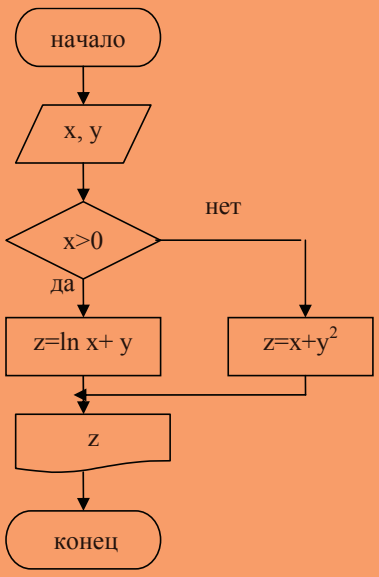
ТОЛЬЯТТИНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Е.В. Панюкова, Э.В. Егорова

ИНФОРМАТИКА

Учебно-методическое пособие



Тольятти
Издательство ТГУ
2012

Министерство образования и науки Российской Федерации
Тольяттинский государственный университет
Институт математики, физики и информационных технологий
Кафедра «Информатика и вычислительная техника»

Е.В. Панюкова, Э.В. Егорова

ИНФОРМАТИКА

Учебно-методическое пособие

Тольятти
Издательство ТГУ
2012

УДК 004
ББК 32.81
П168

Рецензенты:
к. техн. н., доцент Поволжского государственного университета сервиса *С.М. Бобровский*;
Тольяттинского государственного университета

П168 Панюкова, Е.В. Информатика : учеб.-метод. пособие / Е.В. Панюкова, Э.В. Егорова. – Тольятти : Изд-во ТГУ, 2012. – 148 с. : обл.

В учебно-методическом пособии изложены теоретические сведения по дисциплине, рассмотрены примеры по теме, предложены задания для практических и контрольных работ.

Предназначено для студентов, обучающихся по направлениям подготовки 150700.62 «Машиностроение», 190600.62 «Эксплуатация транспортно-технологических машин и комплексов», 280700.62 «Техносферная безопасность», заочной формы обучения.

УДК 004
ББК 32.81

Рекомендовано к изданию научно-методическим советом Тольяттинского государственного университета.

ВВЕДЕНИЕ

Учебно-методическое пособие предназначено для студентов заочной формы обучения и посвящено изучению основ информатики и информационных технологий.

Дисциплина «Информатика» входит в базовую часть математического и естественно-научного цикла. *Цель* освоения дисциплины – приобретение теоретических знаний в области аппаратного и программного обеспечения вычислительной техники, повышение информационной грамотности студентов, а также умений применять информационные технологии в профессиональной деятельности.

Задачи курса:

1) получить знания о современном уровне и направлениях развития аппаратных и программных средств вычислительной техники;
2) приобрести умения работы с вычислительными технологиями;
3) выработать навыки использования средств поиска и обмена информацией.

В ходе изучения дисциплины «Информатика» рассматриваются следующие учебные вопросы:

1) теоретические основы теории информации и кодирования;
2) основы архитектуры ЭВМ;
3) классификация программного обеспечения;
4) применение встроенных функций электронной таблицы Microsoft Excel в профессиональной деятельности;
5) алгоритмизация и программирование;
6) компьютерные сети.

В результате изучения дисциплины «Информатика» студент формирует следующие компетенции:

– целенаправленное применение базовых знаний в области математических, естественных, гуманитарных и экономических наук в профессиональной деятельности (ОК-9);
– осознание сущности и значения информации в развитии современного общества, владение основными методами, способами и средствами получения, хранения и переработки информации

(ОК-11);
- обладание навыками работы с компьютером как средством управления информацией (ОК-12);
- знание основных методов, способов и средств получения, хранения, переработки информации, использование для решения коммуникативных задач современных технических средств и информационных технологий с применением традиционных носителей информации, распределенных баз данных, а также информацией в глобальных компьютерных сетях (ОК-13).

В результате обучения студент должен:

знать:

- теоретические основы теории информации и кодирования;
- основы архитектуры ЭВМ;
- классификацию программного обеспечения;
- вопросы алгоритмизации и программирования;
- современные информационные технологии переработки информации;
- классификацию компьютерных сетей;

уметь:

- работать с программными средствами общего назначения, соответствующими современным требованиям;
- применять встроенные функции электронной таблицы Microsoft Excel в профессиональной деятельности;
- строить алгоритмы типовых задач в виде блок-схем;
- представлять алгоритмы задач в виде программ, вводить их в компьютер и получать результат;
- осуществлять поиск, обработку необходимой информации в компьютере;
- производить запись алгоритмов простейших задач в виде блок-схем и программ;

владеть навыками:

- использования компьютера как средства управления информацией;
- работы с прикладным программным обеспечением.

Глава 1. ОСНОВНЫЕ ПОНЯТИЯ И МЕТОДЫ ТЕОРИИ ИНФОРМАЦИИ И КОДИРОВАНИЯ

Цель – ознакомить студентов с основными понятиями и методами теории информатики.

Учебные вопросы

1. Формы, свойства, показатели качества информации.
2. Меры и единицы представления, измерения и хранения информации.
3. Системы счисления.
4. Кодирование данных в ЭВМ.
5. Основные понятия алгебры логики.
6. Логические основы ЭВМ.

Изучив данную тему, студент должен:

знать:

- основные понятия теории информатики;
- единицы представления, измерения и хранения информации;
- кодирование данных в ЭВМ;
- логические основы ЭВМ;
- информацию, ее измерение, количество и качество;

уметь:

- переводить числа из одной системы счисления в другую;
- пользоваться аппаратом алгебры логики;

владеть навыками:

- определения количества информации;
- установления показателей качества информации.

При освоении темы необходимо:

- изучить материал по теме 1;
- выполнить задание лабораторного практикума по теме 1;
- ответить на контрольные вопросы.

1.1. Формы, свойства, показатели качества информации

Информатика – наука, изучающая способы создания, хранения, обработки и передачи информации с помощью компьютера, а также принципы функционирования компьютеров и методы управления ими.

Термин «информатика» возник в 60-х гг. XX века во Франции для названия области, занимающейся автоматизированной обработкой информации с помощью электронных вычислительных машин. Французский термин *informatigue* (информатика) образован путем слияния слов *information* (информация) и *automatigue* (автоматика) и означает «информационная автоматика или автоматизированная переработка информации». В англоязычных странах этому термину соответствует синоним *computer science* (наука о компьютерной технике).

Информатику можно представить состоящей из трех взаимосвязанных частей:

1) технических средств (*hardware*);
2) программных средств (*software*);
3) алгоритмических средств (*brainware*).

Информация (от лат. *informatio* – сведение, разъяснение, ознакомление) – это сведения, снимающие неопределенность, об окружающем мире, которые являются объектом хранения, преобразования, передачи и использования.

Сведения – знания, выраженные в сигналах, сообщениях, известиях, уведомлениях и т. д.

Сигнал – любой процесс, несущий информацию.

Данные – это информация, представленная в формализованном виде и предназначенная для обработки ее техническими средствами, например, ЭВМ.

Сообщение – это информация, представленная в определенной форме и предназначенная для передачи.

Различают две ***формы представления информации – непрерывную (аналоговую) и дискретную.*** Поскольку носителями информации являются сигналы, то в качестве сигналов могут использовать-

ся физические процессы различной природы: процесс протекания электрического тока в цепи, механического перемещения тела, распространения света и т. д. Информация представляется (отражается) значением одного или нескольких параметров физического процесса (сигнала) либо комбинацией нескольких параметров.

Сигнал называется ***непрерывным***, если его параметр в заданных пределах может принимать любые промежуточные значения.

Сигнал называется ***дискретным***, если его параметр в заданных пределах может принимать отдельные фиксированные значения.

Качество информации является одним из важнейших параметров для потребителя информации. Оно определяется следующими ***свойствами***.

• ***Репрезентативность*** – правильность отбора информации в целях адекватного отражения источника информации.

• ***Достаточность*** – минимальный, но достаточный состав данных для достижения целей, которые преследует потребитель информации. Как неполная, так и избыточная информация снижает эффективность принимаемых пользователем решений.

• ***Доступность*** – простота (или возможность) выполнения процедур получения и преобразования информации. Например, в информационной системе информация преобразовывается к доступной и удобной для восприятия пользователя форме.

• ***Актуальность*** – определяется степенью сохранения ценности информации для управления в момент ее использования и зависит от динамики изменения ее характеристик и от интервала времени, прошедшего с момента возникновения данной информации.

• ***Своевременность*** – означает ее поступление не позже заранее назначенного момента времени, согласованного со временем решения поставленной задачи.

• ***Точность*** – степень близости получаемой информации к реальному состоянию объекта, процесса, явления и т. п.

• ***Адекватность*** – это определенный уровень соответствия создаваемого с помощью полученной информации образа реальному объекту, процессу, явлению и т. п.

• ***Устойчивость*** – способность информации реагировать на изменения исходных данных без нарушения необходимой точности.

Информация передаётся в виде сообщений от некоторого источника информации к её приёмнику посредством канала связи между ними. Источник посылает передаваемое сообщение, которое кодируется в передаваемый сигнал. Этот сигнал посылается по каналу связи. В результате в приёмнике появляется принимаемый сигнал, который декодируется и становится принимаемым сообщением.

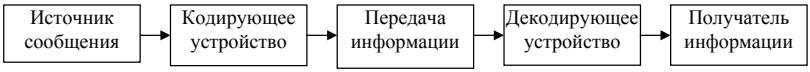


Рис. 1. Схема передачи информации

1.2. Меры и единицы представления, измерения и хранения информации

Количеством информации называют числовую характеристику сигнала, отражающую ту степень неопределенности (неполноту знаний), которая исчезает после получения сообщения в виде данного сигнала. Меру неопределенности в теории информации называют ***энтропией***.

Любая информация может рассматриваться как уменьшение неопределенности наших знаний об окружающем мире (в теории информации принято говорить именно об уменьшении неопределенности, а не об увеличении объема знаний).

Случайность любого события заключается в том, что реализация того или иного исхода имеет некоторую степень неопределенности.

Пусть, например, абсолютно незнакомый нам студент сдает экзамен, результатом которого может служить получение оценок «неудовлетворительно», «удовлетворительно», «хорошо» или «отлично». Поскольку мы ничего не знаем о данном студенте, то степень неопределенности всех перечисленных результатов сдачи экзамена совершенно одинакова. Напротив, если нам известно, как он учится, то уверенность в некоторых исходах будет больше, чем в других.

Наиболее просто определить количество информации в случае, когда все исходы события могут реализоваться с равной долей

вероятности. В этом случае для вычисления информации используется ***формула Хартли***[1]:

$$i = \log_2 N, \tag{1}$$

где i – количество информации; N – множество сообщений.

Согласно этой формуле процесс получения информации рассматривается как выбор одного сообщения из конечного наперёд заданного множества N равновероятных сообщений, а количество информации i, содержащееся в выбранном сообщении, определяется как двоичный логарифм N.

Наиболее простую формулу (1) можно представить следующим образом:

$$2^i = N. \tag{2}$$

Пример. Из колоды выбрали 8 карт и положили на стол рисунком вниз. Верхнюю карту перевернули. Сколько информации будет заключено в сообщении о том, какая карта оказалась сверху?

Решение. Все карты одинаковы, поэтому любая из них могла быть перевернута с одинаковой вероятностью. Событие, заключающееся в открытии карты, для нашего случая могло иметь 8 возможных вариантов. Следовательно, информация о реализации одного из них равняется

$$i = \log_2 8 = 3 \text{ бита}.$$

Пример. Бросают монету. При броске может выпасть «орел» или «решка». Сколько информации будет заключено в сообщении о том, что выпал «орел» или «решка»?

Решение. Воспользуемся формулой Хартли. Для данной задачи $N = 2$, следовательно, $i = \log_2 2 = 1$ бит.

В более сложной ситуации, когда исходы события ожидаются с разной степенью уверенности, требуются более сложные вычисления по ***формуле Шеннона***:

$$\begin{aligned} i = -\big(&p_1 \cdot \mathrm{Log}_2 p_1 + p_2 \cdot \mathrm{Log}_2 p_2 + \ldots \\ &\ldots + p_i \cdot \mathrm{Log}_2 p_i + \ldots + p_n \cdot \mathrm{Log}_2 p_n\big), \end{aligned} \tag{3}$$

где n – количество возможных событий; p_i – вероятности отдельных событий.

[1] Р. Хартли (1928) – американский инженер.

Бит – минимальная единица количества информации (необходимое для различения двух равновероятных сообщений)**.**

При получении информации в 1 бит неопределенность уменьшается в 2 раза. Таким образом, каждое бросание монеты дает нам информацию в 1 бит.

В ЭВМ информация представляется в виде набора бит, позволяющих описывать различную информацию.

Байт – основная единица измерения информации в ЭВМ.

1 байт = 8 бит.

Именно восемь бит требуется для того, чтобы закодировать любой из 256 символов алфавита клавиатуры компьютера ($256 = 2^8$).

Существуют производные единицы информации: ***килобайт*** (Кбайт, Кб), ***мегабайт*** (Мбайт, Мб), ***гигабайт*** (Гбайт, Гб), ***терабайт*** (Тбайт, Тб), ***петабайт*** (Пбайт, Пб).

