Часть первая

Ведущий редактор *О. А. Полежаева*
Методист *И. Л. Сретенская*
Художник *Н. А. Новак*
Технический редактор *Е. В. Денюкова*
Корректор *Е. Н. Клитина*
Компьютерная верстка: *Е. А. Голубова*

Подписано в печать 21.02.12. Формат 70×100/16.
Усл. печ. л. 20,15. Тираж 25 000 экз. Заказ 4225

Издательство «БИНОМ. Лаборатория знаний»
125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: binom@Lbz.ru, http://www.Lbz.ru

При участии ООО Агентство печати «Столица»
тел.: (495) 331-14-38; e-mail: apstolica@bk.ru

Отпечатано с готовых файлов заказчика
в ОАО «Первая Образцовая типография»,
филиал «УЛЬЯНОВСКИЙ ДОМ ПЕЧАТИ»
432980, г. Ульяновск, ул. Гончарова, 14

УДК 004.9
ББК 32.97
Б85

Босова Л. Л.

Б85 Информатика и ИКТ : учебник для 9 класса : в 2 ч. Ч. 1 / Л. Л. Босова, А. Ю. Босова. — М. : БИНОМ. Лаборатория знаний, 2012. — 244 с. : ил.

ISBN 978-5-9963-0827-9 (Ч. 1)
ISBN 978-5-9963-0538-4

Учебник предназначен для продолжения изучения курса «Информатика и ИКТ» в 9 классе общеобразовательной школы. Содержание учебника соответствует Государственному образовательному стандарту основного общего образования по информатике и ИКТ. Выдержан принцип инвариантности к конкретным моделям компьютеров и версиям программного обеспечения.

За счет формирования у учащихся алгоритмического, логического и системного мышления, умений и навыков использования информационных технологий создаются условия для достижения ими метапредметных образовательных результатов, обеспечивается подготовка к сдаче экзамена за курс основной школы в формате ГИА.

Предполагается широкое использование ресурсов федеральных образовательных порталов, в том числе Единой коллекции цифровых образовательных ресурсов (http://school-collection.edu.ru/).

Учебник включен в Федеральный перечень учебников, рекомендуемых Министерством образования и науки Российской Федерации.

УДК 004.9
ББК 32.97

По вопросам приобретения обращаться:
«БИНОМ. Лаборатория знаний»
Телефон: (499) 157-5272
e-mail: binom@Lbz.ru, http://www.Lbz.ru

ISBN 978-5-9963-0827-9 (Ч. 1)
ISBN 978-5-9963-0538-4

Этот учебник входит в УМК по информатике для 8–9 классов. Включен в Федеральный перечень учебников, рекомендованных Министерством образования и науки Российской Федерации.

Состав учебно-методического комплекта:

- авторская программа изучения курса информатики и ИКТ (для обязательного и углубленного изучения предмета);
- учебник для 8 класса;
- учебник для 9 класса;
- рабочая тетрадь для 8 класса;
- рабочая тетрадь для 9 класса;
- методическое пособие для учителя;
- набор электронных образовательных ресурсов.

ISBN 978-5-9963-0827-9