С. В. Мациевский
С. А. Ишанов
С. В. Клевцур

ИНФОРМАТИКА

Учебное пособие

Издательство
Калининградского государственного университета
2003

УДК 681.3(075)
ББК 32.973.2я7
М 367

Рецензент
старший научный сотрудник ЗО ИЗМИРАН
канд. физ.-мат. наук Ф. С. Бессараб

Мациевский С. В., Ишанов С. А., Клевцур С. В.
М 367 Информатика: Учебное пособие / Калининград: Изд-во КГУ, 2003.— 140 с.; ил.: 43; табл.: 45; листингов: 26; библ.: 15 назв.

Настоящее учебное пособие является записью лекций по информатике, прочитанных поступающим в Калининградский государственный университет. Оно достаточно сбалансировано, а его отдельные разделы увязаны между собой.

Учебный материал рассчитан на тех, кто не знаком ни с компьютером, ни с программированием, но, безусловно, будет полезен всем школьникам. При работе с этой книгой компьютер не необходим, но желателен.

Первая часть «Компьютер» может быть использована для получения базовых знаний перед компьютерной практикой, вторая часть «Программирование» — перед программированием на Паскале или другом процедурном языке.

Курс может быть использован не только школьниками, готовящимися к вступительному экзамену по информатике. Эти лекции будут полезны как при обучении информатике в школе, так и при преподавании информатики в вузах. Не исключается работа с материалом при самообразовании.

Содержание соответствует рекомендациям Министерства образования РФ по вступительным экзаменам по информатике.

Замечания и пожелания принимаются по адресу math@kaliningrad.org.

Утверждено ученым советом математического факультета.

ISBN 5-88874-385-2

Оглавление

Введение

Настоящее пособие представляет собой руководство по информатике для абитуриентов, готовящихся к поступлению в Калининградский государственный университет на математический факультет (специальности «прикладная математика и информатика», «математическое обеспечение и администрирование информационных систем») и факультет сервиса.

Компьютерные науки (*computer science*) и информационные технологии стали вездесущими и продолжают сулить перемены, которые еще больше затрагивают практически все сферы нашей жизни. Компьютеры превратились в неотъемлемую часть современной культуры и являются движущей силой экономического роста во всем мире. Информатика развивается с поразительной скоростью. Постоянно появляются новые технологии, а существующие становятся устаревшими. В такой ситуации выигрывают люди, которые могут учиться, умеют учиться и обладают необходимыми базовыми знаниями.

Подавляющее большинство наших абитуриентов не задает наивных вопросов о проблемах своей будущей профессии. Они давно и бесповоротно сделали свой выбор в пользу математики и *computer science*, еще в школе самостоятельно работали на компьютерах, читали специальную литературу, участвовали в олимпиадах и конференциях. Просто диву даешься, как ловко иные школьники обращаются с компьютерами, владеют терминологией, знают современный рынок технических и программных средств.

Но мода на информатику имеет и обратную сторону: некоторые школьники воспринимают ее на чисто внешнем, поверхностном уровне. Рассуждая о мультимедиа и виртуальной реальности, они не могут написать простенькую программу сортировки или редактирования, а с математикой испытывают проблемы.

Традиционно компьютерные науки имеют весьма тесные связи с математикой, причем первые оказывают сильное влияние на последнюю.

Принципиальным отличием вступительного экзамена по информатике является не повышенная сложность предлагаемых заданий, а их комплексность, то есть направленность не на проверку знаний по какой-нибудь одной теме, а на проверку свободы обращения со всем материалом, изученным в курсе школьной информатики. Информатика вместе с математикой закладывает в образование опорный треугольник главных проявлений человеческого интеллекта: способность к обучению, рассуждению и действию.

Формирование алгоритмического мышления как одной из важнейших составляющих информационной культуры всегда было важной задачей курса информатики.

Технология решения задач на компьютере — это не только и не столько кодирование программы и получение загрузочного модуля, но, самое главное, проектирование математической модели, составление алгоритма и тестирование отлаженной программы. В этой книге детально описано, как надо программировать: как разрабатывать программу, как ее кодировать, как тестировать, какие средства выбирать.

Без программирования развитие алгоритмического мышления практически невозможно по причине отсутствия компьютерного эксперимента, только который может проверить работоспособность алгоритма.

Реализация алгоритмических конструкций, рассмотренных в книге, демонстрируется на классическом языке программирования высокого уровня Паскаль, который, несмотря на появление все новых технологий, и на сегодняшний день остается одним из самых удобных средств изучения программирования.

Определим минимальный базовый набор знаний, умений и навыков, необходимый абитуриенту для успешной сдачи экзамена по информатике при поступлении в КГУ.

Абитуриент должен быть хорошо осведомлен о предмете информатики, знать назначение, технические и потребительские характеристики устройств персонального компьютера, состав его программного обеспечения, разбираться в представлении числовой, графической и звуковой информации в компьютере, владеть хотя бы на минимальном уровне каким-нибудь языком программирования.

Абитуриент должен иметь понятие об основах алгоритмизации, уметь описывать алгоритмы и программировать вычислительные процессы.

Не следует также забывать, что хороший результат на вступительных экзаменах не самоцель. Качественная подготовка по информатике — залог дальнейшего успешного обучения в университете на прикладных специальностях.

Представленный учебный материал прошел успешную проверку на легкость усвоения школьниками и студентами.

Авторы выражают искреннюю благодарность доцентам университета С. А. Григорьеву и Н. М. Кащенко за интерес и посильный вклад в процесс разработки и организации учебного материала по проведению вступительного экзамена по информатике.

Часть I. КОМПЬЮТЕР

Глава 1. Информация

§ 1. Числа

1.1. Число

Число — основное понятие математики, которое обычно означает либо количество, размер, вес и т. д., либо порядковый номер, расположение в последовательности, код, шифр и т. д.

В этой главе мы будем иметь дело с множеством целых неотрицательных чисел, которое начинается с нуля и продолжается до бесконечности:

0, 1, 2, 3, 4, …

В информатике эти числа, начинающиеся с нуля, называются ***натуральными*** (но в математике эти числа представляют *расширенное* множество натуральных чисел, т. е. натуральные числа, дополненные нулем).

1.2. Цифра

Для представления и записи чисел используют специальные графические знаки — цифры. Например, число 256 состоит из трех цифр 2, 5 и 6, число 16 состоит из двух цифр 1и 6, а число 0 — из одной цифры 0.

Цифра — условный знак для обозначения чисел. Числа записываются при помощи цифр. ***Цифра*** в узком смысле — один из 10 знаков десятичной системы счисления

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

1.3. Система счисления

Система счисления, или просто счисление, или нумерация,— набор конкретных знаков-цифр вместе с системой приемов записи, которая представляет числа этими цифрами.

Различные системы счисления могут отличаться друг от друга по следующим признакам:

1) разное *начертание* цифр, которые обозначают одни и те же числа;

2) разные *способы записи* чисел цифрами;

3) разное *количество* цифр.

Например, восточные арабы до сих пор используют ту же самую систему счисления, что и в большинстве стран, но *начертание* цифр у них иное.

По способу записи чисел цифрами системы счисления бывают позиционные и непозиционные.

1.4. Непозиционная система счисления

Непозиционная система счисления — это такая система счисления, что в записи числа каждая цифра имеет всегда одно и то же значение, т. е. ее «вес» *не зависит* от местоположения в числе.

Римская система счисления является непозиционной.

Например, число I в римской системе означает один, число II означает 1 + 1, т. е. два, а число III — 1 + 1 + 1 = 3.

1.5. Позиционная система счисления

Позиционная система счисления характеризуется тем, что значение знака-цифры, «вес» цифры *зависит* от ее расположения в записи числа.

Например, число 1 в обычной десятичной системе счисления означает один. В числе 11 первая цифра справа означает 1, а вторая цифра справа — уже 10, поэтому число 11 означает 1 + 10, т. е. одиннадцать. Также число 111 = 100 + 10 + 1.

Основание системы счисления — это *количество* цифр *позиционной* системы счисления. Позиционные системы отличаются друг от друга своим количеством цифр, и поэтому именуется по своему основанию.

Например, десятичная система счисления, двоичная система.

§ 2. Обычные системы счисления

2.1. Использование римских цифр

Римская система счисления — счисление древних римлян, используемое в современной цивилизации (см. прил. 1).

В русском языке это счисление используется для написания:

1) века;
2) порядкового числительного;
3) месяца при указании даты и, очень редко:
4) года н. э. (нашей эры).

2.2. Семь римских цифр

Римская система счисления имеет свое собственное оригинальное начертание цифр. В частности, в этой системе отсутствует нуль.

Римская система основана на употреблении семи особых знаков — ***римских цифр***, которые делятся на четыре знака ***десятичных разрядов***

I = 1, X = 10, C = 100, M = 1000

и три знака ***половин десятичных разрядов***

V = 5, L = 50, D = 500.

2.3. Запись римскими цифрами

Натуральные числа, т. е. целые положительные числа (без нуля), можно записывать при помощи повторения римских цифр, используя три следующие правила.

1. ***Правило сложения***: если все цифры в числе по значению не возрастают, если считать слева направо, то они *складываются*.

Например:

II = 2, VI = 6, XI = 11 — правильно, IV = 6, XL = 60 — неправильно.

2. ***Правило вычитания***: 1) сначала во всех парах, где меньшая цифра стоит перед большей, *вычитается* меньшая цифра из большей; 2) затем полученные результаты вместе с оставшимися цифрами подпадают под принцип сложения и складываются.

Например:

IV = 4, XIV = 14, XXIX = 29 — правильно, IVX = 6, IXX = 1 — неправильно.

3. ***Правило ограничения***: 1) число записывается слева направо максимально возможными цифрами; 2) но четыре одинаковых *десятичных* знака подряд заменяются этим десятичным и следующим половинным; 3) но если при этой замене *этот десятичный* знак оказывается между двумя одинаковыми половинными, то эти три знака заменяются *этим десятичным* и следующим десятичным (т. е. два половинных знака заменяются равноценным десятичным).

Например:

4 = IV, а не IIII; 9 = IX, а не VIIII или VIV; 19 = XIX, а не XVIIII или XVIV.

В качестве примера выпишем все единицы, десятки и сотни, записанные в римской системе (в прил. 1 приведены все римские числа от 1 до 10).

Таблица 1. Единицы, десятки и сотни, записанные римскими цифрами

I	II	III	IV	V	VI	VII	VIII	IX	X
X	XX	XXX	XL	L	LX	LXX	LXXX	XC	C
C	CC	CCC	CD	D	DC	DCC	DCCC	CM	M

2.4. Десятичная система счисления

Десятичная система счисления — это позиционная система счисления, состоящая из 10 разных цифр и изучаемая в школе:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Здесь значение цифры *зависит* от ее положения в записи числа. Например, если цифра 1 стоит в числе на первом месте справа, то она значит *один*, если на 2-м месте справа, то *десять*, на 3-м месте справа — *сто*, и т. д. Так, в числе 512 пять сотен, один десяток и две единицы.

2.5. Количество цифр в числе

Однозначное число записывается 1 цифрой; количество таких чисел совпадает с количеством цифр: 1-значных чисел всего 10: 0, 1, 2, …, 9. ***Двузначное число*** записывается 2 цифрами, ***трехзначное число*** — 3 и т. д.

Заметим, что однозначные числа легко превратить в двузначные без изменения их значения, записав их в виде 00, 01, 02, …, 09 — нули в начале числа не влияют на величину числа. Также однозначные и двузначные числа можно превратить в трехзначные и т. д.

Подсчитаем количество однозначных, двузначных и т. д. до десятизначных чисел и сведем результаты в таблицу.

Таблица 2. Количество однозначных, двузначных и т. д. чисел

Количество цифр в числе	Количество чисел
1	$10 = 10^1$
2	$100 = 10^2$
3	$1000 = 10^3$
4	$10\,000 = 10^4$
5	$100\,000 = 10^5$
6	$1\,000\,000 = 10^6$
7	$10\,000\,000 = 10^7$
8	$100\,000\,000 = 10^8$
9	$1\,000\,000\,000 = 10^9$
10	$10\,000\,000\,000 = 10^{10}$

2.6. Код

Код — это *правило* отображения одного набора объектов или знаков в другой набор знаков без потери информации. Чтобы избежать потерь информации, это отображение должно быть таким, чтобы можно было всегда однозначно возвратиться к прежнему набору объектов или знаков.

Например, любую информацию можно передать русским языком с помощью 33 букв русского алфавита и добавочных знаков препинания.

Кодирование — это *представление*, моделирование одного набора знаков другим с помощью кода. ***Кодовая таблица*** — это соответствие между набором знаков и их кодами, обычно разными числами.

Табл. 2 говорит, что однозначными десятичными числами можно закодировать 10 предметов, приписав каждому предмету одно из 10 однозначных чисел. Двузначными десятичными числами — 100 предметов, и т. д.

2.7. ASCII

Самая распространенная и универсальная компьютерная кодовая таблица ***ASCII (аски-коды)*** (American Standard Code for Information Interchange) приведена в табл. 3.

Таблица 3. ASCII (аски-коды)

Символ / Код					
32	0 48	@ 64	P 80	` 96	p 112
! 33	1 49	A 65	Q 81	a 97	q 113
" 34	2 50	B 66	R 82	b 98	r 114
# 35	3 51	C 67	S 83	c 99	s 115
$ 36	4 52	D 68	T 84	d 100	t 116
% 37	5 53	E 69	U 85	e 101	u 117
& 38	6 54	F 70	V 86	f 102	v 118
' 39	7 55	G 71	W 87	g 103	w 119
(40	8 56	H 72	X 88	h 104	x 120
) 41	9 57	I 73	Y 89	i 105	y 121
* 42	: 58	J 74	Z 90	j 106	z 122
+ 43	; 59	K 75	[91	k 107	{ 123
, 44	< 60	L 76	\ 92	l 108	| 124
- 45	= 61	M 77	] 93	m 109	} 125
. 46	> 62	N 78	^ 94	n 110	~ 126
/ 47	? 63	O 79	_ 95	o 111	□ 127

§ 3. Двоичная система счисления

3.1. Двоичная система счисления

Двоичная система счисления — это позиционная система счисления, состоящая только из двух цифр:

0 и 1.

В компьютерах используется именно эта система счисления из=за своей простоты. Простота выполнения операций в двоичной системе счисления связана с двумя обстоятельствами:

1) простотой аппаратной реализации: 1 — есть сигнал, 0 — нет сигнала;

2) самое сложное действие таблицы умножения — это $1_2 \times 1_2 = 1_2$, а таблицы сложения — $1_2 + 1_2 = 10_2$.

Почему в двоичной системе при сложении двух единиц счисления получается десять? Эта ситуация аналогична той, когда в десятичной системе к девяти прибавляется один: $9_{10} + 1_{10} = 10_{10}$. На девятке цифры десятичной системы заканчиваются, и затем следует наименьшее двузначное число десять 10_{10}. В двоичной системе цифры заканчиваются на единице, и после нее идет наименьшее двузначное число десять 10_2.

Двойка внизу в виде нижнего индекса означает, что числа записаны в двоичной системе. При записи чисел в разных позиционных системах счисления основание системы записывается в виде нижнего индекса. Этот индекс всегда записывается только в *десятичной* системе счисления.

3.2. Таблицы умножения и сложения

Пользуясь описанными выше действиями, можно записать таблицы умножения и сложения для двоичной системы (табл. 4 и 5). При этом таблица сложения оказывается сложнее таблицы умножения.

Таблица 4. Таблицы умножения двоичных чисел

×	0	1
0	**0**	**0**
1	**0**	**1**

Таблица 5. Таблица сложения двоичных чисел

+	0	1
0	**0**	**1**
1	**1**	**10_2**

3.3. Натуральные двоичные числа

Выпишем первые натуральные двоичные числа от 0 до 16. Цифровую запись следующего числа можно получить, вспомнив ***основное свойство натуральных чисел***: следующее число больше предыдущего на 1.

Поэтому для получения следующего двоичного числа после 1_2 прибавим к 1_2 число 1_2, получим $1_2 + 1_2 = 10_2$, т. е. «десять». Отсюда имеем: $2_{10} = 1_2 + 1_2 = 10_2$ (см. рис. 1).

Также столбиком посчитаем следующее двоичное число, т. е. к 11_2 прибавим 1_2 Итак, $4_{10} = 100_2$ (см. рис. 1)

$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 10_2 \end{array} \qquad \begin{array}{r} 10_2 \\ +\ 1 \\ \hline 11_2 \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ 1} \\ 11_2 \\ +\ \ 1 \\ \hline 100_2 \end{array}$$

Рис. 1. Получение первых двоичных чисел

Продолжая аналогичным образом, заполним всю таблицу.

Таблица 6. Первые двоичные натуральные числа от 0 до 16

Десятичное число	Двоичное число
0	0
1	1
2_{10}	10_2
3_{10}	11_2
4_{10}	100_2
5_{10}	101_2
6_{10}	110_2
7_{10}	111_2
8_{10}	1000_2
9_{10}	1001_2
10_{10}	1010_2
11_{10}	1011_2
12_{10}	1100_2
13_{10}	1101_2
14_{10}	1110_2
15_{10}	1111_2
16_{10}	10000_2

3.4. Количество цифр в числе

Рассмотрим внимательно табл. 5. В ней имеется:

2 однозначных двоичных числа 0 и 1;

$4 = 2^2$ двузначных двоичных числа: 00, 01, 10_2 и 11_2;

$8 = 2^3$ трехзначных двоичных чисел от 000 до 111_2

и $16 = 2^4$ четырехзначных двоичных чисел от 0000 до 1111_2.

Рассуждая по аналогии и учитывая подобный опыт подсчета количества таких чисел для десятичной системы, получим следующую таблицу.

Таблица 7. Количество однозначных, двузначных и т. д. двоичных чисел

Количество цифр в двоичном числе	Количество двоичных чисел
1	**$2 = 2^1$**
2	**$4 = 2^2$**
3	**$8 = 2^3$**
4	**$16 = 2^4$**
5	**$32 = 2^5$**
6	**$64 = 2^6$**
7	**$128 = 2^7$**
8	**$256 = 2^8$**
9	**$512 = 2^9$**
10	**$1024 = 2^{10}$**

Из этой таблицы следует, что однозначными двоичными числами можно закодировать только два объекта, двузначными — четыре объекта, трехзначными — восемь объектов и т. д.

3.5. Перевод чисел из двоичной системы в десятичную

Установим простую, но и одностороннюю связь между одним и тем же числом, записанным одновременно в десятичной и двоичной системах.

Перевести любое двоичное число в десятичное можно по формуле

$$\langle a_n \ldots a_3 a_2 a_1 \rangle_2 = \langle a_1 + a_2 \cdot 2 + a_3 \cdot 2^2 + \ldots + a_n \cdot 2^{n-1} \rangle_{10},$$

Например:

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

$$1110_2 = 0_{10} + 1_{10} \cdot 2_{10} + 1_{10} \cdot 4_{10} + 1_{10} \cdot 8_{10} = 2_{10} + 4_{10} + 8_{10} = 14_{10}.$$

$$101010_2 = 2_{10} + 8_{10} + 32_{10} = 42_{10}.$$

3.6. Бит

Бит — минимальная единица количества информации, равная одному двоичному разряду. Его можно представить как выбор ответа «да» или «нет» на поставленный вопрос. Электронным представлением бита на компьютере является ситуация «есть сигнал/нет сигнала». В математических науках и информатике обычно «да» обозначается цифрой 1, «нет» — цифрой 0. Одним битом можно закодировать два объекта.

3.7. Байт

Бит как единица информации слишком мала, поэтому постоянно используется другая более распространенная единица количества информации, производная от бита — байт.

Байт — наименьшая единица памяти компьютера, равная 8 битам, или 8-значному двоичному числу:

1 байт = 8 бит.

Одним байтом можно закодировать 256 объектов, приписав каждому из 256 объектов одно из 256 8-значных двоичных чисел. Поэтому запомните еще одно число, постоянно встречающееся в информатике: $\mathbf{256 = 2^8}$.

3.8. Производные единицы от байта

Работая с информацией на современных компьютерах, следует знать следующие единицы, производные от байта, при составлении которых используется третье замечательное число $\mathbf{1024 = 2^{10}}$.

1 килобайт **= 1 Кб = 1 К = 1024 байта.**

1 мегабайт **= 1 Мб = 1 М = 1024 Кб.**

1 гигабайт **= 1 Гб = 1 Г = 1024 Мб.**

1 терабайт **= 1 Тб = 1 Т = 1024 Гб.**

Заполним таблицу с коэффициентами перевода производных единиц от байта друг в друга. Например, 1 Мб = 2^{10} Кб, 1 Кб = 2^{-10} Мб.

Таблица 8. Коэффициенты переводы производных единиц от байта

	Байт	Килобайт	Мегабайт	Гигабайт	Терабайт
б	1	2^{-10}	2^{-20}	2^{-30}	2^{-40}
Кб	2^{10}	1	2^{-10}	2^{-20}	2^{-30}
Мб	2^{20}	2^{10}	1	2^{-10}	2^{-20}
Гб	2^{30}	2^{20}	2^{10}	1	2^{-10}
Тб	2^{40}	2^{30}	2^{20}	2^{10}	1

Упражнения

1. Запишите римскими цифрами числа от 1 до 100.

2. Запишите римскими цифрами тремя разными неправильными (нарушая третье правило записи римских цифр) способами числа 49 и 99.

3*. Найдите максимальное число, которое можно записать римскими цифрами.

4. В городе может быть построено не более миллиона строений, каждому присваивается инвентарный номер. Какова его минимальная длина?

5. В стране немногим более 100 миллионов человек. Каждому человеку присваивается индивидуальный номер. Какова его минимальная длина?

6. Запишите в двоичной системе числа от 1 до 32.

7. К одной телефонной станции подключено 100 номеров, к другой — 1000. Двоичными числами какой минимальной длины они кодируются?

8*. Программе доступны 5000 ячеек компьютерной памяти. Двоичным числом какой длины можно их закодировать?

9. Переведите числа 100_2 и 111100_2 в десятичную систему счисления.

10. Переведите числа 20_{10} и 30_{10} в двоичную систему счисления.

11*. Выясните алгоритм перевода чисел из десятичной системы счисления в двоичную. Переведите числа 100_{10} и 200_{10} в двоичную систему счисления.

12*. Выясните, какие цифры входят в шестнадцатеричную систему счисления. Выпишите все шестнадцатеричные числа от 0 до 32.

13. Пересчитайте в мегабайты: 10240 Кб, 1024000 Кб, 10 Гб, 1000 Гб.

Глава 2. Аппаратура

§ 4. Процессор и память

4.1. Процессор

Процессор, или ЦПУ (CPU) — это «мозг» компьютера, большая интегральная микросхема, полупроводниковый кристалл, или просто камень. Процессор выполняет арифметические операции с двоичными числами.

Главный параметр процессора — ***частота*** — является основной характеристикой быстродействия компьютера. Величина частоты примерно соответствует количеству арифметических операций, выполняемых в секунду. Частота процессоров измеряется в единицах частоты — *герцах* — и ее производных. Современные процессоры ПК имеют частоты 1—2 гигагерца (Гг).

Серия процессора также существенно влияет на мощность компьютера: при переходе на следующую серию увеличивается скорость обмена данными между процессором и оперативной памятью. Процессор ПК ИБМ вначале был серии 86, затем 286 — «Двойкой», 386 — «Тройкой» и 486 — «Четверкой». Потом пошла серия Пентиумов: просто Пентиум, Пентиум II («Двойка»), Пентиум III («Тройка») и сегодняшний Пентиум IV («Четверка»).

4.2. Оперативная память

С процессором непосредственно, функционально (самый быстрый обмен) и конструктивно (находятся на одной плате), связана ***оперативная, или временная, память (random access memory, RAM)***. Это память произвольного доступа (см. ее английское название).

Объем памяти современного ПК 128 Мб, 256 Мб или 512 Мб.

В оперативной памяти компьютер хранит данные и программы, которые выполняет процессор. Эти программы и обрабатывают эти данные.

Однако информация, которую компьютер записывает во временную память, исчезает при его выключении.

4.3. Магнитная память

Для постоянного хранения информации используется ***постоянная, или магнитная, память*** компьютера, находящаяся в системном блоке в виде отдельного устройства. Это устройство называется ***жесткий диск, или винчестер (hard disk drive, HDD)***. Информация, записанная на магнитной поверхности винчестера, хранится и после выключения компьютера.

Объем памяти современного винчестера составляет 20—80 Гб.

Для переноса информации между компьютерами служит ***гибкий магнитный диск, или дискета, или флоппи,***— разновидность постоянной магнитной памяти. Современные дискеты имеют объем 1,44 Мб и размер 3,5 дюйма. В системном блоке находится устройство для работы с дискетами — ***дисковод гибких дисков, или флоппи-драйв***.

4.4. Сектор

Информация на магнитных дисках записывается в виде очень узких концентрических колец, расположенных очень близко друг к другу. Эти узкие кольца разбиваются на маленькие ***сектора*** — минимальные единицы физической, аппаратной записи. Длина сектора строго фиксирована и обычно составляет 512 байт.

На рис. 2 схематически изображена разбивка диска на секторы.

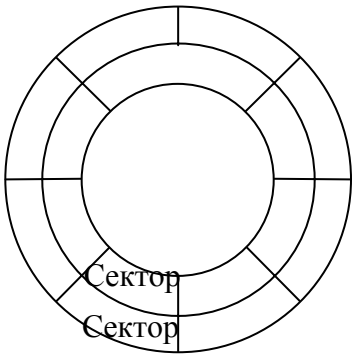


Рис. 2. Схематическое изображение секторов на диске

4.5. Кластер

В свою очередь сектора объединяются в последовательные цепочки — ***кластеры*** — минимальные единицы, кванты логической, программной записи на магнитные диски. Длина кластера тоже фиксирована на одном диске, но эта величина зависит от его объема.

При записи информации на магнитный носитель место под нее выделяется покластерно: сначала информация записывается на один свободный кластер, если его не хватает — то выделяется второй, затем третий и т. д. до полной записи данных на диск.

4.6. Компакт-диск, его виды

Компакт-диски, или компакты (CD, сиди) — это немагнитные оптические хранители информации. Они также представляют постоянную компьютерную память.

Информация на компакте записывается в виде обычно одной длинной спиральной дорожки с очень тесными витками, как на грампластинке.

Объем компакта обычно составляет 650 Мб.

Для проигрывания компакта компьютером в системном блоке должно находиться ***устройство для чтения компактов, или компакт-дисковод, или CD-ROM (сиди-ром) драйв, или CD-ROM***.

Устройства ***CD-writer, или CD-recorder***, позволяют записывать компакты. Запись осуществляется специальные на ***пустые компакты, или болванки***.

На ***аудио-компакте*** записана музыка, которую можно проигрывать и на компьютере, и на обычном лазерном проигрывателе.

Видео-компакт позволяет при помощи специальной программы смотреть на компьютере оцифрованный видеофильм, на нем записанный.

Компьютерный компакт содержит записанные на нем файлы — компьютерную информацию.

4.7. Прямой и последовательный доступ

Вся описанная память являются ***памятью прямого, или произвольного, доступа***. При работе с такой памятью компьютер обращается непосредственно сразу к нужной единице информации в памяти (на магнитных дисках это кластеры или секторы).

Память последовательного доступа представляют накопители на магнитной ленте. Устройство, работающее с магнитной лентой, вынуждено прокручивать, перематывать все предыдущие участки магнитной ленты, чтобы добраться до нужных данных.

§ 5. Основные компоненты персонального компьютера

5.1. Системный блок

Главная часть современного персонального компьютера (ПК) — ***системный блок***, который содержит самые главные части компьютера:

1) ***системную, или материнскую, печатную плату***;

2) ***процессор***, находящийся на этой плате и выполняющий основные вычисления компьютера, в том числе выполнение компьютерных программ;

3) ***оперативную память***, также находящуюся на системной плате, тесно связанную с процессором и хранящей код выполняемых программ.

На системной плате имеются также гнезда, или слоты, для подключения к компьютеру других устройств, находящихся вне системного блока.

На рис. 3 схематически изображена системная плата.

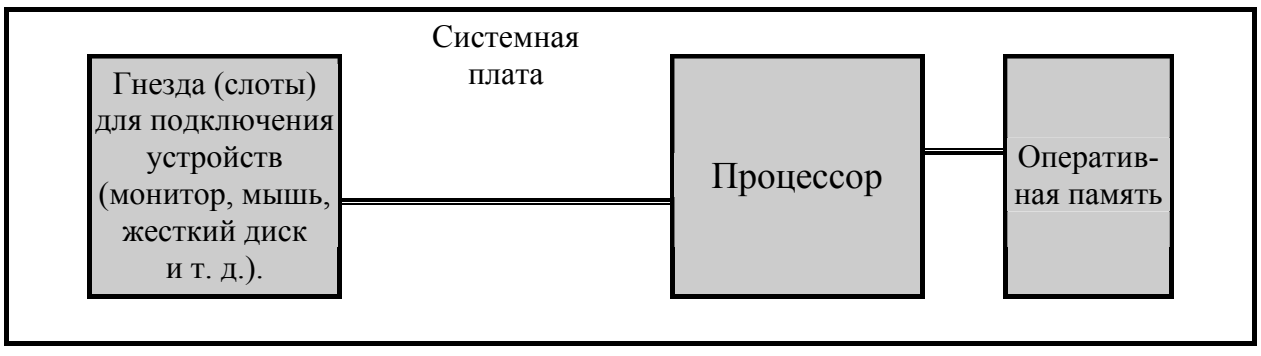


Рис. 3. Схематическое изображение системной платы

В системном блоке также находятся жесткий диск, дисковод для дискет и дисковод для компактов. Все устройства системного блока подключены к его ***блоку питания***, соединенному с электрической сетью.

5.2. Периферия

Компьютерная периферия, или просто периферия,— это все компьютерные устройства, не входящие в состав системного блока.

Устройство ввода — устройство, позволяющее вводить данные в компьютер или управлять им. Устройствами ввода являются клавиатура и мышь, которые управляют компьютером, а также сканер, микрофон и др.

Устройство вывода выводит информацию из компьютера, в том числе и для чтения человеком. Они мощнее устройств ввода, ведь компьютер больше отдает, чем получает. Это монитор, принтер, звуковые колонки и др.

Сигнальный кабель — кабель, по которому компьютерные устройства обмениваются информацией. Системный блок соединяется сигнальными кабелями со всей периферией, в том числе с клавиатурой и мышью (рис. 4).

С другими компьютерами компьютер соединяет *сигнальный* ***сетевой кабель*** (рис. 4) Этот кабель называется сетевым потому, что при связывании между собой компьютеры образуют ***компьютерная сеть, или просто сеть***.

С питанием, т. е. с электрической сетью, системный блок и вообще любая аппаратура соединяется ***силовым кабелем*** (рис. 4).

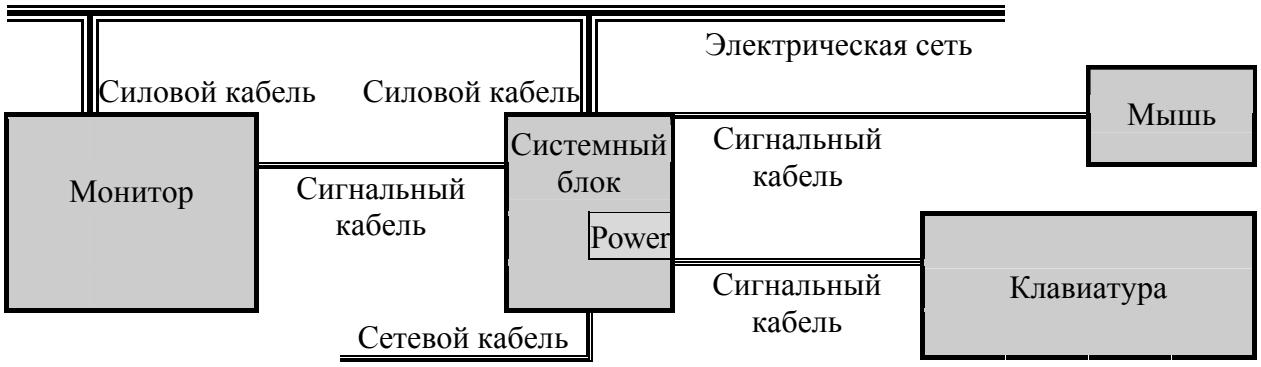


Рис. 4. Схематическое изображение системного блока и периферии

5.3. Клавиатура и ASCII

Клавиатура имеет более 100 клавиш, служащими для ввода текстов и управления компьютером. Клавиатуры в России — американские с нарисованными символами русского алфавита. Клавиатура может иметь несколько языковых ***раскладок***, т. е. может использовать свои клавиши для ввода букв разных алфавитов: русского, английского и т. п.

Рассмотрим ASCII (см. гл. 1, § 2, п. 7). Они состоят из символов следующих трех групп.

1. Прописных (больших) и строчных (маленьких) букв современного латинского алфавита, содержащего 26 символов.

2. 10 цифр.

3. 33 знаков препинания и специальных знаков.

Всего получаем $26 \times 2 + 10 + 33 = 95$ символов. В кодовых таблицах они кодируются числами от 32 до 126 включительно: $126 - 31 = 95$ (см. табл. 3).

На клавиатуре имеется 47 ***алфавитно-цифровых клавиш*** — буквы, цифры, знаки препинания и ***специальные символы***. На каждой в режиме английского языка набирается по два символа, и вместе с клавишей пробела получаем эти же 95 символов: английский язык — это язык ASCII.

5.4. Мышь

Мышь, или манипулятор «мышь» — это устройство для управления компьютером и ввода данных. На экране монитора мыши соответствует ***указатель мыши***, движение которого *по экрану* управляется движением мыши *по коврику*. Компьютером управляют, наводя указатель мыши на *объект на экране* и нажимая при этом различные *кнопки на мыши*.

5.5. Монитор, пиксель

Монитор, или дисплей,— устройство вывода компьютером визуальных данных. На утомляемость глаз влияет ***частота обновления экрана***, т е. количество кадров в секунду. Минимальная приемлемая частота 85 Гц.

Изображение на экране монитора состоит из цветных пикселей. ***Пиксель*** — это единица цвета монитора, точка-зерно, состоящая из точек трех цветов, в сумме дающих цвет пикселя.

5.6. Разрешение

Разрешение аппаратуры — количество точек на единицу длины, измеряемое по вертикали или горизонтали. В качестве длины всегда выступает ***американский дюйм***, равный примерно 2,5 см. Обычно разрешение аппаратуры по обоим направлениям, горизонтали и вертикали, одинаковое, поэтому для обозначения разрешения используется только одно число.

Точка по-английски «dot», дюйм — «inch», точек на дюйм — «dot per inch», поэтому выражение «точек на дюйм» сокращается как «***dpi***».

Итак, ***разрешение*** — это характеристика картинки на плоскости. Схематически разрешение, например, 10 dpi для картинки 40×30 пикселей можно изобразить так, как показано на рис. 5.

Разрешение мониторов составляет 75—100 точек-пикселей на дюйм.

5.7. Основные цвета монитора

Пиксель белого цвета состоит из 3 точек 3 цветов максимальной яркости:

красного (red), ***синего (blue)*** и ***зеленого (green)***

— это ***основные цвета монитора*** (зеленый цвет на самом деле выглядит скорее светло-салатовым на фоне красного и синего цветов).

Каждый цвет обладать ***яркостью*** — силой свечения цвета. Если все три цвета имеет минимальную яркость, т. е. ее отсутствие, то на экране получается ***черный цвет***, максимальную — ***ярко-белый***. Чтобы получить в чистом виде один из трех основных цветов, нужно погасить остальные два.

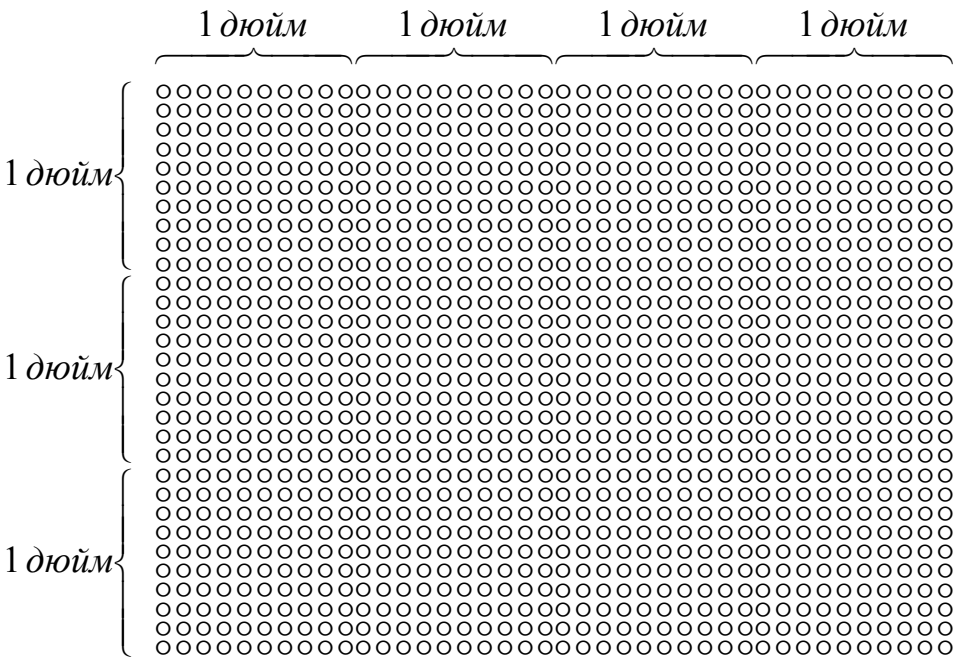


Рис. 5. Схематическое изображение картинки 40 × 30 пикселей с разрешением 10 dpi

5.8. Смешивание цветов монитора

Пары трех основных цветов максимальной яркости выглядят для человеческого глаза как следующие цвета (см. рис. 6):

фиолетовый (пурпурный, лиловый, магента, magenta) = красный + синий;

голубой (синезеленый, бирюзовый, циан, cyan) = синий + зеленый;

желтый (yellow) = красный + зеленый.