1 Кб = 1024 байта = 2^{10} (1024) байта.

1 Мб = 1024 Кбайта = 2^{20} (1024^2) байта.

1 Гб = 1024 Мбайта = 2^{30} (1024^3) байта.

1 Тб = 1024 Гбайта = 2^{40} (1024^4) байта.

1 Пб = 1024 Тбайт = 2^{50} (1024^5) байта.

Контрольные вопросы

1. Что изучает информатика?
2. Дайте определение понятию «информация».
3. Формы представления информации.
4. Перечислите свойства информации.
5. Какова минимальная единица измерения информации?
6. Какова основная единица измерения информации?
7. Как задаются производные единицы измерения информации?
8. Дайте определение понятию «количество информации».
9. Как связаны между собой понятия «энтропия» и «информация»?
10. Что вычисляет формула Хартли?

1.3. Системы счисления

Информация в ЭВМ хранится и обрабатывается в закодированном виде. ЭВМ оперирует числами, представленными в некоторой системе счисления.

Системой счисления называется способ записи чисел с помощью заданного набора специальных знаков (цифр).

Системы счисления принято делить на два вида:

1) позиционные;

2) непозиционные.

В позиционных системах счисления «вес» каждой цифры изменяется в зависимости от ее положения (позиции) в последовательности цифр, изображающих число.

Пример. В числе 555 первая пятерка означает пять сотен, вторая – 5 десятков, а третья – 5 единиц.

В непозиционных системах вес цифры (т. е. тот вклад, который она вносит в значение числа) не зависит от ее позиции в записи числа.

Пример. При записи чисел в Римской системе счисления в числе XXI (двадцать один) «вес» цифры X в любой позиции равен просто десяти.

Любая позиционная система счисления характеризуется ***основанием*** – количеством различных знаков или символов, используемых для изображения чисел в данной системе.

За основание системы можно принять любое натуральное число. Следовательно, возможно бесчисленное множество позиционных систем: двоичная, троичная, четверичная и т. д. Запись чисел в каждой из систем счисления с основанием g означает сокращенную запись выражения

$$a_n a_{n-1} \ldots a_0, a_{-1} \ldots a_{-m} = a_n \cdot g^n + a_{n-1} \cdot g^{n-1} \ldots + a_0 \cdot g^0 + a_{-1} \cdot g^{-1} + \ldots + a_{-m} \cdot g^{-m}, \quad (4)$$

где a_i – цифры системы счисления; n и m – число целых и дробных разрядов соответственно; g – основание системы счисления.

Любая позиционная система счисления должна удовлетворять условию $a < g$.

Наибольшее распространение в ЭВМ получили двоичная, восьмеричная и шестнадцатеричная системы счисления.

Двоичная система счисления – для представления числа применяются две цифры: 0, 1.

Восьмеричная система счисления – для представления числа применяются цифры – от 0 до 7.

Шестнадцатеричная система счисления – для представления числа используются цифры от 0 до 9 и буквы латинского алфавита – A, B, C, D, E, F.

Запись первых двух десятков чисел в этих системах счисления представлена в табл. 1.

Таблица 1

Система представления чисел в системах счисления

Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

Арифметические операции в двоичной системе счисления

В табл. 2 представлены операции сложения, вычитания и умножения в двоичной системе счисления.

Таблица 2

Арифметические операции в двоичной системе счисления

Сложение	Вычитание	Умножение
$0 + 0 = 0$	$0 - 0 = 0$	$0 \times 0 = 0$
$1 + 0 = 1$	$1 - 0 = 1$	$1 \times 0 = 0$
$0 + 1 = 1$	$0 - 1 = 1$	$0 \times 1 = 0$
$1 + 1 = 10$	$1 - 1 = 0$	$1 \times 1 = 1$

Примечание. При сложении двух чисел, равных 1, в данном разряде получается 0, а единица переносится в старший разряд.

Пример. Даны числа $1001_{(2)}$ и $101_{(2)}$. Найти сумму этих чисел.

Решение

$$\begin{array}{r} 100\mathbf{1}_{(2)} \\ +10\mathbf{1}_{(2)} \\ \hline \mathbf{0}_{(2)}. \end{array}$$

1. При сложении двух единиц согласно табл. 2 получаем 10. В младшем разряде записываем **0**, а 1 переносится влево на одну позицию.

$$\begin{array}{r} 10\mathbf{0}1_{(2)} \\ +1\mathbf{0}1_{(2)} \\ \hline \mathbf{1}0_{(2)}. \end{array}$$

2. При сложении двух нолей получаем 0. Не забываем про 1, которую перенесли из младшего разряда. При сложении 0 и 1 получаем **1**.

$$\begin{array}{r} 1\mathbf{0}01_{(2)} \\ +\mathbf{1}01_{(2)} \\ \hline \mathbf{1}10_{(2)}. \end{array}$$

3. При сложении 0 и 1 получаем **1**.

$$\begin{array}{r} \mathbf{1}001_{(2)} \\ +101_{(2)} \\ \hline \mathbf{1}110_{(2).} \end{array}$$

4. В старшем разряде осталась только **1**.

5. Проведем проверку.

$$1001_{(2)} = 9_{(10)},\ 101_{(2)} = 5_{(10)},\ 1110_{(2)} = 14_{(10)},$$

$$9 + 5 = 14.$$

Пример. Даны числа $101_{(2)}$ и $11_{(2)}$. Найти сумму этих чисел.

Решение

$$\begin{array}{r} 101 \\ +11 \\ \hline 1000. \end{array}$$

где $101_{(2)} = 5_{(10)}$, $11_{(2)} = 3_{(10)}$, $1000_{(2)} = 8_{(10)}$.

Проверка: $5 + 3 = 8$.

При вычитании из 0 единицы занимается единица из старшего ближайшего разряда, отличного от 0. При этом единица, занятая в старшем разряде, даёт 2 единицы в младшем разряде и по единице во всех разрядах между старшим и младшим.

Пример. Даны числа $1101_{(2)}$ и $11_{(2)}$. Найти разность этих чисел.

Решение

$$\begin{array}{r} 1101 \\ -\ 11 \\ \hline 1010. \end{array}$$

Проверка

$$1101_2 = 2^3 + 2^2 + 1 = 13_{10};$$

$$11_2 = 2 + 1 = 3_{10};$$

$$1010_2 = 2^3 + 2 = 1\,0_{10}.$$

Пример. Даны числа $101_{(2)}$ и $11_{(2)}$. Найти разность этих чисел.

$$\begin{array}{r} 101 \\ -11 \\ \hline 10, \end{array}$$

где $101_{(2)} = 5_{(10)}$, $11_{(2)} = 3_{(10)}$, $10_{(2)} = 2_{(10)}$.

Проверка: $5 - 3 = 2$.

Операция умножения сводится к многократному сдвигу и сложению.

Пример. Даны числа $11_{(2)}$ и $10_{(2)}$. Найти произведение этих чисел.

$$\begin{array}{r} 11 \\ *10 \\ \hline 00 \\ 11 \\ \hline 110, \end{array}$$

где $11_{(2)} = 3_{(10)}$, $10_{(2)} = 2_{(10)}$, $110_{(2)} = 6_{(10)}$.

Проверка: $3 * 2 = 6$.

Перевод чисел из любой системы счисления в десятичную

Пример. Дано число 1101_2. Необходимо перевести число 1101_2 из двоичной системы счисления в десятичную.

Решение

1. Для перевода числа из любой системы счисления в десятичную необходимо разложить это число по степеням основания этой системы:

$$1101_{(2)} = 1^3 1^2 0^1 1^0{}_{(2)}.$$

2. Каждую цифру числа умножить на основание, возведенное в соответствующую степень:

$$1^3 1^2 0^1 1^0{}_{(2)} = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 8 + 4 + 0 + 1 = 13_{(10)}.$$

3. Число $1101_2 = 13_{(10)}$.

Примечание. При переводе важно помнить, что любое число в нулевой степени равно 1.

Пример. Дано число 13_4. Необходимо перевести число 13_4 из четверичной системы счисления в десятичную.

Решение

1. Для перевода числа из любой системы счисления в десятичную необходимо разложить это число по степеням основания этой системы:

$$13_{(4)} = 1^1 3^0{}_{(4)}.$$

2. Каждую цифру числа умножить на основание, возведенное в соответствующую степень:

$$1^1 3^0{}_{(4)} = 1 * 4^1 + 3 * 4^0 = 4 + 3 = 7_{(10)}.$$

3. Число $13_{(4)} = 7_{(10)}$.

Перевод чисел из десятичной системы счисления в любую другую

Пример. Дано число 13_{10}. Необходимо перевести число 13_{10} из десятичной системы счисления в двоичную.

Решение

1. Для перевода чисел из десятичной системы счисления в любую другую необходимо делить десятичное число на основание системы, в которую переводят, сохраняя при этом остатки от каждого деления. Деление продолжается до тех пор, пока результат деления не станет меньше делителя.

13/2 = 6 (остаток 1). Так как частное 6 больше делителя 2, продолжаем делить частное 6 на 2.

6/2 = 3 (остаток 0). Так как частное 3 больше делителя 2, продолжаем делить частное 3 на 2.

3/2 = 1 (остаток 1). Так как частное 1 меньше делителя 2, записываем полученное число.

13	2		
12	6	2	
1	6	3	2
	0	2	**1**
		1	

$13_{(10)} = 1101_{(2)}$.

2. Результат формируем справа налево (при формировании числа используют остатки деления). – $1101_{(2)}$.

Пример. Дано число 7_{10}. Необходимо перевести число 7_{10} из десятичной системы счисления в четверичную.

Решение

1. Для перевода чисел из десятичной системы счисления в любую другую необходимо делить десятичное число на основание системы, в которую переводят, сохраняя при этом остатки от каждого деления. Деление продолжается до тех пор, пока результат деления не станет меньше делителя.

7/4 = 1 (остаток 3). Так как частное 1 меньше делителя 4, записываем полученное число.

$7_{(10)} = 13_{(4)}$.

2. Результат формируем справа налево (при формировании числа используют остатки деления) – $13_{(4)}$.

Контрольные вопросы

1. Дайте определение понятию «система счисления».
2. Чем отличается позиционная система счисления от непозиционной?
3. Приведите примеры позиционной и непозиционной систем счисления.
4. В какой системе счисления при представлении числа используются буквы латинского алфавита?
5. Приведите примеры перевода чисел из двоичной в восьмеричную систему счисления.
6. Приведите примеры перевода чисел из десятичной в двоичную систему счисления.
7. Правила выполнения арифметических действий в двоичной системе счисления.

1.4. Кодирование данных в ЭВМ

Данные в компьютере имеют вид кода, который состоит из единиц и нулей в разной последовательности.

Код – набор условных обозначений для представления информации.

Кодирование – процесс представления информации в виде кода.

Кодирование текстовой информации

Поскольку текст изначально дискретен (он состоит из отдельных символов), для компьютерного представления текстовой информации используется способ, когда все символы кодируются числами и текст представляется в виде набора чисел – кодов символов, его составляющих. При выводе текста на экран монитора или принтера необходимо восстановить изображения всех символов, составляющих данный текст. Для этого используются так называемые кодовые таблицы символов, в которых каждому коду символа ставится соответствующее изображение.

Кодовая таблица – это внутреннее представление символов в компьютере.

Во всем мире в качестве стандарта принята таблица ASCII (American Standard Code for Information Interchange – Американский стандартный код для обмена информацией). Для хранения двоичного кода одного символа выделен 1 байт = 8 бит. Учитывая, что каждый бит принимает значение 0 или 1, количество их возможных сочетаний в байте равно $2^8 = 256$. Значит, с помощью 1 байта можно получить 256 разных двоичных кодовых комбинаций и отобразить с их помощью 256 различных символов. Эти комбинации и составляют таблицу ASCII. Эта таблица состоит из 16 строк и 16 столбцов, пронумерованных от 0 до *F* в шестнадцатеричной системе счисления. Например, в столбце 4 и строке *D* таблицы расположена заглавная буква *M* латинского алфавита. Таким образом, при записи текста с такой буквой, она будет храниться в памяти в виде кода 4D(16) или 77(10). Другие коды: «,» – 2C; «j» – 6A; «2» – 32. Такая форма кодирования позволяет представлять буквы в более компактном виде по сравнению с двоичным кодом.

Первые 8 столбцов таблицы кодов или первые 128 символов от 0 (двоичный код 00000000) до 127 (01111111) – цифры, буквы латинского алфавита, управляющие символы. Первые 32 символа являются управляющими и предназначены в основном для передачи команд управления. А последние 8 столбцов таблицы кодов, т. е. коды от 128 (двоичный код 10000000) до 255 (11111111) обычно содержат буквы национальных алфавитов, графические знаки. В большом количестве разновидностей таблицы кодов ASCII первая половина таблицы является неизменной, а вторая – переменной.

Однако 8-битовая кодировка (2^8) является недостаточной для кодировки всех символов расширенных алфавитов. Все препятствия могут быть сняты при переходе на 16-битовую (2^{16}) кодировку Unicode, допускающую 65536 кодовых комбинаций.