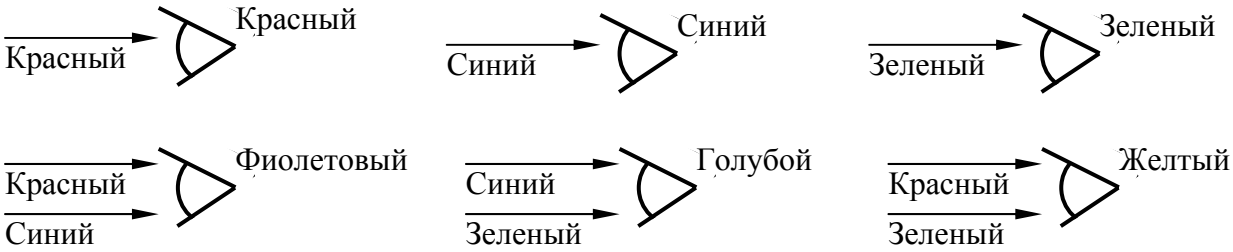


Рис. 6. Схематическое изображение восприятия цвета глазом человека

5.9. Цветовая схема RGB

Для наглядного представления наложения цветов на мониторе рисуют внутри квадрата три пересекающихся круга, соответствующих трем основным цветам. Эти круги, пересекаясь, разбивают квадрат на 8 частей, каждая из которых соответствует одному из 8 цветов. Вне кругов, когда нет ни одного луча света, находится черный цвет. Все три круга пересекаются по белому цвету. Имеем ***цветовую схему монитора RGB***, названную так по первым буквам английских названий трех основных цветов.

Рис. 7. Цветовая схема RGB

§ 6. Сканер, принтер

6.1. Сканер, его виды

Сканер — устройство оптического ввода информации, ее ***сканирования***, фотографирования, служащее для копирования картинок окружающей действительности в компьютер. При работе сканера в компьютере создается графический объект — копия реальной картинки.

Планшетный сканер размещается на столе. При сканировании считывающее устройство перемещается вдоль планшета. При этом на стекло планшета кладется лист носителя или книга. При наличии дополнительной приставки возможно автоподача листов с изображениями.

Рулонный, или барабанный, сканер протягивает лист бумаги вдоль оптического считывателя, подобно принтеру. Только принтер печатает на бумаге, а рулонный сканер сканирует, фотографирует лист бумаги. В него можно вводить информацию с рулонного носителя или осуществлять автоподачу листов одного за другим.

Проекционный сканер снимает окружающие предметы, как фотоаппарат или телекамера. В отличие от предыдущих видов сканера в проекционном сканере нет движущихся частей. ***Цифровые камера*** и ***видеокамера*** представляют собой разновидности проекционного сканера.

6.2. Результат сканирования

Результатом цифрового сканирования является матрица из пикселей, расположенных равномерно по вертикали и горизонтали. Эта картинка имеет размеры и разрешение подобно изображению на экране монитора. Разрешение 600 пикселей на дюйм — минимальное для современного сканера.

Специальной компьютерной программой ***распознавания текстов*** можно прочитать текст, находящийся на картинке, снятой сканером. При этом информация переходит из графической формы в текстовую.

6.3. Принтер, его виды

Принтер — устройство вывода информации, используемое для печати данных на твердом носителе — бумаге или пленке.

Матричный принтер появился первым и назван в противоположность векторному устройству — ***графопостроителю, или плоттеру***. Матричный принтер переносит изображение на бумагу, печатая точки, равномерно расположенные по вертикали и горизонтали. Графопостроитель рисует изображение специальными перьями, фломастерами.

Матричный принтер печатает стальными иголками, бьющими по пишущей ленте, пропитанной типографской краской. Поэтому его более правильное название — ***игольчатый принтер***.

Затем появились ***лазерные*** принтеры, печатающие точками на твердом носителе (матричный принцип) при помощи мелкого черного порошка, который после нанесения на бумагу или пленку вплавляется в нее.

Струйные принтеры печатают краской разных цветов, которая точечными капельками (опять матричный принцип) напыляется на бумагу или другой твердый носитель, разбрызгиваясь через специальные микросопла.

Минимальное разрешение современных принтеров — также 600 dpi.

6.4. Основные цвета при печати

Способ формирования цвета на бумаге при цветной печати отличается от формирования цвета на экране монитора. Более того, эти способы в некотором смысле противоположны, а именно: белый цвет на экране получается как смешивание всех трех цветов монитора, тогда как белый цвет на белой бумаге — это результат отсутствия на ней какой-либо краски.

В самых простых цветных принтерах имеется только три цвета:

голубой (cyan), ***фиолетовый (magenta)*** и ***желтый (yellow)***.

Эти три цвета и черный цвет суть ***основные цвета цветной печати***.

6.5. Особенности цветной печати

Обычно цветные принтеры имеют черный цвет для передачи черного цвета и темных тонов. Более качественные цветные принтеры имеют еще два цвета для светлых тонов: ***светло-голубой*** и ***светло-фиолетовый***.

Печатные машины имеют много цветов. Черно-белые принтеры имеют только один цвет для печати — черный.

При печати насыщенность цвета достигается количеством печатаемых точек — ***плотностью цвета***. При отсутствии цветных точек, т. е. при нулевой плотности цвета, на белой бумаге получается белый цвет. Небольшая плотность цвета при печати дает бледные цвета. Когда же все цветные точки выводятся, т. е. при максимальной плотности цвета, то даже на самых простых принтерах получается очень темный цвет, практически черный.

6.6. Смешивание цветов при печати

Принцип смешивания красок при печати основан на поглощении цвета. Поглощаемые цвета смешиваемых красок складываются, а отражается тот цвет, который отражают все смешиваемые краски.

Голубая краска поглощает все цвета, кроме синего и зеленого, фиолетовая — все цвета, кроме красного и синего и желтая — кроме красного и зеленого. Отсюда следует, что смесь фиолетовой краски с голубой отражает синий, смесь фиолетовой с желтой отражает красный и смесь голубой с желтой — зеленый цвет (см. рис. 8):

синий (blue) = голубой + фиолетовый;

красный (red) = фиолетовый + желтый;

зеленый (green) = голубой + желтый.

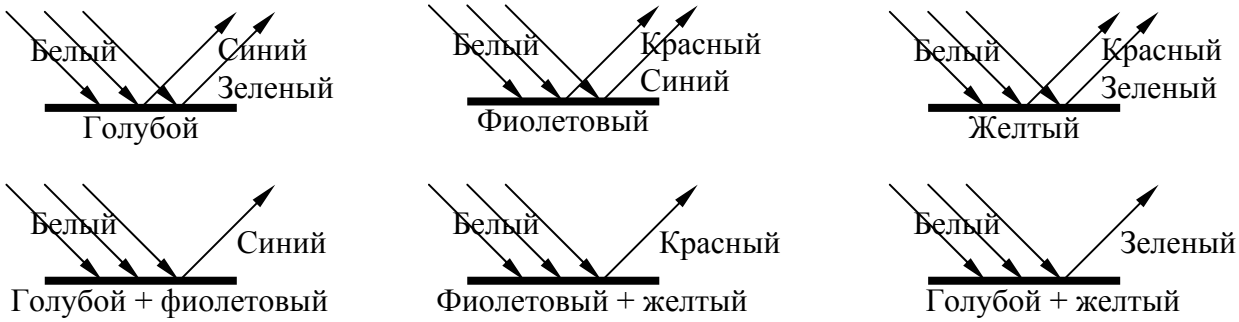


Рис. 8. Схематическое изображение отражения цвета от бумаги

6.7. Цветовая схема CMYK

Для наглядного представления наложения цветов на принтере также рисуют внутри квадрата три пересекающихся круга, соответствующих трем основным цветам. Получают ***цветовую схему цветной печати CMYK***, названную так по первым буквам английских названий трех цветов для печати и последней буквы английского названия черного цвета (рис. 9).

Рис. 9. Цветовая схема CMYK

Упражнения

1. Графическая картинка занимает на экране площадь 15×15 см^2 и содержит 360 тысяч пикселей. Найти разрешение экрана.

2. Графическая картинка имеет ширину 600 пикселей. На экране она имеет ширину 15 см, а при печати — 2,5 см. Определить разрешения экрана и принтера.

3. Размер экрана дисплея составляет 800×600 точек при глубине цвета 24 бита. Учитывая, что частота обновления экрана равна 85 Герц, вычислить количество информации, которое появляется на экране за секунду.

4. Объем дискеты составляет 1,44 Мб. Сколько файлов можно записать на дискету, если длина каждого файла равна 250 Кб?

5. Страница, набранная в современном текстовом редакторе, занимает 4 Кб памяти. Сколько 100 страничных документов может поместиться на дискете, объем которой 1,44 Мб?

6. Объем дискеты составляет 1,44 Мб. Величина сектора дискеты равна величине ее кластера и равна 512 байт. Сколько кластеров на дискете?

7. Объем сектора составляет 512 байт. Максимальное количество секторов в кластере 64. Максимальное количество кластеров на винчестере 2^{16}. Определить максимальную емкость такого винчестера.

Глава 3. Программы

§ 7. Операционные системы

7.1. Формула компьютера

Компьютер = аппаратура + программы. Аппаратура представляет собой «жесткую» часть компьютера и по-английски называется соответственно — ***hard***. Программы — это «мягкая» часть компьютера, по-английски называется ***soft***. Программы находятся в форме файлов. Итак, получаем эквивалентную формулу *компьютер = hard + soft*.

7.2. Интерфейс, его виды

Интерфейс компьютерной программы — это ее внешний вид на экране дисплея, включающий ее оформление, вид и расположение элементов управления работой этой программы.

Текстовый интерфейс состоит только из символов, каждый из которых находится в какой-то текстовой строке и столбце на экране монитора. В этом интерфейсе экран разбит 25 строками и 80 столбцами на 2000 ячеек, в каждой из которых может находиться один символ.

Графический интерфейс гораздо богаче текстового, он состоит из окошек и кнопок, изображенных на экране. В окошках выполняются программы, а кнопки управляют их выполнением. Здесь нет никаких текстовых ячеек, графическое изображение строится с точностью до пикселя.

7.3. Операционная система, ее компоненты

Программы делятся на операционные системы и прикладные программы.

Операционной системой (ОС) называется *комплект программ*, которые совместно управляют ресурсами системы и процессами, использующими эти ресурсы. Выполнение любой программы на компьютере происходит под управлением ОС.

Программы, из которых состоит ОС, делятся на следующие три категории.

1. ***Ядро ОС***, выполняющее основные функции ОС (в основном загрузку ее компонентов и поддержку выполнения компьютерных программ, в том числе и этих компонентов).

2. ***Программу управления файлами и директориями***, служащую для классификации и просмотра информации, с которой имеет дело пользователь на компьютере.

3. ***Драйверы***, которые позволяют ОС работать с аппаратурой: периферийными устройствами (монитор, клавиатура, мышь, принтеры и т. д.) и устройствами, входящими в состав системного блока (видеокарта, жесткий диск и т. д.). Без драйверов невозможно функционирование никаких компьютерных устройств.

7.4. Виды операционных систем

Наиболее распространены в России ИБМ-совместимые ПК с ОС от фирмы Микрософт серии Windows, имеющие графический оконный интерфейс. ДОС — прежняя версия этой ОС с текстовым интерфейсом.

Профессионалы в Интернете широко используют и в России ОС UNIX («юникс»). Ядро ОС UNIX имеет текстовый интерфейс, причем некоторые ее версии обладают графическими оконными оболочками.

Компьютеры Mac, или Макинтош, на которых установлена еще одна ОС — от фирмы Apple — широко используются на западе в учебных заведениях и дома. Она имеет только графический интерфейс.

7.5. Утилиты

Под управлением ОС на компьютерах работают прикладные программы, которыми пользуются пользователи.

Существуют также программы, занимающие промежуточное положение между прикладными программами и ОС — это ***утилиты, или вспомогательные программы***. Большинство утилит поставляется вместе с ОС, также эти утилиты производят другие отдельные фирмы.

Рассмотрим два класса утилит, не входящих в состав ОС. Различные их реализации поставляются разными фирмами.

7.6. Архиватор, архив

Архиватор — программа, которая используется для сокращения объема хранимой или передаваемой информации. Архиватор по алгоритмам сжатия кодирует исходные данные, уплотняя их. Эта уплотненная информация хранится или передается по назначению, и затем при необходимости может быть полностью восстановлена в прежнем объеме.

Результатом работы архиватора является ***архив*** — файл со сжатой информацией. Можно запаковывать не только файлы одного каталога, но и целое дерево, иерархию каталогов со всеми файлами.

7.7. Антивирус, вирус

Антивирусная программа, или антивирус,— программа для борьбы с компьютерными вирусами. ***Компьютерный вирус, или вирус***,— компьютерная программа, которая не имеет своего выполняемого файла, а внедряется, самодописывается в файлы других программ. Все вирусы опасны для нормальной работы компьютера, даже и так называемые «безвредные», поскольку они все равно портят с непредсказуемыми последствиями код программ.

Чтобы вирус активизировался и заработал, он должен попасть в оперативную память компьютера как *программа*. Поэтому при копировании и передаче файла с вирусом, когда он попадает в память как пассивные *данные*, заражения новых файлов и памяти компьютера не происходит.

§ 8. Прикладные программы

8.1. Приложение

Прикладная программа, или приложение, позволяет пользователю делать то, ради чего он использует компьютер, т. е. применять компьютер в разных областях человеческой деятельности. Прикладная программа выполняется на компьютере под управлением ОС.

Прикладные программы, в свою очередь, можно разделить на два класса: 1) программы-автоматы; 2) программы-инструменты.

8.2. Программы-автоматы

Программы-автоматы — это прикладные программы, где пользователь эксплуатирует алгоритмы и данные, а также способы классификации данных и их просмотра, созданные другими людьми.

С помощью этих программ пользователь не создает новой информации. Это самые легкие в использовании программы, не требующие никаких специальных знаний, в том числе игры.

Приведем три разновидности программ-автоматов: обучающие, игры и базы знаний.

8.3. Обучающие программы

Обучающие программы помогают пользователю обучиться какой-нибудь области знания (языки, набор на клавиатуре, математика и т. д.).

Современные обучающие программы обычно являются мультимедийными, включая не только звук и работу с микрофоном, но и отрывки из видеофильмов.

8.4. Игры

Игры используются для отдыха за компьютером, спортивных соревнований, тренировки логического мышления, тренажерной тренировки определенных навыков и умений, а также обучения.

Различают следующие классы игр: логические, стратегические, квесты (бродилки), симуляторы, аркады (стрелялки).

8.5. Базы знаний

Базы знаний — самая разнообразная информация, организованная в логические структуры. Частный случай таких программ — ***экспертные системы***, которые помогают специалистам обрабатывать специальные данные и делать заключения. Эти программы легче перечислить по областям знаний: медицинские, математические, статистические и т. д.

Сайт — организация информации в пространстве Интернета, представляющая собой ряд связанных между собой страниц одной тематики.

8.6. Программы-инструменты

Программы-инструменты — это прикладные программы, с помощью которых пользователь создает новую авторскую информацию, хранящуюся в соответствующих файлах.

Программы-инструменты также делятся на два класса:

1) ***редакторы*** — программы для создания, редактирования, просмотра и изменения новой информации, за исключением компьютерных программ;

2) ***системы программирования, или языки программирования*** — программы для создания компьютерных программ.

Рассмотрим три вида редакторов.

8.7. Текстовые редакторы

Текстовые редакторы служат для создания разнообразных текстов на естественных и компьютерных языках.

Развитые текстовые редакторы с возможностями форматирования текста называются ***текстовыми процессорами***.

Мощные текстовые процессоры используются только для верстки книг и называются ***издательскими системами***.

8.8. Графические и мультимедийные редакторы

Графические редакторы обрабатывают графическую информацию, состоящую из пикселей или формул, позволяют добавлять в нее графические эффекты. Они также обрабатывают анимационную информацию, состоящую из последовательных кадров графической информации..

Мультимедийные редакторы имеют дело с полной коллекцией мультимедиа, в том числе звуком и видео. ***Звуковые редакторы*** позволяют визуально просматривать оцифрованный звук, редактировать и прослушивать его. ***Видео-редакторы*** занимаются с оцифрованным видео: осуществляют покадровый просмотр, редактирование и добавление видео-эффектов, монтаж и озвучивание видео-информации.

8.9. Редакторы баз данных

Редакторы баз данных, или системы управления базами данных (СУБД), занимаются ***базами данных (БД)***, т. е. самой разнообразной информацией, организованной в логические структуры.

Их разновидностью являются ***табличные редакторы***, которые создают и обрабатывают числовые таблицы, в которых хранится исключительно числовая информация и формулы для обработки этих чисел.

Еще одной разновидностью СУБД являются ***специальные программы***, которые легче перечислять по областям знаний: математические, статистические, бухгалтерские и т. д. Эти программы накапливают и редактируют данные в тех специальных областях знания, где они применяются.

8.10. Языки программирования

Системы программирования, или языки программирования — это прикладные программы, которые позволяют программисту создавать любые компьютерные программы.

Этими компьютерными программами являются:

1) прикладные программы, в том числе языки программирования;
2) утилиты;
3) вирусы;
4) операционные системы.

Самые распространенные языки программирования: Бейсик, Паскаль, Си.

§ 9. Файл и дерево

9.1. Файл, его содержание

Файл — это форма организации информация на компьютере. Обычно файл — это именованная область компьютерного диска, т. е. файл имеет имя и содержание. Почти все программы и все данные хранятся в файлах.

Содержанием файла является закодированный результат работы прикладной программы. По виду информации, которую хранят файлы, они делятся на выполняемые, т. е. программы, текстовые, графические, звуковые, и т. д. Объем файлов измеряется в байтах, килобайтах и мегабайтах.

9.2. Стандартное имя файла

Каждый файл имеет ***имя***, составляемое по особым правилам. Стандартное имя, которое «понимает» ОС на любом компьютере и в любой стране, отвечает стандарту 8.3. Стандарт 8.3 заключается в следующем.

1. Имя файла имеет длину от одного до 8 символов. Имя файла составляется так, чтобы передавать его содержание; например: lekcija.

2. Символы имени файла являются либо латинскими буквами, либо цифрами, либо некоторыми специальными символами, в число которых пробел ни в коем случае не входит.

3. Имя файла может иметь ***расширение*** длиной от 1 до трех символов из того же комплекта, причем, если расширение есть, оно отделяется от основного имени файла точкой. Расширение файла составляется так, чтобы отражать тип файла; например, lekcija.doc — текстовый файл.

Таким образом, максимальная длина имени файла, отвечающего стандарту 8.3, равна 12 символов: 8 символов основное имя, 1 точка, 3 символа расширение имени файла (см. рис. 10).

В *некоторых* современных русифицированных компьютерных системах имя файла может иметь большую длину и включать русские буквы.

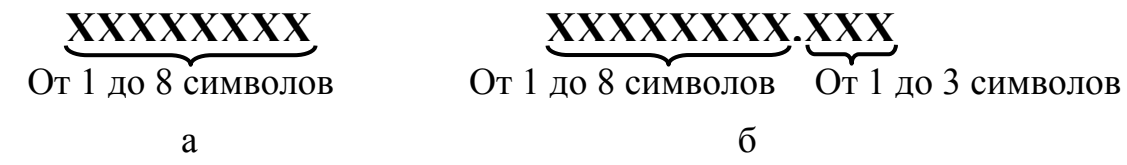


Рис. 10. Структура стандартного имени файла: а) без расширения; б) с расширением

9.3. Имена дисков

Диски на персональных компьютерах именуются отдельными латинскими буквами и дополняются двоеточием, чтобы их можно было отличить от имен файлов. Например: **A:**, **C:** или **D:**.

Обычно на ПК диски имеют следующие имена:

1) дискеты имеют имя **A:**;

2) жесткие диски имеют имя **C:**;

3) дисководы компакт-дисков имеют имя **D:**;

4) имена сетевых дисков, находящихся на другом компьютере, начинаются с имени **F:** и заканчиваются именем **Z:**.

9.4. Директория

На жестких дисках файлов очень много — десятки тысяч. Чтобы работать с таким большим количеством объектов, их необходимо классифицировать, распределить по группам, создать из них структуру. Структура файлов на дисках всегда имеет вид дерева.

Файлы объединены по своему назначению в логические группы (см. рис. 11), расположенные на дисках компьютера и имеющие свои имена. Эти группы называются ***директориями, или каталогами, или папками***, а их имена — именами директорий.

9.5. Дерево директорий, поддерево

Директории нижнего уровня, в свою очередь, объединяются в группы — директории, или ***наддиректории (надкаталоги, надпапки)*** (рис. 11).

Эти директории второго уровня снизу также объединяются в директории и т. д. до самой верхней ***корневой директории (каталога, папки)***.

Корневая директория именуются по названию диска с добавлениям символа бэкслеша \, например, **C:** или **D:** (рис. 11).

Вся эта иерархия называется ***деревом директорий (каталогов, папок)***.

Часть дерева, у которой какая-то директория является самой верхней, называется ***поддеревом***. Эта самая верхняя директория является корневой директорией этого поддерева.

На рис. 11 изображен пример дерева директорий. Прямоугольниками обозначены директории, кружками — файлы, в треугольники заключены примеры поддеревьев. Графически дерево директорий изображается в виде дерева, растущего корнем вверх и ветвями вниз, файлы — листья на дереве.

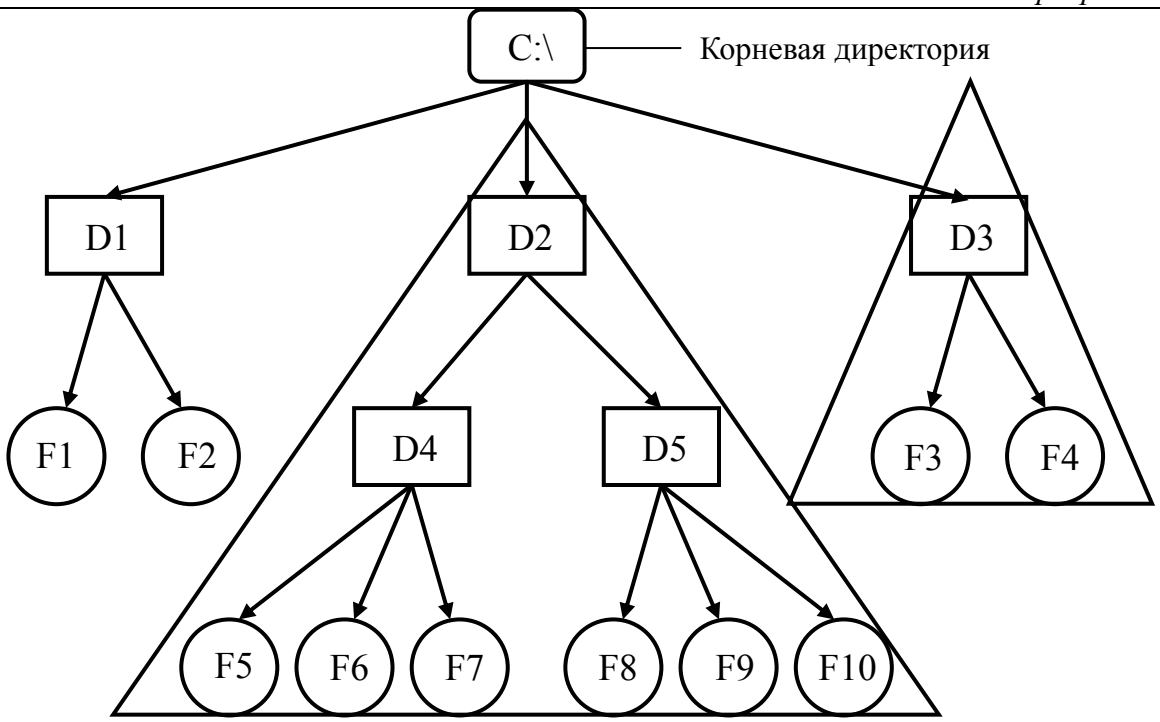


Рис. 11. Дерево директорий. Здесь кружки — файлы, прямоугольники — директории, треугольники — примеры поддеревьев

9.6. Создание дерева директорий, поддиректория

При создании дерева директорий процесс создания директорий идет сверху вниз: сначала на диске в корневой директории создаются директории, называемые ***поддиректориями (подкаталогами, подпапками)***,— это директории первого уровня.

Затем при необходимости в директориях первого уровня создаются поддиректории — директории второго уровня, и т. д. до тех пор, пока вся информация не будет структурирована, классифицирована в виде дерева.

9.7. Добавление новой информации

При появлении новой информации она добавляется:

1) либо в виде отдельного файла или файлов в какую-то уже существующую директорию;

2) либо в какой-нибудь, соответствующей по смыслу, директории создается еще одна новая поддиректория, в которую и записывается поступившая или созданная информация в виде поддерева, для которого эта новая поддиректория является корневой.

9.8. Имена и содержание директорий

Имена директорий составляются по тем же самым правилам, что и имена файлов. Обычно имена директорий не имеют расширения.

Содержанием директорий является:

1) список имен объектов (файлов и директорий), которые и составляют, как группа, данную директорию, входят в нее;

2) указатель на вышестоящую директорию, наддиректорию, которой принадлежит данная директория.

Упражнения

1. Пусть имена файлов можно составлять только из двух цифр 0 и 1. Сколько можно составить имен файлов без расширения, содержащих не более чем два символа?

2*. Тот же вопрос, что и в упр. 1, но файлы могут иметь расширение.

3. На диске **C:** находятся две директории **A** и **B** и один файл 1,doc. Нарисуйте в виде дерева все 9 вариантов их взаимного расположения. Сколько различных поддеревьев можно выделить в каждом случае?

4*. На диске **C:** находятся две директории **A** и **B** и два файла 1,doc и 2,doc. Нарисуйте в виде дерева все 27 вариантов их взаимного расположения. Сколько различных поддеревьев можно выделить в каждом случае?

5. Посчитайте количество директорий и файлов на следующих деревьях на рис. 12.

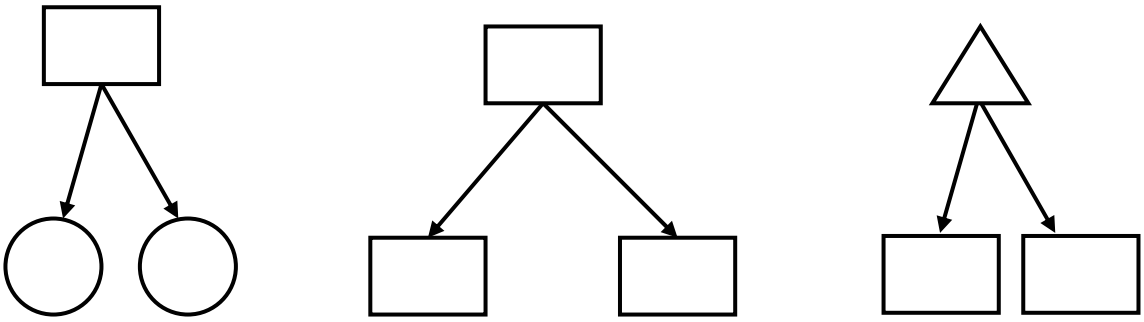


Рис. 12. Примеры деревьев директорий с файлами

6*. Какие из следующих схем на рис. 13 могут быть схемами деревьев директорий, а какие — не могут и почему?

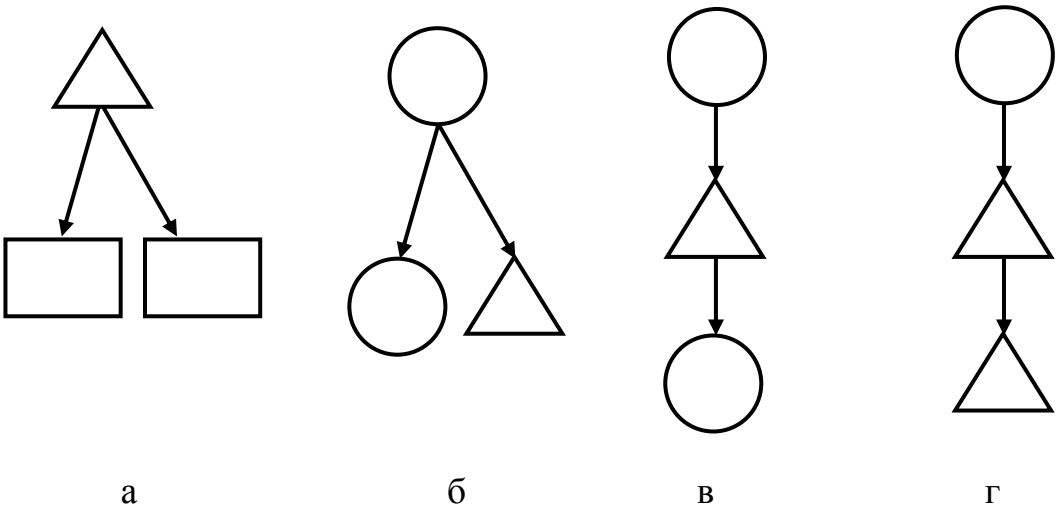


Рис. 13. Схемы, некоторые из которых могут быть,
а некоторые в принципе не могут быть деревьями директорий

Глава 4. Тексты

§ 10. Текстовые редакторы

10.1. Текстовый редактор и процессор

Текстовый редактор — это компьютерная программа, служащая для набора, сохранения, просмотра и редактирования текстов.

Текстовый процессор — это текстовый редактор, обладающий широкими возможностями ***форматирования текста*** — набора на различных языках, разнообразных формул, нот, изменения вида абзацев, начертания символов, вставки в текст рисунков, рисования, создания таблиц и т. д.

Издательская система — это мощный текстовый процессор.

10.2. Виды текстовых процессоров

Самый распространенный текстовый процессор — Word для Windows от фирмы Микрософт (Microsoft). Своим файлам он присваивает расширение .doc. Word работает по принципу ***WYSIWYG*** — что видишь на экране, то примерно и получишь при печати.

Специалисты (математики, химики, филологи и др.) работают в текстовом процессоре $\TeX$, или TeX, в котором очень быстро и качественно набираются разнообразные математические, химические и др. формулы.

10.3. Технологии текстового редактора, набор, курсор

При работе в текстовых редакторах необходимо ознакомиться и освоить системы характерных понятий и специальных приемов работы. Эти системы называются ***технологиями текстового редактора***.

Основная технология — это просто набор текста.

Набирается текст на клавиатуре. При этом на экране компьютера мигает ***курсор*** — вертикальная палочка, на которой набирается текст. Мигание курсора — это признак готовности компьютера к набору текста.

10.4. Сохранение

Сохраняется набранный текст на диске в виде файла с фиксированным расширением, зависящим от вида текста и используемого текстового процессора. В этом файле содержится закодированный по кодовой таблице текст документа и закодированное ***форматирование*** этого текста, т. е. его внешний вид.

Если при сохранении текста сохраняются только его символы в чистом виде, без какого-либо форматирования, за исключением символов конца страницы и конца абзаца, то говорят, что используется не текстовый процессор, а текстовый редактор.

10.5. Просмотр

Текст должен ***просматриваться*** в том же текстовом процессоре, в каком был набран, иначе отдельные элементы форматирования текста будут потеряны, даже если в документации написано, что текстовый процессор может работать с файлами другого текстового процессора.

С файлами, созданными текстовыми редакторами и состоящими только из ASCII, могут работать любые текстовые редакторы и процессоры. С ASCII и набором русских букв могут работать любые текстовые редакторы, «понимающие» русские буквы.

10.6. Редактирование, выделение

Мощь текстовых редакторов проявляется при ***редактировании*** текста. Операции редактирования разделяются на две группы: редактирование символов и редактирование блоков символов.

Редактирование блоков текста базируется на еще одной технологии ***выделения, или высвечивания,*** блоков текста, при которой фиксируется часть текста в виде непрерывной цепочки символов. Можно выделить только один блок. При выделении другого блока выделение предыдущего пропадает. Блоком может быть и один символ: для этого его нужно выделить.

10.7. Редактирование символов

Под ***редактированием символов*** понимаются операции, при которых пользователь имеет дело с отдельными символами текста, которые удаляются и вставляются по одному. Существуют три таких операции:

1) ***удаление*** символов;

2) ***вставка*** в текст новых символов. В этом случае редактор работает в ***режиме вставки***;

3) при одновременном выполнении удаления и вставки получается операция ***замещения*** символов. В этом случае редактор работает в ***режиме замещения***.

10.8. Редактирование блоков символов

Чтобы использовать эти операции, необходимо сначала выделить блок текста. Можно проводить четыре таких операции:

1) ***удаление*** блока текста.

Следующие операции являются специфическими для операций на блоках и работают только с ними:

2) ***копирование*** блока текста в буфер обмена;

3) при одновременном выполнении удаления и копирования получается операция ***вырезания*** блока текста;

4) ***вставка*** в массив набранного текста блока текста, предварительно скопированного в буфере обмена.

§ 11. Страница и абзац

11.1. Страница, ее части

Страница — оптимальная единица представления информации для восприятия человеком. Она состоит из следующих частей.

Страница должна иметь ***поля*** — пустые промежутки между текстом и краями носителя текста, например, листа бумаги. У страницы имеются ***верхнее***, ***нижнее***, ***левое*** и ***правое поля***. Поля отделяют ***поле для набора текста*** от краев страницы (см. рис. 14).

На верхнем и нижнем поле страницы находятся соответственно ***верхний и нижний колонтитулы***, на которых помещается вспомогательная информация, например, ***номер страницы*** (рис. 14).

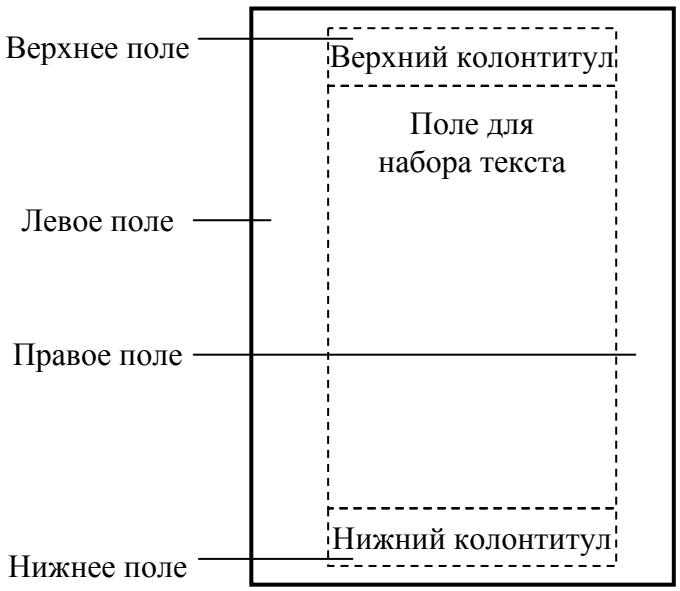


Рис. 14. Страница и ее части

11.2. Виды страниц

При просмотре на экране текст удобно просматривать ***экранами*** — страницами размерами с экран. При этом вертикальное смещение текста вверх и вниз воспринимается человеком естественно. Отметим, что горизонтального перемещение текста следует избегать.

В Интернете широко используются ***веб-страницы***, представляющие собой законченные страницы с информацией по одной теме.

Первая страница, на которой написано название документа и имя автора, называется ***титульной***. Страницы работы нумеруются, но на титульной странице номер не ставится.

Если материал состоит из именованных частей, глав, параграфов, то в нем имеется ***оглавление, или содержание***,— список этих частей с указанием номера страницы, с которой начинается каждая часть работы.

11.3. Печать страницы, формат бумаги

Текст разбивается на страницы для того, что быть в дальнейшем распечатанным на бумаге. Разбивка на страницы обычно проводится текстовым процессором автоматически согласно выставленным в нем размерам бумаги и величинам полей. Однако при необходимости всегда можно насильно задать конец страницы, набрав символ ***конца страницы***. Этот символ размещается в конце страницы и отделяет ее от следующей.

Страница распечатывается на принтере на листе бумаги (или пленке). Листы бумаги имеют разные размеры. Стандартный размер листа бумаги в России имеет ***формат А4*** и составляет 210 × 297 мм. Лист бумаги, в два раза меньший по площади, чем А4, называется ***А5***, в два раза больший — ***А3*** (см. рис. 15).

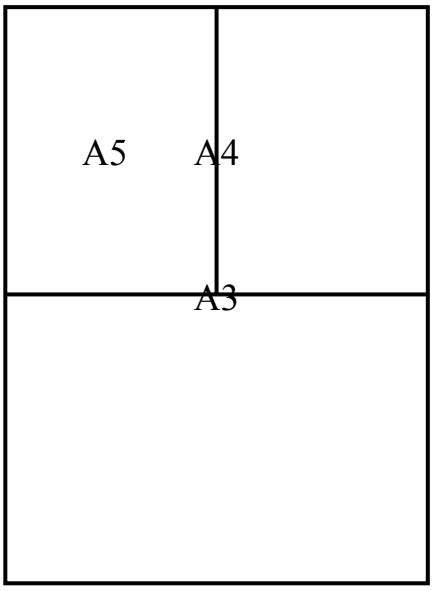


Рис. 15. Соотношение форматов листов бумаги А3, А4 и А5

11.4. Абзац

Абзацем называется законченная по смыслу часть текста. Обычно абзац состоит из нескольких предложений и занимает в тексте документа несколько строк. Абзац может занимать одну строку. Например, название глав и параграфов, стихотворные строки — отдельные абзацы.

Чтобы не слиться друг с другом при наборе текста на компьютере, абзацы при кодировании текста отделяются друг от друга специальным символом ***конца абзаца, или абзацем***. Этот символ в обычных режимах не выводится ни на экран, ни на печать. При наборе и редактировании текста абзац заканчивается нажатием клавиши <Enter>, и в текст документа вставляется специальный ***символ конца абзаца***. Этот символ записывается в конец абзаца и отделяет этот абзац от следующего.

11.5. Красная строка

Чтобы не слиться друг с другом абзацы должны визуально как-то отделяться друг от друга.

В России и некоторых странах абзацы отделяются друг от друга с помощью ***красной строки, или абзаца***,— смещения начала первой строки абзаца вправо (см. рис. 16 а)).

В США и некоторых странах абзацы отделяются друг от друга ***отбивкой*** — пустой строкой после абзаца (см. рис. 16 б)).

Видно, отделение абзацев красной строкой более экономно, чем пустой.

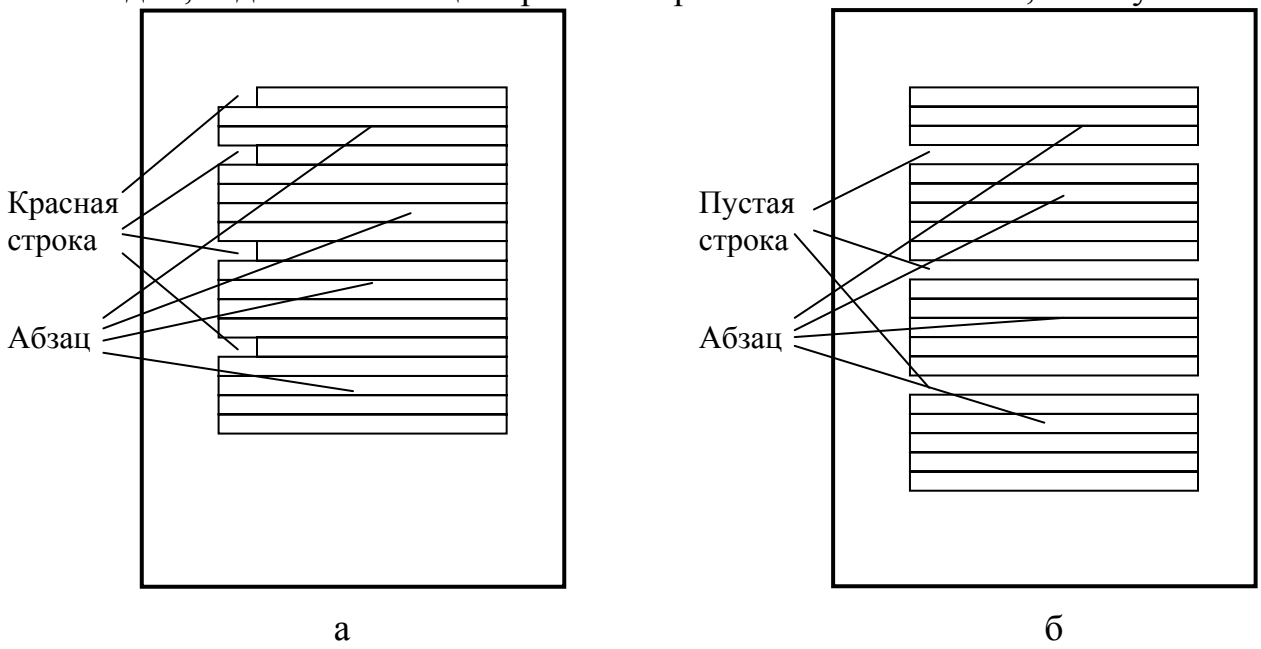


Рис. 16. Отделение абзацев: а) красной строкой в России; б) пустой строкой в США

11.6. Форматирование абзаца

Изменение внешнего вида абзаца в текстовом процессоре называется ***форматированием абзаца***. Отметим главное: форматирование абзаца в текстовых процессорах производится не добавлением пробелов или табуляторов между словами и пустых строк между абзацами, а исключительно специальными средствами форматирования абзацев текстового процессора.

Окончательный вид абзаца является комбинацией всех пяти его параметров. Абзац в основном массиве текста в русском стандарте имеет:

1) выравнивание по ширине;
2) красную строку;
3) края, совпадающие с полями страницы;
4) одинарный или полуторный интервал;
5) отсутствие интервалов перед и после абзаца.

Американский стандарт отличается от русского двумя пунктами:

3) имеет место нулевой отступ первой строки;
5) имеется интервал после абзаца.

11.7. Выравнивание строк абзаца

В абзаце можно по-разному ***выровнять строки*** относительно левого и правого краев абзаца (см. рис. 17):

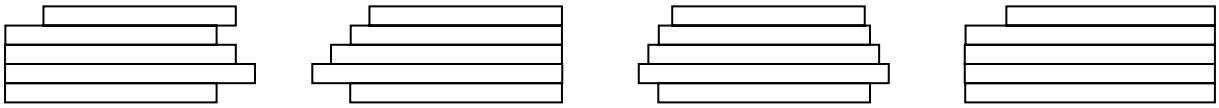
1) только ***по левому краю***;

2) только ***по правому краю***;

3) ***по центру*** — сделать равными расстояния от левого и правого краев;

4) ***по ширине***, т. е. сразу по левому и правому краям, *при этом в строках немного увеличивается расстояние между словами*. При остальных видах выравнивания строк расстояние между словами не увеличивается.

Из рис. 17 видно, что эффект красной строки пропадает при выравнивании строк абзаца по правому краю и по центру.



а б в г

Рис. 17. Выравнивание строк абзаца: а) по левому краю; б) по правому краю; в) по центру; г) по ширине, т. е. по левому и правому краям. Первая строка во всех примерах имеет отступ. Пропорции соблюдены.

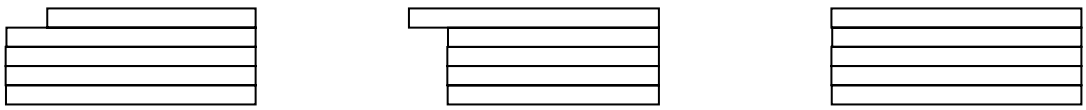
11.8. Сдвиг первой строки абзаца

Может иметь место ***сдвиг первой строки*** абзаца в три положения (см. рис. 18):

1) сдвиг вправо первой строки относительно левой границы абзаца называется ***отступ, или красная строка***;

2) сдвиг влево первой строки, а точнее, сдвиг вправо относительно левой границы абзаца всех строк абзаца, кроме первой строки, называется ***выступ, или висячая строка***;

3) ***нулевой отступ***.



а б в

Рис. 18. Сдвиг первой строки абзаца: а) отступ; б) выступ; в) нулевой отступ. Пропорции соблюдены

11.9. Сдвиг краев абзаца

Обычно края абзаца совпадают с границами полей страницы. Но у эпиграфов и стихотворений наблюдается ***сдвиг краев*** абзацев (см. рис. 19).

У эпиграфа левый край абзаца сильно сдвинут вправо так, что эпиграф оказывается в левой части страницы.

У стихотворений левый и правый края абзаца сдвинуты к центру так, что строфы оказываются расположенными в середине страницы.

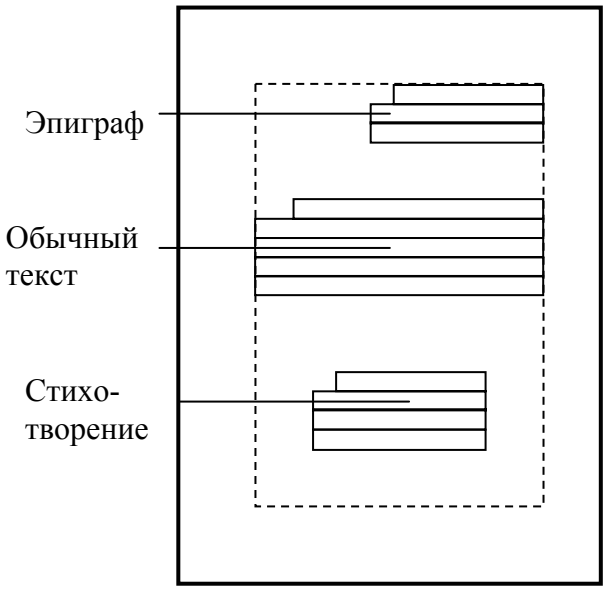


Рис. 19. Сдвиг краев абзацев. Все три абзаца выровнены по ширине

11.10. Абзацный интервал

Расстояние между строками абзаца называется ***абзацным, или междустрочным интервалом***. Расстояние между строками считается в единицах расстояния между их основаниями (см. рис. 20):

1) если строки вплотную прилегают друг к другу, то это ***одинарный интервал***;

2) если они раздвинуты на полстроки — ***полуторный***;

3) раздвинуты на одну полную строку — ***двойной***.

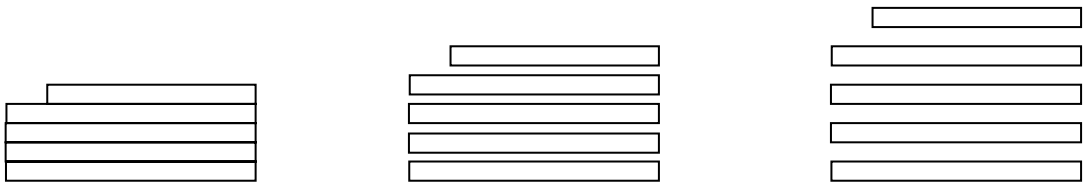


Рис. 20. Абзацный интервал: а) одинарный; б) полуторный; в) двойной

11.11. Межабзацный интервал

Можно установить два вида промежутков *перед* или *после* абзаца, не прибегая к добавлению пустых строк, а пользуясь специальными средствами форматирования абзацев, имеющимися в текстовом процессоре:

1) промежуток перед абзацем называется ***интервалом перед абзацем***;

2) промежуток после абзаца называется ***интервалом после абзаца***.

§ 12. Набор текстов

12.1. Арабские и римские цифры

При наборе текстов используются цифры двух видов: арабские и римские. Эти цифры не зависят ни от языков, ни от компьютеров, поскольку набираются символами из стандартного множества ASCII.

На русской пишущей машинке римские цифры набирают буквами русского алфавита, на компьютере — прописными латинскими буквами.

12.2. Целые и дробные числа

При наборе целых чисел, начиная с числа 10 000, можно отделять *пробелом или точкой* по три цифры, считая справа налево. При наборе исходных текстов компьютерных программ цифры на группы не разбиваются.

Дробная часть десятичных дробей при наборе на группы не делится. В русских текстах дробная часть отделяется от целой запятой, в английских текстах — точкой. Поэтому в исходных текстах языков программирования дробная часть десятичной дроби отделяется от целой точкой.

12.3. Структура кодовой таблицы

Символы указанного в текстовом процессоре шрифта набираются на клавиатуре и кодируются в текстовый файл в соответствие с кодовой таблицей. ***Кодовая таблица*** — это соответствие между символами и их кодами, числами. Кодовая таблица обычно рисуется в виде таблицы из 16 строк. В кодовой таблице содержится 256 символов с кодами 0—255, потому что при наборе каждый символ кодируется одним байтом.

Приведем структуру кодовых таблиц на любом компьютере (см. прил. 2).

Первые 32 кода 0—31 отведены под управляющие символы, отвечающие за форматирование теста и другие вспомогательные функции. Например, символ конца абзаца, получающийся при нажатии клавиши <Enter>, имеет код 13. Эти символы обычно в таблицах не рисуются.

Символы с кодами 32—126 являются ASCII и содержат все символы латинского алфавита, цифры, знаки препинания и специальные символы. Остальные коды 128—255 соответствуют символам других национальных алфавитов.

12.4. Русские кодовые таблицы

При передаче русских текстов с компьютера на компьютер, особенно в Интернете, возникает проблема перекодировки кириллицы в другую кодовую таблицу. Наиболее распространены русские кодировки windows, koi8-r и dos, которые существенно отличаются друг от друга расположением русских букв в соответствующих им кодовых таблицах (см. прил. 2).

Таблица 9. Расположение русских букв в кодовых таблицах

№	Кодировка	ОС	Коды русских букв		Сортировка русских букв
			Прописных	Строчных	
1	windows	Windows	192—223	224—255	Русская
2	koi8-r	UNIX	224—255	192—223	Латинская
3	dos	DOS	128—159	160—175, 224—239	Русская

12.5. Русские знаки препинания

Каждая страна имеет свои национальные правила: 1) начертания знаков препинания; 2) их набора.

Часто путают русское тире с американской черточкой, он же знак минус. Американская черточка в два раза длиннее дефиса, а русское тире в два раза длиннее американской черточки.

В русском языке используют кавычки «елочки», а „лапки“ используют как вложенные кавычки внутри «елочек». Вложение кавычек можно использовать и наоборот, только нужно следить, чтобы в одном документе был один способ употребления вложенных кавычек.

Когда порядковое числительное пишется арабскими цифрами, то через дефис ставится *окончание* его названия, но когда римскими — не ставится.

Точка после заголовка не ставится, а после номера заголовка и чисел в списке — ставится.

12.6. Отбивка русских знаков препинания

Отбивка — набор пробела между словами и знаками препинания. Правило набора, т. е. отбивки, слов и знаков препинания в русском языке:

знаки препинания «приклеиваются» к тому слову, к которому относятся. Между словами с «приклеенными» к ним знаками препинания набивается один и только один пробел.

Это общее правило. Из него имеют место следующие два исключения.

1. Символы ***дефиса*** - и ***диапазона*** — не отбивается ни справа, ни слева. Символ диапазона соединяет 2 числа или слова и между ними нет пробела.

2. Символ ***тире*** — отбивается пробелами с обеих сторон, поэтому между словами, разделенными тире, получается два пробела.

3. Исключение из исключений: тире не отбивается от точки и от запятой, т. е. набивают .— и ,—.

Рассмотрим некоторые частные простые случаи.

9 одиночных знаков препинания «приклеиваются» к предыдущему слову.

. , : ; ! ? !.. ?.. .,,

Парные знаки препинания, скобки и кавычки, «приклеиваются» к словам, в них заключенным.

Многоточие … «приклеивается» к тому фрагменту текста, к которому относится.

Упражнения

1. Полностью опишите параметры форматирования абзацев этого упражнения. При этом следует учесть следующие параметры форматирования абзацев: 1) выравнивание строк; 2) сдвиг краев; 3) сдвиг первой строки; 4) абзацный интервал.

Этот текст относится ко второму аб-

зацу этого упражнения.

На этом месте набрано од-

но-единственное предложе-

ние третьего абзаца этого

упражнения.

Это последний, четвертый абзац данного упражнения. Необязательно выражать параметры форматирования абзацев этого упражнения в точных числах. Достаточно перечислить все особенности форматирования этих четырех абзацев.

2. Перепишите текст этого упражнения полностью , правильно отбив все знаки препинания , которые здесь встречаются .Для обозначения пробелов —символа разделителя слов— используйте следующий специальный знак пробела :␣.

3. Перепишите текст этого упражнения полностью, исправив парные знаки препинания, если они неправильно набраны. К парным знакам препинания относятся скобки (конечно, круглые; квадратные, фигурные и угловые скобки знаками препинания не являются(, кавычки “елочки” и кавычки »лапки«. При этом используйте на верхнем уровне только кавычки „елочки“! Иногда к парным знакам препинания причисляют также два тире - … -, разделенные текстом.

Глава 5. Мультимедиа

§ 13. Графика

13.1. Размер экрана монитора

На мониторе картинка выводится в виде пикселей. Рассмотрим две характеристики этого изображения.

Размер экрана — это количество пикселей по горизонтали и вертикали матрицы экрана, заполняющей весь экран. Минимальный размер современного экрана в пикселях равен 640 × 480 точек.

Современные мониторы с диагональю 17 дюймов имеют оптимальный размер матрицы 800 × 600 точек. Более большой размер матрицы не вписывается в медицинские стандарты по размерам надписей на экране.

13.2. Глубина цвета

Глубина цвета — это количество цветов, приходящихся на один пиксель. Это разнообразие оттенков достигается разной степенью яркости люминофоров трех цветов, составляющих пиксель. Для хранения информации о яркости каждого люминофора требуется дополнительная память.

16 ($= 2^4$) цветов на один пиксель при 4-битной кодировке цвета недостаточно для просмотра фотографий. 256 ($= 2^8$) цветов на один пиксель при 8-битном цвете приемлемо. 16-битный цвет, называемый ***High Color*** (хороший цвет) (2^{16} цветов), вполне достаточен для непрофессиональных работ.

24-битный цвет имеет 1 байт памяти на каждый из трех цветов пикселя. Он необходим, например, для правильной цветопередачи страниц Интернета. Поэтому в настройках монитора выставляют ***True Color*** (истинный цвет) (2^{24} или 2^{32} цветов).

Когда картинка попадает на экран, она может закрыть часть фонового изображения. ***Прозрачный пиксель*** имеет цвет пикселя фона, который он перекрыл. ***RGBA*** — цветовая схема с прозрачными пикселями.

13.3. Растровая графика, растр

Выводимая на экран и печать графика хранится в виде файлов. Имеется два способа представления графики в компьютере: растровый и векторный.

Рисунок в растровом формате можно создать в программе на компьютере и получить путем сканирования картинки со сканера. Графика в векторном формате создается только в векторных графических процессорах.

Фотографии и другие ***полутоновые картинки*** существуют только в растровом формате. ***Растр*** — способ представления и хранения графической информации в виде матрицы пикселей, т. е. точно в таком виде, в каком эта информация выводится на экран монитора или принтер.

13.4. Растровые графические форматы

Самый распространенный графический формат Windows — это растровый формат BMP. Его файлы имеют расширение ***.bmp***. Обычно в них хранится первоначальная графическая информация, полученная, например, путем сканирования. Эти файлы не сжаты, поэтому имеют большой размер.

В Интернете на веб-страницах распространены два растровых формата.

1. Формат GIF имеет файлы с расширением ***.gif***. Он позволяет хранить прозрачные пиксели и анимацию, сохраняя в одном файле несколько последовательных кадров. Файлы в этом формате сжаты без потерь информации.

2. Формат JPEG имеет файлы с расширением ***.jpg***. Он позволяет при сжатии файлов задавать степень сжатия, т. е. процент потерь информации. Чем больше потерь, тем меньший размер будет иметь файл. Фотографии с 50 % потерь обычно смотрятся в Интернете вполне удовлетворительно.

13.5. Векторная графика

При ***векторном*** способе хранения графической информации данные хранятся не в виде точек, а в виде формул. Формулы описывают линии, ограничивающие элементарные фигуры, из которых состоят рисунки. Например, простейший отрезок хранится в виде координат его начала, угла наклона, длины, толщины и цвета.

Компьютерные шрифты, используемые в текстовых редакторах и других программах, созданы и хранятся в векторном формате.

13.6. Дискретизация и сглаживание

В векторном графическом редакторе работают на растровом мониторе. На экран или матричную печать графика выводится в виде растра, даже если она задана в векторном виде. Для этого векторный рисунок пересчитывают в растр специальными алгоритмами, т. е. производят его ***дискретизацию***. При этом линии векторной графики принимают вместо непрерывного ступенчатый вид с минимальным шагом один пиксель (см. рис. 21).

При изменении размеров растровой графики приходится ***пересчитывать растр***. Если картинка при этом увеличивается, то появляются ступеньки с шагом большим, чем один пиксель. Такие ступени можно ***сгладить*** специальными алгоритмами и перейти к ступеням с минимальным шагом один пиксель.

Рис. 21. а) Линия в векторном формате;
б)—г) линия в растре, или разные способы дискретизации линии

§ 14. Графические редакторы

14.1. Графический редактор и процессор

Графический редактор позволяет создавать, редактировать простыми средствами, просматривать и сохранять растровые рисунки.

Графический процессор — это графический редактор, обладающий интеллектуальными средствами, основанными на использовании формул. Например, в нем можно повернуть картинку на любой угол или изменить ее масштаб на любую величину.

Графический процессор может преобразовывать картинки с помощью разнообразных графических эффектов. Графический процессор также позволяет сканировать, если к компьютеру подключен сканер.

Любой графический редактор включает в себя текстовый редактор и позволяет набирать тексты.

14.2. Виды графических редакторов

Перечислим распространенные графические редакторы под Windows.

Редактор (растровый): ***Paint***.

Простой растровый процессор: ***Microsoft Photo Editor***.

Простой векторный процессор: ***MS Word*** рисует в своих документах.

Мощные растровые процессоры: ***Adobe Photoshop***, ***Corel Photo-Paint***.

Мощные векторные процессоры: ***Corel Draw***, ***Adobe Illustrator***.

14.3. Графические технологии

При работе в графических редакторах необходимо ознакомиться и освоить системы характерных понятий и специальных приемов работы. Эти системы называются ***технологиями графического редактора***.

Основная технология — это рисование элементарных фигур.

Элементы картинки рисуются в два этапа: 1) мышью выбирается из меню инструмент или фигура; 2) с помощью указателя мыши, вид которого соответствует выбранному инструменту (в случае фигуры указатель имеет вид креста), производится рисовательное действие. В дальнейшем то, что получилось, корректируется с помощью мыши или специального меню.

14.4. Сохранение и просмотр

Сохраняется нарисованная картинка на диске в виде файла с расширением, соответствующим графическому формату файла. Каждый графический процессор имеет свой оригинальный графический формат.

Картинка должна просматриваться в том же графическом процессоре, в каком была создана, если ее файл имеет расширение, соответствующее этому процессору. Распространенные форматы BMP, GIF или JPEG могут просматриваться и редактироваться в любом графическом редакторе.

14.5. Редактирование, выделение

Графические работы трудно разделить на рисование и редактирование, как можно разделить работу с текстами на набор и редактирование.

Можно сказать, что рисование — это графическое редактирование. В начале редактируется пустая картинка.

При графическом редактировании имеется универсальная компьютерная технология ***выделения, или высвечивания,*** частей картинки. В случае растровой картинки можно выделить только одну прямоугольную часть картинки. В векторном формате выделяются несколько элементов рисунка.

С выделенными объектами можно проделать следующие операции:

1) ***удаление*** выделенных объектов;

2) ***копирование*** выделенных объектов в буфер обмена;

3) при одновременном выполнении удаления и копирования получается операция ***вырезания*** выделенных объектов;

4) ***вставка*** в картинку выделенных объектов, предварительно скопированных в буфер обмена;

5) ***группирование*** выделенных объектов в один объект;

6) ***геометрические преобразования*** выделенных объектов (поворот, увеличение, растяжение и т. д.);

7) применение графических эффектов к выделенным объектам.

14.6. Основные инструменты

Перечислим некоторые простейшие графические инструменты, используемые в графических редакторах. На рис. 22 изображены эти инструменты так, как они выглядят в графическом редакторе Paint.

1. ***Карандаш***. Рисует тонкую линию заданного цвета.

2. ***Кисть***. Рисует линию, используя фигуру заданной конфигурации и заданного цвета. Имитирует рисование кистью.

3. ***Ластик***. Рисует линию, используя квадрат заданного размера и заданного цвета. Обычно используется для «стирания» части рисунка.

4. ***Заливка***. Заливает связную область одного цвета (из любого пикселя области можно перейти по пикселям такого же цвета в любой другой пиксель области) заданным цветом.

а б в г

Рис. 22. Кнопки а) карандаш, б) кисть, в) ластик и г) заливка
с панели **Набор инструментов** графического редактора Paint

14,7. Элементарные фигуры

Перечислим некоторые простейшие графические элементарные фигуры, которые рисуются в графических редакторах:

1) ***линия***; 2) ***стрелка***; 3) ***прямоугольник***; 4) ***овал***.

В фигурах можно задавать толщину линии, ее цвет и цвет заливки внутренней области. На рис. 23 представлены различные варианты этих фигур, которые можно нарисовать в текстовом процессоре Word.

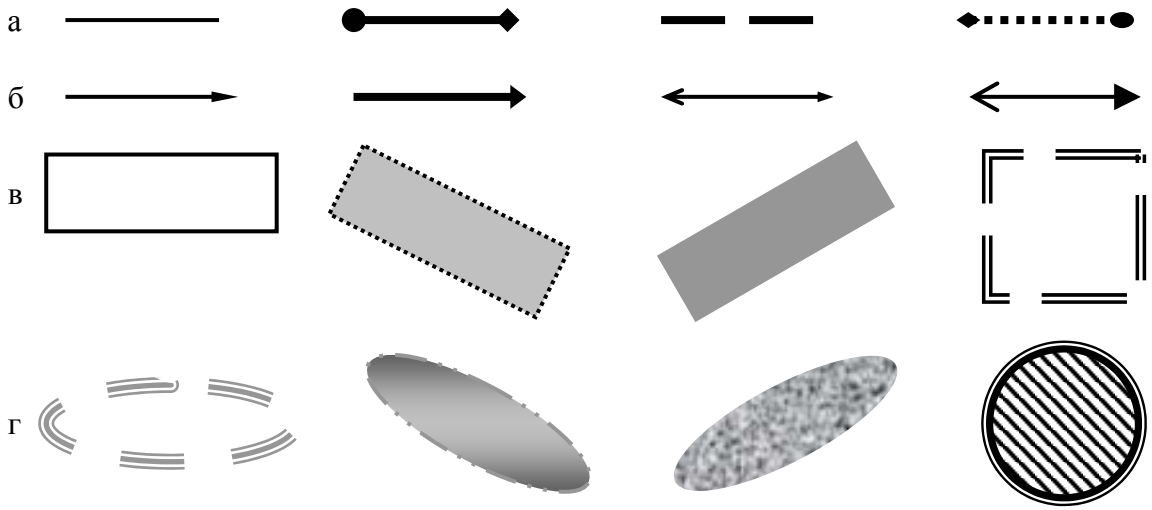


Рис. 23. Различные варианты а) линий, б) стрелок, в) прямоугольников и г) овалов из встроенного графического редактора текстового процессора Word

§ 15. Мультимедиа

15.1. Звуковая периферия

Для работы со звуком нужны следующие периферийные устройства:

1) ***звуковая карта, или звуковая плата***, для оцифровки звука. Она размещается внутри системного блока в гнезде на системной плате;

2) ***колонки*** и ***головные телефоны*** для прослушивания звука. Колонки размещаются вне компьютера; можно использовать колонки от звуковых центров. Головные телефоны надеваются на голову;

3) ***микрофон*** для записи звука. Микрофон обычно размещается на столе. Головные телефоны с прикрепленным микрофоном называют ***гарнитурой***.

15.2. Характеристики звука, оцифровка

Звук имеет следующие две основные характеристики.

1. ***Громкость*** звука в данный момент времени.

2. ***Частота*** звука как частота колебаний воздуха в данный момент времени. Человек обычно слышит звуки с частотами 20—20 тысяч Герц.

При ***оцифровке*** звука обе его характеристики записываются в отдельные моменты времени. Количество этих моментов в секунду называется ***частотой дискретизации*** звука. Чтобы передать высокие частоты, частота дискретизации звука должна быть достаточно высокой. Оцифрованный звук высокого качества имеет частоту дискретизации 48 Килогерц (в два раза выше верхней границы восприятия звука человеком).

Звуковые редакторы редактируют файлы с оцифрованным звуком.

15.3. Непосредственная оцифровка

При хранении на компьютере звук цифруется, кодируется и хранится в файлах. Оцифровка звука может происходить двумя разными способами. Первый способ ***оцифровки*** — это дискретизация непосредственно колебаний, которые генерируют звук. В простейшем случае получаются несжатые файлы с расширением .wav. Этот формат содержит оцифрованный звук и всегда готов для его воспроизведения.

Однако звук в этих файлах не сжат, поэтому они занимают много места. В Интернете очень много файлов со сжатым звуком формата .mp3, можно найти практически любое произведение. Появились файлы формата .asf, могущие вместе со звуком хранить и сжатую видеоинформацию.

Звук занимает большой объем: 1 минута звучания даже в сжатом виде занимает от 150 до 1000 Кб в зависимости от качества оцифровки.

15.4. Запись музыки

Цифровую музыку обычно создают сразу на компьютере. В этом случае звук записывается в виде ***MIDI***-файла. MIDI-файл занимает сравнительно маленький объем, поскольку формат MIDI хранит не готовый к воспроизведению звук, а лишь его нотную запись сразу для всех различных инструментов, участвующих в исполнении. При воспроизведении звук инструментов либо синтезируется компьютером, либо берется из специальной базы звучания инструментов.

15.5. Запись звука

Звук можно создать самому, записав его через ***микрофон***, подключенный непосредственно к компьютеру. В этом случае сразу получают файл с оцифрованным звуком.

Звук можно записать на магнитофоне (или диктофоне), а затем переписать его с магнитофона на компьютер как на другой магнитофон. Также можно переписать звук с компьютера на магнитофон.

Можно исполнить и записать музыку прямо на компьютере, запустив компьютерную программу, превращающую клавиатуру в фортепьянные клавиши и даже в любой инструмент.

К компьютеру подключается ***MIDI-клавиатура*** как периферийное устройство. Запустив специальную программу, играют на этой клавиатуре, как на фортепиано, записывая оцифрованную музыку и слушая ее через колонки.

15.6. Аудио-диски

Стандартные аудио-диски, используемые на аудио-проигрывателях и магнитолах, проигрываются на компьютерах, имеющих CD-дисководы.

На аудио-дисках информация записана не в виде файлов, а в виде звуковой дорожки, как на грампластинке. Поэтому прослушать аудио-диски можно на компьютере без звуковой карты. В этом случае музыку слушают на головных телефонах, подключенных непосредственно к CD-дисководу. На некоторых CD-дисководах для этого имеется регулятор громкости звука.

15.7. Анимация

Расширением графики являются анимация и видео.

Анимация — это просто картинки, сменяющие друг друга. Любая компьютерная игра включает в себя анимационные эффекты. Как уже отмечалось, графический GIF-формат может хранить анимацию.

Анимацию также можно получить, запрограммировав смену картинок с помощью языков программирования. В Интернете такими языками являются Ява, Ява-скрипт и Flash.

15.8. Видео

Видео — демонстрация на компьютере видеоинформации — фильмов, клипов и их фрагментов,— предназначенной для просмотра на телевизоре. В отличие от аудио-дисков видео-диски содержат запись видео-информации в виде файла. Эти файлы могут иметь ничего не говорящее расширение ***.dat*** или мультимедийные, т. е. видео-звуковые, расширения ***.asf*** или ***.avi***. Видеофайлы читаются специальными программами воспроизведения видео или мультимедиа, например, ***Windows Media***.

15.9. Мультимедиа

Мультимедийный компьютер — это компьютер, имеющий CD-дисковод, звуковую карту и колонки. Мультимедиа имеет большой объем, и мультимедийные программы хранятся обычно на компактах.

Мультимедиа — использование *компьютерной программой* в комплексе трех компонент: текста, графики и звука. Графика обычно расширяется до анимации или видео.

Мультимедиа существует только на компьютере; на телевидении пока мультимедиа нет. Связано это с тем, что непременный компонент мультимедиа — обратная связь, или ***интерактивность***, которую может обеспечить только компьютер. Мультимедиа является управляемой средой.

Глава 6. Сети

§ 16. Локальная компьютерная сеть

16.1. Локальная компьютерная сеть

Отдельные компьютеры связаны между собой не только дискетами, но и помощью других средств передачи информации. Многие компьютеры входят в состав локальных компьютерных сетей.

Локальная компьютерная сеть, или локальная сеть, или локально-вычислительная сеть (ЛВС) — это множество компьютеров, соединенных между собой специальными кабелями, используемыми для передачи информации между компьютерами.

16.2. Функции локальной сети

Локальная сеть выполняет следующие функции:

1) совместное использование данных;

2) обмен данными, например, ***электронную почту*** — отправку сообщений в сети указанным адресатам. Электронная почта изначально появилась в локальных сетях; с развитием глобальных сетей глобальная электронная почта заняла лидирующее положение;

3) совместное использование программ;

4) совместное использование модемов, принтеров и других устройств.

16.3. Оборудование

Для организации локальной сети необходимо специальное оборудование и программное обеспечение. Из оборудования назовем:

1) ***сетевую карту, или сетевой контроллер***, которая вставляется в компьютер в системную плату и позволяет компьютеру получать данные из локальной сети и передавать данные в сеть;

2) кабели с сетевыми разъемами для подключения к сетевым картам;

3) ***хабы, или концентраторы***, к которым и подключаются непосредственно *все компьютеры*.

Скорость обмена информацией составляет 10 Мбит/с и 100 Мбит/с.

16.4. Одноранговая сеть

Когда все компьютеры в сети равноправны, одного ранга, такая сеть называется ***одноранговой***.

В качестве программного обеспечения в одноранговых сетях может использоваться Windows. Их основной недостаток — слабая или низкая защиты передаваемой по сети информации от другого пользователя, работающего за другим компьютером. Такие сети — для дружных коллективов.

16.5. Принт-сервер

Как уже отмечалось, принтер можно совместно использовать в локальной сети. Эффективней, когда он подключен к сети независимо, как равноправное устройство, и называется ***принт-сервером***. Совместное использование принтера, подключенного к какому-нибудь компьютеру, ведет к дополнительной нагрузке на компьютер, к которому он подключен.

Такая же дополнительная нагрузка на компьютер происходит, если на компьютере хранятся какие-либо данные общего пользования.

16.6. Сервер и локальные станции

Чаще в локальных сетях ставится ***сервер*** — выделенный компьютер, который занимается исключительно обслуживанием локальной сети и совместно используемых данных. В этом случае получаем локальную сеть с выделенным сервером.

На сервере пользователи не работают, поэтому он обычно размещается в недоступном для них месте. Простые компьютеры, подключенные к серверу, называются ***локальными, или рабочими, станциями***.

В вышеприведенную схему объединения компьютеров в локальную сеть сервер вписывается очень просто: один из подключенных к хабу компьютеров, обычно специализированный и самый мощный, назначается сервером, после чего на него ставится специальная операционная система — серверное программное обеспечение.

На сервере используются операционные системы Novell NetWare, Windows NT Server, UNIX.

16.7. Функции сервера

Сервер организует общий доступ к ресурсам следующим образом.

1. *Файлы пользователя* хранятся на сервере, поэтому он может сесть работать за любой компьютер, подключенный к сети. Доступ к файлам пользователя осуществляется по паролю, который знает только пользователь. Однако программист-администратор сети имеет доступ ко всем файлам, расположенным на сервере.

2. На жестком диске сервера находятся *файлы с данными* общего пользования: библиотеки текстов, рисунков, музыки, видеофильмов. В системном блоке сервера может иметься CD-дисковод с общим доступом из сети.

3. На сервере могут находиться *программы общего пользования*: мультимедийные обучающие курсы, юридические базы данных, приложения (например, Word). Локальная станция может даже не иметь жесткого диска, а загружать операционную систему и все приложения с сервера.

§ 17. Глобальная компьютерная сеть

17.1. Глобальная сеть и адресное пространство

Отдельные компьютеры и локальные сети объединены в ***глобальные компьютерные сети***, которые осуществляют передачу данных между компьютерами, расположенными в любых странах. Любая глобальная сеть имеет свое ***адресное пространство***,— любой компьютер, подключенный к сети, имеет свой собственный уникальный адрес, хотя бы временный.