Необходимо помнить, что в настоящее время для кодировки русских букв используют пять различных кодовых таблиц (КОИ – 8, CP1251, CP866, Mac, ISO), причем тексты, закодированные при помощи одной таблицы, не будут правильно отображаться в другой кодировке. Наглядно это можно представить в виде фрагмента объединенной таблицы кодировки символов.

Одному и тому же двоичному коду ставятся в соответствие различные символы.

Пример

Двоичный код	Десятичный код	КОИ8	CP1251	CP866	Mac	ISO
11000010	194	б	В	-	-	Т

Впрочем, в большинстве случаев о перекодировке текстовых документов заботится не пользователь, а специальные программы – конверторы, которые встроены в приложения.

Начиная с 1997 года последние версии Microsoft Windows & Office поддерживают новую кодировку Unicode. Чтобы определить числовой код символа, можно воспользоваться кодовой таблицей или работать в текстовом редакторе MS Word. Для этого в меню нужно выбрать пункт «Вставка» – «Символ», после чего на экране появляется диалоговая панель «Символ». В диалоговом окне появляется таблица символов для выбранного шрифта. Символы в этой таблице располагаются построчно, последовательно слева направо, начиная с символа «Пробел» (левый верхний угол) и кончая буквой «я» (правый нижний угол).

Для определения числового кода символа в кодировке Windows (CP1251) нужно при помощи мыши или клавиш управления курсором выбрать нужный символ, затем щелкнуть по кнопке «Клавиша». После этого на экране появляется диалоговая панель «Настройка», в которой в нижнем левом углу содержится десятичный числовой код выбранного символа.

Кодирование чисел

Для кодирования числа, участвующего в вычислениях, используется специальная система правил перевода из десятичной системы счисления в двоичную. В результате число будет записано двоичным кодом, т. е. представлено различным сочетанием всего двух цифр: 0; 1.

Есть два основных формата представления чисел в памяти компьютера. Один из них используется для кодирования целых чисел, второй (так называемое представление числа в формате с плаваю-

щей точкой) используется для задания некоторого подмножества действительных чисел.

Например, целое число в зависимости от типа может кодироваться одним, двумя или четырьмя байтами. Для получения кода положительного целого числа достаточно перевести его из десятичной в двоичную систему счисления, например, десятичное число 12 кодируется как двоичное 00001100 (при однобайтовом типе числа). Отрицательные целые числа часто кодируются в так называемом дополнительном коде, когда старший двоичный разряд используется как признак отрицательности числа, а остальные разряды должны быть такими, чтобы сумма отрицательного числа и его модуля равнялась нулю.

Для вещественных (действительных с плавающей точкой) чисел система кодирования является более сложной. Обычно для каждого числа часть байтов отводится для хранения мантиссы числа (M, причем $1 < |M| < 10$), а часть – для порядка числа (b – целое число). Вещественное число представляется в виде $R = \pm M * 10^{\pm b}$.

Чаще всего в ЭВМ используют нормализованное представление числа в форме с плавающей точкой. Мантисса в таком представлении должна удовлетворять условию $0{,}1_p < M < 1_p$.

Иначе говоря, мантисса меньше 1 и первая значащая цифра – не ноль (p – основание системы счисления).

В памяти компьютера мантисса представляется как целое число, содержащее только значащие цифры (0 целых и запятая не хранятся), так для числа 12,345 в ячейке памяти, отведенной для хранения мантиссы, будет сохранено число 12345. Для однозначного восстановления исходного числа остается сохранить только его порядок. В данном примере порядок равен 2.

Кодирование графической информации

В видеопамяти находится двоичная информация об изображении, выводимом на экран. Почти все создаваемые, обрабатываемые или просматриваемые с помощью компьютера изображения можно разделить на две большие части – растровую и векторную графику.

Растровое изображение

Форму представления на экране дисплея графического изображения, состоящего из отдельных точек (пикселей), называют растровой. Минимальным объектом в растровом графическом редакторе является точка (пиксель). Растровые изображения представляют собой однослойную сетку точек, называемых пикселями (*pixel*, от англ. *picture element*). Код пикселя содержит информацию о его цвете.

Для черно-белого изображения (без полутонов) пиксель может принимать только два значения: белый и черный (светится – не светится), а для его кодирования достаточно одного бита памяти: 1 – белый, 0 – черный.

Пиксель на цветном дисплее может иметь различную окраску, поэтому одного бита на пиксель недостаточно. Для кодирования 4-цветного изображения требуются два бита на пиксель, поскольку два бита могут принимать 4 различных состояния. Может использоваться, например, такой вариант кодировки цветов: 00 – черный, 10 – зеленый, 01 – красный, 11 – коричневый.

Если говорить о кодировании цветных графических изображений, то нужно рассмотреть принцип декомпозиции произвольного цвета на основные составляющие. Применяют несколько систем кодирования: HSB, RGB и CMYK. Первая цветовая модель проста и интуитивно понятна, т. е. удобна для человека, вторая наиболее удобна для компьютера, а последняя модель CMYK – для типографий. Использование этих цветовых моделей связано с тем, что световой поток может формироваться излучениями, представляющими собой комбинацию «чистых» спектральных цветов: красного, зеленого, синего или их производных. Различают аддитивное цветовоспроизведение (характерно для излучающих объектов) и субтрактивное цветовоспроизведение (характерно для отражающих объектов). В качестве примера объекта первого типа можно привести электронно-лучевую трубку монитора, второго типа – полиграфический отпечаток.

Рассмотрим кратко основные цветовые модели.

1. Модель HSB характеризуется тремя компонентами: оттенок цвета (Hue), насыщенность цвета (Saturation) и яркость цвета (Brightness). Можно получить большое количество произвольных цветов, регулируя эти компоненты. Эту цветовую модель лучше

применять в тех графических редакторах, в которых изображения создают сами, а не обрабатывают уже готовые. Значение цвета выбирается как вектор, выходящий из центра окружности. Направление вектора задается в угловых градусах и определяет цветовой оттенок. Насыщенность цвета определяется длиной вектора, а яркость цвета задается на отдельной оси, нулевая точка которой имеет черный цвет. Точка в центре соответствует белому (нейтральному) цвету, а точки по периметру – чистым цветам.

2. Принцип метода RGB заключается в том, что любой цвет можно представить в виде комбинации трех цветов: красного (Red, R), зеленого (Green, G), синего (Blue, B). Другие цвета и их оттенки получаются за счет наличия или отсутствия этих составляющих. По первым буквам основных цветов система и получила свое название – RGB. Данная цветовая модель является аддитивной, то есть любой цвет можно получить сочетанием основных цветов в различных пропорциях. При наложении одного компонента основного цвета на другой яркость суммарного излучения увеличивается. Если совместить все три компонента, получим ахроматический серый цвет, при увеличении яркости которого происходит приближение к белому цвету.

При 256 градациях тона (каждая точка кодируется 3 байтами) минимальные значения RGB (0, 0, 0) соответствуют черному цвету, а белому – максимальные с координатами (255, 255, 255). Чем больше значение байта цветовой составляющей, тем этот цвет ярче. Например, темно-синий кодируется тремя байтами (0, 0, 128), а ярко-синий (0, 0, 255).

3. Принцип метода CMYK. Эта цветовая модель используется при подготовке публикаций к печати. Каждому из основных цветов ставится в соответствие дополнительный цвет (дополняющий основной до белого). Получают дополнительный цвет за счет суммирования пары остальных основных цветов. Значит, дополнительными цветами для красного является голубой (Cyan, C) = зеленый + синий = белый – красный, для зеленого – пурпурный (Magenta, M) = красный + синий = белый – зеленый, для синего – желтый (Yellow, Y) = красный + зеленый = белый – синий. Причем принцип декомпозиции произвольного цвета на составляющие можно применять как

для основных, так и для дополнительных. Любой цвет можно представить или в виде суммы красной, зеленой, синей составляющей или же в виде суммы голубой, пурпурной, желтой составляющей. В основном такой метод принят в полиграфии. Но там еще используют черный цвет (Black, так как буква *B* уже занята синим цветом, обозначают буквой *K*). Это связано с тем, что наложение друг на друга дополнительных цветов не дает чистого черного цвета.

Различают несколько режимов представления цветной графики:
1) полноцветный (True Color);
2) High Color;
3) индексный.

При полноцветном режиме для кодирования яркости каждой из составляющих используют по 256 значений (восемь двоичных разрядов), то есть на кодирование цвета одного пикселя (в системе RGB) надо затратить $8 * 3 = 24$ разряда. Это позволяет однозначно определять 16,5 млн цветов. Это довольно близко к чувствительности человеческого глаза. При кодировании с помощью системы CMYK для представления цветной графики надо иметь $8 * 4 = 32$ двоичных разряда.

Режим High Color – это кодирование при помощи 16-разрядных двоичных чисел, то есть уменьшение количества двоичных разрядов при кодировании каждой точки. Но при этом значительно уменьшается диапазон кодируемых цветов.

При индексном кодировании цвета можно передать всего лишь 256 цветовых оттенков. Каждый цвет кодируется при помощи восьми бит данных. Но так как 256 значений не передают весь диапазон цветов, доступный человеческому глазу, то подразумевается, что к графическим данным прилагается палитра (справочная таблица), без которой воспроизведение будет неадекватным: море может получиться красным, а листья – синими. Сам код точки растра в данном случае означает не сам по себе цвет, а только его номер (индекс) в палитре. Отсюда и название режима – индексный.

Количество различных отображаемых цветов K и битовая глубина (число разрядов, используемых для кодировки цвета) b связаны формулой:

$$K = 2^b, \quad (5)$$

где K – количество цветов; b – битовая глубина.

Зависимость цветовой палитры монитора от информационной емкости одного пикселя: 4 бита – 16 цветов, 8 бит – 256 цветов.

Разрешающая способность монитора (количество точек по горизонтали и вертикали), а также число возможных цветов каждой точки определяются типом монитора. Например: 640 ∗ 480 = 307 200 точек, 800 ∗ 600 = 480 000 точек.

Объем памяти, необходимой для хранения графического изображения, занимающего весь экран, равен произведению количества пикселей (разрешающей способности) на число бит, кодирующих одну точку. Объем графического файла в битах определяется как произведение количества пикселей $N * M$ на разрядность цвета C (битовую глубину):

$$V = M * M * C. \quad \text{с} \quad (6)$$

Например, при разрешении 640 ∗ 480 и количестве цветов 16 (4 бита) объем памяти равен:

640 ∗ 480 ∗ 4 = 1 228 800 (бит) = 153 600 (байт) = 150 (Кбайт).

Векторное изображение

В противоположность растровой графике векторное изображение многослойно. Каждый элемент векторного изображения – линия, прямоугольник, окружность или фрагмент текста – располагается в своем собственном слое, пиксели которого устанавливаются независимо от других слоев. Каждый элемент векторного изображения является объектом, который описывается с помощью специального языка (математических уравнений, линий, дуг, окружностей и т. д.) Сложные объекты (ломаные линии, различные геометрические фигуры) представляются в виде совокупности элементарных графических объектов.

Базовым элементом изображения является линия. Как и любой объект, она обладает свойствами: формой (прямая, кривая), толщиной, цветом, начертанием (пунктирная, сплошная). Замкнутые линии имеют свойство заполнения (или другими объектами, или выбранным цветом). Все прочие объекты векторной графики состав-

ляются из линий. Так как линия описывается математически как единый объект, то и объем данных для отображения объекта средствами векторной графики значительно меньше, чем в растровой графике. Информация о векторном изображении кодируется как обычная буквенно-цифровая и обрабатывается специальными программами.

Объекты векторного изображения, в отличие от растровой графики, могут изменять свои размеры без потери качества (при увеличении растрового изображения увеличивается зернистость).

К программным средствам создания и обработки векторной графики относятся следующие графические редакторы: Corel Draw, Adobe Illustrator, а также векторизаторы (трассировщики) – специализированные пакеты преобразования растровых изображений в векторные.

Кодирование звуковой информации

Из физики известно, что звук – это колебания воздуха. Если преобразовать звук в электрический сигнал (например, с помощью микрофона), то видно плавно изменяющееся с течением времени напряжение. Для компьютерной обработки такой – аналоговый – сигнал нужно каким-то образом преобразовать в последовательность двоичных чисел.

Делается это, например, так – измеряется напряжение через равные промежутки времени и полученные значения записываются в память компьютера. Этот процесс называется дискретизацией (или оцифровкой), а устройство, выполняющее его – аналого-цифровым преобразователем (АЦП).

Чтобы воспроизвести закодированный таким образом звук, нужно сделать обратное преобразование (для этого служит цифро-аналоговый преобразователь – ЦАП), а затем сгладить получившийся ступенчатый сигнал.

Чем выше частота дискретизации и чем больше разрядов отводится для каждого отсчета, тем точнее будет представлен звук, но при этом увеличивается и размер звукового файла. В настоящее время при записи звука в мультимедийных технологиях применяются частоты 8, 11, 22 и 44 кГц. Так, частота дискретизации 44 килогерца означает, что одна секунда непрерывного звучания заменяется набором из сорока четырех тысяч отдельных отсчетов сигнала.

Чем выше частота дискретизации, тем лучше качество оцифрованного звука. Поэтому в зависимости от характера звука, требований, предъявляемых к его качеству и объему занимаемой памяти, выбирают некоторые компромиссные значения.