Адрес в Интернете может быть также задан в унифицированном виде, состоящем из 4 чисел, разделенных точками, например, 255.255.255.255.

17.2. Каналы связи и независимость

Для обмена множеством данных в глобальных сетях нужны каналы связи с большой пропускной способностью. В глобальных сетях используют мощные многоканальные спутниковые линии и оптоволоконные кабели, проложенные между компьютерами отдельных организаций и стран. Используются также обычные телефонные и телевизионные каналы связи.

Глобальные сети не принадлежат целиком никакому лицу или организации.

17.3. BBS и Интернет

Бесплатная и некоммерческая глобальная компьютерная сеть ***BBS*** функционирует на чужих телекоммуникациях и осуществляется силами отдельных энтузиастов и специалистов-профессионалов.

Платная и коммерческая глобальная сеть ***Интернет*** часто использует свои собственные телекоммуникационные линии: прокладывает оптоволоконные кабели, устанавливает спутниковое оборудование и другие антенны.

17.4. Устойчивость Интернет

Глобальные компьютерные сети могут осуществлять обмен данными между группами пользователей по разным маршрутам, по разным линиям связи. Это делает компьютерные сети очень надежными в эксплуатации, поскольку выход из строя отдельного телекоммуникационного канала не ведет к полному прекращению связи. В условиях путча в России или бомбежки Ирака, несмотря на попытки отключения каналов связи, Интернет, в отличие от других видов связи, в этих странах все равно работал.

17.5. Провайдер, модем

Чтобы подключить компьютер к глобальной сети, пользователю нужно договориться с ***провайдером***, который назначит пользователю имя для подключения к Интернету. При подключении к провайдеру через телефонную линию компьютер соединяется с Интернетом посредством ***модема*** — специального периферийного устройства. Модем может быть ***внутренним*** —

в виде компьютерной карты, ставящейся внутри компьютера на системную плату и соединяющейся телефонным кабелем с телефонной линией. Внешний модем выполнен в виде отдельного устройства, соединенного телефонным кабелем с телефонной линией, компьютерным — с компьютером.

17.6. Сервер

Компьютер, к которому подключается пользователь при входе в глобальную сеть, также называется ***сервером***.

Понятие сервера в глобальных сетях более широко. Войдя в сеть через провайдера, пользователь может соединиться с какому-нибудь другим компьютером из этой сети, который также называется ***сервером*** и оказывает дополнительные услуги, часто бесплатные. Платный провайдер нужен только для входа в Интернет; все остальное можно найти в Сети бесплатно.

17.7. Электронная почта

Электронная почта, или E-mail (е-мэйл), позволяет послать сообщение любому пользователю глобальных сетей. Принцип ее работы аналогичен функционированию обычной почты, где пользователь бросает письмо в почтовый ящик, его вынимает почтальон и оно доставляется в почтовый ящик адресата, из которого его достает адресат в удобное для себя время. В электронной почте пользователь сам передает письмо автоматической системе доставки, которая практически немедленно помещает его в почтовый ящик адресата, откуда его можно достать в удобное время.

С электронным письмом могут быть посланы ***вложения*** — дополнительные файлы с *любой* информацией.

В Интернете функционирует самая массовая электронная почта со своими собственными адресами.

17.8. Адреса почты, домен

Адреса электронной почты в Интернете организованы иерархически, как дерево ***доменов***. Домены первого уровня могут обозначать страну, например, ru — Россия, de — Германия, или область деятельности, например, com — коммерция, org — учебные организации, net — провайдеры. Домен второго уровня обозначает одно из имен провайдера, который оказывает услуги по подключению к Интернету, например, gazinter.net — Газкомплектимпекс в Калининграде, newmail.ru — ОРЦ в Москве.

Пользователь обычно подключается к домену второго уровня. При образовании адреса электронной почты к зарегистрированному имени пользователя добавляется символ @ (а-коммерческое, или собака) и затем — имя домена узла провайдера. Пользователь сам выбирает себе имя при регистрации электронной почты при соблюдении условия его уникальности. Например, автор курса имеет электронный адрес matsievsky@newmail.ru.

17.9. Телеконференции

Телеконференции представляют собой обмен информацией в Интернете по электронной почте. Существуют десятки тысяч тем телеконференций для обмена информацией. При подписке на телеконференции пользователю по электронной почте будут приходить все письма, поступающие на эти телеконференции. Пользователь может также посылать свои письма на телеконференции. Пользователь может создать новую телеконференцию, предложив тему, которая еще не освещена на существующих телеконференциях и которая интересует других пользователей.

Телеконференция может быть ***модерируемой***, когда каждое письмо читает специальный пользователь — ***модератор*** — и принимает решение, помещать ли его в телеконференцию или удалить. Телеконференция может быть и немодерируемой.

§ 18. Гипертекст и WWW

18.1. Ссылка, гипертекст

Гипертекст — это текст с перекрестными ссылками. ***Перекрестная ссылка, или ссылка*** — выделенная область текста или картинки, при выборе которой на компьютере происходит переход на другой текст или участок этого же текста. Гипертекст лишь просматривается, на нем отсутствует курсор редактирования, и выбор ссылки производится специальным указателем.

Современные электронные справочники — гипертекстовые документы.

18.2. WWW, веб-страница

WWW, или World Wide Web, или всемирная паутина, или веб-пространство — всемирная система гипертекстовых документов — ***веб-страниц***. которые расположены на ***веб-серверах*** и доступны пользователю, вошедшему в Интернет. Любой компьютер, подключенный к Интернету, может быть веб-сервером. Пользователь может создавать веб-страницы и сайты как на своем собственном сервере, так и на готовом сервере, уже работающем в Сети.

18.3. Режимы off-line и on-line

Электронная почта работает в режиме ***off-line*** (офф-лайн) по полной аналогии с обычной почтой: свои письма адресат получает в удобное время, а отправляет ответ, когда сочтет это нужным.

Интернет работает в непосредственном интерактивном, диалоговом режиме, т. е. в режиме ***on-line*** (он-лайн): пользователь общается с Интернетом в ***реальном времени***.

18.4. Обозреватель, его виды

Для доступа к Интернету необходимо на компьютере подключиться к провайдеру и затем запустить специальную компьютерную программу просмотра веб-страниц — ***броузер, или обозреватель***. В Windows это обычно: 1) ***Internet Explorer, или IE*** — бесплатная программа от фирмы Микрософт; 2) ***Netscape Navigator, или NN*** — платная программа от фирмы Netscape Communications.

В окне броузера нет курсора: броузер не предназначен для ввода текста или иного создания какой-либо информации. Броузер служит для просмотра готовой информации в Интернете на локальном компьютере.

18.5. Сайт, портал

В Интернете гипертекстовые веб-страницы объединены в ***сайты*** — наборы веб-страниц одной тематики, связанные между собой гиперссылками. Каждый сайт имеет свой адрес в веб-пространстве. При подключении к сайту в броузере набирают его адрес, после чего открывается главная страница сайта. На веб-странице могут быть ссылки на другое место этой страницы, на другую страницу этого сайта или на другой сайт.

Большие сайты или их объединения называются ***порталами***.

18.6. Поисковый сервер, адрес сайта

Если на сайте оказываются услуги по поиску информации в Интернете, то это — ***поисковый сервер, или поисковая машина***, а если по регистрации пользователей для организации электронной почты и сайта — то просто ***сервер***. На поисковом сервере можно найти адрес сайта, на котором находится нужная информация.

Отметим два самых лучших поисковых русскоязычных сервера: 1) yandex.ru; 2) google.ru.

Адреса сайтов составляются по тем же правилам, что и адреса электронной почты, только вместо собаки @ используется точка. Например, адрес сайта автора этого курса matsievsky.newmail.ru.

18.7. Бесплатные серверы

Зарегистрироваться и получить бесплатную электронную почту и организовать свой сайт в Интернете можно, например, на двух бесплатных серверах со следующими адресами:

1) ***www.newmail.ru***. Здесь можно бесплатно зарегистрироваться на доменах newmail.ru, hotmail.ru, nm.ru и nightmail.ru;

2) ***narod.yandex.ru***. Для бесплатной регистрации здесь предназначены домен narod.ru.

Ответы на упражнения

Глава 1. Числа

3*. MMMCMXCIX.

8*. 13.

11*. 1100100; 11001000.

12*. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Глава 3. Программы

2*. 90.

6*. а) и б) могут.

Часть II. ПРОГРАММИРОВАНИЕ

Глава 7. Алгоритм

§ 19. Технологии программирования

19.1. Алгоритм, его характеристики и свойства

Алгоритм — описание последовательности действий для решения задачи. ***Исполнитель алгоритма*** может быть человеком или автоматом.

Алгоритм имеет две характеристики.

1. ***Конечность, или результативность***. Алгоритм приводит к получению результата за конечное число шагов..

2. ***Однозначность, или определенность***. При одинаковых входных данных алгоритм выдает одинаковый результат.

Алгоритм также обладает следующими свойствами.

1. ***Массовость, или универсальность***. Алгоритм выдает результат при любых однотипных входных данных.

2. ***Модульность, или дискретность***. Алгоритм можно представить в виде последовательности более элементарных алгоритмов.

19.2. Архитектура фон Неймана

При создании первых компьютеров возникла проблема выбора: где хранить алгоритм и где хранить данные. У этого вопроса имеется два различных ответа:

1) алгоритм и данные хранятся в одном устройстве;

2) алгоритм и данные хранятся в разных устройствах.

Принцип работы компьютера по первому способу называется ***архитектурой фон Неймана*** и применяется практически во всех современных компьютерах: алгоритм и данные при выполнении программы хранятся в оперативной памяти.

19.3. Проектирование сверху вниз

Основной метод создания алгоритмов — ***проектирование, или программирование, сверху вниз, или пошаговая детализация***. Он заключается в разбиении исходной задачи на последовательность нескольких меньших подзадач. Эти подзадачи, в свою очередь, тоже распадаются на подзадачи и т. д. до тех пор, пока не останутся только элементарные алгоритмы.

Например, чтобы написать число 512, сначала пишут цифру 5, затем 1 и, наконец, 2. При этом цифры рисуют, последовательно прорисовывая линии, из которых они состоят. Принтер напечатает это число точками.

19.4. Модульность алгоритмов, головная программа

При программировании сверху вниз алгоритмы и данные делятся на относительно независимые части, называемые ***модулями***. Некоторые из модулей являются стандартными и поставляются в составе языков программирования, например, вычисление элементарных математических функций квадратный корень, логарифм, синус и т. д. Но главные модули все равно приходится проектировать программистам.

Таким образом, алгоритм является ***деревом модулей***: одни модули вызывают другие модули, начиная с самого верхнего первого модуля, называемого ***корневым модулем, или головной программой***.

19.5. Принцип черного ящика

При проектировании сколько-нибудь больших алгоритмов невозможно держать в памяти одновременно детали всех модулей алгоритма. Если модуль составлен правильно, то с ним можно обращаться как с черным ящиком.

Принцип черного ящика означает, что не имеет значения, *как* модуль выполняет свою функцию, какие алгоритмы скрыты у него внутри. Это не важно для остальных модулей алгоритма. Для модулей, которые обращаются к этому модулю, имеет значение только следующее:

1) какова функция модуля, т. е. *что* он делает;

2) описание входных и выходных данных модуля.

Правильное проектирование алгоритмов позволяет абстрагироваться от внутренней структуры модулей и рассматривать при сборке полного алгоритма только функции модулей.

19.6. Структурное программирование

Структурное программирование является технологией простого и прозрачного создания алгоритмов. Это единственный способ строить алгоритмы быстро и в последующем легко вносить в них изменения. ***Структурное программирование*** позволяет проектировать алгоритмы только из трех элементарных алгоритмов.

Эти три элементарных алгоритма являются не только самыми нижними модулями на дереве модулей. Каждый модуль является иерархией этих элементарных алгоритмов. Алгоритм, состоящий только из этих трех элементарных алгоритмов, присутствующих на всех его уровнях, называется ***структурным***.

Основная теорема структурного программирования утверждает, что любой алгоритм можно преобразовать к структурному виду.

Тремя элементарными структурными алгоритмами являются следующие.

1. ***Следование, или цепочка, или составная инструкция***.
2. ***Выбор, или ветвление, или условная инструкция***.
3. ***Цикл, или возврат, или циклическая инструкция***.

19.7. Объектно-ориентированное программирование

Компьютер = аппаратура + программы, а программа = алгоритм + данные. После открытия структурного проектирования алгоритмов стали разбираться с программированием данных.

Объектно-ориентированное программирование (ООП) организует данные и алгоритмы, обрабатываемые программой. При этом программист создает формы данных и алгоритмы, соответствующие основным характеристикам решаемой проблемы. Модели данных и алгоритмы, их обрабатывающие, называются ***классами***, а ***объекты*** — это конкретные их представители, используемые в программе.

Из общих объектов создаются другие, более специализированные. Механизм создания таких подобъектов называется ***наследованием***. В итоге данные программы представляют из себя ***объектную модель*** — дерево объектов, начиная с самого верхнего наиболее абстрактного и общего объекта.

19.8. Визуальное программирование

Визуальное программирование существенно облегчает программирование для графического интерфейса типа Windows, который состоит из множества графических объектов: кнопок, окошек, меню и т. д. Система ***визуального программирования*** предоставляет программисту:

1) готовую объектную визуальную модель, содержащую множество графических диалоговых объектов (кнопки, окошки, меню и т. д.) и программных модулей, которые их реализуют;

2) среду визуального программирования, в которой графические диалоговые объекты, которые будут определять интерфейс программы, просто размещаются на экране мышью.

§ 20. Языки программирования

20.1. Язык программирования, программа

Чтобы создать компьютерную программу, нужно записать алгоритм по специальным правилам на ***языке программирования***, который понимает и человек, и компьютер. Такая запись называется ***исходным текстом программы, или программой***. Программа пишется в простом текстовом редакторе. Затем программа переводится в машинные коды, выполняемые процессором компьютера, специальной программой-переводчиком.

20.2. Компилятор, система программирования

Компилятор — программа-переводчик с языка программирования в машинные коды, а процесс перевода — это ***компилирование*** программы.

Комплекс программ, включающий компилятор и другие средства написания программ, называется ***системой программирования***.

20.3. Транслятор, выполняемый файл

Возможны два способа компиляции.

Первый способ называется ***трансляцией*** и заключается в компилировании сразу всей программы в машинные коды. Затем строится ***выполняемый файл***, содержащий эту программу. И только потом программа выполняется путем запуска выполняемого файла.

Благодаря трансляции получается более быстрая по времени работы программа, но выполняемый файл имеет большой объем. Кроме того, выполняемый файл запускается только на компьютере того типа, где программа была транслирована.

20.4. Интерпретатор, виртуальная машина

При ***интерпретации*** компьютер читает программу по одной строке и сразу выполняет эту строку.

При интерпретации программу не надо переводить всю сразу в машинные коды, и поэтому она имеет маленький объем, равный объему исходного текста программы. Такая программа запускается на любом компьютере, на котором находится ***интерпретатор, или виртуальная машина***.

20.5. Сборщик, приложение

Большинство современных компиляторов работают в режиме трансляции. При трансляции модулей исходных текстов, оформленных специальным образом и называемых ***подпрограммами***, получается набор оттранслированных подпрограмм. Подпрограмма, в которую входит корневой модуль, называется ***головной программой***.

Выполняемый файл, или приложение создается из этих подпрограмм с помощью еще одной специальной программы — ***сборщика, или компоновщика, или линковщика, или редактора связей***. Сборщик связывает на уровне машинных кодов подпрограммы в цельную программу.

Таким образом, получается дерево подпрограмм, начиная с головной программы. Эти подпрограммы при выполнении вызывают друг друга. Головная программа вызывает свои подпрограммы, те, в свою очередь, подпрограммы следующего уровня и т. д., пока вся программа не выполнится.

20.6. Ошибки программирования

Ошибки в программах бывают двух видов.

1. ***Синтаксические ошибки*** — несоответствие формальным требованиям языка программирования. На них указывает транслятор при трансляции и линковщик при сборке программы.

2. ***Семантические ошибки*** — смысловые ошибки; при них программа «работает», но работает неправильно. Поиск этих ошибок происходит с помощью логического анализа работы программы и ее тестирования.

20.7. Этапы создания программы, программирование

Программирование — это анализ проблемы, постановка задачи, проектирование алгоритмов и получение правильно работающего программного кода. Программирование состоит из следующих компонентов.

1. Написание программы.

1.1. ***Проектирование, или алгоритмирование*** — описание требований, составление алгоритмов, блок-схем.

1.2. ***Реализация, или кодирование*** — написание программы, кодирующей алгоритмы на языке программирования.

2. Устранение ошибок.

2.1. ***Отладка*** — устранение синтаксических и других элементарных ошибок в программах на этапах трансляции и сборки.

2.2. ***Тестирование*** — проверка правильности работы программы на заранее подготовленных тестах, для которых известен точный результат.

§ 21. Блок-схема алгоритма

21.1. Исполнитель

Исполнитель алгоритмов — это устройство, выполняющее алгоритмы. Одним из наиболее гибких и универсальных исполнителей алгоритмов, наряду с человеком, является компьютер.

Когда программист составляет алгоритм, то при этом он является первым его исполнителем.

21.2. Процессор и оперативная память

Самые основные части компьютера с точки зрения проектирования алгоритмов — это процессор и оперативная память.

Процессор является активной частью компьютера и выполняет действия с данными.

Оперативная память необходима для хранения данных и алгоритмов, с которыми компьютер работает в данный момент.

21.3. Проектирование программы

Для успешного создания программ требуется использовать ***технологию программирования***. Технология состоит из набора правил, которые необходимо строго соблюдать.

Одним из правил является ***проектирование программы***, заключающееся в предварительном построении ее алгоритма. Алгоритм строится и записывается на бумаге, опытные программисты могут держать простые алгоритмы в уме.

21.4. Блок-схема

Наглядным изображением алгоритмов является блок-схема. ***Блок-схема, или диаграмма,*** кодирует алгоритм наглядными графическими средствами. Она состоит из следующих графических элементов:

1) кружков, кодирующих начало и конец, вход и выход из алгоритма;
2) параллелограммов, описывающих ввод и вывод данных;
3) прямоугольников, в которых описывается действие с данными;
4) ромбов, определяющих проверяемые алгоритмом условия.

Эти фигуры соединяются линиями со стрелками, указывающими последовательность действий.

Блок-схемы, как и алгоритмы, также имеют иерархическую структуру. В блок-схеме, описывающей алгоритм решения какой-нибудь задачи, на место прямоугольников подставляются блок-схемы алгоритмов, описывающих решение подзадач этой задачи.

21.5. Текущее положение исполнителя

При выполнении алгоритма по его схеме движется исполнитель, который находится в каждый момент в каком-то месте алгоритма, называемом ***активной точкой***. Для проверки правильности простого алгоритма достаточно его протестировать, т. е. пройти алгоритм, моделируя исполнителя.

При тестировании алгоритма берут простые входные данные и по ним рассчитывают результат. Затем вручную проходят алгоритм и получают его выходные данные. Если полученные выходные данные совпали с рассчитанным результатом, то алгоритм, может быть, построен правильно.

21.6. Следование

Элементарный алгоритм ***следование*** — последовательное, линейное выполнение действий. Этот элементарный алгоритм можно изобразить в виде блок-схемы из двух прямоугольников (рис. 24). Поскольку вместо прямоугольника можно подставить снова эту же блок-схему, и т. д., то последовательность прямоугольников может быть сколь угодно большой (рис. 24).

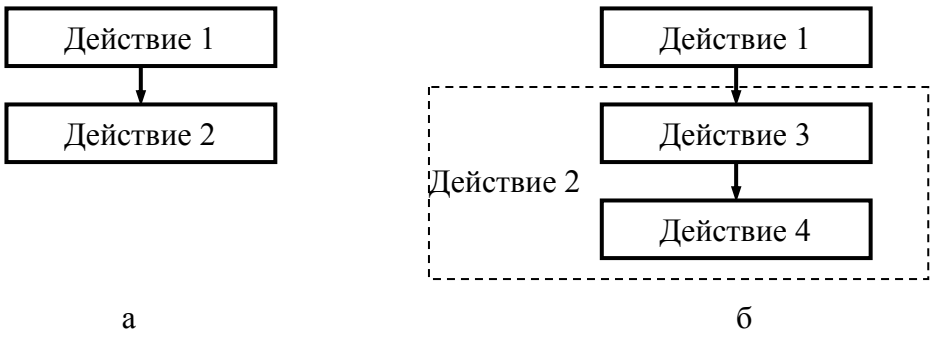


Рис. 24. Блок-схема элементарного алгоритма «следование»:
а) в простейшем виде из двух прямоугольников 1 и 2;
б) вместо прямоугольника 2 подставлены два прямоугольника 3 и 4

21.7. Выбор

Элементарный алгоритм ***выбор*** — это выбор по условию одного действия из двух. Когда исполнитель доходит до ромба с условием, то он выбирает среди двух действий. Если условие выполняется, исполнитель осуществляет переход к прямоугольнику действия по стрелке с пометкой «да», иначе — к действию «нет». После выполнения действия, определенного условием, исполнитель продолжает движение дальше по алгоритму.

Этот элементарный алгоритм изображается в виде блок-схемы из двух прямоугольников и одного ромба (рис. 25). Следует иметь в виду, что вместо прямоугольников можно подставлять любые элементарные алгоритмы.

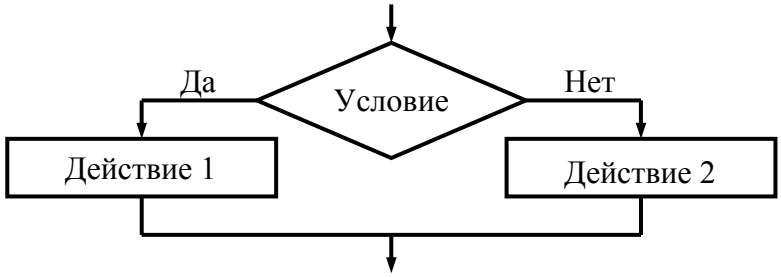


Рис. 25. Блок-схема элементарного алгоритма «выбор»:
если условие выполняется, то исполнитель выбирает действие 1, если нет — то 2

21.8. Цикл

Элементарный алгоритм ***цикл*** — это повторение одного и того же действия, пока выполняется условие. Когда исполнитель доходит до ромба с условием, то решает, выполнить действие или пройти мимо и больше сюда не возвращаться. Если условие выполняется, исполнитель выполняет действие и *возвращается обратно* на ромб с условием. Естественно, что действие должно, помимо прочего, изменять условие. Когда условие перестает выполняться, исполнитель продолжает движение дальше по алгоритму.

Этот элементарный алгоритм изображается в виде блок-схемы из одного прямоугольника и одного ромба (рис. 26). Следует иметь в виду, что вместо прямоугольника можно подставлять любые элементарные алгоритмы.

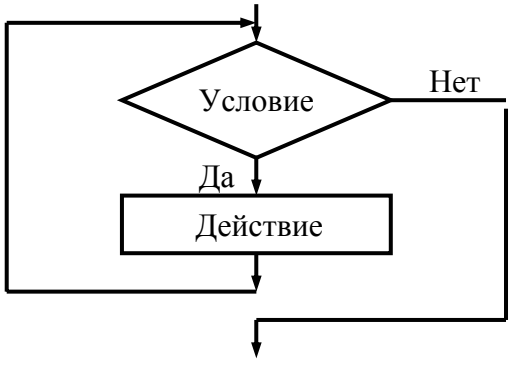


Рис. 26. Блок-схема элементарного алгоритма «выбор»:
пока условие выполняется, действие исполняется

Клиент либо жив, либо мертв. Если клиент мертв, то его либо можно оживить, либо нельзя.

А. Толстой. Золотой ключик.

Когда программист ложится спать, то на тумбочку рядом с кроватью он ставит два стакана: один с водой, другой пустой. Стакан с водой на тот случай, если пить захочется, пустой — если не захочется.

Анекдот.

Глава 8. Примеры блок-схем

§ 22. Функции

22.1. Вычисление длины гипотенузы по катетам

Задача. Даны два вещественных числа, являющихся величинами катетов некоторого прямоугольного треугольника. Вычислить длину гипотенузы этого треугольника.

Решение.

Анализ задачи.

1. Формулы. Будем вычислять гипотенузу по формуле Пифагора

$$z = \sqrt{x^2 + y^2}\ . \qquad (1)$$

2. Входные данные. Из формулы (1) следует, что входными данными являются длины катетов x и y.

3. Результат. Результатом работы алгоритма является вычисленное значение гипотенузы z.

Проектирование алгоритма.

Применим проектирование сверху вниз. Сначала нарисуем самую общую блок-схему решения задачи, являющуюся блок-схемой решения любой задачи. Это блок-схема типа «следование». Она изображена на рис. 27.

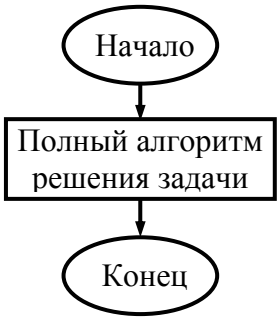


Рис. 27. Самая общая блок-схема решения любой задачи

В простых задачах, с которыми мы всегда будем иметь дело, единственное действие «полный алгоритм решения задачи» всегда можно заменить последовательностью трех действий. Соответствующая блок-схема изображена на рис. 28.

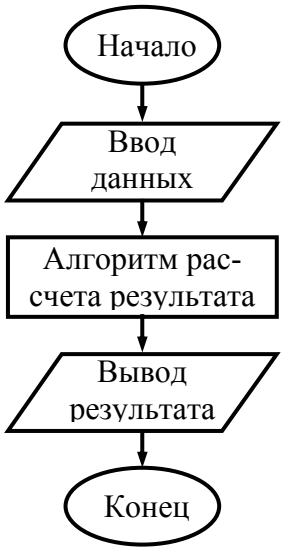


Рис. 28. Самая общая блок-схема решения простой задачи

Детализируем главный алгоритм. Для этого: 1) напишем входные данные и результат в явном виде; 2) алгоритм расчета результата выделим в отдельный алгоритм. Этот отдельный алгоритм представляет функцию.

В главном алгоритме не нужно описывать, как функция будет вычислять значение гипотенузы. Это внутреннее дело функции, она — «черный ящик» для главного алгоритма. На блок-схеме главного алгоритма вызов функции обозначается прямоугольником с двойными боковыми сторонами (см. рис. 29 а)). Вместо этого прямоугольника и подставляется алгоритм функции (см. рис. 29 б)). Функция реализует формулу Пифагора (1).

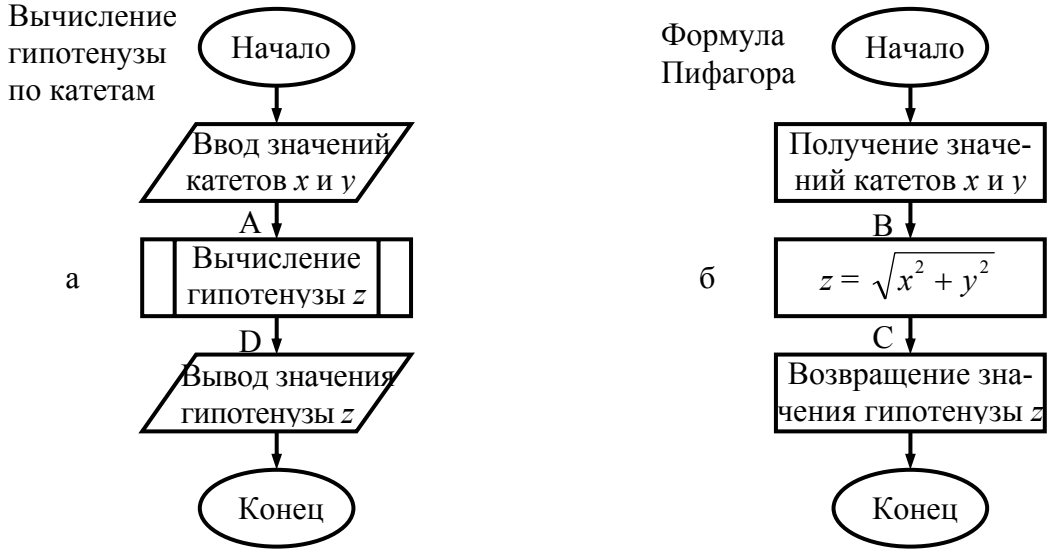


Рис. 29. Вычисление гипотенузы по катетам: а) главный алгоритм; б) функция

Тестирование алгоритма.

Осталось протестировать этот алгоритм вычисления гипотенузы. Расставим на алгоритме контрольные точки A, B, C и D (см. рис. 29). Контрольные точки ставятся во всех критических местах, везде, где вводятся или изменяются данные, т. е. после каждого ввода и каждого действия алгоритма (на блок-схеме контрольные точки ставятся после всех параллелограммов и прямоугольников, кроме вывода данных).

Исполнитель начинает выполнять алгоритм с его начала. При движении по алгоритму его активная точка проходит сначала контрольную точку A, затем B, потом C и, наконец, D.

Зададим входные данные. Пусть катет $x = 3$, а катет $y = 4$. Тогда, по формуле Пифагора, гипотенуза такого прямоугольного треугольника должна быть $z = 5$.

Посмотрим, что вычислит алгоритм, представленный на рис. 29. Для этого составим таблицу 10 из значений данных в контрольных точках.

Таблица 10. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$x = 3, y = 4$
B	$x = 3, y = 4$
C	$x = 3, y = 4, z = 5$
D	$x = 3, y = 4, z = 5$

Итак, алгоритм вычислил значение гипотенузы 5, что совпадает с расчетным значением. Поскольку алгоритм простой, то, скорее всего, он работает правильно. Для большей уверенности в правильной работе алгоритма желательно проверить его еще на нескольких более сложных тестах.

22.2. Вычисление площади треугольника по сторонам

Задача. Даны три вещественных числа, являющихся величинами сторон некоторого треугольника. Вычислить площадь этого треугольника.

Решение.

Анализ задачи.

1. Формулы. Вычислим площадь треугольника по формуле Герона

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \qquad (2)$$

где p — полупериметр.

2. Входные данные. Из формулы (2) следует, что входными данными являются длины сторон треугольника a, b и c.

3. Результат. Результатом работы алгоритма является вычисленное значение площади треугольника S.

Проектирование алгоритма.

Поскольку задача очень проста, пропустим первые шаги, описанные при решении предыдущей задачи, и начнем проектирование сразу с детализации общей блок-схемы на рис. 28: 1) напишем входные данные и результат в явном виде; 2) алгоритм расчета результата выделим в отдельный алгоритм. Этот отдельный алгоритм представляет функцию.

Результат проектирования представлен на рис. 30.

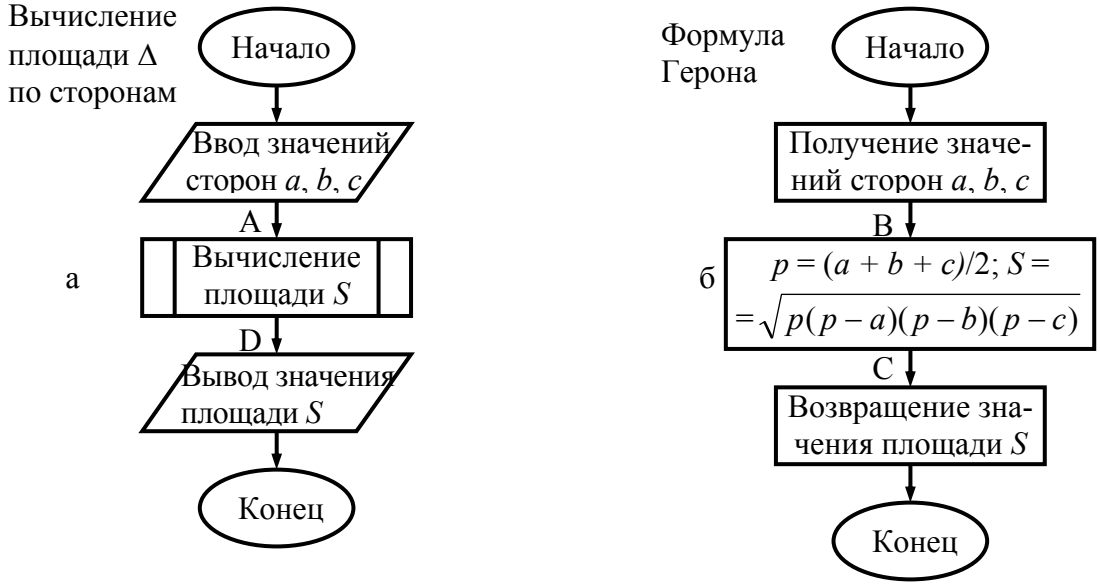


Рис. 30. Вычисление площади треугольника по сторонам: а) главный алгоритм; б) функция

Тестирование алгоритма.

При тестировании расставим на алгоритме контрольные точки A, B, C и D (см. рис. 30). Исполнитель начинает выполнять алгоритм с его начала. При движении по алгоритму его активная точка проходит сначала контрольную точку A, затем B, потом C и, наконец, D.

Зададим входные данные. Пусть стороны треугольника $a = 3$, $b = 4$ и $c = 5$. Тогда, по формуле Герона, его площадь должна быть $S = 6$. Просканируем, какое число вычислит алгоритм, представленный на рис. 30. Для этого сведем значения всех данных в контрольных точках в таблицу 11.

Таблица 11. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$a = 3, b = 4, c = 5$
B	$a = 3, b = 4, c = 5$
C	$a = 3, b = 4, c = 5, S = 6$
D	$a = 3, b = 4, c = 5, S = 6$

Алгоритм вычислил значение 6, что совпадает с расчетным значением.

22.3. Определение вида треугольника по сторонам, версия 1

Задача. Даны три вещественных числа, являющихся длинами сторон треугольника. Определить, является ли треугольник прямоугольным.

Решение.

Анализ задачи.

1. Формулы. Как узнать, является ли треугольник с заданными сторонами прямоугольным? Один из способов заключается в проверке трех равенств, основанных на формуле Пифагора

$$c^2 = a^2 + b^2,\ a^2 = b^2 + c^2,\ b^2 = c^2 + a^2. \qquad (3)$$

2. Входные данные. Из равенств (3) следует, что входными данными являются длины сторон треугольника a, b и c.

3. Результат. Результатом работы алгоритма является вывод сообщения о том, является ли рассматриваемый треугольник прямоугольным.

Проектирование алгоритма.

Детализируем общую блок-схему на рис. 28:

1) напишем входные данные в явном виде;

2) детализируем вывод сообщения о виде треугольника — заменим действие вывода (пунктирный прямоугольник, рис. 31) на выбор. Для этого введем переменную *vid*, которая вычисляется в функции определения вида треугольника. Если *vid* = 1, то выводим сообщение «Треугольник прямоугольный», иначе выводим сообщение «Треугольник не прямоугольный»;

3) алгоритм определения вида треугольника и вычисления переменной *vid* выделим в отдельный модуль, где также воспользуемся выбором.

Результат проектирования представлен на рис. 31.

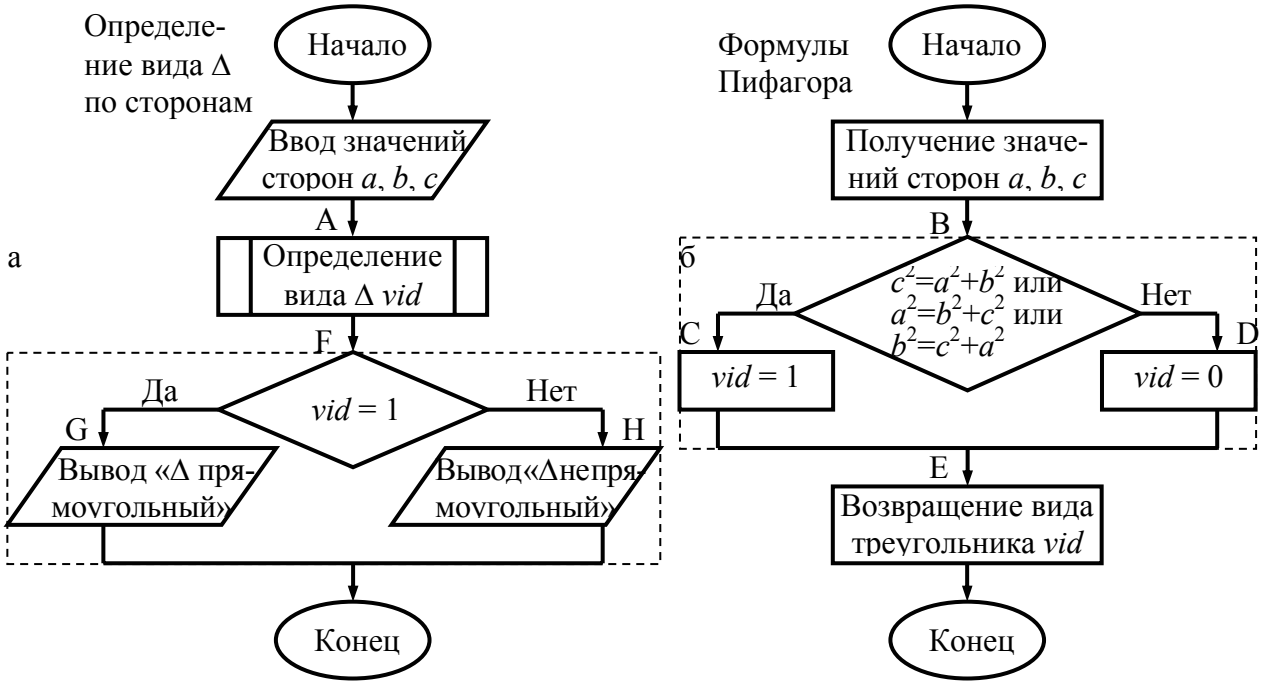


Рис. 31. Определение вида Δ по сторонам, версия 1: а) главный алгоритм; б) функция

Тестирование алгоритма.