Как отмечалось выше, каждый отдельный отсчет можно описать некоторой совокупностью чисел, которые затем можно представить в виде некоторого двоичного кода. Качество преобразования звука в цифровую форму определяется не только частотой дискретизации, но и количеством битов памяти, отводимых на запись кода одного отсчета. Этот параметр принято называть разрядностью преобразования. В настоящее время обычно используется разрядность 8, 16 и 24 бит. На описанных выше принципах основывается формат WAV (от WAVeform-audio – волновая форма аудио) кодирования звука. Получить запись звука в этом формате можно от подключаемых к компьютеру микрофона, проигрывателя, магнитофона, телевизора и других стандартно используемых устройств работы со звуком. Однако формат WAV требует очень много памяти. Так, при записи стереофонического звука с частотой дискретизации 44 килогерца и разрядностью 16 бит – параметрами, дающими хорошее качество звучания, – на одну минуту записи требуется около десяти миллионов байтов памяти.

Описанный способ кодирования звуковой информации достаточно универсален, он позволяет представить любой звук и преобразовывать его самыми разными способами. Но бывают случаи, когда выгодней действовать по-иному.

Издавна используется довольно компактный способ представления музыки – нотная запись. В ней специальными символами указывается, какой высоты звук, на каком инструменте и как сыграть. Фактически, ее можно считать алгоритмом для музыканта, записанным на особом формальном языке. В 1983 году ведущие производители компьютеров и музыкальных синтезаторов разработали стандарт, определивший такую систему кодов. Он получил название MIDI.

Конечно, такая система кодирования позволяет записать далеко не всякий звук, она годится только для инструментальной музыки. Но есть у нее и неоспоримые преимущества: чрезвычайно

компактная запись, естественность (практически любой MIDI-редактор позволяет работать с музыкой в виде обычных нот), легкость замены инструментов, изменения темпа и тональности мелодии.

Есть и другие, чисто компьютерные, форматы записи музыки. Среди них – формат MP3, позволяющий с очень большим качеством и степенью сжатия кодировать музыку, при этом вместо 18–20 музыкальных композиций на стандартном компакт-диске (CD-ROM) помещается около 200. Одна песня занимает примерно 3,5 Mb, что позволяет пользователям сети Интернет легко обмениваться музыкальными композициями.

Кодирование видеоинформации

Видеоинформация включает последовательность кадров и звуковое сопровождение, поэтому кодирование видеоинформации еще более сложная проблема, чем кодирование звуковой информации, так как нужно позаботиться не только о дискретизации непрерывных движений, но и о синхронизации изображения со звуковым сопровождением. В настоящее время для этого используется формат, который называется AVI (Audio-Video Interleaved – чередующееся аудио и видео).

Основные мультимедийные форматы AVI и WAV очень требовательны к памяти. Объем видеофайла примерно равен произведению количества информации в каждом кадре на число кадров. Число кадров вычисляется как произведение длительности видеоклипа Δt на скорость кадров v, то есть их количество в 1 с:

$$V = N * M * C * v * \Delta t. \quad (7)$$

При разрешении 800×600 точек, разрядности цвета $C = 16$, скорости кадров $v = 25$ кадров/с, видеоклип длительностью 30 с будет иметь объем: $800 * 600 * 16 * 25 * 30 = 576 * 10^7$ (бит) $= 72 * 10^7$ (байт) $= 687$ (Мбайт). Это много для такого короткого видеофрагмента, поэтому на практике применяются различные способы компрессии, то есть сжатия звуковых и видеокодов. В настоящее время стандартными стали способы сжатия, предложенные MPEG (Moving Pictures Experts Group – группа экспертов по движущимся изображениям). В частности, стандарт MPEG описывает несколько популярных в настоящее время форматов записи звука. Например,

при записи в формате MP3 при практически том же качестве звука требуется в десять раз меньше памяти, чем при использовании формата WAV. Существуют специальные программы, которые преобразуют записи звука из формата WAV в формат MP3. Не так давно был разработан стандарт MPEG-4, применение которого позволяет записать полнометражный цветной фильм со звуковым сопровождением на компакт-диск обычных размеров и качества.

Контрольные вопросы

1. Как представляются данные в компьютере?
2. Для чего используется кодовая таблица?
3. Как кодируются символы в памяти компьютера?
4. Что собой представляет таблица ASCII-кодов?
5. Как определить числовой код символа?
6. Как кодируются целые положительные числа в памяти компьютера?
7. Как кодируются вещественные числа в памяти компьютера?
8. Какие системы кодирования применяются для цветных графических изображений?

1.5. Основные понятия алгебры логики

Алгебра логики (булева алгебра) изучает высказывания, рассматриваемые со стороны их логических значений (истинности или ложности), и логические операции над ними.

Основным предметом алгебры логики являются ***высказывания***.

Под ***высказыванием*** понимается имеющее смысл языковое выражение, относительно которого можно утверждать, что оно либо истинно, либо ложно.

Примеры

- «5 есть простое число» – это высказыванием является истинным.
- «$4 + x = 6$» – это уравнение не является высказыванием. Однако, придавая переменной x определенное числовое значение, получим высказывание.
- «Все углы – прямые» – это высказывание является ложным.

Истинностные значения новых высказываний определяются при этом только истинностными значениями входящих в них высказываний. Построение из данных высказываний (или из данного высказывания) нового высказывания называется ***логической операцией***. Знаки логических операций называются ***логическими связками***.

Пример

- Из высказываний «$x > 2$», «$x < 3$» при помощи связки ***и*** можно получить высказывание «$x > 2$ и $x < 3$».
- Из высказываний «$y > 10$», «$x < 3$» при помощи связки ***или*** можно получить высказывание «$y > 10$ или $x < 3$».

Истинность или ложность получаемых таким образом высказываний зависит от истинности и ложности исходных высказываний и соответствующей трактовки связок как операций над высказываниями.

Одной из основных операций алгебры логики является операция ***отрицания***. Отрицание высказывания A (т. е. не А) обозначается $\overline{A}$ и читается: «отрицание A», «не A» или «A с чертой».

В табл. 3 приведены основные бинарные логические операции и связки.

Таблица 3

Основные бинарные логические операции и связки

Обозначение логической операции	Другие обозначения логической операции	Название логической операции и связки	Логические связки
A∧B	A&B A·B AB	**конъюнкция**, логическое умножение, логическое «и»	A **и** B; (A ***** B)
A∨B	A+B	**дизъюнкция**, логическое сложение, логическое «или»	A **или** B; (A +B)
A→B	A⊇B A⇒B	**импликация**, логическое следование	**если** A, **то** B;
A⊕B	AΔB	сумма по модулю, **разделительная дизъюнкция**, разделительное «или»	**либо** A, **либо** B
A~B	A≡B A↔B A⇔B	**эквиваленция**, тождественность равнозначность	A **тогда** **и только тогда, когда** B;

Обозначение логической операции	Другие обозначения логической операции	Название логической операции и связки	Логические связки
A | B	$\overline{A \wedge B}$	штрих Шеффера, **антиконъюнкция**	**неверно, что** A **и** B;
A↓B	$\overline{A \vee B}$	стрелка Пирса, **антидизъюнкция**	**ни** A, **ни B**

Примечание. ***A*** и ***B*** являются высказываниями.

Инверсия

Пример. Дано высказывание **A** =< Киев – столица Франции>.

Тогда **не** ***A*** = «**не** Киев – столица Франции». Высказывание **не** ***A*** означает – не верно, что *A*, т. е. не верно, что <Киев **–** столица Франции>.

Конъюнкция

Результатом операции конъюнкции для высказывания **A∧B** будет истина только тогда, когда истинны одновременно оба высказывания.

Пример. Даны высказывания ***A*** = «Москва – столица России» и ***B*** = «Рим – столица Италии».

Сложное высказывание **A∧B** = «Москва – столица России **и** Рим – столица Италии» истинно, так как истинны оба высказывания.

Дизъюнкция

Результатом операции дизъюнкции для высказывания **A∨B** будет истина тогда, когда истинно хотя бы одно высказывание, входящее в него.

Пример. Даны высказывания ***A*** = «2 + 3 = 5» и ***B*** = «3 + 3 = 5».

Сложное высказывание **A∨B** = «2 + 3 = 5 **или** 3 + 3 = 5» истинно, так как истинно высказывание ***A***.

Эквиваленция

Результатом операции эквиваленции для высказывания ***A~B*** будет истина тогда, когда истинны или ложны одновременно оба высказывания. Отличие эквиваленции от конъюнкции состоит в том, что вне зависимости от смысла, равнозначными являются как истинные, так и ложные высказывания.

Пример. Даны высказывания ***A*** = «2 + 2 = 7» и **B** = «1 – 8 = 5».

Сложное высказывание ***A~B*** = «2 + 2 = 7 **тогда и только тогда, когда** 1 – 8= 5» истинно, так как оба высказывания ложны.

Импликация

Результатом операции импликации для высказывания **A→B** будет ложь только тогда, когда первое высказывание (***A***) истинно, а второе (***B***) ложно. При этом ***A*** – предпосылка, а ***B*** – следствие. В остальных случаях результатом операции всегда будет истина.

Пример. Даны высказывания ***A*** = «2 + 2 = 4» и ***B*** = «1 – 8 = 5».

Сложное высказывание ***A→B*** = «**если** 2 + 2 = 4, **то** 1 – 8 = 5» ложно, так как высказывание *A* истинно, а *B* – ложно.

Антиконъюнкция

Результатом операции антиконъюнкции для высказывания **A | B** будет ложь только тогда, когда оба высказывания истинны. В остальных случаях результатом операции всегда будет истина.

Пример. Даны высказывания ***A*** = «Москва – столица России» и ***B*** = «Рим – столица Италии».

Сложное высказывание **A | B** = «**неверно, что** Москва – столица России **и** Рим – столица Италии» ложно, так как истинны оба высказывания.

Антидизъюнкция

Результатом операции антидизъюнкции для высказывания **A↓B** будет истина только тогда, когда оба высказывания ложны. В остальных случаях результатом операции всегда будет ложь.

Пример. Даны высказывания ***A*** = «Рим – столица России» и ***B*** = «Москва – столица Италии».

Сложное высказывание **A↓B** = «**ни** Рим – столица России, **ни** Москва – столица Италии» истинно, так как ложны оба высказывания.

Связки и частица «не» рассматриваются в алгебре логики как операции над величинами, принимающими значения 0 (ложь/false) и 1 (истина/true), и результатом применения этих операций также являются числа 0 или 1.

В алгебре логики логические операции чаще всего описываются при помощи ***таблиц истинности***.

В табл. 4 представлена таблица истинности для операции ***отрицания*** (***инверсия***).

Таблица 4

Таблица истинности для операции «отрицания»

A	**не** A
0	1
1	0

Пример. Дана переменная ***A* = 1** (истина). После применения операции инверсии для переменной ***A*** ее значение станет равным **0 (**ложь).

В табл. 5 представлены все наборы значений переменных ***A*** и ***B*** и значения операций на этих наборах.

Таблица 5

Таблица истинности для основных бинарных логических операций

A	***B***	$\wedge$	$\vee$	$\rightarrow$	$\oplus$	~	$\mid$	$\downarrow$
0	0	0	0	1	0	1	1	1
0	1	0	1	1	1	0	1	0
1	0	0	1	0	1	0	1	0
1	1	1	1	1	0	1	0	0

Пример. Даны высказывания ***A*** = «Москва – столица России» и ***B*** = «Рим – столица Италии». Следовательно, ***A*** = 1 (истина) и ***B*** = 1.

Чтобы определить значение операции **A$\wedge$B** для данных высказываний, необходимо:

- в табл. 6 в столбцах с именами ***A*** и ***B*** найти строку для ***A*** = 1 и ***B*** = 1;
- затем найти пересечение этой строки со столбцом с именем $\wedge$**;**
- получим **A$\wedge$B = 1.**

Таблица 6

A	***B***	$\wedge$	$\vee$	$\rightarrow$	$\oplus$	~	$\mid$	$\downarrow$
0	0	0	0	1	0	1	1	1
0	1	0	1	1	1	0	1	0
1	0	0	1	0	1	0	1	0
1	**1**	**1**	1	1	0	1	0	0

Пример. Даны высказывания A = «2 + 3 = 5» и B = «3 + 3 = 5». Тогда $A = 1$ и $B = 0$.

Высказывание $\mathbf{A \vee B = 1}$ демонстрируется табл. 7.

Таблица 7

A	B	$\wedge$	$\vee$	$\rightarrow$	$\oplus$	~	$\mid$	$\downarrow$
0	0	0	0	1	0	1	1	1
0	1	0	1	1	1	0	1	0
1	**0**	0	**1**	0	1	0	1	0
1	1	1	1	1	0	1	0	0

Пример. Даны высказывания A = «2 + 2 = 4» и B = «1 – 8 = 5». Тогда $A = 1$ и $B = 0$.

Высказывание $\mathbf{A \sim B} = 0$ можно увидеть в табл. 8.