При тестировании расставим на алгоритме контрольные точки A—H (см. рис. 31). При движении по алгоритму активная точка исполнителя проходит не все контрольные точки!

Тест 1. Зададим стороны прямоугольного треугольника $a = 3$, $b = 4$ и $c = 5$. Протрассируем алгоритм и запишем значения переменных в контрольных точках; получим таблицу 12. Полученная последовательность контрольных точек также несет полезную информацию. Она показывает, по каким ветвям выбора проходит исполнитель: точки D и H пропущены.

Таблица 12. Значения данных в контрольных точках: треугольник прямоугольный

Контрольная точка	Значения данных
A	$a = 3, b = 4, c = 5$
B	$a = 3, b = 4, c = 5$
C	$a = 3, b = 4, c = 5$
E	$a = 3, b = 4, c = 5, vid = 1$
F	$a = 3, b = 4, c = 5, vid = 1$
G	$a = 3, b = 4, c = 5, vid = 1$

Итак, мы убедились, что алгоритм вычислил правильное значение переменной *vid* и вывел правильный текст.

Но в случае рассматриваемой задачи одного теста не достаточно. Мы не знаем, правильно ли будет работать алгоритм, если входной треугольник *не* будет прямоугольным.

Тест 2. Зададим стороны прямоугольного треугольника $a = 3$, $b = 4$ и $c = 6$. Протрассируем алгоритм и запишем значения переменных в контрольных точках; получим таблицу 13.

Таблица 13. Значения данных в контрольных точках: треугольник не прямоугольный

Контрольная точка	Значения данных
A	$a = 3, b = 4, c = 6$
B	$a = 3, b = 4, c = 6$
D	$a = 3, b = 4, c = 6$
E	$a = 3, b = 4, c = 6, vid = 0$
F	$a = 3, b = 4, c = 6, vid = 0$
H	$a = 3, b = 4, c = 6, vid = 0$

В этом тесте алгоритм также вычислил правильное значение переменной *vid* и вывел правильный текст.

Мы протестировали алгоритм двумя тестами. В одном тесте входной треугольник был прямоугольным, в другом — нет. Является ли эта система тестов полной?

Нет, не является. При работе функции проверка формулы Пифагора осуществляется тремя сравнениями $c^2 = a^2 + b^2$, $a^2 = b^2 + c^2$ и $b^2 = c^2 + a^2$. Мы же протестировали только первое, в котором длину гипотенузы передает переменная c, а также рассмотрели тест, в котором ни одно из трех сравнений не выполняется.

Тест 3. Зададим стороны прямоугольного треугольника $b = 3$, $c = 4$ и $a = 5$. Протрассируем алгоритм и запишем значения переменных в контрольных точках; получим таблицу 12 (значения a, b и c будут отличаться).

Тест 4. Зададим стороны прямоугольного треугольника $c = 3$, $a = 4$ и $b = 5$. Протрассируем алгоритм и запишем значения переменных в контрольных точках; получим таблицу 12 (значения a, b и c будут отличаться).

Теперь система тестов полна, и в том, что алгоритм работает правильно, гораздо больше уверенности.

22.4. Определение вида треугольника по сторонам, версия 2

Задача. Даны три вещественных числа, являющихся длинами сторон треугольника. Определить, является ли треугольник прямоугольным.

Решение.

Анализ задачи.

Анализ задачи полностью совпадает с анализом в предыдущем пункте.

Проектирование алгоритма.

Детализируем общую блок-схему на рис. 28, но сделаем это немного по другому, чем это было сделано в предыдущем пункте. Отличия, конечно, будут содержаться только в том, как функция вычисляет вид треугольника:

3) Алгоритм определения вида треугольника и вычисления переменной *vid* выделим в отдельный модуль, где также воспользуемся выбором. Но, в отличие от предыдущего пункта, где все три сравнения $c^2 = a^2 + b^2$, $a^2 = b^2 + c^2$ и $b^2 = c^2 + a^2$ были сделаны в одном блоке выбора, будем проверять эти равенства по очереди.

Сначала проверим первое равенство $c^2 = a^2 + b^2$. Если оно верно, то треугольник прямоугольный, и остальные равенства не нуждаются в проверке (тем более что нам уже стало известно, что они ложны).

Если первое равенство неверно, то проверим второе $a^2 = b^2 + c^2$. Если оно верно, то треугольник прямоугольный, и функцию можно закончить.

Если и второе равенство неверно, то проверяем оставшееся $b^2 = c^2 + a^2$. Если оно верно, то треугольник все-таки прямоугольный, если неверно — то точно не прямоугольный, поскольку были проверены все три равенства, и ни одно не было верно.

Результат проектирования представлен на рис. 32 и 33. В этой задаче продемонстрировано также преимущество технологии модульности при построении алгоритмов: во втором варианте решения изменилась только функция, главный алгоритм остался абсолютно без изменений

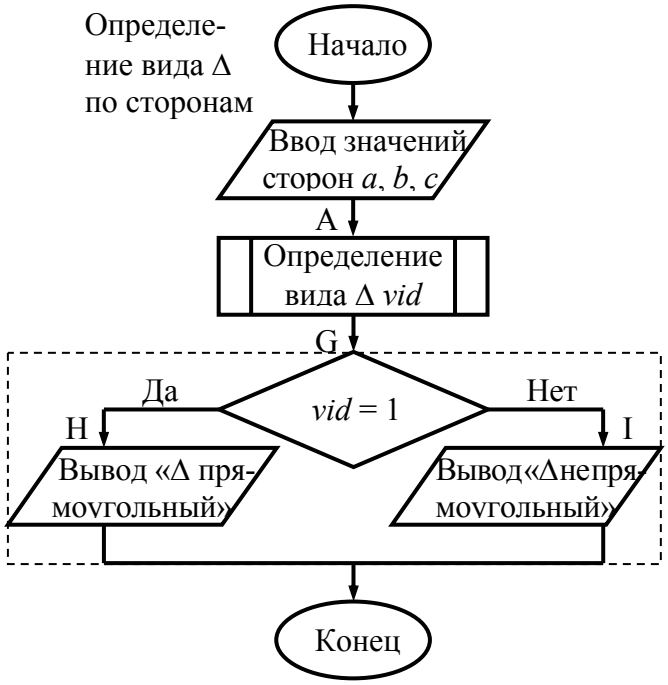


Рис. 32. Определение вида треугольника по сторонам, версия 2: главный алгоритм

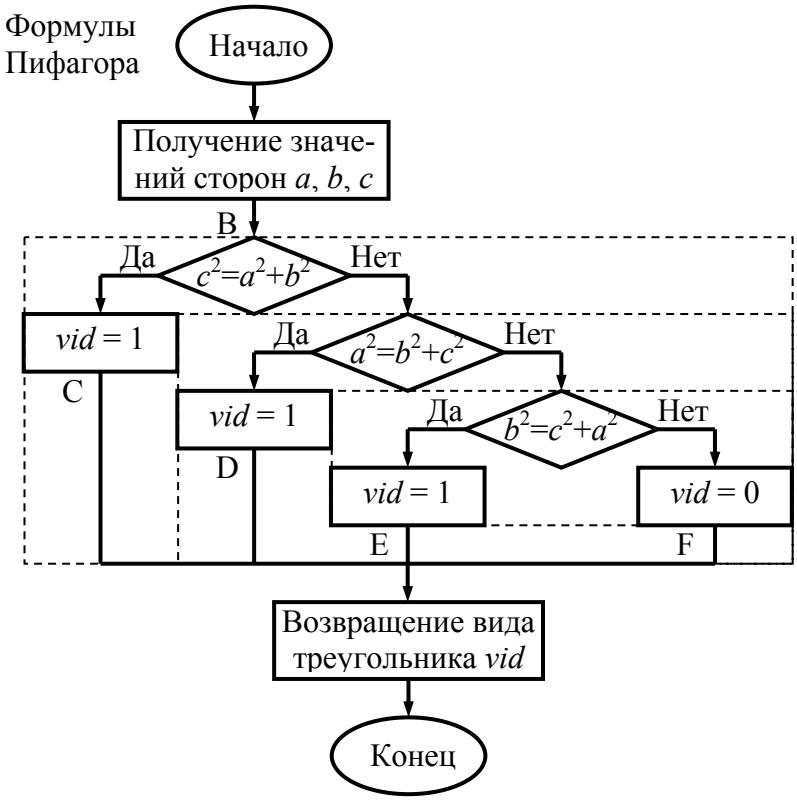


Рис. 33. Определение вида треугольника по сторонам, версия 2: функция

Тестирование алгоритма.

При тестировании расставим на алгоритме контрольные точки A—I (см. рис. 32 и 33). Как и в предыдущем пункте, организуем четыре теста. Результаты трассировок алгоритма при этих тестах сведем в таблицу 14.

Таблица 14. Значения данных в контрольных точках для четырех тестов

Контрольная точка	Значения данных			
	Тест 1	Тест 2	Тест 3	Тест 4
A	$a = 3, b = 4, c = 5$	$a = 3, b = 4, c = 6$	$a = 5, b = 3, c = 4$	$a = 4, b = 5, c = 3$
B	$a = 3, b = 4, c = 5$	$a = 3, b = 4, c = 6$	$a = 5, b = 3, c = 4$	$a = 4, b = 5, c = 3$
C	$a = 3, b = 4, c = 5, vid = 1$	—	—	—
D	—	—	$a = 5, b = 3, c = 4, vid = 1$	—
E	—	—	—	$a = 4, b = 5, c = 3, vid = 1$
F	—	$a = 3, b = 4, c = 6, vid = 0$	—	—
G	$a = 3, b = 4, c = 5, vid = 1$	$a = 3, b = 4, c = 6, vid = 0$	$a = 5, b = 3, c = 4, vid = 1$	$a = 4, b = 5, c = 3, vid = 1$
H	$a = 3, b = 4, c = 5, vid = 1$	—	$a = 5, b = 3, c = 4, vid = 1$	$a = 4, b = 5, c = 3, vid = 1$
I	—	$a = 3, b = 4, c = 6, vid = 0$	—	—

Итак, мы убедились, что алгоритм вычислил правильное значение переменной *vid* и вывел правильный текст на всех четырех тестах.

22.5*. Определение вида треугольника по сторонам, версия 3

Задача. Даны три вещественных числа, являющихся длинами сторон треугольника. Определить, является ли треугольник прямоугольным.

Решение.

Анализ задачи.

Анализ задачи полностью совпадает с анализом в предыдущих пунктах.

Проектирование алгоритма.

В версии 1 функция всегда проводила три пифагоровых сравнения, в версии 2 — в двух случаях из четырех (см. табл. 14). Снова по-другому построим функцию вычисления вида треугольника, так, чтобы пифагорово сравнение проводилось функцией при любых данных только один раз.

i. Сначала найдем наибольшую сторону треугольника, т. е. вероятную гипотенузу. Например, проверим неравенство $a < b$. Если $a < b$ верно, то проверим $b < c$: если верно, то гипотенуза c, иначе гипотенуза b. Если $a < b$ не верно, то проверим $a < c$: если верно, то гипотенуза c, иначе гипотенуза a.

ii. Теперь мы знаем, какая сторона может быть гипотенузой, и сразу проверяем на истинность нужную формулу Пифагора.

§ 23. Циклы

23.1. Сумма натурального ряда

Задача. Найти сумму первых n натуральных чисел.

Решение.

Анализ задачи.

1. Формулы. Нужно найти сумму n чисел

$$1 + 2 + 3 + \ldots + n. \qquad (4)$$

2. Входные данные. Из суммы (4) следует, что входным данным является количество первых натуральных чисел n.

3. Результат. Результатом работы алгоритма является вывод суммы (4).

Проектирование алгоритма.

Детализируем общую блок-схему на рис. 28:

1) напишем входные данные в явном виде;

2) выведем сообщение о величине суммы;

3) алгоритм вычисления суммы S первых n натуральных чисел выделим в отдельный модуль. В этом модуле используем цикл для накопления суммы в сумматоре S.

Для этого заметим, что текущее слагаемое суммы (4) имеет вид i. Тогда сумму можно накопить в сумматоре, если:

1) присвоить ему начальное значение $S = 0$;

2) прибавлять к сумматору в цикле переменную i, изменяя ее от 1 до n.

Результат проектирования представлен на рис. 34.

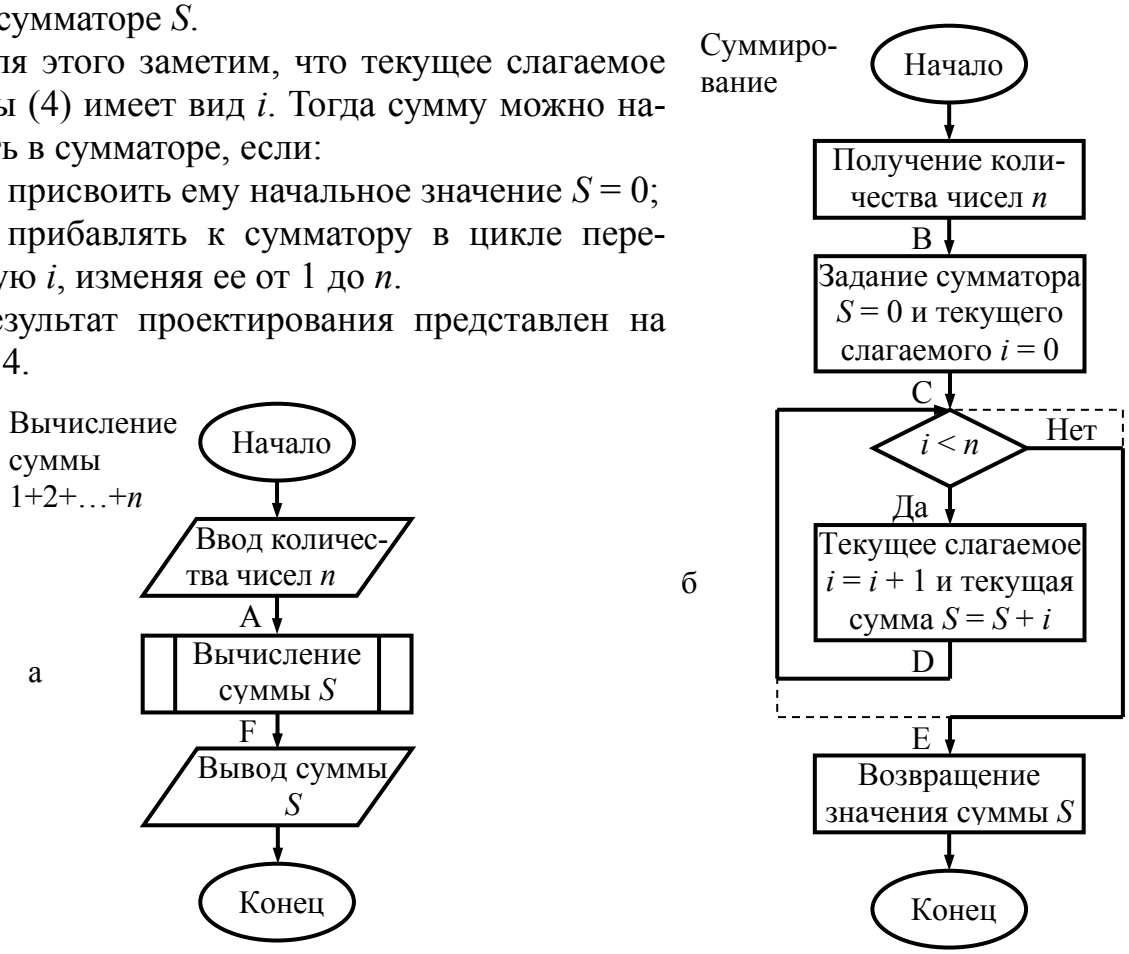


Рис. 34. Вычисление суммы первых n натуральных чисел:
а) главный алгоритм; б) функция

Тестирование алгоритма.

При тестировании расставим на алгоритме контрольные точки A—F (см. рис. 34). Зададим количество суммируемых первых натуральных чисел $n = 5$. Мы должны получить значение суммы первых пяти натуральных чисел $S = 15$. Протрассируем алгоритм и запишем значения переменных в контрольных точках в таблицу 15.

Таблица 15. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 5$
B	$n = 5$
C	$n = 5, S = 0, i = 0$
D	$n = 5, S = 1, i = 1$
D	$n = 5, S = 3, i = 2$
D	$n = 5, S = 6, i = 3$
D	$n = 5, S = 10, i = 4$
D	$n = 5, S = 15, i = 5$
E	$n = 5, S = 15, i = 5$
F	$n = 5, S = 15, i = 5$

Итак, мы убедились, что построенный алгоритм вычислил правильное значение суммы первых пяти натуральных чисел.

23.2. Сумма обратных величин натурального ряда

Задача. Найти сумму первых n обратных величин натуральных чисел.

Решение.

Анализ задачи.

1. Формулы. Нужно найти сумму n чисел

$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}. \qquad (5)$$

2. Входные данные. Из суммы (5) следует, что входным данным является количество первых обратных натуральных чисел n.

3. Результат. Результатом работы алгоритма является вывод суммы (5).

Проектирование алгоритма.

Алгоритм совпадает с предыдущим алгоритмом, исключая функцию.

В алгоритме вычисления функции используем цикл для накопления суммы в сумматоре S. Текущее слагаемое суммы (5) имеет вид $\frac{1}{i}$.

Тогда сумму можно накопить в сумматоре, если:

1) присвоить ему начальное значение $S = 0$;

2) прибавлять к сумматору в цикле число $\frac{1}{i}$, изменяя i от 1 до n.

Результат проектирования представлен на рис. 35.

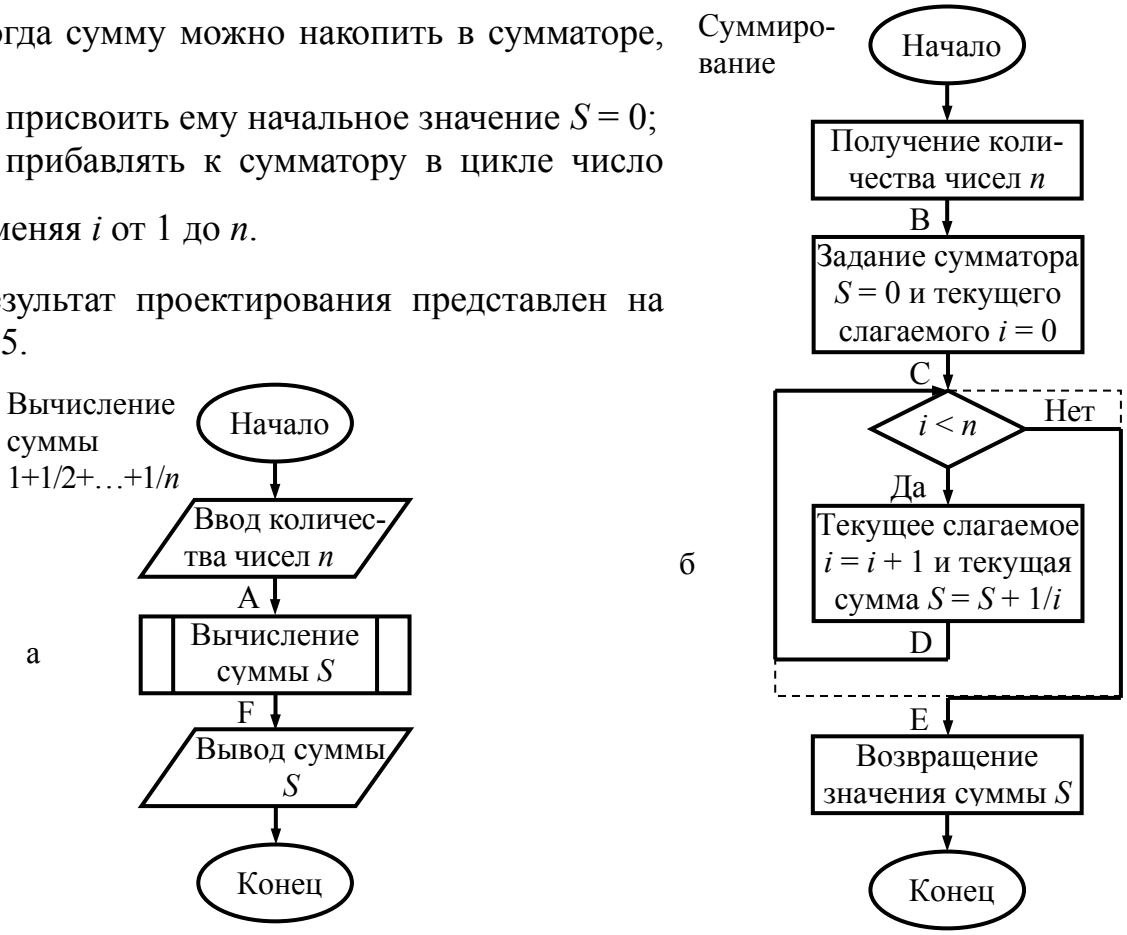


Рис. 35. Вычисление суммы первых n обратных натуральных чисел:
а) главный алгоритм; б) функция

Тестирование алгоритма.

Расставим контрольные точки A—F (см. рис. 35). Пусть $n = 3$. Мы должны получить $S = 11/6$. Протрассируем алгоритм в таблицу 16.

Таблица 16. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 3$
B	$n = 3$
C	$n = 3, S = 0, i = 0$
D	$n = 3, S = 1, i = 1$
D	$n = 3, S = 3/2, i = 2$
D	$n = 3, S = 11/6, i = 3$
E	$n = 3, S = 11/6, i = 3$
F	$n = 3, S = 11/6, i = 3$

Построенный алгоритм работает правильно.

23.3. Нахождение произведения нечетных чисел, версия 1

Задача. Найти произведение n чисел вида $1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots$

Решение.

Анализ задачи.

1. Формулы. Нужно найти произведение n первых нечетных чисел. Чему равно n-е нечетное число? В решение вопроса о том, чему равно n-е число, и заключается один из трудных моментов в программировании.

В данном случае можно рассуждать так. Нечетное число меньше четного на единицу. Очевидно, что n-е четное число равно $2n$. Можно предположить, что в нашем случае n-е число будет равно $2n - 1$.

Проверим нашу гипотезу. Методом математической индукции можно доказать, что n-е нечетное число равно $2n - 1$. Но в простых случаях, подобных этому, можно поступить проще. Проверим гипотезу для первых *трех* чисел. Если первые три числа ей удовлетворяют, то можно считать, что проблема решена.

При $n = 1$ получаем, что $2 \cdot 1 - 1 = 1$. При $n = 2$ имеем: $2 \cdot 2 - 1 = 3$. Наконец, когда $n = 3$, то $2 \cdot 3 - 1 = 5$. Следовательно, формула для выражения n-го нечетного числа верна.

Итак, нужно найти произведение

$$1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots \cdot (2n - 1). \qquad (6)$$

2. Входные данные. Из суммы (6) следует, что входным данным является количество сомножителей n.

3. Результат. Результатом работы алгоритма является вывод вычисленного произведения (6).

Проектирование алгоритма.

Алгоритм совпадает с предыдущим алгоритмом, исключая функцию. Только сумму нужно заменить на произведение.

В алгоритме вычисления функции используем цикл для накопления произведения в умножителе P. Текущий сомножитель произведения (6) имеет вид $2i - 1$.

Тогда произведение можно накопить в умножителе, если:

1) присвоить ему начальное значение $P = 1$;

2) домножать умножитель в цикле на число $2i - 1$, изменяя i от 1 до n.

Результат проектирования представлен на рис. 36.

Тестирование алгоритма.

Расставим контрольные точки A—F (см. рис. 36). Пусть $n = 5$. Мы должны получить $P = 945$. После трассирования алгоритма получаем таблицу 17.

Из анализа таблицы 17 можно сделать заключение, что построенный алгоритм работает правильно.

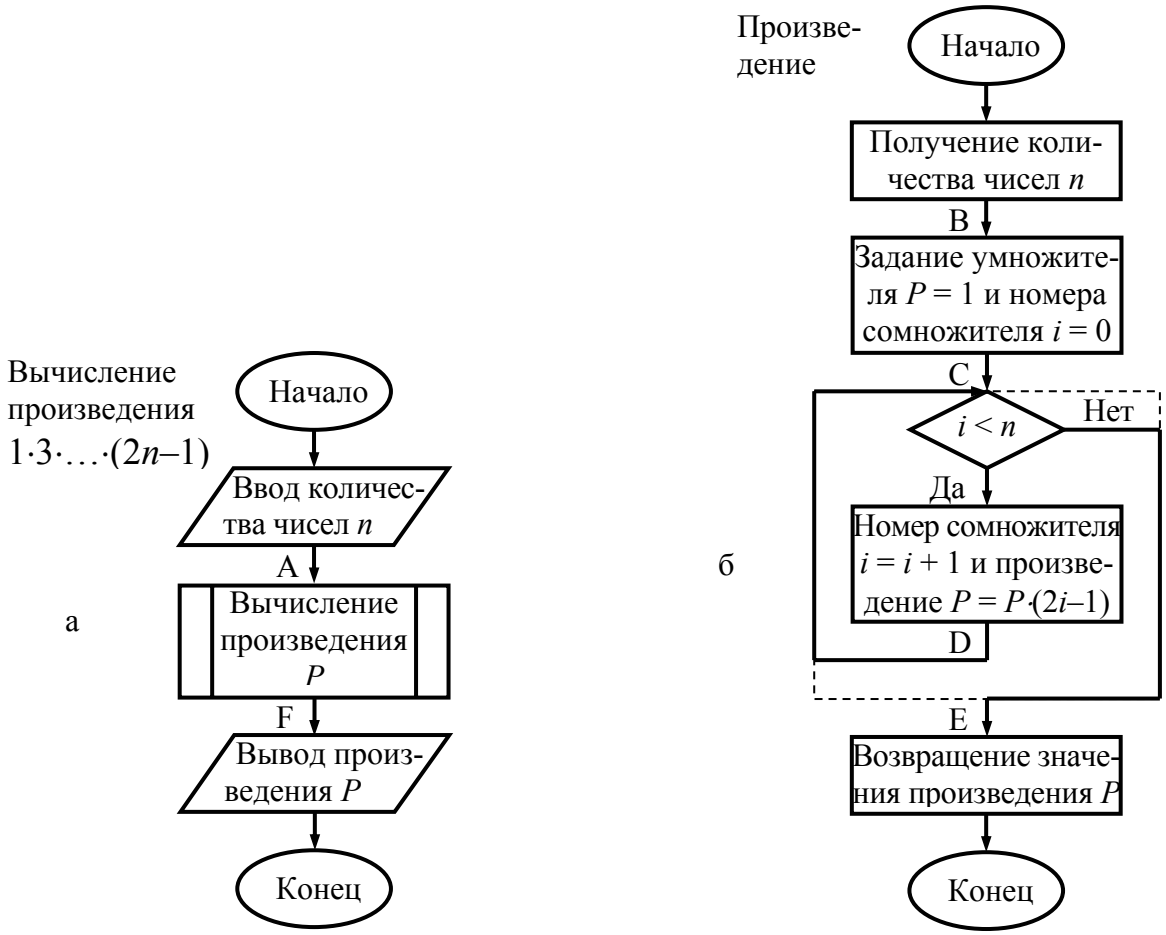


Рис. 36. Вычисление произведения первых n нечетных натуральных чисел, версия 1: а) главный алгоритм; б) функция

Таблица 17. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 5$
B	$n = 5$
C	$n = 5, P = 1, i = 0$
D	$n = 5, P = 1, i = 1$
D	$n = 5, P = 3, i = 2$
D	$n = 5, P = 15, i = 3$
D	$n = 5, P = 105, i = 4$
D	$n = 5, P = 945, i = 5$
E	$n = 5, P = 945, i = 5$
F	$n = 5, P = 945, i = 5$

23.4. Нахождение произведения нечетных чисел, версия 2

Задача. Найти произведение n чисел вида $1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots$

Решение.

Анализ задачи.

1. Формулы. Нужно найти произведение n первых нечетных чисел. Не обязательно знать, чему равно n-е нечетное число. Вычисление этого произведения в функции можно организовать по другому.

Итак, нужно просто найти произведение n чисел

$$1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots \qquad (7)$$

2. Входные данные. Из комментария к сумме (7) следует, что входным данным является количество сомножителей n.

3. Результат. Результатом работы алгоритма является вывод вычисленного произведения (7).

Проектирование алгоритма.

Алгоритм совпадает с предыдущим алгоритмом, исключая функцию. Построим функцию без формулы, задающей текущий множитель.

Для этого заметим, что следующее нечетное число больше предыдущего на 2. Поэтому, прибавив к первому нечетному числу два, получим второе нечетное число, прибавив снова два — третье нечетное число и т. д. до n-го нечетного числа. Лучше понять это поможет трассировка алгоритма.

Алгоритм функции можно записать в следующем виде:

1) присвоить умножителю начальное значение $P = 1$;

2) присвоить сумматору значение $S = 1$;

3) присвоить номеру числа предварительное значение $i = 0$;

4) проверить условие i < n. Если оно справедливо, перейти к шагу 5). Иначе закончить алгоритм;

5) получить номер текущего сомножителя операцией $i = i + 1$. Теперь этот номер нужен только для подсчета количества проходов цикла;

6) умножить умножитель на i-е нечетное число S: $P = P \cdot S$;

7) вычислить следующее нечетное число $S = S + 2$. Обратите внимание, что следующее нечетное число вычисляется *после* умножения;

8) перейти к шагу 4).

Результат проектирования представлен на рис. 37.

Тестирование алгоритма.

Расставим контрольные точки A—F (см. рис. 37). Пусть $n = 5$. Мы должны получить произведение $P = 945$. После трассирования алгоритма получаем таблицу 18.

Внимательное пошаговое изучение трассировки алгоритма поможет понять, что происходит при выполнении этого алгоритма.

Из анализа таблицы 18 можно также сделать заключение, что построенный алгоритм работает правильно.

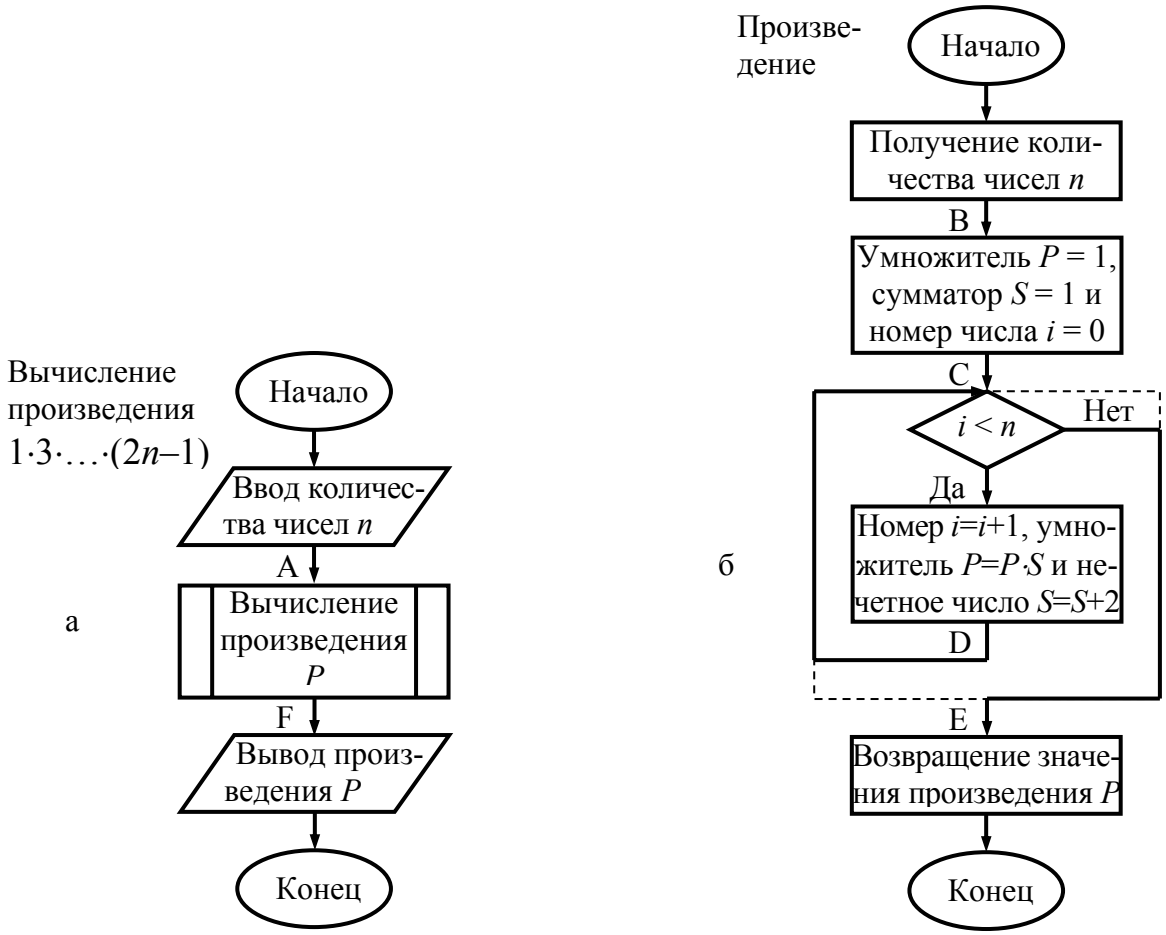


Рис. 37. Вычисление произведения первых n нечетных натуральных чисел, версия 2: а) главный алгоритм; б) функция

Таблица 18. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 5$
B	$n = 5$
C	$n = 5, i = 0, P = 1, S = 1$
D	$n = 5, i = 1, P = 1, S = 3$
D	$n = 5, i = 2, P = 3, S = 5$
D	$n = 5, i = 3, P = 15, S = 7$
D	$n = 5, i = 4, P = 105, S = 9$
D	$n = 5, i = 5, P = 945, S = 11$
E	$n = 5, i = 5, P = 945, S = 11$
F	$n = 5, i = 5, P = 945, S = 11$

23.5*. Нахождение произведения сумм чисел

Задача. Найти произведение следующих *n* чисел в виде суммы $(1 + 1/10) \cdot (-1 - 1/100) \cdot (1 + 1/1000) \cdot (-1 - 1/10000) \cdot \ldots$

Решение.

Анализ задачи.

1. Формулы. Нужно найти произведение *n* первых чисел, представляющих собой некоторые суммы. Не обязательно знать, чему равен *n*-й сомножитель. Вычисление суммы, которую представляет собой *n*-й сомножитель, можно организовать с помощью специального сумматора, как и в задаче 23.4.

Итак, нужно просто найти произведение *n* чисел

$$(1 + 1/10) \cdot (-1 - 1/100) \cdot (1 + 1/1000) \cdot (-1 - 1/10000) \cdot \ldots \qquad (8)$$

2. Входные данные. Из комментария к сумме (8) следует, что входным данным является количество сомножителей *n*.

3. Результат. Результатом работы алгоритма является вывод вычисленного произведения (8).

Проектирование алгоритма.

Алгоритм совпадает с предыдущим алгоритмом 23.4, исключая функцию, которая имеет немного другой вид. Эта функция также не содержит формулу, задающую текущий множитель.

Для этого заметим, что сомножители имеют вид $1 + 0{,}1$, $-(1 + 0{,}01)$, $1 + 0{,}001$, $-(1 + 0{,}0001)$. Отсюда следует, что:

1) второе слагаемое суммы каждый раз умножается на 0,1;

2) весь сомножитель каждый раз умножается на –1.

Поэтому получить следующий сомножитель можно, умножив его второе слагаемое на 0,1 и затем умножив весь сомножитель на –1. Лучше понять это поможет трассировка алгоритма.