Таблица 8

A	B	$\wedge$	$\vee$	$\rightarrow$	$\oplus$	~	$\mid$	$\downarrow$
0	0	0	0	1	0	1	1	1
0	1	0	1	1	1	0	1	0
1	**0**	0	1	0	1	**0**	1	0
1	1	1	1	1	0	1	0	0

Пример. Дана логическая формула $F = (A \wedge \overline{B}) \rightarrow A$. Построить таблицу истинности для данной формулы.

Решение

1. Расставляем приоритеты выполнения операций:

1) $X = \overline{B}$ – операция отрицания высказывания B. Результат выполнения операции присваиваем переменной X;
2) $Y = A \wedge X$ – операция логического умножения (конъюнкция) высказываний A и X. Результат выполнения операции присваиваем переменной Y;
3) $F = Y \rightarrow A$ – операция логического следования (импликация) высказываний Y и A. Результат выполнения операций присваиваем переменной F.

2. Строим таблицу, состоящую из пяти столбцов:

Исходные данные		X	Y	F
A	B			

В **bсходные данные** таблицы записываем имена высказываний *A* и *B*. В остальные три столбца записываем имена переменных, которым присваиваем результаты логических операций.

3. **Исходные данные** таблицы заполняем возможными комбинациями значений высказываний *A* и *B* (первый вариант – когда оба высказывания истинны; второй и третий варианты – когда одно из высказываний истинно, а другое – ложно; четвертый вариант – когда оба высказывания ложны).

Исходные данные		X	Y	F
A	B			
A	**B**			
1	**1**			
0	**1**			
1	**0**			
0	**0**			

Примечание. Истина обозначается – 1, ложь – 0.

4. Заполняем значениями столбец с именем *X*. Для этого по таблице истинности основных логических операций (см. Курс лекций, пункт 2.1.5) определяем значение операции инверсия $X = 0$ (при начальном значении $B = 1$) и т. д.

Исходные данные		**X**	Y	F
A	B			
1	1	**0**		
0	1	**0**		
1	0	**1**		
0	0	**1**		

5. Заполняем значениями столбец с именем *Y*. Для этого по таблице истинности основных логических операций определяем значение операции конъюнкции $Y = 0$ (при $A = 1$ и $X = 0$) и т. д.

Исходные данные		X	**Y**	F
A	B			
1	1	0	**0**	
0	1	0	**0**	
1	0	1	1	
0	0	1	**0**	

6. Заполняем значениями столбец с именем *F*. Для этого по таблице истинности основных логических операций определяем значение операции логическое следование $F = 1$ (при $Y = 0$ и $A = 1$) и т. д.

Исходные данные		X	Y	**F**
A	B			
1	1	0	0	**1**
0	1	0	0	**1**
1	0	1	1	**1**
0	0	1	0	**1**

1.6. Логические основы ЭВМ

Для описания функционирования аппаратных средств компьютера очень удобен математический аппарат алгебры логики, поскольку основной системой счисления в компьютере является двоичная, в которой используются цифры «1» и «0».

Одни и те же устройства компьютера могут применяться для обработки и хранения как числовой информации, представленной в двоичной системе счисления, так и логических переменных.

На этапе конструирования аппаратных средств алгебра логики позволяет значительно упростить логические функции, описывающие функционирование схем компьютера, и, следовательно, уменьшить число элементарных логических элементов, из десятков тысяч которых состоят основные узлы компьютера.

В логической схеме компьютера выделяют ***логические элементы***. ***Логический элемент*** компьютера – это часть электронной логической схемы, которая реализует элементарную логическую формулу.

Логическими элементами компьютеров являются электронные схемы «И», «ИЛИ», «НЕ», «И-НЕ», «ИЛИ-НЕ». С помощью этих схем можно реализовать любую логическую формулу, описывающую работу устройств компьютера.

Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую формулу, но не указывает на то, какая именно электронная схема в нем реализована. Это упрощает запись и понимание сложных логических схем.

Схема «И» реализует конъюнкцию двух или более логических значений. Условное обозначение структурной схемы «И» представлена на рис. 2.

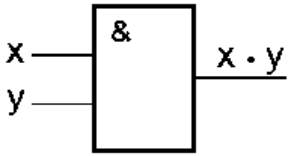


Рис. 2. Схема «И»

На выходе схемы «И» значение «1» будет тогда и только тогда, когда на всех входах будет «1». Когда хотя бы на одном входе будет «0», на выходе также будет «0».

Операция конъюнкции на функциональных схемах обозначается знаком «&» (читается как «амперсэнд»), являющимся сокращенной записью английского слова *and*.

Схема «ИЛИ» реализует дизъюнкцию двух логических значений. Условное обозначение схемы «ИЛИ» представлено на рис. 3.

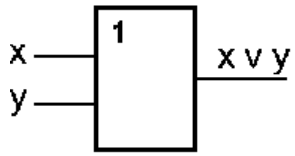


Рис. 3. Схема «ИЛИ»

На выходе схемы «ИЛИ» значение «0» будет тогда и только тогда, когда на всех входах будут «0». Когда хотя бы на одном входе будет «1», на выходе также будет «1».

Операция дизъюнкции на функциональных схемах обозначается знаком «1».

Схема «НЕ» (инвертор) реализует операцию отрицания. Условное обозначение схемы НЕ представлено на рис. 4.

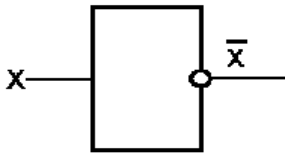


Рис. 4. Схема «НЕ»

Если на входе схемы – «0», то на выходе будет «1». Когда на входе – «1», на выходе будет «0».

Схема «И-НЕ» состоит из элемента «И» и инвертора и осуществляет отрицание результата схемы «И». Условное обозначение схемы «И-НЕ» представлено на рис. 5.

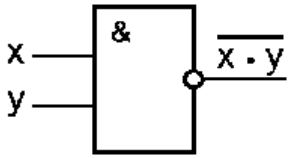


Рис. 5. Схема «И-НЕ»

На выходе схемы «И-НЕ» значение «0» будет тогда и только тогда, когда на всех входах будут «1».

Схема «ИЛИ-НЕ» состоит из элемента «ИЛИ» и инвертора и осуществляет отрицание результата схемы «ИЛИ». Условное обозначение схемы «ИЛИ-НЕ» представлено на рис. 6.

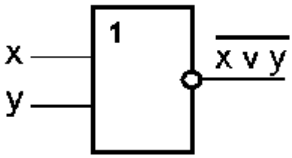


Рис. 6. Схема «ИЛИ-НЕ»

На выходе схемы «ИЛИ-НЕ» значение «1» будет тогда и только тогда, когда на всех входах будут «0».

Контрольные вопросы

1. Что изучает алгебра логики?
2. Что понимается под *высказыванием*?
3. Перечислите основные логические операции.
4. Назовите соответствующие логические связки для каждой логической операции.
5. Для чего используется таблица истинности?
6. Для высказываний A = «На улице светит солнце» и B = «Идет дождь» примените операцию конъюнкции. Какое новое высказывание получилось?
7. Для высказываний A = «У меня в зачетке стоят одни пятерки» и B = «Я добросовестно выполняю задания» примените операцию эквиваленции. Какое новое высказывание получилось?
8. Как изображается логическая схема «И-НЕ»?
9. Как изображается логическая схема «ИЛИ»?
10. Как изображается логическая схема «НЕ»?

1.7. Практические работы

Арифметические операции в двоичной системе счисления

Цель работы – научиться выполнять арифметические операции в двоичной системе счисления.

Порядок выполнения работы

1. Изучите пункт 1.3 учебно-методического пособия.
2. Выполните задания.
3. Оформите отчет о работе.

Задание 1. Даны числа $1101_{(2)}$ и $111_{(2)}$. Найти сумму этих чисел и провести проверку.

Задание 2. Даны числа $1110_{(2)}$ и $101_{(2)}$. Найти разность этих чисел и провести проверку.

Задание 3. Даны числа $110_{(2)}$ и $111_{(2)}$. Найти произведение этих чисел и провести проверку.

Построение таблиц истинности для логических формул

Цель работы – научиться строить таблицы истинности для заданных логических формул

Порядок выполнения работы

1. Изучите пункт 1.5 учебно-методического пособия.
2. Выполните задания.
3. Оформите отчет о работе.

Задание 4. Дана логическая формула $F = \left(\overline{A} \to \overline{B}\right) \wedge A \sim B$. Построить таблицу истинности для данной формулы.

Глава 2. ТЕХНИЧЕСКИЕ СРЕДСТВА РЕАЛИЗАЦИИ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

Цель – ознакомить студентов с основными принципами работы аппаратных средств ЭВМ.

Учебные вопросы

1. Архитектура ЭВМ.
2. Назначение основных элементов персонального компьютера.

Изучив данную тему, студент должен знать

– основные принципы построения ЭВМ;
– структуру и характеристики центрального процессора компьютера;
– виды памяти компьютера.

2.1. Архитектура ЭВМ

Под архитектурой ЭВМ понимается совокупность общих принципов организации аппаратно-программных средств и их характеристик, определяющая функциональные возможности ЭВМ при решении соответствующих классов задач.

Компьютер является электронным устройством для хранения, обработки информации, переработки и создания данных.

1. Внешние устройства персонального компьютера

1) системный блок;
2) клавиатура (ввод);
3) монитор (вывод);
4) принтер (вывод);
5) мышь – манипулятор ввода;
6) трекбол – шаровой манипулятор ввода;
7) сканер – считывает информацию с бумаги в файл (ввод);
8) плоттер – устройство, выводящее рисунки и графику;
9) стример – ленточный накопитель – устройство для быстрой перезаписи данных с жёсткого диска на магнитную ленту;
10) модем – устройство, выполняющее модуляцию и демодуляцию

информационных сигналов при передаче их из ЭВМ в канал связи и при приёме в ЭВМ из канала связи.

2. Принципы фон Неймана

Принципы фон Неймана – это схема классического компьютера, предложенная американским математиком Джоном фон Нейманом в 1945 году (рис. 7).

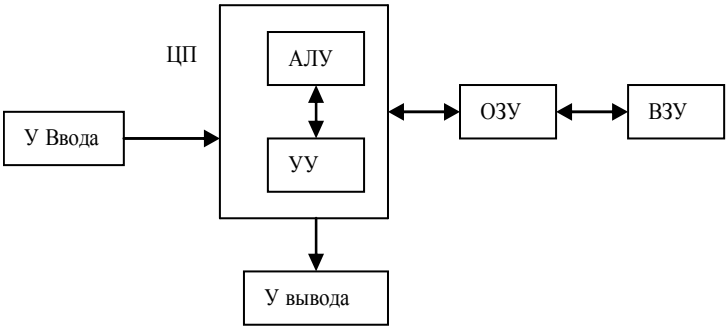


Рис. 7. Схема фон Неймана

Основные принципы построения ЭВМ, сформулированные Джоном фон Нейманом:

1) любую ЭВМ образуют три компонента: процессор, память и устройства ввода-вывода (УВВ);
2) информация, с которой работает ЭВМ, делится на два типа: набор команд по обработке (программы) и данные, подлежащие обработке;
3) один из важнейших принципов – принцип хранимой программы – требует, чтобы *программа закладывалась в память* машины так же, как в нёе закладывается исходная информация. Команды и данные вводятся в память (ОЗУ);
4) руководит обработкой процессор, устройство управления (УУ) которого выбирает команды из ОЗУ и организует их выполнение, а арифметико-логическое устройство (АЛУ) проводит арифметические и логические операции над данными;
5) с процессором и ОЗУ связаны устройства ввода-вывода (УВВ);
6) использование двоичной системы для представления чисел в ЭВМ (ранее хранили в десятичном виде).

ЭВМ неймановской архитектуры содержит следующие основные устройства:

1) центральный процессор (ЦП), включающий арифметико-логическое устройство (**АЛУ**) и устройство управления (**УУ**);
2) оперативное (**ОЗУ**);
3) внешнее запоминающее устройство (**ВЗУ**);
4) устройства ввода-вывода (**УВВ**).

Архитектура современных персональных компьютеров основана на магистрально-модульном принципе. Информационная связь между устройствами компьютера осуществляется через системную шину (другое название – системная магистраль).

Шина – это кабель, состоящий из множества проводников. По одной группе проводников – шине данных передаётся обрабатываемая информация, по другой – шине адреса передаются адреса памяти или внешних устройств, к которым обращается процессор. Третья часть магистрали – шина управления, по ней передаются управляющие сигналы (например, сигнал готовности устройства к работе, сигнал к началу работы устройства и др.).

Системная шина характеризуется тактовой частотой и разрядностью. Количество одновременно передаваемых по шине бит называется разрядностью шины. Тактовая частота характеризует число элементарных операций по передаче данных в одну секунду. Разрядность шины измеряется в битах, тактовая частота – в мегагерцах.

2.2. Состав и назначение основных элементов персонального компьютера

Системный блок является центральной частью ПК. Внутри корпуса системного блока размещены электронные схемы, смонтированные на нескольких печатных платах. Кроме того, в системном блоке находится ***блок питания***, преобразующий поступающий из сети переменный ток напряжением 220 V в постоянный ток низкого напряжения, вентилятор, жесткий магнитный диск, дисководы для магнитных дискет, устройства для чтения/записи CD (DVD) дисков.