Теперь алгоритм функции можно записать в следующем виде:

1) присвоить умножителю всего произведения начальное значение $P = 1$;

2) присвоить умножителю второго слагаемого начальное значение $P_1 = 1$;

3) присвоить умножителю сомножителя начальное значение $P_2 = 1$;

4) присвоить номеру числа предварительное значение $i = 0$;

5) проверить условие i < n. Если оно справедливо, перейти к шагу 6). Иначе закончить алгоритм;

6) получить номер текущего сомножителя операцией $i = i + 1$. Теперь этот номер нужен только для подсчета количества проходов цикла;

7) умножить умножитель второго слагаемого 0,1: $P_1 = P_1 \cdot 0{,}1$;

8) умножить умножитель всего произведения на текущий сомножитель: $P = P \cdot (P_2 \cdot (1 + P_1))$;

9) умножить умножитель сомножителя на –1: $P_2 = P_2 \cdot (-1)$. Обратите внимание, что следующее нечетное число вычисляется *после* умножения;

10) перейти к шагу 5).

§ 24. Массивы

24.1. Максимум данных положительных чисел 1

Задача. Найти максимальное число из n данных положительных чисел.

Решение.

Анализ задачи.

1. Формулы. Данные n чисел обозначим через k_i, где $i = 1, 2, 3, \ldots, n$. Итак, нужно найти максимальное среди положительных чисел $k_1, k_2, \ldots, k_n$.

2. Входные данные. Из условия задачи вытекает, что входными данными являются количество чисел n, а также все эти n чисел k_i, $i = 1, 2, 3, \ldots, n$.

3. Результат. Результатом работы алгоритма является вывод одного числа и его номера во входной последовательности чисел. В программировании последовательности называются ***массивами***.

Проектирование алгоритма.

Детализируем общую блок-схему на рис. 28:

1) напишем входные данные n и k_i, $i = 1, 2, 3, \ldots, n$, в явном виде;

2) выведем максимальное число *max* и его номер m;

3) алгоритм вычисления максимального числа *max* и его номера m выделим в отдельный модуль. В этом модуле используем цикл для поиска максимального числа.

В качестве начального значения максимума m нужно взять число, которое заведомо меньше всех данных. Поскольку данные числа положительные, то возьмем начальное значение $max = 0$.

Далее в цикле от 1 до n данные числа по очереди сравниваются с числом *max*, и если текущее число больше *max*, то *max* присваивается значение этого числа. При этом также запоминается номер m этого числа в массиве.

Результат проектирования представлен на рис. 38.

Тестирование алгоритма.

Расставим контрольные точки A—G (см. рис. 38). Протестируем алгоритм, когда максимальное число стоит первым, последним и в середине.

Тест 1. Пусть $n = 5$, $k_1 = 10$, $k_2 = 8$, $k_3 = 7$, $k_4 = 3$, $k_5 = 9$. Тогда должно получиться $max = 10$, $m = 1$.

Тест 2. Пусть $n = 5$, $k_1 = 1$, $k_2 = 8$, $k_3 = 7$, $k_4 = 3$, $k_5 = 9$. Тогда должно получиться $max = 9$, $m = 5$.

Тест 3. Пусть $n = 5$, $k_1 = 1$, $k_2 = 8$, $k_3 = 7$, $k_4 = 3$, $k_5 = 2$. Тогда должно получиться $max = 8$, $m = 2$.

Внимательное пошаговое изучение трассировки алгоритма поможет понять, что происходит при выполнении этого алгоритма.

Из анализа таблиц 19—21 можно также сделать заключение, что построенный алгоритм работает правильно.

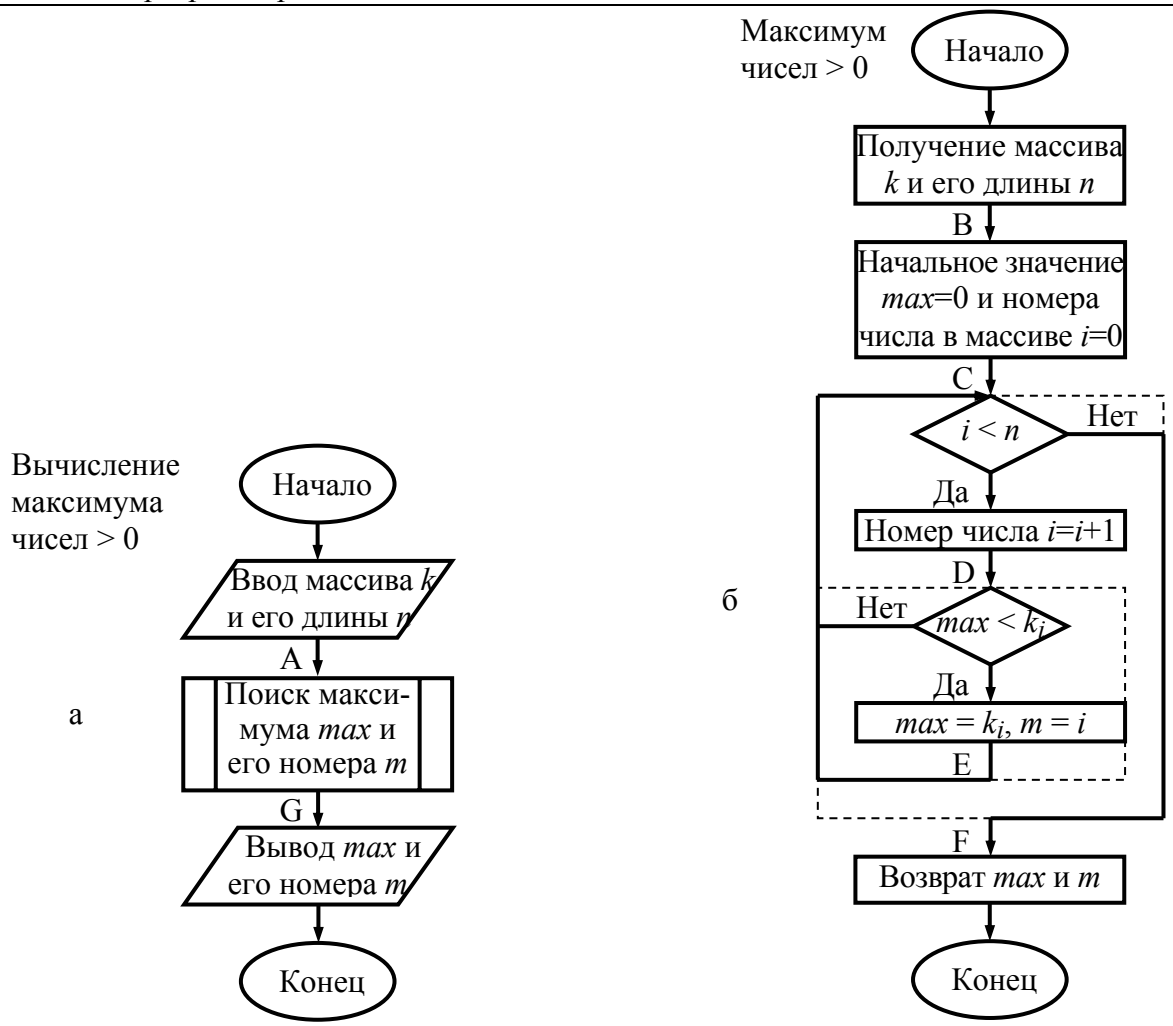


Рис. 38. Вычисление максимального числа в массиве: а) главный алгоритм; б) функция

Таблица 19. Значения данных в контрольных точках: тест 1

Контрольная точка	Значения данных
A	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9$
B	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9$
C	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 0, i = 0$
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 0, i = 1$
E	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 10, i = 1, m = 1$
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 10, i = 2, m = 1$
E	—
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 10, i = 3, m = 1$
E	—
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 10, i = 4, m = 1$
E	—
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 10, i = 5, m = 1$
E	—
F	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 10, i = 5, m = 1$
G	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 10, m = 1$

Таблица 20. Значения данных в контрольных точках: тест 2

Контрольная точка	Значения данных
A	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9$
B	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9$
C	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 0, i = 0$
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 0, i = 1$
E	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 1, i = 1, m = 1$
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 1, i = 2, m = 1$
E	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 8, i = 2, m = 2$
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 8, i = 3, m = 2$
E	—
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 8, i = 4, m = 2$
E	—
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 8, i = 5, m = 2$
E	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 9, i = 5, m = 5$
F	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 9, i = 5, m = 5$
G	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, max = 9, m = 5$

Таблица 21. Значения данных в контрольных точках: тест 3

Контрольная точка	Значения данных
A	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2$
B	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2$
C	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 0, i = 0$
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 0, i = 1$
E	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 1, i = 1, m = 1$
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 1, i = 2, m = 1$
E	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 8, i = 2, m = 2$
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 8, i = 3, m = 2$
E	—
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 8, i = 4, m = 2$
E	—
D	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 8, i = 5, m = 2$
E	—
F	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 8, i = 5, m = 2$
G	$n = 5, k_1 = 1, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 2, max = 8, m = 2$

24.2. Максимум данных положительных чисел 2

Задача. Найти максимальное число из *n* данных положительных чисел.

Решение.

Анализ задачи совпадает с предыдущим анализом.

Проектирование алгоритма.

Отличие от предыдущего алгоритма только в проектировании функции.

В качестве начального значения максимума не всегда удается взять число, заведомо меньшее всех данных. Но в качестве начального значения максимума всегда можно взять первое число массива, что мы и сделаем.

Далее в цикле от 2 до *n* данные числа по очереди сравниваются с числом *max*, и если текущее число больше *max*, то *max* присваивается значение этого числа. При этом также запоминается номер *m* этого числа в массиве.

Результат проектирования представлен на рис. 39.

Тестирование алгоритма совпадает с предыдущим тестированием.

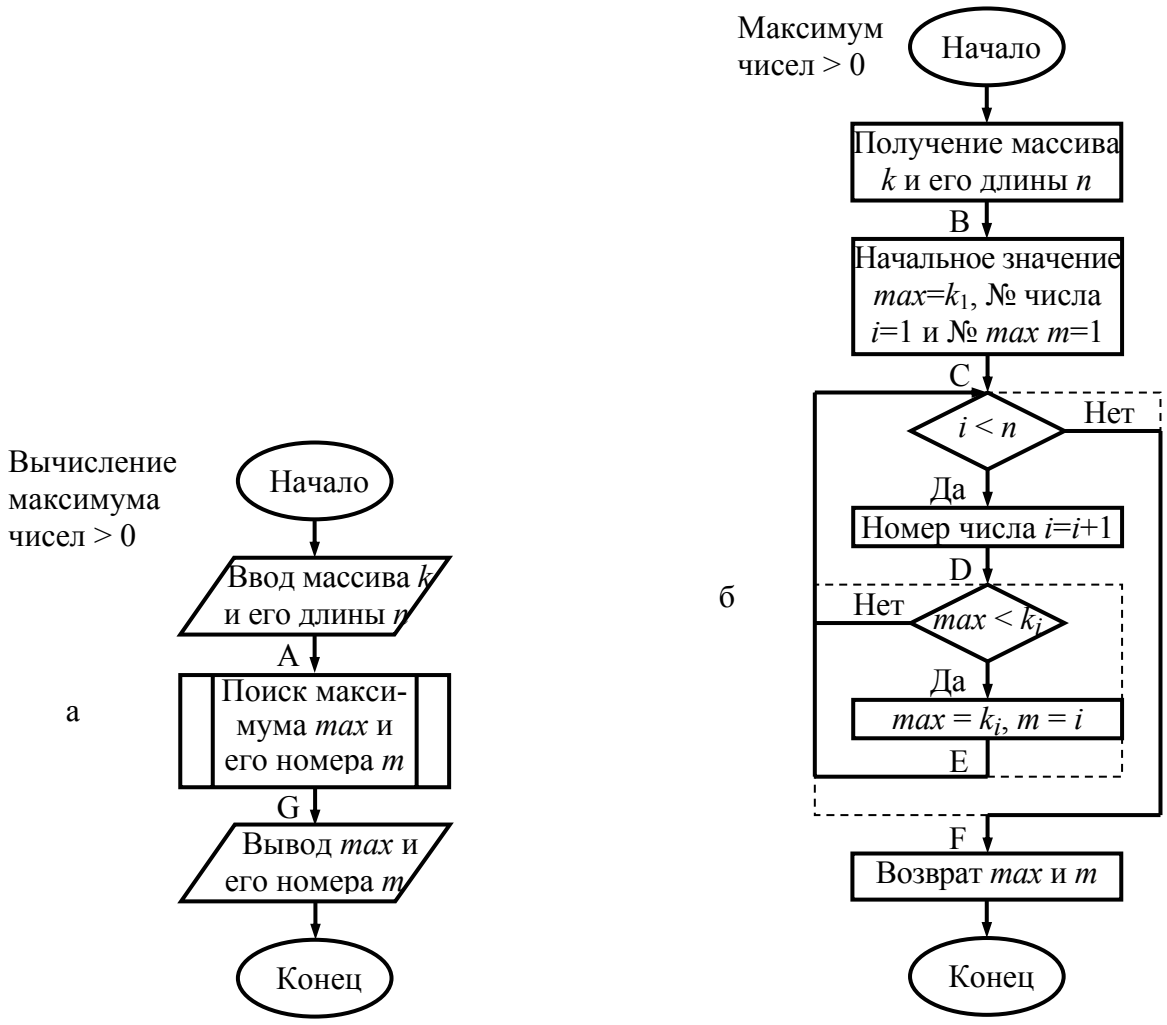


Рис. 39. Вычисление максимального числа в массиве: а) главный алгоритм; б) функция

24.3. Нахождение количества элементов массива по условию

Задача. Посчитать в массиве длины n количество положительных чисел.

Решение.

Анализ задачи.

1. Формулы. Данные n чисел обозначим через k_i, где $i = 1, 2, 3, \ldots, n$. Тогда нужно найти количество чисел $k_i > 0$.

2. Входные данные. Из условия задачи вытекает, что входными данными являются количество чисел n, а также все эти n чисел k_i, $i = 1, 2, 3, \ldots, n$.

3. Результат. Результатом работы алгоритма является вывод одного числа — количества положительных чисел массива.

Проектирование алгоритма.

Спроектируем функцию.

В цикле от 1 до n текущее число сравнивается с числом 0, и если число положительно, то счетчик увеличивается на единицу.

Результат проектирования представлен на рис. 40.

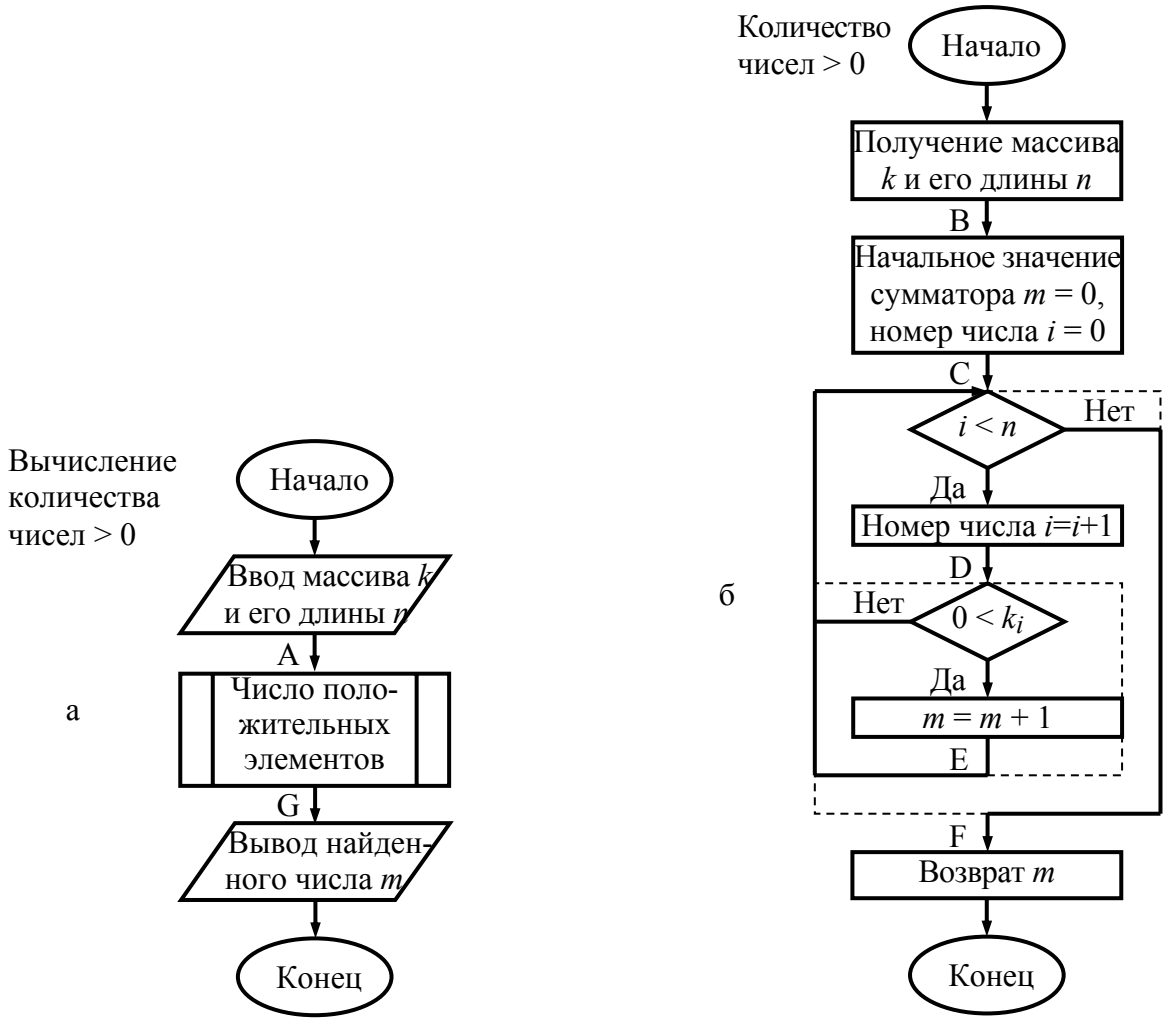


Рис. 40. Вычисление числа положительных элементов: а) главный алгоритм; б) функция

Тестирование алгоритма.

Расставим контрольные точки A—G (см. рис. 40). Протестируем алгоритм в трех разных случаях, а именно:

1) в массиве все положительные числа;

2) несколько положительных чисел;

3) совсем нет положительных чисел.

Тест 1. Пусть $n = 5$, $k_1 = 10$, $k_2 = 8$, $k_3 = 7$, $k_4 = 3$, $k_5 = 9$. Тогда должно получиться $m = 5$.

Тест 2. Пусть $n = 5$, $k_1 = –10$, $k_2 = 8$, $k_3 = —7$, $k_4 = 3$, $k_5 = 9$. Тогда должно получиться $m = 3$.

Тест 3. Пусть $n = 5$, $k_1 = 0$, $k_2 = –8$, $k_3 = –7$, $k_4 = –3$, $k_5 = –2$. Тогда должно получиться $m = 0$.

Внимательное пошаговое изучение трассировки алгоритма поможет понять, что происходит при выполнении этого алгоритма.

Из анализа таблиц 22—24 можно также сделать заключение, что построенный алгоритм работает правильно.

Таблица 22. Значения данных в контрольных точках: тест 1

Контрольная точка	Значения данных
A	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9$
B	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9$
C	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 0, i = 0$
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 0, i = 1$
E	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 1, i = 1$
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 1, i = 2$
E	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 2, i = 2$
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 2, i = 3$
E	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 3, i = 3$
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 3, i = 4$
E	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 4, i = 4$
D	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 4, i = 5$
E	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 5, i = 5$
F	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 5, i = 5$
G	$n = 5, k_1 = 10, k_2 = 8, k_3 = 7, k_4 = 3, k_5 = 9, m = 5, i = 5$

Таблица 23. Значения данных в контрольных точках: тест 2

Контрольная точка	Значения данных
A	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9$
B	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9$
C	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 0, i = 0$
D	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 0, i = 1$
E	—
D	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 0, i = 2$
E	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 1, i = 2$
D	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 1, i = 3$
E	—
D	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 1, i = 4$
E	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 2, i = 4$
D	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 2, i = 5$
E	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 3, i = 5$
F	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 3, i = 5$
G	$n = 5, k_1 = -10, k_2 = 8, k_3 = -7, k_4 = 3, k_5 = 9, m = 3, i = 5$

Таблица 24. Значения данных в контрольных точках: тест 3

Контрольная точка	Значения данных
A	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2$
B	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2$
C	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2, m = 0, i = 0$
D	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2, m = 0, i = 1$
E	—
D	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2, m = 0, i = 2$
E	—
D	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2, m = 0, i = 3$
E	—
D	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2, m = 0, i = 4$
E	—
D	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2, m = 0, i = 5$
E	—
F	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2, m = 0, i = 5$
G	$n = 5, k_1 = 0, k_2 = -8, k_3 = -7, k_4 = -3, k_5 = -2, m = 0, i = 5$

24.4. Суммирование квадратов элементов массива

Задача. Просуммировать квадраты чисел массива длины n.

Решение.

Анализ задачи.

1. Формулы. Данные n чисел обозначим через k_i, где $i = 1, 2, 3, \ldots, n$. Тогда нужно найти сумму чисел $(k_i)^2$.

2. Входные данные. Из условия задачи вытекает, что входными данными являются количество чисел n, а также все эти n чисел k_i, $i = 1, 2, 3, \ldots, n$.

3. Результат. Результатом работы алгоритма является вывод одного числа — суммы квадратов чисел массива.

Проектирование алгоритма.

Проектирование главного алгоритма не представляет трудностей.

Спроектируем функцию.

В цикле от 1 до n квадрат текущего числа складывается с сумматором.

Результат проектирования представлен на рис. 41.

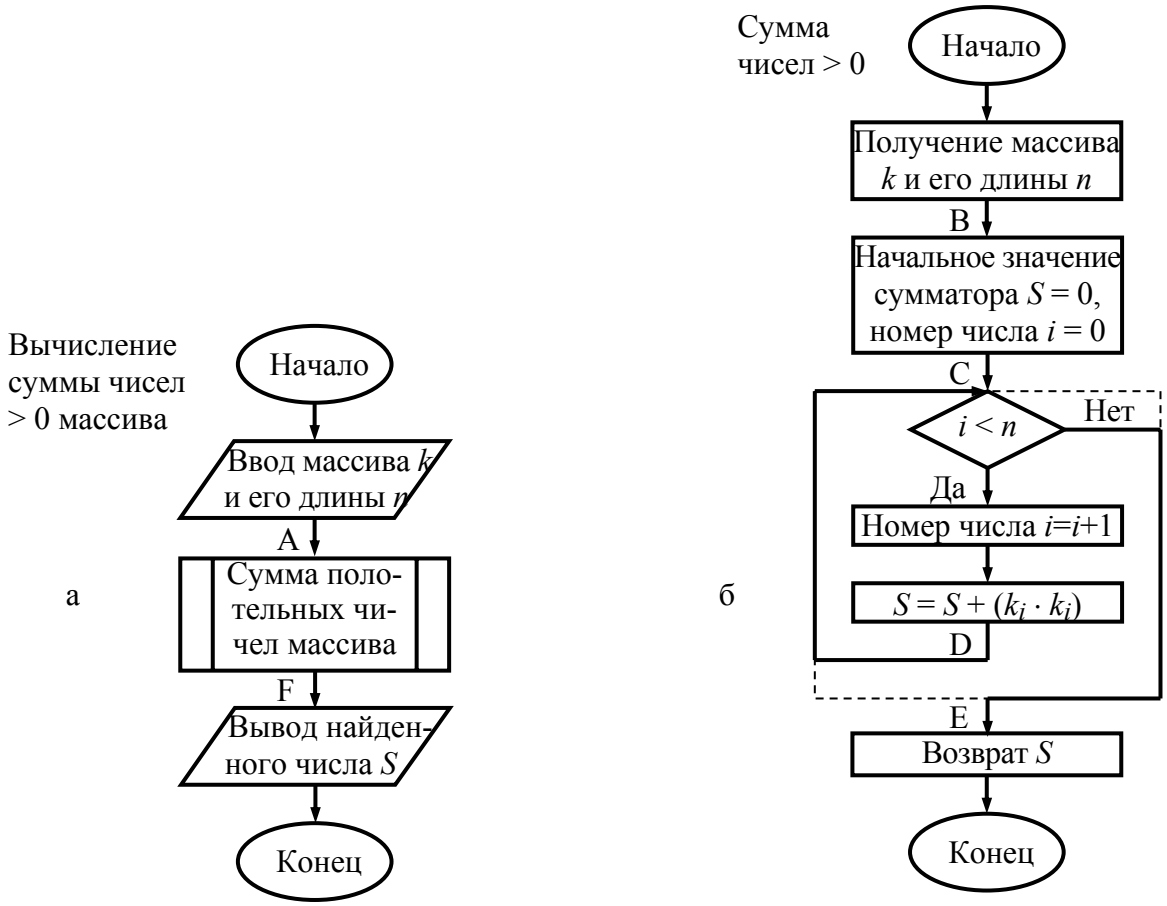


Рис. 41. Вычисление суммы элементов массива: а) главный алгоритм; б) функция

Тестирование алгоритма.

Расставим контрольные точки A—F (рис. 41). Протестируем алгоритм при $n = 5$, $k_1 = -2$, $k_2 = -1$, $k_3 = 0$, $k_4 = 1$, $k_5 = 2$. Должно получиться $S = 10$.

Из таблицы 25 следует, что построенный алгоритм работает правильно.

Таблица 25. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2$
B	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2$
C	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2, S = 0, i = 0$
D	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2, S = 4, i = 1$
D	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2, S = 5, i = 2$
D	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2, S = 5, i = 3$
D	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2, S = 6, i = 4$
D	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2, S = 10, i = 5$
E	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2, S = 10, i = 5$
F	$n = 5, k_1 = -2, k_2 = -1, k_3 = 0, k_4 = 1, k_5 = 2, S = 10, i = 5$

24.5*. Суммирование четных чисел массива

Задача. Просуммировать четные числа положительного массива длины n.

Решение.

Анализ задачи.

1. Формулы. Данные n чисел обозначим через k_i, где $i = 1, 2, 3, \ldots, n$. Тогда нужно найти сумму четных чисел k_i, *если они есть*. Положительное число x является четным, если $x = 2 \cdot \text{int}(x/2)$, где int — стандартная математическая функция вычисления целой части числа.

2. Входные данные. Из условия задачи вытекает, что входными данными являются количество чисел n, а также все эти n чисел k_i, $i = 1, 2, 3, \ldots, n$.

3. Результат. Результатом работы алгоритма являются два случая: 1) если есть хотя бы одно четное число, то вывод суммы четных чисел массива; 2) если четных чисел нет, то вывод сообщения об этом.

Проектирование алгоритма.

В алгоритме можно предусмотреть также вычисление количества четных элементов массива. Тогда в главном алгоритме при выводе результата проверяется условие, равно ли нулю количество четных элементов: если равно, то выводится сообщение об этом, если нет — сумма четных чисел.

В цикле от 1 до n текущее положительное число проверяется на четность, и в случае успеха накапливаются сумма и количество четных чисел.

Упражнения

1. Построить и протестировать алгоритм нахождения среднего арифметического двух чисел.

2. Построить и протестировать алгоритм определения модуля числа.

3. Построить и протестировать алгоритм определения четности числа. Использовать то, что при делении целых чисел получается целое число.

4. Построить и протестировать алгоритм определения знака числа.

5. Построить и протестировать алгоритм сравнения двух чисел.

6*. Расположить три числа в порядке возрастания.

7. Построить и протестировать алгоритм нахождения суммы первых n нечетных чисел.

8. Построить и протестировать алгоритм нахождения суммы первых n степеней двойки, начиная с нулевой степени.

9. Построить и протестировать алгоритм нахождения суммы n чисел

$$4/(1 \cdot 2 \cdot 3) + 4/(2 \cdot 3 \cdot 4) + 4/(3 \cdot 4 \cdot 5) + \ldots$$

10. Построить и протестировать алгоритм нахождения суммы n чисел

$$4 - 4/3 + 4/5 - 4/7 + \ldots$$

11. Найти произведение n чисел

$$\big(1 + 1/2\big) \cdot \big(1 + (1 \cdot 3)/(2 \cdot 4)\big) \cdot \big(1 + (1 \cdot 3 \cdot 5)/(2 \cdot 4 \cdot 6)\big) \cdot \ldots$$

12*. Найти произведение n чисел

$$\big(1 + 1/2\big) \cdot \big(1 - (1 - 3)/(2 \cdot 4)\big) \cdot \big(1 + (1 - 3 + 5)/(2 \cdot 4 \cdot 6)\big) \cdot \ldots$$

13. Построить и протестировать алгоритм нахождения минимального числа из массива длиной n.

14. Построить и протестировать алгоритм нахождения среднего арифметического чисел массива длиной n.

15. Построить и протестировать алгоритм нахождения среднего арифметического чисел массива длиной n с нечетными номерами.

16. Построить и протестировать алгоритм нахождения числа с максимальным синусом из массива длиной n.

17. Построить и протестировать алгоритм обнуления отрицательных чисел массива длиной n.

18*. Найти максимальное число и его номер в массиве длиной n. Если максимальных чисел несколько, вывести все их номера.

19*. Найти в массиве длиной n наименьшее нечетное число.

20*. Найти в массиве длиной n два одинаковых числа.

21*. Отсортировать массив n чисел.

Глава 9. Кодирование

§ 25. Составные части программы

25.1. Язык программирования Паскаль

Одним из популярных языков является язык программирования Паскаль, названный в честь французского математика XVII века Блеза Паскаля, который считается первым конструктором механического калькулятора. Язык программирования Паскаль был разработан в начале 70-х годов прошлого века профессором Цюрихского университета Никлаусом Виртом.

Язык Паскаль сделал языки программирования более легкими для изучения и использования, а компьютеры более доступными для массового пользователя.

Описание любого языка программирования — это прежде всего описание его синтаксиса. Дадим краткое описание основных структур, которые понадобятся нам при написании программ на языке Паскаль.

25.2. Описание переменных

Листинг, т. е. исходный текст, любой программы, написанной на языке Паскаль содержит два раздела.

1. Раздел ***описания переменных, или раздел описаний***, который начинается с зарезервированного слова **var**. В нем объявляются переменные. Объявление переменных заканчивается знаком точка с запятой **;**.

Это слово произошло от английского слова «variable» — «переменная».

25.3. Тело программы

2. Затем идет ***тело программы, или раздел операторов*** — основной текст программы. Тело программы является записью на языке Паскаль алгоритма задачи, которую решает программа. Тело программы обрамляется парой ***логических, или инструктивных, скобок***: **begin**—**end**. После скобки «end» главной программы, кодирующей главный алгоритм, обязательно ставится точка. Больше нигде, ни в каких функциях, точка не ставится.

Слово «begin» по-английски означает «начало», «end» — «конец».

25.4. Блок

Когда строчки языка программирования выполняются последовательно, то они составляют ***блок***. Любой блок обрамляется логическими скобками **begin**—**end** и обычно заканчивается точкой с запятой **;**.

Блок может включать также другие элементарные алгоритмы «выбор» и «цикл». Все составляющие блока выполняются компьютером последовательно, друг за другом.

25.5. Комментарий

В любом месте программы могут находиться комментарии. ***Комментарий*** на Паскале — это любой текст, в том числе и на русском языке, обрамленный открывающейся и закрывающейся фигурными скобками { и }.

Комментарий вместе со своими фигурными скобками игнорируется транслятором Паскаля и никак *не влияет на выполнение программы*. Комментарий служит для пояснения действий программы.

25.6. Общий вид программы

Итак, структура программы на Паскале имеет следующий вид.

Листинг 1. Структура программы на Паскале

```
  {Первые две строки - это комментарии}
  {Секция описания переменных}
var
  {Здесь должны находиться описания переменных}

  {Тело программы}
  {Логические скобки, обрамляющие блок программы}
begin
  {Здесь должны находиться операторы программы}
end.
  {В конце программы стоит точка}
```

25.7. Пример простейшей программы

Приведем пример простейшей программы на языке Паскаль. Эта программа ничего не делает. Поэтому в ней отсутствуют переменные. В этом исключительном тривиальном случае секцию описания переменных можно опустить.

Пустое тело программы, не содержащее ни одного оператора, обрамляется логическими скобками begin—end. После скобки end стоит точка.

Листинг 2. Простейшая пустая программы на Паскале

```
  {Эта программа ничего не делает}
begin
end.
```

§ 26. Переменные

26.1. Переменная

Кодирование, или написание программы на языке программирования, представляет собой управление двумя синтаксическими объектами.

1. В алгоритмах и программах обязательно содержатся переменные, которые являются именами данных.

2. Действия над переменными кодируются операторами.

Переменные в языке Паскаль похожи на буквенные обозначения в алгебре. Они имеют имена, которые придумывают сами программисты.

Переменная — это именованная область в оперативной памяти компьютера. Эта область выделяется для хранения данных программы.

Значением переменной является содержание памяти компьютера, именованной этой переменной. Значение переменной может изменяться в ходе выполнения программы.

26.2. Имя переменной

Переменная как именованная область памяти имеет имя. ***Имя, или идентификатор, переменной*** подчиняется следующим трем правилам.

1. *Обязательное правило*. Все идентификаторы в головной программе уникальны, т. е. имя переменной дается только одной переменной и не повторяется. Внутри модулей, оформленных как функции, находятся свои уникальные переменные.

2. *Рекомендуемое правило*. Имя переменной (как и имя файла) отражает ее значение.

3. *Правило вкуса*. Длина имени не превышает 8 символов (как и имени файла в стандарте 8.3).

Трансляторы языка программирования обычно накладывают на имена переменных следующие четыре жестких ограничения:

1) в имени могут использоваться только латинские буквы, цифры и знак подчеркивания;

2) знак пробела недопустим;

3) первым символом не может быть цифра;

4) имя не может совпадать с зарезервированными ***ключевыми словами*** Паскаля, входящими в состав языка (такими, как var, begin, end и т. д.).

26.3. Тип переменной

Переменные бывают разного типа. ***Тип переменной*** определяет следующие ее характеристики:

1) количество байт, которое эта переменная будет занимать в памяти;

2) набор операций, который можно производить над этой переменной в программе.

Самые элементарные типы переменных называются ***стандартными***. Стандартные типы переменных — это кирпичики программирования, из которых строятся более сложные структуры данных.

К ***стандартным типам*** переменных относятся:

1) логические значения;
2) целые типы;
3) вещественные типы;
4) множество литер, или символов, выводимые на экран или печать.

26.4. Целые числа

Число, записанное в листинге программы в явном виде, называется ***константой***. Перед константой можно записывать положительный или отрицательный знак числа. пробелы в константах не допускаются.

Целые числа, или целые константы, записываются в Паскале так же, как и в математике.

Примеры положительных целых констант: 12345, 001, +128, +5.

Примеры отрицательных целых констант: –777, –03.

26.5. Целые переменные

Первое, что делает программист при написании программы, это задает переменные. Опишем, как их задавать.

Целочислительный тип обозначает конечное множество целых чисел, действующих на данной вычислительной машине. Целочислительный тип называется типом ***integer***. Слово «integer» в переводе с английского означает «целое число».

Три переменные типа **integer** можно объявить в разделе описания переменных следующим образом. По традиции целым переменным могут отводиться имена, начинающиеся с латинских букв i—n.

Листинг 3. Примеры объявления целых переменных

```
  {Целые переменные i, j и n можно объявить так}
var i,j,n : integer;
```

```
  {Целые переменные i, j и n можно объявить и так}
var i : integer;
    j : integer;
    n : integer;
```

26.6. Вещественные числа

Вещественные числа, или вещественные константы, записываются в Паскале по-разному, но всегда дробная часть отделяется от целой *точкой*. Нуль перед или после точки можно опустить.

«Маленькие» ***константы с фиксированной точкой*** записываются так же, как и в английской математике.

Примеры констант с фиксированной точкой:

123.45, 0.01, +128., –.5.

«Большие» ***константы с плавающей точкой*** записываются с использованием английской буквы e. Буква e означает степень десяти. Знак плюс или минус может находиться как непосредственно перед числом, так и перед показателем степени десяти.

Примеры констант с плавающей точкой:

$1.23e45 = 1{,}23 \times 10^{45}$;

$+0.01e–011 = 0{,}01 \times 10^{-11}$;

$–128.e+32 = –128{,}0 \times 10^{32}$.

$9.5e–34 = 9{,}5 \times 10^{-34}$.

26.7. Вещественные переменные

Вещественный тип называется типом ***real***. Вещественный тип определяет в машине вещественные числа.

Слово «real» в переводе с английского означает «вещественное число».

Три переменные типа **real** можно объявить в разделе описания переменных следующим образом. По традиции вещественным переменным могут отводиться имена, начинающиеся с латинских букв a—h и o—z.

Листинг 4. Примеры объявления вещественных переменных

```
  {Вещественные переменные a, t и x объявляются так}
var a,t,x : real;
```

```
  {Вещественные переменные объявляются и так}
var a : real;
    t : real;
    x : real;
```

26.8. Массив

Структурой называется составной тип данных, составленный из стандартных типов данных.

Самой распространенной структурой, реализованной во всех языках программирования, является массив. ***Массив*** состоит из ***элементов, или компонент,*** которые все имеют один и тот же стандартный тип integer или real. Номер элемента массива называется ***индексом***.