Материнская плата – относится к конструктивной части компьютера и является основной платой ПК. На ней размещаются процессор, микропроцессорный комплект, шины, ОЗУ, ПЗУ, разъёмы

для подключения дополнительных устройств. Материнская плата предназначена для взаимодействия её устройств и обмена информацией между ними.

Центральный процессор (ЦП) – функциональная часть ЭВМ, выполняющая основные операции по обработке данных и управлению работой других блоков. Центральный процессор (микропроцессор) предназначен для обеспечения общего управления ЭВМ.

Это наиболее сложный компонент ЭВМ как с точки зрения электроники, так и функциональных возможностей. Центральный процессор состоит из следующих взаимосвязанных составных элементов: арифметико-логического устройства, устройства управления и регистров.

Микропроцессор выполнен в виде СБИС (сверхбольшой интегральной схемы), содержит около 10^6 элементов.

Центральный процессор – это «мозг» ЭВМ, основная микросхема, выполняющая арифметические и логические операции, и управляющая другими устройствами компьютера.

Основные характеристики процессора

1. *Разрядность* показывает, сколько бит данных процессор может принять и обработать в своих регистрах за один такт.

2. *Рабочая тактовая частота* – это число операций в секунду (Гц). Рабочая частота некоторых процессоров превосходит 3 миллиарда тактов в секунду (3 ГГц).

3. *Коэффициент внутреннего умножения тактовой частоты* может быть от 10 до 20 и выше.

4. *Размер кэш-памяти*. Внутри процессора существует буферная область для быстродействия – это кэш-память.

Арифметико-логическое устройство (АЛУ) входит в состав процессора, выполняет основную работу по переработке информации, хранимой в оперативной памяти. В нём выполняются арифметические и логические операции.

Операции выполняются с помощью электронных схем, каждая из которых состоит из нескольких тысяч элементов.

Устройство управления (УУ) – это функциональная часть центрального процессора. Оно вырабатывает последовательность управляющих сигналов, обеспечивающих выборку и выполнение команд.

Запоминающие устройства: классификация, принцип работы, основные характеристики

Запоминающие устройства (ЗУ) служат для хранения программ. Память персонального компьютера (ПК) подразделяется на внутреннюю и внешнюю. Внутренняя память подразделяется:

1) на оперативную;
2) постоянную;
3) буферную.

В табл. 9 представлены основные характеристики, назначение каждой из видов памяти.

Таблица 9

Память			
Внешняя (ВЗУ)	Внутренняя		
	ОЗУ (RAM)	ПЗУ (ROM)	КЭШ (буфер)
1. Характеристика – энергозависимость			
энерго**не**зависимая	энергозависимая	энерго**не**зависимая	энергозависимая
2. Назначение			
Для длительного хранения программ и данных, любой информации внутри компьютера	Для кратковременного хранения программ и данных во время работы компьютера	Для проверки исправности ПК и первоначальной загрузки. В момент включения компьютера стартовый адрес указывает на ПЗУ. Хранит и выдаёт программу «БИОС» в момент включения ПК	Для ускорения доступа к оперативной памяти

ПЗУ – постоянное запоминающее устройство

Медленная память необходима для запуска компьютера при включении, она энергонезависима.

ОЗУ – оперативное запоминающее устройство. или оперативная память (ОП)

Время доступа – определяющая характеристика оперативной памяти (ОП). Оно измеряется в миллиардных долях секунды (наносекундах, нс). 5 нс – для современных модулей памяти. Память состоит из конечного числа ячеек, каждая из которых имеет свой уникальный номер или адрес. Доступ к ячейке осуществляется указанием её адреса.

КЭШ-память – сверхоперативная память

КЭШ-память располагается между процессором и оперативной памятью. При обращении микропроцессора к памяти сначала производится поиск нужных данных в кэш-памяти, благодаря чему уменьшается среднее время доступа к памяти.

Часть оперативной памяти отводится для хранения изображений, получаемых на экране монитора, и называется ***видеопамять***. Чем больше видеопамять, тем более сложные и качественные изображения можно получать на дисплее.

ОЗУ, КЭШ являются энергозависимыми, т. е. очищаются при отключении питания.

ВЗУ – внешнее запоминающее устройство (энергонезависимо)

Типы ВЗУ:

1) винчестер – накопитель на жёстких магнитных дисках. Принцип записи данных на винчестер заключается в намагничивании поверхности диска;
2) дискеты – накопитель на гибких магнитных дисках;
3) лазерные диски. Компакт-диск (CD) – это оптический диск, информация с которого считывается лазерным лучом;
4) флэш-память (Flash) – съемные накопители данных, отличающиеся постоянно растущим от одной модели к другой объемом памяти.

Порты – это устройства, через которые периферийные устройства присоединяются к системному блоку. Аппаратно порты реализуются в виде разъемов на задней стенке системного блока. Обычно выделяются следующие типы портов:

- последовательный порт (COM, PS/2) – осуществляет передачу символов данных по одному биту. Через COM-порты подключают мышь и модем. Через PS/2-порт – клавиатуру и мышь;
- параллельный порт (LPT) – одновременно передается байт данных. Используется для принтеров и сканеров. Порт USB – универсальный порт, к которому можно присоединить до 12 внешних устройств, поддерживающих стандарт USB. Это может быть принтер, сканер, монитор, клавиатура, мышь и т. д.

Помимо названных портов существуют и другие.

Основные характеристики вычислительной техники

1. Быстродействие, которое измеряется количеством элементарных операций, выполняемых центральным процессором в секунду (герц). В зависимости от области применения выпускаются ЭВМ с быстродействием от нескольких сотен тысяч до миллиардов операций в секунду.

2. Объем оперативной памяти определяется максимальным количеством информации, которое можно разместить в памяти ЭВМ.

3. Точность вычислений зависит от количества разрядов (бит), используемых для представления одного числа. Современные ЭВМ комплектуются 32- или 64-разрядными микропроцессорами, что вполне достаточно для обеспечения высокой точности расчетов в самых разнообразных приложениях

4. Надёжность ЭВМ – это способность машины сохранять свои свойства при заданных условиях эксплуатации в течение определенного промежутка времени.

Контрольные вопросы

1. Основные принципы построения ЭВМ, сформулированные Джоном фон Нейманом.
2. Внешние устройства персонального компьютера для ввода информации.
3. Внешние устройства персонального компьютера для вывода информации.
4. Основные характеристики центрального процессора.
5. Типы ВЗУ.
6. Арифметико-логическое устройство (АЛУ), структура и назначение.
7. Основные компоненты любой ЭВМ.
8. Назначение центрального процессора.
9. Виды памяти компьютера и назначение.
10. Отличия между внешней и внутренней памятью.
11. Характеристики микропроцессора.
12. Основные характеристики вычислительной техники.

Глава 3. ПРОГРАММНЫЕ СРЕДСТВА РЕАЛИЗАЦИИ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

Цель – ознакомить студентов с классификацией и назначением программного обеспечения.

Учебные вопросы

1. Классификация программного обеспечения компьютеров.
2. Понятие и назначение операционной системы.
3. Файлы и файловая система Windows.
4. Сервисные системные программы.
5. Прикладное программное обеспечение.

Изучив данную тему, студент должен

знать:

– классификацию программного обеспечения компьютеров;
– назначение операционных систем;
– понятие и функции файловой системы;
– назначение и возможности сервисного программного обеспечения;
– классификацию и возможности прикладного программного обеспечения;

уметь:

– создавать папки и файлы, копировать, переименовывать, удалять их;
– выполнять любые операции с папками и файлами.

3.1. Классификация программного обеспечения

Программное обеспечение (ПО) – это совокупность всех программ и необходимой для их эксплуатации документации.

Программное обеспечение в настоящее время составляет сотни тысяч программ, которые предназначены для обработки разнообразной информации с различными целями.

Классификация программного обеспечения представлена в табл. 10.

Таблица 10

№	Программное обеспечение (ПО)			
	Прикладное программное обеспечение (ППО)	Системное программное обеспечение (СПО)		Системы программирования (инструментальные)
		Базовое СПО	Сервисное СПО (служебные программы)	
1	Текстовые процессоры	Операционные системы	Средства диагностики	Трансляторы
2	Табличные процессоры	Операционные оболочки	Антивирусные программы	Среда разработки программ
3	Базы данных	Сетевые операционные системы	Средства контроля	Отладчики
4	Интегрированные пакеты		Средства сжатия данных (архиваторы)	Библиотеки справочных программ (функций, процедур)
5	Обучающие программы и т. д.		Средства коммуникации и т. Д.	Редакторы связей и др.

1. Системное программное обеспечение (СПО)

Системное ПО – это совокупность программ для обеспечения работы компьютера, оно подразделяется на базовое и сервисное.

Сервисное программное обеспечение (служебные программы)

Сервисные или вспомогательные программы (утилиты) предназначены для обслуживания и повышения эффективности вычислительной системы, их называют служебными программами.

Программы-архиваторы позволяют сжимать информацию на дисках, создавать копии файлов меньшего размера для их хранения. Примеры программ –**WinRar** и **WinZip**.

Программы для создания резервных копий информации позволяют периодически копировать важную информацию, находящуюся на жестком диске компьютера, на дополнительные носители. Примеры программ резервного копирования – **APBackUp**, **Acronis True Image**.

Антивирусные программы предназначены для предотвращения заражения компьютерными вирусами и ликвидации последствий

заражения вирусом. Представители антивирусного семейства программ – **Kaspersky Antivirus**, **DrWeb**, **Norton Antivirus**.

Коммуникационные программы предназначены для обмена информацией между компьютерами. Эти программы позволяют удобно пересылать файлы с одного компьютера на другой при соединении кабелем их последовательных портов. Другой вид таких программ обеспечивает возможность связи компьютеров по телефонной сети (при наличии модема). Они дают возможность посылать и принимать телефаксные сообщения. Примеры коммуникационных программ – **Venta Fax**, **Cute FTP**.

Программы для диагностики компьютера позволяют проверить конфигурацию компьютера (количество памяти, ее использование, типы дисков и т. д.), проверить работоспособность устройств компьютера, оценить его производительность. Представители программ диагностики компьютеров – **Sisoft Sandra**, **Norton System Information**.

Программы для оптимизации дисков позволяют обеспечить более быстрый доступ к информации на диске за счет оптимизации размещения данных на диске. Эти программы перемещают все участки каждого файла друг к другу (устраняют фрагментацию), собирают все файлы в начале диска и т. д., за счет чего уменьшается число перемещений головок диска (т. е. ускоряется доступ к данным) и снижается износ диска. Представители программ для оптимизации дисков – **Norton Disk Doctor**, **Microsoft Scandisk**.

2. Системы программирования

Системы программирования (инструментарий технологии программирования) – это совокупность программ для разработки, отладки и внедрения новых программных продуктов. Примеры таких систем: Turbo Pascal, Qbasic, Delphi и др. Системы программирования обычно содержат:

1) трансляторы;
2) среду разработки программ;
3) библиотеки справочных программ (функций, процедур);
4) отладчики;
5) редакторы связей и др.

3. Прикладное программное обеспечение

Прикладное программное обеспечение (ППО) – это совокупность программ для решения задач конкретной предметной области. Прикладное ПО – это пакеты прикладных программ (ППП), работает только при наличии системного ПО.

Прикладное программное обеспечение включает:

1) текстовые редакторы и процессоры (Microsoft Word, Wordpad, Notepad);
2) табличные процессоры (Microsoft Excel, Quatro Pro);
3) графические редакторы (Paint, Corel Draw, Adobe Photoshop);
4) системы управления базами данных (Access, FoxPro, Paradox, Clipper);
5) системы автоматизированного проектирования (САПР)? позволяющие осуществлять черчение и конструирование (система AutoCad фирмы AutoDesk, отечественный пакет с аналогичными функциями «Компас»);
6) интегрированные пакеты (Microsoft Office);
7) презентации (Microsoft PowerPoint);
8) web-редакторы;
9) браузеры (обозреватели) (Internet Explorer, Navigator, Opera, Mozilla);
10) экспертные системы;
11) правовые, юридические (Гарант, Консультант-Плюс);
12) бухгалтерские системы (1С: Предприятие, Инфо-бухгалтер);
13) финансовые аналитические системы;
14) обучающие программы;
15) программы математических расчетов, моделирования и анализа (MatCad);
16) игры;
17) коммуникационные программы и т. д.

4. Базовое программное обеспечение. Операционные системы (ОС)

Ядром программного обеспечения являются операционные системы.

Операционная система – это комплекс программ, обеспечивающих:

1) управление ресурсами, заключающееся в согласованной работе всех аппаратных средств компьютера;
2) управление процессами, которое обеспечивает выполнение программ, их взаимодействие с устройствами компьютера;
3) пользовательский интерфейс, который создаёт возможность диалога пользователя с компьютером, выполнения определённых команд, обеспечивающих операции по обработке информации.

Наиболее известные из операционных систем: MS DOS, Windows, Unix. В настоящее время используется много различных типов операционных систем для ЭВМ различных видов. Однако в структуре всех операционных систем существуют общие принципы. В любой операционной системе можно выделить главную часть, которую называют ядром, в состав которой входят наиболее часто используемые модули:

1) управление системой прерываний;
2) средства по распределению ресурсов оперативной памяти и процессора.

Программы, входящие в состав ядра, при загрузке ОС помещаются в оперативную память, где они постоянно находятся и используются при работе ЭВМ. Такие программы называют резидентными. Важной частью операционных систем является командный процессор – программа, отвечающая за выполнение простейших команд, подаваемых пользователем, и его взаимодействие с ядром ОС.

К операционной системе относятся наборы утилит – сервисных программ, обслуживающих различные устройства компьютера, которые называют служебными программами.

3.2. Понятие файла, файловой структуры

Одной из важных задач, решаемых ОС, является организация хранения информации во внешней памяти. Долговременно информация хранится во внешней памяти в виде файлов. Правила по их хранению определяет используемая файловая система.

Файл – это поименовм анная область информационного пространства на устройствах внешней памяти. Файл может хранить текст программы, документы, закодированные графические, аудио-, видеоизображения и т. д. Любой файл имеет имя и расшире-

ние. Имя файла может содержать от 1 до 255 символов (латинские или русские буквы, цифры) и может состоять из нескольких слов. Нельзя использовать в имени файла 9 символов:

1) / (слэш!);
2) : (двоеточие);
3) \ (обратный слэш);
4) | (вертикальная черта);
5) * (звёздочка);
6) < (меньше);
7) > (больше);
8) ” (кавычки);
9) ? (вопросительный знак).

Расширение содержит не более трех символов. Расширение отделяется от имени файла точкой, например: stud. doc, lab. pas, и предназначено для определения типа данных в файле. Как правило, расширение является характеристикой файла, указывая программу, в которой создан этот файл, или способ организации информации в файле.

Формат – это способ организации информации в файле. Одна и та же информация может быть сохранена в различных форматах. Например, программы, созданные и сохранённые в Паскале, имеют расширение (тип) *.pas. Эти же программы, текст которых набран и сохранён в Word, имеют расширение *.doc, а в блокноте *.txt.

Файл имеет *атрибуты*: размер файла, измеряющийся в байтах, дата и время создания или последнего редактирования и иные атрибуты в зависимости от типа операционной системы.

Катало́г (папка) – это справочник, содержащий сведения о местоположении, размере, дате и времени создания или обновления файлов. Корневой каталог не имеет имени и в его обозначение включается наименование устройства. Для вновь созданного каталога даётся имя в его родительском каталоге. Такая организация позволяет создавать древовидную (иерархическую) структуру каталогов. На вершине этой структуры находится корневой каталог.

Устройства, где хранятся файлы и каталоги, именуются одной прописной латинской буквой и двоеточием. Различают следующие устройства:

A: , B: – для гибких дискет.

C: – для жёсткого диска, винчестера. Все остальные буквы латинского алфавита могут использоваться для логических дисков, которые выделяются на жёстком диске.

Указание имени диска, каталога, подкаталогов и имени файла в последовательном порядке образует текущий путь к искомому файлу. Это позволяет установить расположение файла на диске. Все файлы и каталоги (папки) объединены в одну файловую систему.

Файловая система – это система управления данными, которая обеспечивает основные операции над файлами (их открытие, копирование, перемещение, объединение, удаление, закрытие). Файловая система определяет способ организации данных на диске. Файловая система позволяет найти файл, указав к нему путь. Например:

C:\stud\pascal\lab1.pas,

где:

- lab1.pas – имя файла и расширение (тип);
- C – винчестер, устройство, на котором хранится файл;
- stud, pascal – каталоги (папки).

Путь доступа к файлу начинается с имени устройства, где хранится файл, а затем перечисляются все имена каталогов (папки), через которые можно попасть к данному файлу. В результате принято считать, что *полное имя файла* включает собственное имя файла с путём доступа к нему.

Данные о местоположении файлов хранятся в табличной структуре. Однако пользователю они представляются в иерархической структуре, которая именуется как *дерево каталогов* (папок). Под управлением операционной системы выполняются операции, которые относятся к функциям обслуживания файловой структуры:

1) создание файлов и присвоение им имён;
2) создание каталогов (папок) и присвоение им имён;
3) переименование файлов и каталогов;
4) копирование и перемещение файлов с одного устройства или каталога на другое;
5) удаление файлов и каталогов;
6) указание пути доступа к данному файлу, каталогу (навигация);
7) управление атрибутами файлов.

3.3. Операционная система MS Windows

Операционная система (ОС) Windows ориентирована на организацию удобной среды работы пользователя на ПК в графическом интерфейсе. До её появления операционные системы работали в основном в командном интерфейсе, что требовало от пользователя знания языка команд по управлению ЭВМ. ОС Windows позволила изменить облик ПК и правила работы с ним.

Контрольные вопросы

1. Классификация программного обеспечения.
2. Системное программное обеспечение (СПО). Структура и назначение.
3. Назначение инструментального программного обеспечения.
4. Программное обеспечение, к которому относятся антивирусные программы.
5. Программное обеспечение, к которому относятся графические редакторы.
6. Понятие операционной системы. Назначение.
7. Компоненты операционной системы.
8. Основные этапы загрузки операционной системы.
9. Прикладное программное обеспечение (ППО). Назначение.
10. Понятие файла. Правила образования имен файлов.
11. Понятие каталога. Организация хранения каталогов и файлов на диске.
12. Представление файловой системы компьютера в графическом интерфейсе Windows.

Глава 4. ПРИМЕНЕНИЕ ВСТРОЕННЫХ ФУНКЦИЙ ЭЛЕКТРОННОЙ ТАБЛИЦЫ MICROSOFT EXCEL В ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

Цель – ознакомить студентов с назначением и возможностями табличного процессора Microsoft Excel.

Учебные вопросы

1. Функциональные возможности Microsoft Excel.
2. Создание и редактирование электронных таблиц.
3. Использование мастера функций.
4. Вычисления с использованием встроенных функций.

Изучив данную тему, студент должен:

знать:

– типы данных, обрабатываемых в электронной таблице;
– способы адресации ячеек таблицы в формулах;
– назначение встроенных функций MS Excel;

уметь:

– создавать документ в MS Excel;
– выполнять изменение формата ячеек таблицы;
– применять встроенные функции для вычислений.

Excel – табличный процессор, входящий в комплект Microsoft Office и предназначенный для обработки информации, представленной в табличной форме. В отличие от текстового процессора Word, предназначенного для оформления текстовых документов, Excel разработан для выполнения вычислений с табличными данными.

Excel имеет большое количество встроенных функций для математических, статистических, финансовых и других вычислений. С другой стороны, Excel – это среда, ориентированная на непрограммирующего пользователя, что делает его популярным среди экономистов, бухгалтеров, юристов, лингвистов и других специалистов, обрабатывающих табличные данные.

Excel позволяет автоматизировать расчеты, связанные с большим количеством исходных данных, а также производить с большой ско-

ростью вычисления при помощи различных встроенных в программу функций, менять значения исходных параметров и отслеживать изменения расчетных результатов, наглядно представлять данные в виде графиков и диаграмм. Основные документы – рабочие таблицы для записи, вычислений и анализа данных, диаграммы для графического представления данных рабочей таблицы. Широко используются графические фигуры и рисунки, улучшающие внешний вид рабочей таблицы. Документ Excel называется рабочей книгой, состоящей из набора рабочих листов. Книга хранится в виде файла с расширением .xls. Одна книга может содержать до 256 рабочих листов.

4.1. Ячейка – основной элемент таблицы

Основным элементом таблицы является ячейка. Ячейки образуются на пересечении строк и столбцов. Каждая ячейка имеет свой уникальный адрес, состоящий из обозначений столбца и строки, на пересечении которых эта ячейка находится. Например, F4 – ячейка, расположенная на пересечении столбца F и строки 4. Столбцы помечаются слева направо, начиная от A до Z. После Z отметка продолжается от AA до AZ, затем от BA до BZ, и т. д., в сумме 256. Строки нумеруются вниз от 1 до 65536. Ячейка может вмещать до 255 символов.

Данные, вводимые в ячейки, имеют тип:

1) текстовый;
2) денежный;
3) финансовый;
4) числовой;
5) общий;
6) процентный.

Одна из ячеек таблицы всегда является активной. Чтобы сделать ячейку активной, необходимо в этой ячейке щелкнуть мышью, в результате она выделится по контуру рамкой.

Для выделения нескольких смежных ячеек необходимо установить указатель мыши в одну из ячеек, нажать левую кнопку мыши и, не отпуская ее, растянуть выделение на всю область. Для выделения нескольких несмежных групп ячеек следует выделить одну группу, нажать клавишу «Ctrl» и, не отпуская ее, выделить другие ячейки.

Чтобы выделить целый столбец или строку таблицы, необходимо щелкнуть мышью на имени соответствующей строки или столбца. Для выделения нескольких столбцов или строк следует щелкнуть на имени первого столбца или строки и растянуть выделение на всю область.

В активной ячейке в нижнем правом углу выделяется в виде небольшого черного квадрата маркер заполнения, который имеет определенное назначение. Установив курсор мыши (в виде черного перекрестья) в маркер, можно всю информацию ячейки копировать в соседние ячейки, не отпуская кнопки мыши.

Если выделить несколько ячеек, то маркер будет виден на весь выделенный фрагмент, который можно аналогично одной ячейке копировать. Такой способ применяется для копирования формул из активной ячейки в соседние.

Установив курсор мыши (в виде стрелки) на контур активной ячейки можно всю информацию ячейки перенести в любую другую ячейку, не отпуская кнопки мыши. Аналогично можно копировать данные из активной ячейки, но с добавлением клавиши «Ctrl». Данные, вводимые в ячейку, могут быть текстом, формулой или ссылкой. Excel может определять тип данных автоматически. Если введен текст, он обычно выравнивается по левому краю ячейки.

В главном меню команда: **«Формат»/ «Ячейки…»** позволяет установить в одной ячейке или выделенной группе ячеек типы вводимых данных, шрифты, границы, различные виды выравнивания и отображения.

Эту команду можно вызвать правой кнопкой мыши, щёлкнув по выделенным ячейкам, выбрав в контекстном меню команду: **«Формат ячеек…»**

4.2. Вычисления в Excel. Формулы и функции

Основным достоинством электронной таблицы Excel является наличие мощного аппарата формул и функций.

Формулы Excel

Вычисления в таблицах выполняются с помощью формул. Для ввода формулы в ячейку следует ввести знак «=» и соответствующее

арифметическое выражение для вычисления. После нажатия клавиши Enter в ячейке появится результат вычисления. При выделении ячейки, содержащей формулу, она появляется в строке редактирования. Формула может состоять из математических операторов, функций, значений, ссылок на ячейку. В формулах в качестве аргументов могут использоваться как числа, так и адреса ячеек. Причём по ряду причин использование в качестве аргументов адресов ячеек предпочтительнее.

Результатом выполнения формулы есть некоторое новое значение, содержащееся в ячейке, где находится формула. В формулах используются известные арифметические операторы:

- сложение (+);
- умножение (*);
- вычитание (–);
- деление (/);
- процент (%);
- возведение в степень (^),

а также операторы сравнения:

- равно (=);
- больше (>);
- меньше (<);
- не равно (<>);
- больше или равно (=>);
- меньше или равно (<=).

Порядок вычислений определяется обычными математическими законами.

Примеры формул:

- =3*(A2+B5)*C8;
- =F7*C14+B12.

В первой формуле перед скобкой записана константа, равная 3.

Константы – текстовые или числовые значения, которые вводятся в ячейку и не могут изменяться во время вычислений.

Ссылка – это адрес ячейки, используемый в формуле. Когда формула использует в качестве аргументов адреса ячеек, говорят, что она ссылается на эти ячейки

Ссылка на ячейку или группу ячеек – способ, которым указывается конкретная ячейка или несколько ячеек. Ссылка на отдельную ячейку – её координаты. Значение пустой ячейки равно нулю. Ссылки на ячейки бывают двух типов – абсолютные и относительные.

Часто требуется, чтобы какая-то часть формулы не изменялась при копировании. Например, когда несколько формул должны ссылаться на одну и ту же ячейку. В этом случае необходимо выполнить действия по защите формулы от изменения при копировании. Защита сводится к защите тех адресов ячеек, используемых в формуле, которые должны оставаться постоянными.

Для защиты адреса ячейки необходимо установить знаки «$» перед обозначением столбца и перед обозначением строки, образующими данную ячейку.

A1	Относительная ссылка
A1	Абсолютная ссылка

Комбинация предыдущих типов (например: F$7 или $F7) – смешанные ссылки. Смешанные ссылки представляют собой комбинацию абсолютных и относительных ссылок. Например, $A1 – смешанная ссылка. В смешанной ссылке знаком $ защищается строка или столбец.

Для обращения к группе ячеек используются специальные символы:

1) : (двоеточие) – формирует обращение к блоку ячеек. Через двоеточие указывается левая верхняя и правая нижняя ячейки блока. Например, C4:D6 – обращение к ячейкам C4, C5, C6, D4, D5, D6;
2) ; (точка с запятой) – обозначает объединение ячеек. Например, D2: D4; D6: D8; – обращение к ячейкам D2, D3, D4, D6, D7, D8.

Если значения в ячейках, на которые есть ссылки в формулах, меняются, то результат изменится автоматически.