Программа при выполнении может сразу получить нужный ей элемент массива сразу по номеру, что является прямым доступом к памяти.

26.9. Объявление массива

Массив в листинге объявляется так же, как и переменная, только перед типом добавляется конструкция ***array [0..n] of***, где 0 и *n* — соответственно первый и последний индексы массива, разделенные двумя точками.

Английское слово «array» означает «ряд».

Для работы с массивом нужен цикл.

По два массива типа **integer** и типа **real** можно объявить в разделе описания переменных следующим образом. По традиции имена массивов могут начинаться с тех же букв, что и имена переменных: целых — с латинских букв i—n, вещественных — с латинских букв a—h и o—z.

Листинг 5. Примеры объявления целых и вещественных массивов длиной 10

```
  {Массивы с индексами от 0 до 9 объявляются так}
var k,l : array [0..9] of integer;
var v,w : array [0..9] of real;
```

```
  {Массивы с индексами от 0 до 9 объявляются и так}
var k,l : array [0..9] of integer;
    v,w : array [0..9] of real;
```

```
  {Массивы с индексами от 0 до 9 объявляются и так}
var k : array [0..9] of integer;
    l : array [0..9] of integer;
    v : array [0..9] of real;
    w : array [0..9] of real;
```

§ 27. Операторы и выражения

27.1. Оператор

Оператор, или инструкция,— это конструкция языка программирования, служащая для задания какого-либо действия или последовательности действий в программе над данными. Те объекты, с которыми манипулирует оператор, называются его ***операндами***. Количество операндов у операторов бывает разное, в том числе и нуль. Операторы записываются в теле программы между логическими скобками **begin—end**.

Результаты работы каждой программы должны быть выведены для просмотра человеком. И каждая программа должна иметь какие-то входные данные.

В наших простейших программах ввод данных будет осуществляться операторами присваивания. Поэтому специальные операторы ввода рассматривать не будем.

Представим только оператор вывода.

27.2. Операторы вывода

В языке Паскаль существуют две стандартные операции вывода данных на экран: **write** и **writeln**. Первая выводит на экран указанные в ней данные, расположенные в скобках после слова write, и все. Вторая после вывода указанных данных посылает на дисплей символ абзаца, или конца строки, в результате чего следующий вывод начинается с новой строки.

Слово «write» в переводе с английского означает «записать».

Теперь можно привести пример действительно простейшей программы на языке Паскаль, просто выводящей на дисплей одно сообщение. Это сообщение записано ***символьной константой***, которая представляет собой символы, заключенные в апострофы.

Листинг 6. Пример простейшей программы

```
  {Программа выводит одно сообщение}
  {В скобках находится символьная константа}
  {После вывода сообщения программа перейдет
   на новую строку}
begin
  writeln ('Это программа на Паскале. Ура!');
end.
```

27.3. Оператор присваивания

Присваивание — процесс засылки значения в переменную. Присваивание значения переменной осуществляется при помощи специальной конструкции языка — ***оператора присваивания***.

В Паскале оператор присваивания:

1) представляет из себя двоеточие с равенством **:=**;
2) имеет следующие два операнда, разделенные этим знаком операции:
   а) в левой части оператора присваивания стоит имя переменной, *которой* присваивается значение;
   б) в правой части — то значение, *которое* присваивается (например: $y := \sin(x)/\cos(x)$).

Обратите внимание, что в математике, когда вводят обозначения, дела обстоят наоборот: слева от знака равенства стоит обозначаемое, справа — обозначающая переменная (например: пусть $(1 + \operatorname{tg} x)/(1 - \operatorname{tg} x) = y$).

Заканчивается оператор присваивания точкой с запятой **;**.

Операторы присваивания выполняются последовательно согласно алгоритму «следование».

Приведем примеры операторов присваивания.

Листинг 7. Примеры операторов присваивания

```
  {Переменной n присваивается значений 10}
  {Переменной x присваивается то значение,
   которое имеет переменная y}
n := 10;
x := y;
```

С помощью оператора присваивания можно, например, поменять значения двух переменных с помощью третьей. Это весьма распространенный стандартный прием программирования. В примере меняются значениями переменные *x* и *y*.

Листинг 8. Примеры операторов присваивания

```
  {Переменные x и y обмениваются своими значениями}
z := x;
x := y;
y := z;
```

27.4. Выражение

Выражение — формула для вычисления результата. Выражение состоит из операторов и операндов — переменных или чисел.

Если выражение содержит несколько операторов, обычно указывается порядок их выполнения с помощью круглых скобок.

Выражения бывают двух видов:

1) ***арифметическими***, вычисляющими числовой результат;

2) ***логическими***, вычисляющими логический результат.

27.5. Арифметическое выражение

В языке Паскаль есть следующие ***арифметические операции***:

1) ***сложения*** +; 2) ***вычитания*** –; 3) умножения *; 4) деления /.

Как и оператор присваивания, они двуместны, т. е. имеют два операнда. Арифметическое выражение записывается по правилам, похожими на алгебраические, но со следующими отличиями.

1. Арифметическое выражение записывается в одну строку.
2. Знаки действий не пропускаются.
3. Применяются только круглые скобки.

Арифметическое выражение стоит справа от оператора присваивания.

Листинг 9. Пример операции суммирования

```
  {Переменные i и j суммируются в переменную k
   и затем выводятся на экран в разных строках}
var i,j,k : integer;
begin
  i := 2;
  j := 3;
  k := i + j;
  writeln ('Переменная i = ', i);
  writeln ('Переменная j = ', j);
  writeln ('Переменная k = i + j = ', k);
end.
```

27.6. Логическое выражение

В Паскале имеются и ***логические выражения***, которые могут принимать только два значения: ИСТИНА и ЛОЖЬ. Логические выражения могут состоять из операций сравнения и логических операций.

Достаточно использовать три двуместные ***операции сравнения***:

1) ***равно*** =; 2) ***меньше*** <; 3) ***больше*** >.

Например, выражение 1 < 5 истинно, а 5 < 5 ложно.

Также опишем три ***логические операции***.

1. ***Отрицание*** **not**. Отрицание истины является ложью, а отрицание лжи — истиной. Например, выражение **not** (1 < 5) ложно, а **not** (5 < 5) истинно.

2. ***Логическое «и»*** **and**. Связывает два или более операндов. Выражение с логическим «и» истинно, если истинны все операнды. Например, выражение (1 < 5) **and (**2 < 5) истинно, (1 < 5) **and (**2 < 5) **and (**5 < 5) ложно, а выражение (1 < 5) **and (**2 < 5) **and not (**5 < 5) снова истинно.

3. ***Логическое «или»*** **or**. Связывает два или более операндов. Выражение с логическим «или» истинно, если истинен хотя бы один операнд. Например, выражение (1 < 5) **or (**2 < 5) истинно, (1 < 5) **or** (2 < 5) **or** (5 < 5) тоже истинно, а вот выражение **not** (1 < 5) **or not (**2 < 5) **or (**5 < 5) наконец ложно.

27.7. Функция

При разбиении программы на независимые модули эти модули называются ***подпрограммами***. ***Функция*** — вид подпрограммы, одна из основных логических единиц программы на Паскале. Функция в программировании аналогична функции в математике.

Представим правила использования функции на примере программы.

Листинг 10. Пример программы с функцией

```
  {Описание глобальных переменных}
var n,n2 : integer;

  {Функция вычисления квадрата целого числа}
  {Формальную переменную k "видит" только функция}
function kvadrat (var k : integer) : integer;
begin
  kvadrat := k*k;
end;

{Головная программа}
begin
  n := 10;
  n2 := kvadrat(n);
  write ('Квадрат целого числа ', n);
  writeln (' равен ', n2);
end.
```

В Паскале программист либо *задает* свою функцию, либо использует готовую ***встроенную, или стандартную***, всегда имеющуюся в Паскале Примеры стандартных функций: квадратный корень числа **sqrt** (*x*); экспонента числа **exp** (*x*); натуральный логарифм числа **ln** (*x*); синус числа **sin** (*x*); косинус числа **cos** (*x*) (см. прил. 3).

27.8. Задание функции

В Паскале задание функции начинается с заголовка, состоящего из:

1) слова **function**; 2) имени функции; 3) описания в скобках входных переменных; 4) описания типа функции.

Затем может быть секция описания переменных и, наконец, тело функции между скобками **begin—end**, заканчивающимися точкой с запятой.

Переменные, указанные в скобках при *описании* функции, называются ***формальными аргументами***.

При *вызове* функции аргументы называются ***фактическими***.

Функция может использоваться во всех случаях, когда используется обычная переменная, за исключением левой части оператора присваивания.

27.9. Условный оператор

Условный оператор полностью соответствует алгоритму «выбор» и имеет следующий вид:

```
if <выражение> then <оператор 1> else <оператор 2>;
```

Здесь <выражение> — логическое выражение, <оператор 1> — оператор, выполняющийся, когда <выражение> истинно, а оператор <оператор 2> выполняется, когда значение выражения <выражение> ложно.

Обратите внимание, что точка с запятой перед словом **else** не ставится.

Слово «if» по-английски означает «если», «then» — «то», «else» — иначе, поэтому этот оператор называется оператором «если—то—иначе».

Листинг 11. Пример программы с условным оператором

```
  {Определение положительности целого числа}
var n : integer;
begin
  n := 1;
  if (n > 0) then
    writeln ('Переменная n положительна')
  else
    writeln ('Переменная n не положительна');
end.
```

Иногда, при отсутствии последней части условного оператора, используется его редуцированная форма

```
if <выражение> then <оператор 1>;
```

27.10. Цикл

Цикл воплощает всю мощь компьютера, способного без устали совершать огромное количество одних и тех же действий.

Оператор цикла полностью соответствует алгоритму «цикл» и имеет следующий вид:

```
while <выражение> do <оператор>;
```

Здесь <выражение> — логическое выражение, <оператор> — оператор, выполняющийся, пока <выражение> истинно.

Слово «while» по-английски означает «пока», «do» — делать.

Зацикливание — семантическая ошибка, при которой внутри цикла изменение значений переменных никогда не приводит <выражение> к ложному значению. При этом оператор, стоящий после ключевого слова **do**, выполняется до бесконечности.

Найдем факториал заданного числа ($n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$).

Листинг 12. Пример программы с циклом

```
  {Вычисление факториала числа}
  {i - переменная цикла, m - умножитель}
var n,i,m : integer;
begin
  n := 7;
  i := 0;
  m := 1;
  while (i < n) do
  begin
    i := i+1;
    m := m*i;
  end;
  writeln (n,'! = ',m);
end.
```

Сущности не следует умножать без необходимости.

Бритва Оккама.

Запрограммировать можно все.
В любой программе есть ошибка.

Программистский фольклор.

Глава 10. Примеры программ

§ 28. Функции

28.1. Вычисление длины гипотенузы по катетам

Задача. Даны два вещественных числа, являющихся величинами катетов некоторого прямоугольного треугольника. Вычислить длину гипотенузы этого треугольника.

Решение.

В главе 8 «Примеры блок-схем» в одноименных разделах проведен анализ этой и всех следующих задач. Воспользуемся полученными в главе 8 алгоритмами и блок-схемами задач.

Все листинги наших простых программ будут состоять из трех частей.

1. Описание переменных головной программы.
2. Листинг функции.
3. Листинг головной программы.

В комментариях листинга программы сделаны важные замечания по поводу написания программы. Написание программы в нашем случае представляет собой перевод с языка блок-схемы на язык Паскаль. Причем листинг программы соответствует вышеприведенному тесту.

Листинг программы приведен на листинге 13.

Отладку программы можно проводить только на компьютере. Поэтому, если в листинге программы и встретятся какие-нибудь синтаксические ошибки, они не повлияют на тестирование. Потому что тестирование будем проводить также без компьютера.

В этом случае тестирование программы совпадает с тестированием блок-схемы. В листинге программы 13 расставлены контрольные точки, совпадающие с контрольными точками блок-схемы. Результат тестирования приведен в таблице 26. В ней же приведен текст, который программа выводит на дисплей при этом тесте.

Итак, программа вычислила значение гипотенузы 5, что совпадает с расчетным значением. Поскольку программа простая, то, скорее всего, она работает правильно.

Листинг 13. Программа вычисления длины гипотенузы по катетам

```
  {Вычисление длины гипотенузы по катетам}

  {Описание переменных головной программы}
  {Здесь собраны все переменные главного алгоритма}
  {Эти переменные должны быть вещественного типа}
var x,y,z : real;

  {Функция вычисления гипотенузы по катетам}
  {Возьмем для переменных функции и ее имени другие имена}
  {a и b - это катеты, gipoten - гипотенуза}
function gipoten (var a,b : real) : real;
  {Вычислим формулу Пифагора поэтапно}
  {Для этого введем вспомогательную переменную c}
var c : real;
  {a, b и c - внутренние переменные функции}
begin
  {Контрольная точка B}
  c := a*a + b*b;
  c := sqrt(c);
  gipoten := c;
  {Контрольная точка C}
end;

  {Головная программа задает катеты,
   вычисляет гипотенузу и выводит результат}
begin
  x := 3.0;
  y := 4.0;
  writeln ('Первый катет равен ', x);
  writeln ('Второй катет равен ', y);
  {Контрольная точка A}
  z := gipoten (x,y);
  {Контрольная точка D}
  write ('Гипотенуза равна ', z);
end.
```

Таблица 26. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$x = 3, y = 4$
B	$a = 3, b = 4$
C	$a = 3, b = 4, gipoten = 5$
D	$x = 3, y = 4, gipoten = 5, z = 5$

```
Первый катет равен 3
Второй катет равен 4
Гипотенуза равна 5
```

28.2. Вычисление площади треугольника по сторонам

Задача. Даны три вещественных числа, являющихся величинами сторон некоторого треугольника. Вычислить площадь этого треугольника.

Решение.

Листинг 14. Программа вычисления площади треугольника по сторонам

```
  {Вычисление площади треугольника по сторонам}

  {Описание переменных головной программы}
  {Здесь собраны все переменные главного алгоритма}
  {Эти переменные должны быть вещественного типа}
var a,b,c,s : real;

  {Функция вычисления площади треугольника по сторонам}
  {Возьмем для переменных функции и ее имени другие имена}
  {x, y и z - это стороны, plo - площадь}
function plo (var x,y,z : real) : real;
  {Вычислим формулу Герона поэтапно}
  {Для этого введем вспомогательную переменную p}
var p : real;
  {x, y, z и p - внутренние переменные функции}
begin
  {Контрольная точка B}
  p := (x + y + z)/2;
  p := p*(p - x)*(p - y)*(p - z);
  p := sqrt(p);
  plo := p;
  {Контрольная точка C}
end;

  {Головная программа задает стороны,
   вычисляет площадь треугольника и выводит результат}
begin
  a := 3.0;
  b := 4.0;
  c := 5.0;
  writeln;
  writeln ('Первая сторона = ', a);
  writeln ('Вторая сторона = ', b);
  writeln ('Третья сторона = ', c);
  {Контрольная точка A}
  s := plo (a,b,c);
  {Контрольная точка D}
  writeln ('Площадь треугольника = ', s);
  writeln;
end.
```

Таблица 27. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$a = 3, b = 4, c = 5$
B	$x = 3, y = 4, z = 5$
C	$x = 3, y = 4, z = 5, plo = 6$
D	$a = 3, b = 4, c = 5, plo = 6, s = 6$

```
Первая сторона = 3
Вторая сторона = 4
Третья сторона = 5
Площадь треугольника = 6
```

28.3. Определение вида треугольника по сторонам 1

Задача. Даны три вещественных числа, являющихся длинами сторон треугольника. Определить, является ли треугольник прямоугольным.

Решение.

Листинг 15. Программа определения вида треугольника по сторонам 1

```
  {Определение вида треугольника по сторонам 1}

  {Описание переменных головной программы}
  {Здесь собраны все переменные главного алгоритма}
  {Эти переменные должны быть вещественного типа}
var a,b,c : real;

  {Функция определяет вид треугольника по сторонам}
  {Возьмем для переменных функции и ее имени другие имена}
  {x, y и z - это стороны, vid - вид треугольника}
function vid (var x,y,z : real) : integer;
  {Введем вспомогательные переменные - квадраты сторон}
var x2,y2,z2 : real;
  {x, y, z, x2, y2 и z2 - внутренние переменные функции}
begin
  {Контрольная точка B}
  x2 := x*x;
  y2 := y*y;
  z2 := z*z;
  if ((z2=y2+x2) or (y2=x2+z2) or (x2=z2+y2)) then
  {Контрольная точка C}
    vid := 1
  else
  {Контрольная точка D}
    vid := 0;
  {Контрольная точка E}
end;
```

```
  {Головная программа задает стороны,
   вызывает функцию сравнения и выводит результат}
begin
  a := 3.0;
  b := 4.0;
  c := 5.0;
  writeln;
  writeln ('Первая сторона = ', a);
  writeln ('Вторая сторона = ', b);
  writeln ('Третья сторона = ', c);
  {Контрольная точка A}
  if (vid (a,b,c) = 1) then
  {Контрольная точка G}
    writeln ('Треугольник прямоугольный')
  else
  {Контрольная точка H}
    writeln ('Треугольник не прямоугольный');
  writeln;
end.
```

На этом листинге представлена программа определения вида треугольника по сторонам для теста 1. Чтобы протестировать программу тестами 2—4, необходимо в головной программе изменить значения сторон треугольника *a*, *b* и *c*.

Результаты работы теста 1 представлены в таблице 28, теста 2 — в таблице 29. Результаты тестов 3—4 совпадают с результатами теста 2.

Таблица 28. Значения данных в контрольных точках: тест 1

Контрольная точка	Значения данных
A	$a = 3, b = 4, c = 5$
B	$x = 3, y = 4, z = 5$
C	$x = 3, y = 4, z = 5, vid = 1$
D	—
E	$x = 3, y = 4, z = 5, vid = 1$
G	$a = 3, b = 4, c = 5, vid = 1$
H	—

```
Первая сторона = 3
Вторая сторона = 4
Третья сторона = 5
Треугольник прямоугольный
```

Таблица 29. Значения данных в контрольных точках: тест 2

Контрольная точка	Значения данных
A	$a = 3$, $b = 4$, $c = 6$
B	$x = 3$, $y = 4$, $z = 6$
C	—
D	$x = 3$, $y = 4$, $z = 6$, $vid = 0$
E	$x = 3$, $y = 4$, $z = 6$, $vid = 0$
G	—
H	$a = 3$, $b = 4$, $c = 6$, $vid = 0$

```
Первая сторона = 3
Вторая сторона = 4
Третья сторона = 6
Треугольник не прямоугольный
```

28.4. Определение вида треугольника по сторонам 2

Задача. Даны три вещественных числа, являющихся длинами сторон треугольника. Определить, является ли треугольник прямоугольным.

Решение.

Листинг 16. Программа определения вида треугольника по сторонам 2

```
  {Определение вида треугольника по сторонам 2}

  {Переменные должны быть вещественного типа}
var a,b,c : real;

  {Функция определяет вид треугольника по сторонам}
function vid (var x,y,z : real) : integer;
var x2,y2,z2 : real;
begin
  {Контрольная точка B}
  x2 := x*x; y2 := y*y; z2 := z*z;
  if (z2=x2+y2) then
    vid := 1
  {Контрольная точка C}
  else
    if (x2=y2+z2) then
      vid := 1
  {Контрольная точка D}
    else
      if (y2=z2+x2) then
        vid := 1
  {Контрольная точка E}
      else
        vid := 0;
  {Контрольная точка F}
end;
```

```
  {Основная программа задает стороны,
   вызывает функцию сравнения и выводит результат}
begin
  a := 3.0; b := 4.0; c := 5.0;
  writeln;
  writeln ('Стороны = ', a, b, c);
  {Контрольная точка A}
  if (vid(a,b,c) = 1) then
  {Контрольная точка H}
    writeln ('Треугольник прямоугольный')
  else
  {Контрольная точка I}
    writeln ('Треугольник не прямоугольный');
  writeln;
end.
```

На листинге 16 представлена программа определения вида треугольника по сторонам для теста 1. Для тестирования программы тестами 2—4 необходимо в головной программе изменить значения сторон треугольника.

Результаты работы тестов 1—4 представлены в таблице 30.

Таблица 30. Значения данных в контрольных точках для четырех тестов

Контрольная точка	Значения данных			
	Тест 1	Тест 2	Тест 3	Тест 4
A	$a = 3, b = 4, c = 5$	$a = 3, b = 4, c = 6$	$a = 5, b = 3, c = 4$	$a = 4, b = 5, c = 3$
B	$x = 3, y = 4, z = 5$	$x = 3, y = 4, z = 6$	$x = 5, y = 3, z = 4$	$x = 4, y = 5, z = 3$
C	$x = 3, y = 4, z = 5,$ $vid = 1$	—	—	—
D	—	—	$x = 5, y = 3, z = 4,$ $vid = 1$	—
E	—	—	—	$x = 4, y = 5, z = 3,$ $vid = 1$
F	—	$x = 3, y = 4, z = 6,$ $vid = 0$	—	—
H	$a = 3, b = 4, c = 5,$ $vid = 1$	—	$a = 5, b = 3, c = 4,$ $vid = 1$	$a = 4, b = 5, c = 3,$ $vid = 1$
I	—	$a = 3, b = 4, c = 6,$ $vid = 0$	—	—
	Стороны = 3, 4, 5 Треугольник прямоугольный	**Стороны = 3, 4, 6 Треугольник не прямоугольный**	**Стороны = 5, 3, 4 Треугольник прямоугольный**	**Стороны = 4, 5, 3 Треугольник прямоугольный**

§ 29. Циклы

29.1. Сумма натурального ряда

Задача. Найти сумму первых *n* натуральных чисел.

Решение.

На листинге 17 приведена программа суммирования первых *n* натуральных чисел, полностью совпадающая с соответствующим алгоритмом из главы 8.

Результат работы теста представлен в таблице 31.

Листинг 17. Программа вычисления суммы первых *n* натуральных чисел

```
  {Суммирование натурального ряда}

  {Переменная целого типа}
var n : integer;

  {Функция summa суммирует ряд натуральных чисел}
function summa (var k : integer) : integer;
var i, s : integer;
begin
  {Контрольная точка B}
  i := 0;
  s := 0;
  {Контрольная точка C}
  {Начало цикла}
  while (i < k) do
  begin
    i := i + 1;
    s := s + i;
  {Контрольная точка D}
  end;
  {Конец цикла}
  summa := s;
  {Контрольная точка E}
end;

  {Основная программа задает число слагаемых в сумме,
   вызывает функцию суммирования и выводит результат}
begin
  n := 5;
  {Контрольная точка A}
  writeln;
  write ('Сумма натуральных чисел от 1 до ', n);
  writeln (' = ', summa(n));
  writeln;
end.
```

Таблица 31. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 5$
B	$k = 5$
C	$k = 5, i = 0, s = 0$
D	$k = 5, i = 1, s = 1$
D	$k = 5, i = 2, s = 3$
D	$k = 5, i = 3, s = 6$
D	$k = 5, i = 4, s = 10$
D	$k = 5, i = 5, s = 15$
E	$n = 5, i = 5, s = 15, summa = 15$

```
Сумма натуральных чисел от 1 до 5 = 15
```

29.2. Сумма обратных величин натурального ряда

Задача. Найти сумму первых *n* обратных величин натуральных чисел.

Решение.

Листинг 18. Программа вычисления суммы первых *n* обратных величин натурального ряда

```
  {Суммирование обратного натурального ряда}

  {Переменная целого типа}
var n : integer;

  {Функция summa суммирует ряд обратных натуральных чисел}
function summa (var k : integer) : real;
  {Сумма должна быть вещественной переменной}
var i : integer;
    s : real;
begin
  {Контрольная точка B}
  i := 0;
  s := 0.0;
  {Контрольная точка C}
  while (i < k) do
  begin
    i := i + 1;
    s := s + 1.0/i;
  {Контрольная точка D}
  end;
  summa := s;
  {Контрольная точка E}
end;
```

```
  {Основная программа задает число слагаемых в сумме,
   вызывает функцию суммирования и выводит результат}
begin
  n := 3;
  {Контрольная точка A}
  writeln;
  write ('Сумма обратных натуральных чисел от 1 до ', n);
  writeln (' = ', summa(n));
  writeln;
end.
```

На листинге 18 приведена программа суммирования первых n обратных натуральных чисел.

Результат работы теста представлен в таблице 32.

Таблица 32. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 3$
B	$k = 3$
C	$k = 3$, $i = 0$, $s = 0.0$
D	$k = 3$, $i = 1$, $s = 1.0$
D	$k = 3$, $i = 2$, $s = 1.5$
D	$k = 3$, $i = 3$, $s = 1.833333$
E	$k = 3$, $i = 3$, $s = 1.833333$, $summa = 1.833333$

```
Сумма обратных натуральных чисел от 1 до 5 = 1.833333
```

29.3. Нахождение произведения нечетных чисел 1

Задача. Найти произведение n чисел вида $1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots$

Решение.

Листинг 19. Программа вычисления произведения первых n нечетных натуральных чисел

```
  {Произведение первых n нечетных натуральных чисел}

  {Переменная целого типа}
var n : integer;

  {Функция pro перемножает первые нечетные натуральные числа}
function pro (var k : integer) : real;
  {Произведение лучше сделать вещественной переменной}
var i : integer;
    p : real;
```

```
begin
  {Контрольная точка B}
  i := 0;
  p := 1.0;
  {Контрольная точка C}
  while (i < k) do
  begin
    i := i + 1;
    p := p * (2.0*i - 1);
  {Контрольная точка D}
  end;
  pro := p;
  {Контрольная точка E}
end;

  {Основная программа задает число сомножителей,
   вызывает функцию перемножения и выводит результат}
begin
  n := 5;
  {Контрольная точка A}
  writeln;
  write ('Произведение первых ', n);
  write (' нечетных натуральных чисел');
  writeln (' = ', pro(n));
  writeln;
end.
```

На листинге 19 приведена программа нахождения произведения первых *n* нечетных натуральных чисел.

Результат работы теста представлен в таблице 33.

Таблица 33. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 5$
B	$k = 5$
C	$k = 5, i = 0, p = 1.0$
D	$k = 5, i = 1, p = 1.0$
D	$k = 5, i = 2, p = 3.0$
D	$k = 5, i = 3, p = 15.0$
D	$k = 5, i = 4, p = 105.0$
D	$k = 5, i = 5, p = 945.0$
E	$k = 5, i = 5, p = 945.0, pro = 945.0$

```
Произведение первых 5 нечетных натуральных чисел = 945
```

29.4. Нахождение произведения нечетных чисел 2

Задача. Найти произведение n чисел вида $1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots$

Решение.

Листинг 20. Программа вычисления произведения нечетных натуральных чисел

```
  {Произведение первых n нечетных натуральных чисел}

var n : integer;

  {Функция pro перемножает первые нечетные натуральные числа}
function pro (var k : integer) : real;
var i,s : integer; p : real;
begin
  {Контрольная точка B}
  i := 0; s := 1; p := 1.0;
  {Контрольная точка C}
  while (i < k) do
  begin
    i := i + 1; p := p * s; s := s + 2;
  {Контрольная точка D}
  end;
  pro := p;
  {Контрольная точка E}
end;

  {Основная программа задает число сомножителей,
   вызывает функцию перемножения и выводит результат}
begin
  n := 5;
  {Контрольная точка A}
  writeln;
  write ('Произведение первых ', n);
  writeln (' нечетных натуральных чисел = ', pro(n));
  writeln;
end.
```

Таблица 34. Значения данных в контрольных точках

Контрольная точка	Значения данных
A	$n = 5$
B	$k = 5$
C	$k = 5, i = 0, s = 1, p = 1.0$
D	$k = 5, i = 1, s = 3, p = 1.0$
D	$k = 5, i = 1, s = 5, p = 3.0$
D	$k = 5, i = 1, s = 7, p = 15.0$
D	$k = 5, i = 1, s = 9, p = 105.0$
D	$k = 5, i = 1, s = 11, p = 945.0$
E	$k = 5, i = 5, s = 11, p = 945.0, pro = 945.0$

```
Произведение первых 5 нечетных натуральных чисел = 945
```

§ 30. Массивы

30.1. Максимум данных положительных чисел. Версия 1

Задача. Найти максимальное число из *n* данных положительных чисел.

Решение.

Листинг 21. Программа вычисления максимума положительных чисел

```
  {Максимум n положительных чисел}

var n,m : integer; k : array[0..5] of integer;

  {Функция maxim ищет максимум в массиве положительных чисел}
function maxim (var n,m : integer;
                var k: array of integer) : integer;
var i, max : integer;
begin
  {Контрольная точка B}
  i := 0; max := 0;
  {Контрольная точка C}
  while (i < n) do
  begin
    if (max < k[i]) then
    begin
      max := k[i]; m := i+1;
  {Контрольная точка D}
    end;
    i := i + 1;
  {Контрольная точка E}
  end;
  maxim := max;
  {Контрольная точка F}
end;

  {Основная программа задает элементы массива и их число,
   вызывает функцию поиска максимума и выводит результат}
begin
  n := 5;
  k[0] := 10; k[1] := 8; k[2] := 7; k[3] := 3; k[4] := 9;
  writeln;
  write ('Элементы массива длиной ', n, ' равны ');
  writeln (k[0], ' ', k[1], ' ', k[2], ' ', k[3], ' ', k[4]);
  {Контрольная точка A}
  write ('Максимальный элемент массива длиной ', n);
  write (' равен ', maxim(n,m,k));
  writeln (' и имеет номер ', m);
  writeln;
  {Контрольная точка G}
end.
```

Таблица 35. Значения данных в контрольных точках: тест 1

Контрольная точка	Значения данных
A	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$
B	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$
C	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 0$, $i = 0$
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 0$, $i = 1$
E	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 10$, $i = 1$, $m = 1$
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 10$, $i = 2$, $m = 1$
E	—
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 10$, $i = 3$, $m = 1$
E	—
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 10$, $i = 4$, $m = 1$
E	—
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 10$, $i = 5$, $m = 1$
E	—
F	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 10$, $i = 5$, $m = 1$, $maxim = 10$
G	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $max = 10$, $m = 1$, $maxim = 10$

```
Элементы массива длиной 5 равны 10 8 7 3 9
Максимальный элемент массива длиной 5 равен 10 и имеет номер 1
```

Таблица 36. Значения данных в контрольных точках: тест 2

Контрольная точка	Значения данных
A	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$
B	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$
C	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 0$, $i = 0$
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 0$, $i = 1$
E	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 1$, $i = 1$, $m = 1$
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 1$, $i = 2$, $m = 1$
E	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 8$, $i = 2$, $m = 2$
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 8$, $i = 3$, $m = 2$
E	—
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 8$, $i = 4$, $m = 2$
E	—
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 8$, $i = 5$, $m = 2$
E	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 9$, $i = 5$, $m = 5$
F	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 9$, $i = 5$, $m = 5$, $maxim = 9$
G	$n = 5$, $k[1..5] = 1, 8, 7, 3, 9$, $max = 9$, $m = 5$, $maxim = 9$

```
Элементы массива длиной 5 равны 1 8 7 3 9
Максимальный элемент массива длиной 5 равен 9 и имеет номер 5
```

Таблица 37. Значения данных в контрольных точках: тест 3

Контрольная точка	Значения данных
A	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$
B	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$
C	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 0$, $i = 0$
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 0$, $i = 1$
E	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 1$, $i = 1$, $m = 1$
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 1$, $i = 2$, $m = 1$
E	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 8$, $i = 2$, $m = 2$
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 8$, $i = 3$, $m = 2$
E	—
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 8$, $i = 4$, $m = 2$
E	—
D	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 8$, $i = 5$, $m = 2$
E	—
F	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 8$, $i = 5$, $m = 2$, $maxim = 8$
G	$n = 5$, $k[1..5] = 1, 8, 7, 3, 2$, $max = 8$, $m = 2$, $maxim = 8$

```
Элементы массива длиной 5 равны 1 8 7 3 2
Максимальный элемент массива длиной 5 равен 8 и имеет номер 2
```

30.2. Максимум данных положительных чисел. Версия 2

Задача. Найти максимальное число из *n* данных положительных чисел.

Решение.

Листинг 22. Программа вычисления максимума положительных чисел

```
  {Максимум n положительных чисел}

var n,m : integer; k : array[0..4] of integer;

  {Функция maxim ищет максимум в массиве положительных чисел}
function maxim (var n,m : integer;
                var k: array of integer) : integer;
var i, max : integer;
begin
  {Контрольная точка B}
  i := 1; max := k[0]; m := 1;
  {Контрольная точка C}
  while (i < n) do
  begin
    if (max < k[i]) then
    begin
      max := k[i];  m := i + 1;
```

```
  {Контрольная точка D}
    end;
    i := i + 1;
  {Контрольная точка E}
  end;
  maxim := max;
  {Контрольная точка F}
end;

  {Основная программа задает элементы массива и их число,
   вызывает функцию поиска максимума и выводит результат}
begin
  n := 5;
  k[0] := 10; k[1] := 8; k[2] := 7; k[3] := 3; k[4] := 9;
  writeln;
  write ('Элементы массива длиной ', n, ' равны ');
  writeln (k[0], ' ', k[1], ' ', k[2], ' ', k[3], ' ', k[4]);
  {Контрольная точка A}
  writeln;
  write ('Максимальный элемент массива длиной ', n);
  write (' равен ', maxim(n,m,k));
  writeln (' и имеет номер ', m);
  writeln;
  {Контрольная точка G}
end.
```

30.3. Нахождение количества элементов массива по условию

Задача. Посчитать в массиве длины *n* число положительных чисел.

Решение.

Листинг 23. Программа вычисления количества положительных чисел

```
  {Количество положительных чисел}

var n,m : integer; k : array[0..4] of integer;

  {Функция kol определяет количество положительных чисел}
function kol (var n : integer;
              var k: array of integer) : integer;
var i, m : integer;
begin
  {Контрольная точка B}
  i := 0; m := 0;
  {Контрольная точка C}
  while (i < n) do
  begin
    if (0 < k[i]) then
      m := m + 1;
  {Контрольная точка D}
```

```
    i := i + 1;
  {Контрольная точка E}
  end;
  kol := m;
  {Контрольная точка F}
end;

  {Основная программа задает элементы массива и их число,
   вызывает функцию определения количества положительных
   чисел и выводит результат}
begin
  n := 5;
  k[0] := 10; k[1] := 8; k[2] := 7; k[3] := 3; k[4] := 9;
  writeln;
  write ('Элементы массива длиной ', n, ' равны ');
  writeln (k[0], ' ', k[1], ' ', k[2], ' ', k[3], ' ', k[4]);
  {Контрольная точка A}
  write ('Количество положительных элементов массива ');
  writeln ('длиной ', n, ' равно ', kol(n,k));
  writeln;
  {Контрольная точка G}
end.
```

Таблица 38. Значения данных в контрольных точках: тест 1

Контрольная точка	Значения данных
A	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$
B	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$
C	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 0$, $m = 0$
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 1$, $m = 0$
E	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 1$, $m = 1$
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 2$, $m = 1$
E	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 2$, $m = 2$
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 3$, $m = 2$
E	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 3$, $m = 3$
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 4$, $m = 3$
E	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 4$, $m = 4$
D	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 5$, $m = 4$
E	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 5$, $m = 5$
F	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, $i = 5$, $m = 5$, kol = 5
G	$n = 5$, $k[1..5] = 10, 8, 7, 3, 9$, kol = 5

```
Элементы массива длиной 5 равны 10 8 7 3 9
Количество положительных элементов массива длиной 5 равно 5
```

Таблица 39. Значения данных в контрольных точках: тест 2

Контрольная точка	Значения данных
A	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$
B	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$
C	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 0$, $m = 0$
D	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 1$, $m = 0$
E	—
D	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 2$, $m = 0$
E	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 2$, $m = 1$
D	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 3$, $m = 1$
E	
D	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 4$, $m = 1$
E	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 4$, $m = 2$
D	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 5$, $m = 2$
E	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 5$, $m = 3$
F	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, $i = 5$, $m = 3$, kol = 3
G	$n = 5$, $k[1..5] = -10, 8, -7, 3, 9$, kol = 3

```
Элементы массива длиной 5 равны -10 8 -7 3 9
Количество положительных элементов массива длиной 5 равно 3
```

Таблица 40. Значения данных в контрольных точках: тест 3

Контрольная точка	Значения данных
A	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$
B	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$
C	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$, $i = 0$, $m = 0$
D	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$, $i = 1$, $m = 0$
E	
D	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$, $i = 2$, $m = 0$
E	
D	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$, $i = 3$, $m = 0$
E	
D	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$, $i = 4$, $m = 0$
E	
D	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$, $i = 5$, $m = 0$
E	
F	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$, $i = 5$, $m = 0$, kol = 0
G	$n = 5$, $k[1..5] = 0, -8, -7, -3, -2$, kol = 0

```
Элементы массива длиной 5 равны 0 -8 -7 -3 -2
Количество положительных элементов массива длиной 5 равно 0
```

30.4. Суммирование квадратов элементов массива

Задача. Просуммировать квадраты чисел массива длины *n*.

Решение.

Листинг 24. Программа суммирования квадратов чисел

```
  {Суммирование квадратов чисел массива}

var n : integer; k : array[0..4] of integer;

  {Функция s суммирует квадраты чисел массива}
function s (var n : integer; var k: array of integer) : integer;
var i, sum : integer;
begin
  {Контрольная точка B}
  i := 0; sum := 0;
  {Контрольная точка C}
  while (i < n) do
  begin
    sum := sum + k[i]*k[i]; i := i + 1;
  {Контрольная точка D}
  end;
  s := sum;
  {Контрольная точка E}
end;

  {Головная программа задает элементы массива и их число,
   вызывает функцию определения количества положительных
   чисел и выводит результат}
begin
  n := 5;
  k[0] := -2; k[1] := -1; k[2] := 0; k[3] := 1; k[4] := 2;
  writeln;
  write (' Элементы массива длиной ', n, ' равны ');
  writeln (k[0], ' ', k[1], ' ', k[2], ' ', k[3], ' ', k[4]);
  {Контрольная точка A}
  write ('Сумма квадратов элементов массива ');
  writeln ('длиной ', n, ' равна ', s(n,k));
  writeln;
end.
```

Таблица 41. Значения данных в контрольных точках: тест 2

Контрольная точка	Значения данных
A	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$
B	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$
C	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$, $i = 0$, $s = 0$
D	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$, $i = 1$, $s = 4$
D	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$, $i = 2$, $s = 5$
D	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$, $i = 3$, s = 5
D	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$, $i = 4$, s = 6
D	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$, $i = 5$, $s = 10$
E	$n = 5$, $k[1..5] = -2, -1, 0, 1, 2$, $i = 5$, s = 10

```
Элементы массива длиной 5 равны -2 -1 0 1 2
Сумма квадратов элементов массива длиной 5 равна 10
```

Приложения
Приложение 1

Сто первых римских чисел

Таблица 42. Сто первых римских чисел

№ п/п	Римское число	№ п/п	Римское число	№ п/п	Римское число	№ п/п	Римское число	№ п/п	Римское число
1	I	21	XXI	41	XLI	61	LXI	81	LXXXI
2	II	22	XXII	42	XLII	62	LXII	82	LXXXII
3	III	23	XXIII	43	XLIII	63	LXIII	83	LXXXIII
4	IV	24	XXIV	44	XLIV	64	LXIV	84	LXXXIV
5	V	25	XXV	45	XLV	65	LXV	85	LXXXV
6	VI	26	XXVI	46	XLVI	66	LXVI	86	LXXXVI
7	VII	27	XXVII	47	XLVII	67	LXVII	87	LXXXVII
8	VIII	28	XXVIII	48	XLVIII	68	LXVIII	88	LXXXVIII
9	IX	29	XXIX	49	XLIX	69	LXIX	89	LXXXIX
10	X	30	XXX	50	L	70	LXX	90	XC
11	XI	31	XXXI	51	LI	71	LXXI	91	XCI
12	XII	32	XXXII	52	LII	72	LXXII	92	XCII
13	XIII	33	XXXIII	53	LIII	73	LXXIII	93	XCIII
14	XIV	34	XXXIV	54	LIV	74	LXXIV	94	XCIV
15	XV	35	XXXV	55	LV	75	LXXV	95	XCV
16	XVI	36	XXXVI	56	LVI	76	LXXVI	96	XCVI
17	XVII	37	XXXVII	57	LVII	77	LXXVII	97	XCVII
18	XVIII	38	XXXVIII	58	LVIII	78	LXXVIII	98	XCVIII
19	XIX	39	XXXIX	59	LIX	79	LXXIX	99	XCIX
20	XX	40	XL	60	LX	80	LXXX	100	C

Приложение 2

Русские (кириллические) кодовые таблицы

Кодовая таблица win

Таблица 43. Русская кодовая таблица win

Символ / Код													
32	0 48	@ 64	P 80	` 96	p 112	Ђ 0128	ђ 0144	 0160	° 0176	А 0192	Р 0208	а 0224	р 0240
! 33	1 49	A 65	Q 81	a 97	q 113	Ѓ 0129	‘ 0145	Ў 0161	± 0177	Б 0193	С 0209	б 0225	с 0241
" 34	2 50	B 66	R 82	b 98	r 114	, 0130	’ 0146	ў 0162	І 0178	В 0194	Т 0210	в 0226	т 0242
# 35	3 51	C 67	S 83	c 99	s 115	ѓ 0131	“ 0147	Ј 0163	і 0179	Г 0195	У 0211	г 0227	у 0243
$ 36	4 52	D 68	T 84	d 100	t 116	„ 0132	” 0148	¤ 0164	ґ 0180	Д 0196	Ф 0212	д 0228	ф 0244
% 37	5 53	E 69	U 85	e 101	u 117	… 0133	• 0149	Ґ 0165	µ 0181	Е 0197	Х 0213	е 0229	х 0245
& 38	6 54	F 70	V 86	f 102	v 118	† 0134	– 0150	¦ 0166	¶ 0182	Ж 0198	Ц 0214	ж 0230	ц 0246
' 39	7 55	G 71	W 87	g 103	w 119	‡ 0135	— 0151	§ 0167	· 0183	З 0199	Ч 0215	з 0231	ч 0247
(40	8 56	H 72	X 88	h 104	x 120	€ 0136	□ 0152	Ё 0168	ё 0184	И 0200	Ш 0216	и 0232	ш 0248
) 41	9 57	I 73	Y 89	i 105	y 121	‰ 0137	™ 0153	© 0169	№ 0185	Й 0201	Щ 0217	й 0233	щ 0249
* 42	: 58	J 74	Z 90	j 106	z 122	Љ 0138	љ 0154	Є 0170	є 0186	К 0202	Ъ 0218	к 0234	ъ 0250
+ 43	; 59	K 75	[91	k 107	{ 123	‹ 0139	› 0155	« 0171	» 0187	Л 0203	Ы 0219	л 0235	ы 0251
, 44	< 60	L 76	\ 92	l 108	| 124	Њ 0140	њ 0156	¬ 0172	j 0188	М 0204	Ь 0220	м 0236	ь 0252
- 45	= 61	M 77	] 93	m 109	} 125	Ќ 0141	ќ 0157	- 0173	S 0189	Н 0205	Э 0221	н 0237	э 0253
. 46	> 62	N 78	^ 94	n 110	~ 126	Ћ 0142	ћ 0158	® 0174	s 0190	О 0206	Ю 0222	о 0238	ю 0254
/ 47	? 63	O 79	_ 95	o 111	□ 127	Џ 0143	џ 0159	Ї 0175	ї 0191	П 0207	Я 0223	п 0239	я 0255

Продолжение приложения 2

Кодовая таблица koi8

Таблица 44. Русская кодовая таблица koi8

Символ / Код													
32	0 48	@ 64	P 80	` 96	p 112	) 0128	! 0144	4 0160	K 0176	ю 0192	п 0208	Ю 0224	П 0240
! 33	1 49	A 65	Q 81	a 97	q 113	* 0129	" 0145	5 0161	: 0177	а 0193	я 0209	А 0225	Я 0241
" 34	2 50	B 66	R 82	b 98	r 114	+ 0130	# 0146	? 0162	I 0178	б 0194	р 0210	Б 0226	Р 0242
# 35	3 51	C 67	S 83	c 99	s 115	, 0131	⌠ 0147	ё 0163	Ё 0179	ц 0195	с 0211	Ц 0227	С 0243
$ 36	4 52	D 68	T 84	d 100	t 116	− 0132	■ 0148	C 0164	M 0180	д 0196	т 0212	Д 0228	Т 0244
% 37	5 53	E 69	U 85	e 101	u 117	. 0133	• 0149	6 0165	< 0181	е 0197	у 0213	Е 0229	У 0245
& 38	6 54	F 70	V 86	f 102	v 118	/ 0134	√ 0150	@ 0166	L 0182	ф 0198	ж 0214	Ф 0230	Ж 0246
' 39	7 55	G 71	W 87	g 103	w 119	0 0135	≈ 0151	D 0167	H 0183	г 0199	в 0215	Г 0231	В 0247
(40	8 56	H 72	X 88	h 104	x 120	1 0136	□ 0152	7 0168	; 0184	х 0200	ь 0216	Х 0232	Ь 0248
) 41	9 57	I 73	Y 89	i 105	y 121	2 0137	□ 0153	B 0169	N 0185	и 0201	ы 0217	И 0233	Ы 0249
* 42	: 58	J 74	Z 90	j 106	z 122	3 0138	0154	F 0170	J 0186	й 0202	з 0218	Й 0234	З 0250
+ 43	; 59	K 75	[91	k 107	{ 123	& 0139	⌡ 0155	9 0171	= 0187	к 0203	ш 0219	К 0235	Ш 0251
, 44	< 60	L 76	\ 92	l 108	| 124	(0140	° 0156	A 0172	P 0188	л 0204	э 0220	Л 0236	Э 0252
- 45	= 61	M 77	] 93	m 109	} 125	$ 0141	² 0157	E 0173	O 0189	м 0205	щ 0221	М 0237	Щ 0253
. 46	> 62	N 78	^ 94	n 110	~ 126	% 0142	· 0158	8 0174	> 0190	н 0206	ч 0222	Н 0238	Ч 0254
/ 47	? 63	O 79	_ 95	o 111	⌂ 127	' 0143	÷ 0159	G 0175	© 0191	о 0207	ъ 0223	О 0239	Ъ 0255

Продолжение приложения 2

Кодовая таблица dos

Таблица 45. Русская кодовая таблица dos

Символ / Код													
32	0 48	@ 64	P 80	` 96	p 112	А 0128	Р 0144	а 0160	! 0176	· 0192	J 0208	р 0224	Ё 0240
! 33	1 49	A 65	Q 81	a 97	q 113	Б 0129	С 0145	б 0161	" 0177	2 0193	L 0209	с 0225	ё 0241
" 34	2 50	B 66	R 82	b 98	r 114	В 0130	Т 0146	в 0162	# 0178	0 0194	H 0210	т 0226	□ 0242
# 35	3 51	C 67	S 83	c 99	s 115	Г 0131	У 0147	г 0163	* 0179	/ 0195	F 0211	у 0227	□ 0243
$ 36	4 52	D 68	T 84	d 100	t 116	Д 0132	Ф 0148	д 0164	1 0180	) 0196	B 0212	ф 0228	⌠ 0244
% 37	5 53	E 69	U 85	e 101	u 117	Е 0133	Х 0149	е 0165	I 0181	3 0197	? 0213	х 0229	⌡ 0245
& 38	6 54	F 70	V 86	f 102	v 118	Ж 0134	Ц 0150	ж 0166	M 0182	G 0198	C 0214	ц 0230	÷ 0246
' 39	7 55	G 71	W 87	g 103	w 119	З 0135	Ч 0151	з 0167	D 0183	K 0199	O 0215	ч 0231	≈ 0247
(40	8 56	H 72	X 88	h 104	x 120	И 0136	Ш 0152	и 0168	@ 0184	9 0200	P 0216	ш 0232	° 0248
) 41	9 57	I 73	Y 89	i 105	y 121	Й 0137	Щ 0153	й 0169	< 0185	6 0201	- 0217	щ 0233	• 0249
* 42	: 58	J 74	Z 90	j 106	z 122	К 0138	Ъ 0154	к 0170	5 0186	= 0202	+ 0218	ъ 0234	· 0250
+ 43	; 59	K 75	[91	k 107	{ 123	Л 0139	Ы 0155	л 0171	7 0187	; 0203	$ 0219	ы 0235	√ 0251
, 44	< 60	L 76	\ 92	l 108	| 124	М 0140	Ь 0156	м 0172	8 0188	: 0204	(0220	ь 0236	n 0252
- 45	= 61	M 77	] 93	m 109	} 125	Н 0141	Э 0157	н 0173	E 0189	4 0205	% 0221	э 0237	² 0253
. 46	> 62	N 78	^ 94	n 110	~ 126	О 0142	Ю 0158	о 0174	A 0190	> 0206	' 0222	ю 0238	■ 0254
/ 47	? 63	O 79	_ 95	o 111	⌂ 127	П 0143	Я 0159	п 0175	, 0191	N 0207	& 0223	я 0239	 0255

Приложение 3

Некоторые встроенные математические функции Паскаля

1. a Mod b — остаток от деления целого числа a на целое число b.

2. Abs (x) — абсолютная величина $|x|$ числа x.

3. Sqr (x) — квадрат x^2 числа x.

4. Sqrt (x) — квадратный корень $\sqrt{x}$ числа x.

5. Exp (x) — экспонента e^x (или $\exp x$) числа x.

6. Ln (x) — натуральный логарифм $\ln x$ числа x.

7. Sin (x) — синус $\sin x$ числа x.

8. Cos (x) — косинус $\cos x$ числа x.

9. Random (x), где x — натуральное число,— натуральное случайное число в диапазоне от 0 до $x-1$, включая 0 и $x-1$.

Приложение 4

Примеры экзаменационных билетов

Билет

1. Двоичная система счисления. Таблицы умножения и сложения. Натуральные двоичные числа. Количество цифр в числе. Перевод чисел из двоичной системы в десятичную. Бит. Байт. Производные единицы от байта.

2. Текстовый редактор и процессор. Виды текстовых процессоров. Технологии текстового редактора, набор, курсор. Сохранение. Просмотр. Редактирование, выделение. Редактирование символов. Редактирование блоков символов.

3. Использовать функцию, нарисовать блок-схему алгоритма и написать программу на языке программирования, решающую следующую задачу:

найти произведение n чисел вида
$(1 + 1/10) \cdot (1 + 1/100) \cdot (1 + 1/1000) \cdot \ldots$

Билет

1. Звуковая периферия. Характеристики звука, оцифровка. Непосредственная оцифровка. Запись музыки. Запись звука. Аудио-диски. Анимация. Видео. Мультимедиа.

2. Алгоритм, его характеристики и свойства. Архитектура фон Неймана. Проектирование сверху вниз. Модульность алгоритмов, головная программа. Принцип черного ящика. Структурное программирование. Объектно-ориентированное программирование. Визуальное программирование.

3. Использовать функцию, нарисовать блок-схему алгоритма и написать программу на языке программирования, решающую следующую задачу:

найти среднее арифметическое элементов
целочисленного массива длиной 100.

Приложение 5

Образцы решения задач

Задача 1.

Найти произведение *n* чисел вида

$$(1 + 1/10) \cdot (1 + 1/100) \cdot (1 + 1/1000) \cdot \ldots$$

Решение.

Анализ задачи.

1. Входные данные. На входе имеем число сомножителей *n*.

2. Выходные данные. Программа должна посчитать произведение *n* указанных сомножителей.

3. Функция. Используем в программе функцию pro, которая с помощью цикла вычисляет произведение первых *n* чисел

$$(1 + 1/10) \cdot (1 + 1/100) \cdot (1 + 1/1000) \cdot \ldots$$

4. Формулы. Заметим, что *i*-е число можно записать в виде $1 + 1/10^i$. Само произведение и каждое слагаемое являются вещественными числами.

5. План функции. Функция pro использует цикл от 1 до *n* с переменной цикла *i*. В этом цикле результат каждый раз умножается на число $1 + 1/10^i$, которое должно быть вещественным. Чтобы не возводить каждый раз 10 в степень *i*, будем в цикле домножать переменную *des* на 1/10.

Блок-схема алгоритма. См. рис. 42.

Текст компьютерной программы.

Листинг 25. Программа нахождения произведения *n* чисел заданного вида

```
  {Произведение первых n чисел указанного вида}

var n : integer;

  {Функция pro перемножает первые n чисел указанного вида}
function pro (var k : integer) : real;
var i : integer;
    p,des : real;
begin
  i := 0;
  p := 1.0;
  des := 1.0;
  while (i < k) do
  begin
    i := i + 1;
    des := des/10.0;
    p := p * (1.0 + des);
  end;
```

Продолжение приложения 5

```
  pro := p;
end;

  {Головная программа задает число сомножителей,
   вызывает функцию перемножения и выводит результат}
begin
  n := 100;
  writeln;
  write ('Произведение 1+1/10^i от 1 до ', n);
  writeln (' = ', pro(n));
  writeln;
end.
```

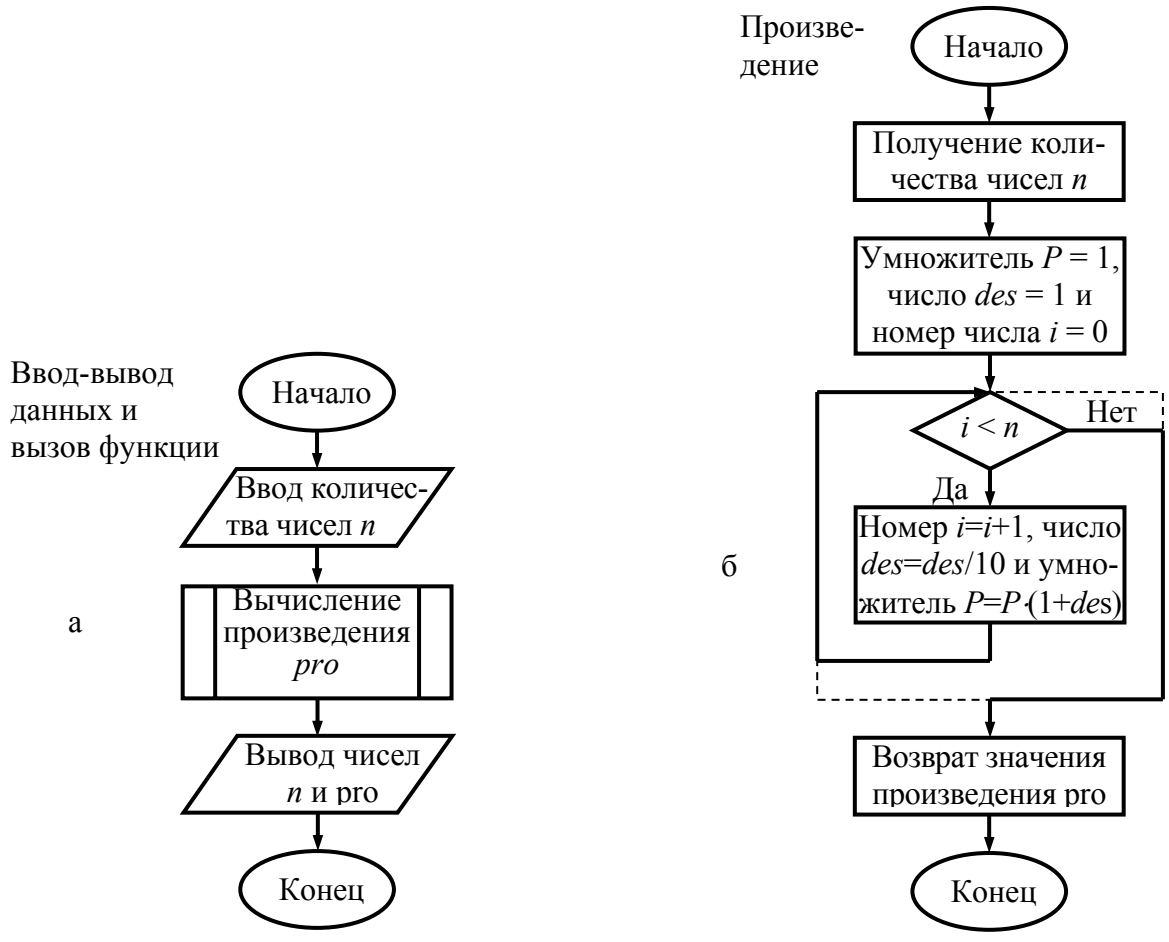


Рис. 42. Вычисление произведения первых *n* чисел указанного вида:
а) главный алгоритм; б) функция

Продолжение приложения 5

Задача 2.

Найти среднее арифметическое элементов целочисленного массива длиной 100.

Решение.

Анализ задачи.

1. Входные данные. Нужно ввести элементы целочисленного массива длиной 100. Сделаем это с помощью функции случайных чисел.

2. Выходные данные. Программа должна посчитать среднее арифметическое элементов массива. Это число может быть вещественным.

3. Функция. Используем в программе функцию sr, которая с помощью цикла вычисляет сумму элементов массива *s* и делит ее на количество этих элементов *n*. При этом должно получиться среднее арифметическое — вещественное число. Поскольку длина массива *n* — целое число, то, чтобы при делении получилось не целое, а вещественное число, сумма элементов массива *s* должна быть вещественной.

4. Формулы. Случайные числа будем генерировать в диапазоне от 0 до 10.

5. План функции. Функция sr использует цикл от 1 до *n* с целой переменной цикла *i*. В этом цикле элементы массива *n* суммируются в вещественную переменную суммирования *s*. После этого переменная суммирования делится на *n*.

Блок-схема алгоритма. См. рис. 43.

Текст компьютерной программы.

Листинг 26. Программа нахождения среднего арифметического

```
  {Нахождение среднего арифметического элементов массива}

var n,i : integer; k : array[0..99] of integer;

  {Функция sr находит среднее арифметическое элементов массива}
function sr (var n : integer; var k: array of integer) : real;
var i : integer;
    s : real;
begin
  i := 0;
  s := 0.0;
  while (i < n) do
  begin
    s := s + k[i];
    i := i + 1;
  end;
  sr := s/n;
end;
```

Продолжение приложения 5

```
  {Головная программа задает случайным образом элементы
   массива и их число, вызывает функцию нахождения среднего
   арифметического элементов массива и выводит результат}
begin
  n := 100;
  i := 0;
  while (i < n) do
  begin
    k[i] := random(11);
    i := i + 1;
  end;
  writeln;
  write ('Среднее арифметическое массива ');
  writeln ('длиной ', n, ' равно ', sr(n,k));
  writeln;
end.
```

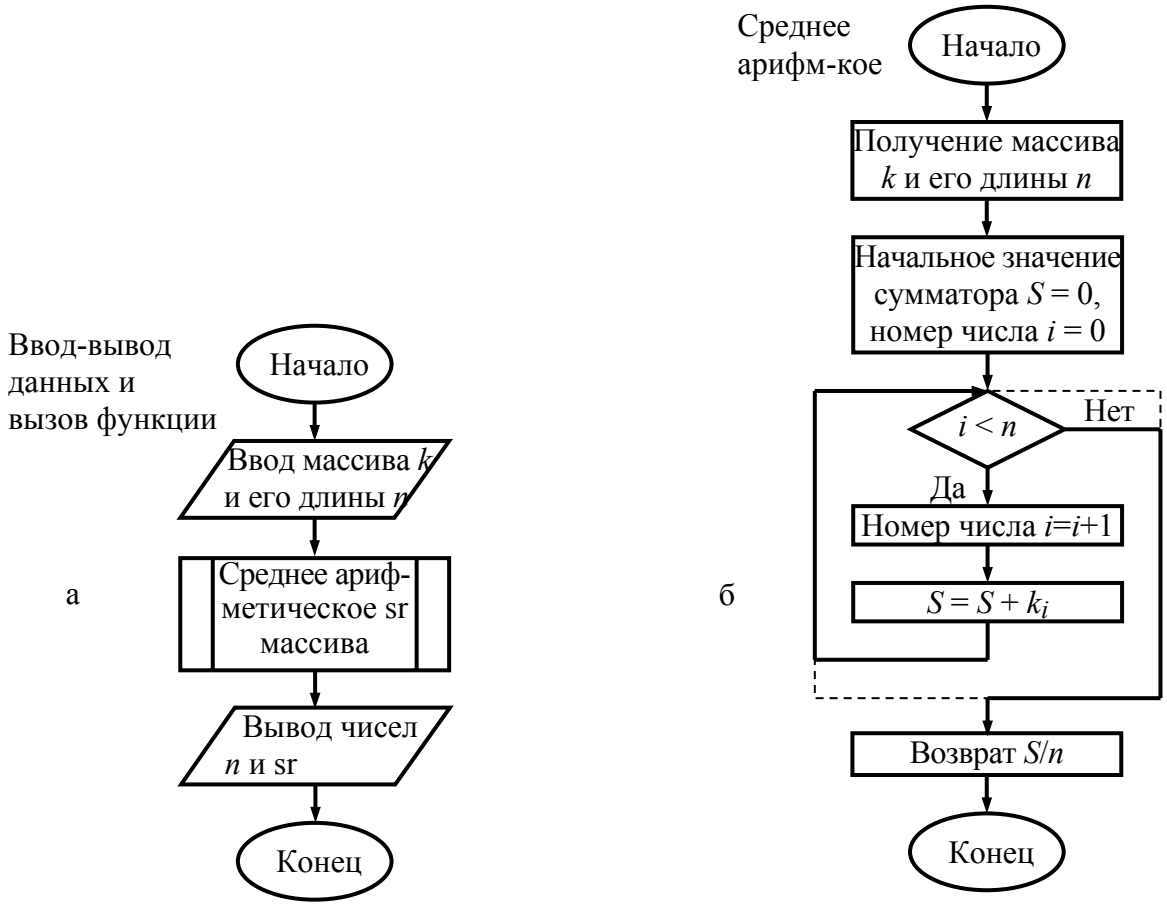


Рис. 43. Вычисление среднего арифметического элементов массива:
а) главный алгоритм; б) функция

Приложение 6

Задачи устного экзамена 2002 г.

А

1. Даны два вещественных числа, являющихся величинами гипотенузы и одного из катетов прямоугольного треугольника. Нарисовать блок-схему и написать компьютерную программу, вычисляющую площадь этого треугольника. Использовать при этом функцию.

2. Дана длина окружности. Нарисовать блок-схему и написать компьютерную программу, вычисляющую площадь круга, ограниченного этой окружностью. Использовать при этом функцию.

3. Даны три вещественных числа, являющихся длинами сторон треугольника. Нарисовать блок-схему и написать компьютерную программу, определяющую, является ли этот треугольник правильным. Использовать при этом функцию.

4. Даны три вещественных числа, являющихся длинами сторон треугольника. Нарисовать блок-схему и написать компьютерную программу, определяющую, является ли этот треугольник равнобедренным. Использовать при этом функцию.

5. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую сумму n чисел вида $1 - 1/2^3 + 1/3^3 - 1/4^3 + \ldots$ Использовать при этом функцию.

6. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую сумму n чисел вида $1 - 2/3^3 + 3/5^3 - 4/7^3 + \ldots$ Использовать при этом функцию.

7. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую произведение n чисел вида $2 \cdot 4 \cdot 6 \cdot 8 \cdot \ldots$ Использовать при этом функцию.

8. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую произведение следующих n чисел вида $(1 + 2/10) \cdot (1 + 3/100) \cdot (1 + 4/1000) \cdot \ldots$ Использовать при этом функцию.

9. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую в целочислительном массиве длиной 100 количество положительных чисел. Использовать при этом функцию.

10. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую в целочислительном массиве длиной 100 сумму положительных чисел. Использовать при этом функцию.

11. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую в целочислительном массиве длиной 100 количество положительных и количество отрицательных чисел. Использовать при этом функцию.

12. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую среднее арифметическое положительных чисел целочислительного массива длиной 100. Использовать при этом функцию.

13. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую среднее арифметическое положительных и среднее арифметическое отрицательных чисел целочислительного массива длиной 100. Использовать при этом функцию.

Б

1. Даны четыре вещественных числа, являющихся длинами сторон четырехугольника. Нарисовать блок-схему и написать компьютерную программу, определяющую, является ли этот четырехугольник ромбом или параллелограммом. Использовать при этом функцию.

2. Даны три вещественных числа, являющихся длинами сторон треугольника. Нарисовать блок-схему и написать компьютерную программу, определяющую, является ли этот треугольник остроугольным, прямоугольным или тупоугольным. Использовать при этом функцию.

3. Даны расстояние между центрами двух окружностей и их радиусы. Нарисовать блок-схему и написать компьютерную программу, определяющую, пересекаются ли эти окружности. Использовать при этом функцию.

4. Даны три вещественных числа, являющихся углами треугольника. Нарисовать блок-схему и написать компьютерную программу, определяющую, является ли этот треугольник остроугольным, прямоугольным или тупоугольным. Использовать при этом функцию.

5. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую произведение следующих n чисел вида $(1 + 1) \cdot (1 + 1/(1 \cdot 2)) \cdot (1 + 1/(1 \cdot 2 \cdot 3)) \cdot \ldots$ Использовать при этом функцию.

6. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую сумму n чисел вида $1 - 1/(1 \cdot 2) + 1/(1 \cdot 2 \cdot 3) - \ldots$ Использовать при этом функцию.

7. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую сумму n чисел вида $1 - (1 + 3)/(1 \cdot 2) + (1 + 3 + 5)/(1 \cdot 2 \cdot 3) - \ldots$ Использовать при этом функцию.

8. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую произведение следующих n чисел вида $(1 + 1) \cdot (1 + (1 + 2)/(1 \cdot 3)) \cdot (1 + (1 + 2 + 3)/(1 \cdot 3 \cdot 5)) \cdot \ldots$ Использовать при этом функцию.

9. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую среднее арифметическое всех чисел целочислительного массива длиной 100 и вычитающую это среднее арифметическое из каждого числа массива. Использовать при этом функцию.

10. Нарисовать блок-схему и написать компьютерную программу, складывающую и перемножающую пары чисел целочислительного массива длиной 100, находящихся на равных расстояниях от концов массива, и помещающую сумму на место числа пары из 1-й половины массива, а произведение — из 2-й половины массива. Использовать при этом функцию.

11. Нарисовать блок-схему и написать компьютерную программу, находящую максимум чисел целочислительного массива длиной 100, обнуляющую элементы массива с отрицательными числами и заменяющую положительные на максимум. Использовать при этом функцию.

12. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую среднее арифметическое всех чисел целочислительного массива длиной 100 и обнуляющую все его элементы с числами, меньшими этого среднего арифметического. Использовать при этом функцию.

13. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую среднее арифметическое всех чисел целочислительного массива длиной 100 и умножающую каждое число массива на это среднее арифметическое. Использовать при этом функцию.

14. Даны четыре вещественных числа, являющихся величинами углов выпуклого четырехугольника, идущих по порядку. Нарисовать блок-схему и написать компьютерную программу, определяющую, является ли этот четырехугольник прямоугольником или параллелограммом. Использовать при этом функцию.

15. Нарисовать блок-схему и написать компьютерную программу, подсчитывающую произведение следующих n сомножителей вида $(1 + 1) \cdot (1 + 2 – 1/(1\cdot3)) \cdot (1 + 2 + 3 + 1/(1\cdot3\cdot5)) \cdot \ldots$ Использовать при этом функцию.

16. Нарисовать блок-схему и написать компьютерную программу, находящую минимум чисел целочислительного массива длиной 100, заменяющую элементы массива с положительными числами на противоположные им по знаку, а с отрицательными — на этот минимум. Использовать при этом функцию.

Аннотированная литература

1. Абрамов С. А., Зима Е. В. Начала программирования на языке паскаль.— М.: Наука. Гл. ред. физ.-мат. лит., 1987.— 112 с. Предлагается сокращенный вариант языка программирования Паскаль. Разбирается большое число примеров и предлагаются задачи для самостоятельного решения.

2. Богданов-Катьков Н. В., Хайт А. М. Самоучитель работы на персональном компьютере.— СПб.: Сова; М.: Изд-во ЭКСМО-Пресс, 2002.— 656 с., ил. Великолепно изложена аппаратная часть компьютера и ее история. Компьютерные программы представлены обычным комплектом.

3. Введение в программирование: Учеб. пособие для уч. сред. и ст. шк. возраста / Авт.-сост. В. А. Гольденберг.— Минск: ООО «Харвест», 1997.— 528 с. (Библиотека школьника). Подробно изложено начало истории развития компьютерной техники. Хорошо описаны алгоритмы и данные, имеются сведения о MS-DOS. Представлены четыре языка программирования: Бейсик, Паскаль, Си и ассемблер. Уделено внимание вирусам и играм, приведен исходный текст Го-Моку на Паскале.

4. Григорьев С. А. Программирование на языке Паскаль для математиков: Учебное пособие / Изд. 2-е, испр. и доп.; Калинингр. ун-т.— Калининград, 2000. ISBN 5-88874-189-2. Представлен опыт преподавания языка Паскаль на математическом факультете Калининградского госуниверситета.

5. Грошев С. В., Коцюбинский А. О. Аудио и видео на компьютере: Экспресс-курс.— М.: ТЕХНОЛОДЖИ-3000, 2002.— 256 с., ил. ISBN 5-94472-008-5. Данная книга предназначена для широкого круга читателей, начинающих знакомство с мультимедийными технологиями. Описаны физические основы звука и видео, а также принципы их оцифровки. Представлены проигрыватели звука и видео, встроенные в Windows. Приведено описание мощных редакторов звука Sound Forge и видео Ulead MediaStudio.

6. Ефимова О., Морозов В., Угринович Н. Курс компьютерной технологии с основами информатики: Учебное пособие для старших классов.— М., 1999. Один из авторов этого курса является также соавтором примерных программ вступительных испытаний по информатике в высшие учебные заведения Российской Федерации.

7. Леонтьев В. П. Новейшая энциклопедия Интернет.— М.: ОЛМА-–ПРЕСС, 2002.— 697 с., ил. ISBN 5-224-03203-2. Эта книга предназначена как для тех, кто только готовится подключиться к сети Интернет, так и для более опытных пользователей Сети. Написанная живо и увлекательно, она выгодно отличается от занудных и скучных «технических талмудов» и способна стать настольной книгой.

8. Леонтьев В. П. Новейшая энциклопедия персонального компьютера 2003.— М.: ОЛМА-ПРЕСС, 2003.— 920 с., ил. ISBN 5-224-04031-0. Книга известного автора. Энциклопедия 2002 под новой обложкой. Описаны бо-

лее 100 программ, самоучитель по Windows и Microsoft Office, справочник по комплектующим, основы работы в Интернет, словари компьютерных терминов. Весь материал проверен автором.

9. Окулов С. М. Основы программирования.—М.: Лаборатория Базовых Знаний, 2002.— 424 с. Книга является достаточно полным учебником по программированию, реализующим сложную задачу — формирование у читателя структурного стиля мышления.

10. Примерные программы вступительных испытаний в высшие учебные заведения Российской Федерации.— http://www.ed.gov.ru (Профессиональное образование: высшее).

11. Румянцев Д. Г., Монастырский Л. Ф. Путь программиста: Опыт создания личности программиста.— М.: ИНФРА-М, 2000.— 864 с., ил. ISBN 5-16-000507-2. Предпринята попытка дать не только базовые знания по программированию, но также рассмотрены методологические и логические основы формирования личности программиста. Материал подается не с точки зрения теоретика, а с позиции практика. В книге в основном используется язык программирования Паскаль, но разъясняются также языки Си и Ассемблер.

12. Степанов А. Н. Информатика.— СПб.: Питер, 2002.— 608 с. с ил. ISBN 5-94723-313-4. В книге рассматриваются основные понятия информатики: информация, информационная модель, алгоритм, архитектура персональных компьютеров и их программное обеспечение. Содержит материал, предусмотренный государственными стандартами по информатике для студентов гуманитарных специальностей вузов. Объем сведений даже превышает объем информатики для гуманитариев и книга может быть рекомендована для технических вузов.

13. Угринович Н. Д. Информатика и информационные технологии: Учебник для 10—11классов.— М.: Бином. Лаборатория знаний, 2002.— 512 с., ил. ISBN 5-94774-016-8. Классическая книга по подготовке к экзамену по информатике. Автор этого учебного пособия является одним из авторов примерных программ вступительных испытаний по информатике в высшие учебные заведения Российской Федерации.

14. Угринович Н. Д. Информатика и информационные технологии: Учебное пособие для 10—11 классов. Углубленный курс.— М.: Лаборатория базовых знаний, 2000. Классическая книга по подготовке к экзамену по информатике.

15. Частиков А. П. Архитекторы компьютерного мира.— СПб.: БХВ-Петербург, 2002.— 384 с., ил. ISBN 5-94157-138-0. Кто есть кто в компьютерном мире с XVII до XX века. История компьютеров в лицах с фотографиями. Разбита на пять разделов: предшественники и концептуалисты, первые изобретатели, выдающиеся конструкторы, знаменитые программисты и создатели компьютерных технологий.

Учебное издание

Сергей Валентинович Мациевский
Сергей Александрович Ишанов
Сергей Владимирович Клевцур

ИНФОРМАТИКА

Учебное пособие

Под редакцией С. В. Мациевского.
Набор и верстка С. В. Мациевского.

Подписано в печать 10.03.2003 г.. Формат $60\times90^{1}/_{16}$.
Бумага для множительных аппаратов. Усл. печ. л. 8,75.
Уч.-изд. л. 5,25. Тираж 300 экз. Заказ

Издательство Калининградского государственного университета
236041, г. Калининград, ул. А. Невского, 14.