Имя – это легко запоминающийся идентификатор, который можно использовать для ссылки на ячейку, группу ячеек, значение или формулу. Создать имя для ячейки можно в поле имени или через меню: **Вставка | Имя | Присвоить...** Использование имен обеспечивает следующие преимущества:

1) формулы, использующие имена, легче воспринимаются и запоминаются, чем формулы, использующие ссылки на ячейки;
2) при изменении структуры рабочего листа достаточно обновить ссылки лишь в одном месте (в определении имен), и все формулы, использующие эти имена, будут использовать корректные ссылки. После того как имя определено, оно может использоваться в любом месте рабочей книги. Доступ ко всем именам из любого рабочего листа можно получить с помощью окна имени в левой части строки формул.

Функции Excel

Функции в Excel используются для выполнения стандартных вычислений в рабочих книгах. Значения, которые используются для вычисления функций, называются аргументами. В качестве аргументов можно использовать числа, текст, логические значения, массивы, ссылки. Аргументы могут быть как константами, так и формулами. В свою очередь эти формулы могут содержать другие функции. Функции, являющиеся аргументом другой функции, называются вложенными. В формулах Excel можно использовать до семи уровней вложенности функций.

Функция может быть выбрана двумя способами.

1. *В меню «Вставка»* выбрать команду «f_x Функция...». В окне «Мастер функций» из списка выбрать нужную функцию рабочей таблицы и затем нажать кнопку «OK».

2. *Мастер функций*. Работа с Мастером функций начинается с нажатия кнопки «fx» в строке формул.

Для выбора функций используется окно с двумя полями. В левом поле задается категория функции, в правом – сама функция. Всего имеется 14 категорий. При выделении функции в правом списке в нижней части окна появляется краткое описание ее назначения.

Количество стандартных (встроенных) функций в Excel около 400. Использование стандартных функций позволяет значительно упростить процесс вычислений. Для ввода функции необходимо ввести вначале знак равенства, затем название функции и диапазон ячеек с нужными данными. В табл. 11 приведены часто используемые в Excel функции.

Таблица 11

№	Функция	Назначение	Синтаксис	Пример
1	**МАКС()**	Максимум из указанного диапазона	МАКС (число 1; число 2; ...)	=МАКС (A3:A7)
2	**МИН()**	Минимум из указанного диапазона	МИН (число1 ; число2; ...)	=МИН (A3:A7)
3	**СРЗНАЧ()**	Вычисление среднего арифметического значения для ячеек из указанного диапазона	СРЗНАЧ (число 1; число 2; ...)	=СРЗНАЧ (B7:B10)
4	**СУММ()**	Суммирование значений из ячеек указанного диапазона	СУММ (число 1; число 2; ...)	=СУММ (A1:E1)
5	**СЧЁТ()**	Подсчёт количества числовых значений в указанном диапазоне ячеек	СЧЁТ (значение 1; значение 2;)	=СЧЁТ (E2:E12)
6	**ОСТАТ()**	Возвращает остаток от деления аргумента (число) на делитель	ОСТАТ (число; делитель)	=ОСТАТ (A1; 3)
7	**СУММЕСЛИ()**	Суммирует ячейки, заданные критерием	СУММЕСЛИ (диапазон; критерий; диапазон_ суммирования)	=СУММЕСЛИ (A1:A9; >5; B1:B9)
8	**СЧЁТЕСЛИ()**	Подсчет количества ячеек в указанном диапазоне, удовлетворяющих заданному условию	СЧЁТЕСЛИ (диапазон; критерий)	=СЧЁТЕСЛИ (A1:A10; "ВЫПОЛНЕНО")
9	**ЕСЛИ()**	Возвращает одно значение, если заданное условие при вычислении дает значение ИСТИНА, и другое значение, если ЛОЖЬ	ЕСЛИ (лог_выражение; значение_если_истина; значение_если_ложь)	=ЕСЛИ (x>0; 5; 7)
10	**ПРОИЗВЕД()**	Умножение	ПРОИЗВЕД (число1, число2,..)	=Произвед (B2:B6)
11	**КОРЕНЬ()**	Квадратный корень	КОРЕНЬ (число)	=Корень (F5)

На панели инструментов символ ∑ – автосуммирование, позволяющее складывать числовые значения одним щелчком мыши, даёт возможность сделать это одновременно и со строками, и со столбцами. По этой команде можно подводить общие итоги даже в таблицах с промежуточными итогами.

В Excel широко представлены математические функции. С помощью статистических функций возможно проводить статистическое моделирование. Кроме того, возможно использовать элементы регрессионного анализа.

Логические функции помогают создавать сложные формулы, которые в зависимости от выполнения тех или иных условий будут совершать различные виды обработки данных.

Логические функции находятся в Excel в меню: **Вставка / Функция / Категория / Логические.**

Функция «ЕСЛИ»

Эта функция позволяет задать некоторое условие и определить значение, если условие истинно или ложно. Вид функции в ячейке может быть, например, следующим: =Если (A2>50; A5+C6; A5+C7)

Это означает, что если в ячейку A2 введено число, большее 50, в данную ячейку (ячейку, в которую введена данная функция) будет помещена сумма данных из ячеек A5 и C6, иначе – сумма из ячеек A5 и C7.

Другой пример логической функции в ячейке:

=Если(и (A2>50; B2>20; C2>70); A2+B2+C2; A2-B2-C2).

Это означает, что если будут выполнены все условия во внутренних скобках, в данную ячейку будет помещена сумма данных из ячеек A2, B2, C2, иначе – разность этих ячеек.

Некоторые изменения в логической функции предыдущего примера дают другую формулу:

=Если(или (A2>50; B2>20; C2>70); A2+B2+C2; A2-B2-C2).

Это означает, что если будет выполнено хотя бы одно из условий во внутренних скобках, в данную ячейку будет помещена сумма данных из ячеек A2, B2, C2, иначе – разность этих ячеек.

В табл. 12 представлены типовые ошибки в формулах.

Таблица 12

№	Ошибка	Пояснения
1	#ДЕЛ/0!	Функция содержит деление на ноль
2	#ИМЯ?	В функции используется неопределенное либо некорректное имя ячейки (Excel не может найти путь к ячейке)
3	#ЗНАЧ!	Функция содержит недопустимый тип аргумента, например, пытается произвести арифметические действия не над числами, а над текстом
4	#ССЫЛ!	Функция ссылается на несуществующую ячейку или интервал ячеек (возможно, они были удалены)
5	#ЧИСЛО!	Функция содержит некорректную математическую операцию, нарушение математических правил. Например, корень из отрицательного числа

Для облегчения поиска ошибок в формулах целесообразно перейти в режим показа формул в ячейках через **Сервис/Параметры/Вид** и установить флажок «Формулы». При этом ширина ячеек таблицы будет автоматически увеличена и вместо результатов и сообщений об ошибках будут показаны формулы.

Контрольные вопросы

1. Основной элемент в электронной таблице.
2. Виды адресации ячеек в электронной таблице и их назначение.
3. Понятие абсолютной адресации ячейки. Привести пример.
4. Типы данных, которые можно использовать в электронной таблице.
5. Правила ввода формул в ячейку.
6. Логические функции в Excel. Их назначение
7. Виды стандартных функций в Excel.

Глава 5. АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Цель – ознакомить студентов с основами алгоритмизации и программирования на языке Паскаль.

Учебные вопросы

1. Алгоритм и его свойства. Способы записи алгоритмов.
2. Основные типы алгоритмов.
3. Блок-схемы типовых алгоритмов.
4. Конструкция алгоритмического языка Паскаль.
5. Структура программы на алгоритмическом языке Паскаль.
6. Основные операторы языка Паскаль.

Изучив данную тему, студент должен:

знать:

– свойства алгоритма;
– блоки для построения схем;
– основные типы алгоритмов;
– структуру программы на языке Паскаль;
– правила записи арифметических выражений на языке Паскаль;
– основные операторы языка Паскаль;

уметь:

– строить алгоритмы по условию задачи;
– применять основные алгоритмические структуры для решения задач.

5.1. Понятие алгоритма

Алгоритм – одно из фундаментальных понятий информатики, изучаемых самостоятельной дисциплиной *теорией алгоритмов*, близкой другой дисциплине *математической логике*. С другой стороны, дисциплину «Теория алгоритмов» можно рассматривать как промежуточную между двумя дисциплинами: математикой и информатикой, связанной с разделом программирования.

Алгоритмизация относится к общим методам информатики, имеет большое значение при решении сложных задач. Прежде чем написать программу решения задачи на ЭВМ, необходимо просмотреть последовательность действий, которые должны быть выполнены для правильного решения рассматриваемой задачи.

Алгоритм – это последовательность арифметических, логических и прочих операций, необходимых для выполнения на ЭВМ.

Для получения правильного результата алгоритм должен быть составлен так, чтобы при его исполнении все команды трактовались однозначно. Поэтому появились обязательные требования, которые должны учитываться при составлении алгоритмов. Требования формулируются в виде свойств.

Алгоритм должен быть всегда результативным, иметь свойство повторяемости и рассчитан на конкретного исполнителя. В технике таким исполнителем является ЭВМ. Для обеспечения возможности реализации на ЭВМ алгоритм должен быть описан на языке, понятном ЭВМ, то есть на машинном языке. Однако прежде чем представить алгоритм на языке, понятном для ЭВМ (машинном языке), необходимо написать программу с помощью алгоритмического языка программирования.

Алгоритм может быть представлен различными способами, в частности:

1) словесно (вербальное описание);
2) таблично;
3) в виде блок-схемы;
4) на алгоритмическом языке.

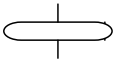
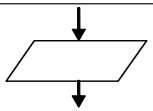
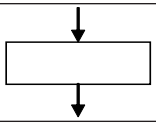
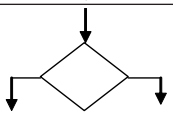
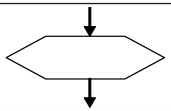
Достаточно распространенным способом представления алгоритма является его запись на алгоритмическом языке, представляющем в общем случае систему обозначений и правил для единообразной и точной записи алгоритмов и их исполнения. Этот способ представления алгоритма предусматривает запись его в виде программы.

Программа – это запись алгоритма на языке программирования, приводящая к конечному результату за конечное число шагов.

Предпочтительнее до записи на алгоритмическом языке представить алгоритм в виде блок-схемы. Для построения алгоритма

в виде блок-схемы необходимо знать назначение каждого из блоков. В табл. 13 приводятся типы блоков и их назначение.

Таблица 13

№	Блок	Назначение блока	Комментарий (блоку соответствует оператор)
1		Начало или конец блок-схемы	–
2		Ввод или вывод данных	ввода / вывода
3		Процесс (в частности, вычислительный)	присваивания
4		Решение	условия
6		Модификатор цикла	цикла

5.2. Основные типы алгоритмов

Алгоритмизация выступает как набор определенных практических приёмов, особых специфических навыков рационального мышления в рамках заданных языковых средств. Алгоритмизация вычислений предполагает решение задачи в виде последовательности действий, т. е. решение, представленное в виде блок-схемы. Можно выделить типичные алгоритмы:

1) линейные;
2) разветвляющиеся;
3) циклические.

Линейные алгоритмы

Линейный алгоритм является наиболее простым. В нём предполагается последовательное выполнение операций. В этом алгоритме не предусмотрены проверки условий или повторений.

Пример. Вычислить функцию $z = (x - y)/x + y^2$.

Составить блок-схему вычисления функции по линейному алгоритму. Значения переменных ***x, y*** могут быть любые, кроме нуля, вводятся с клавиатуры.

Решение. Линейный алгоритм вычисления функции задан в виде блок-схемы на рис. 8. При выполнении линейного алгоритма значения переменных вводятся с клавиатуры, подставляются в заданную функцию, вычисляется результат, а затем выводится результат.

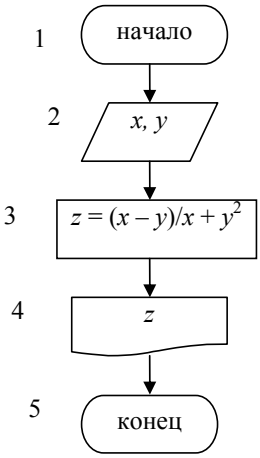


Рис. 8. Линейный алгоритм

Назначение блоков в схеме на рис. 8:

– блок 1 в схеме служит в качестве логического начала;
– блок 2 соответствует вводу данных;
– блок 3 представляет арифметическое действие. ;
– блок 4 выводит результат. ;
– блок 5 в схеме служит в качестве логического завершения схемы.

Алгоритмы ветвлений

Разветвляющийся алгоритм предполагает проверку условий для выбора решения. Соответственно, в алгоритме появятся две ветви для каждого условия.

В примере рассматривается разветвляющийся алгоритм, где в зависимости от условия выбирается один из возможных вариантов решений. Алгоритм представляется в виде блок-схемы.

Пример. При выполнении условия $x > 0$ вычисляется функция: $z = \ln x + y$, иначе, а именно, когда $x = 0$ или $x < 0$, вычисляется функция: $z = x + y^2$.

Составить блок-схему вычисления функции по алгоритму ветвления. Значения переменных ***x***, ***y*** могут быть любые, вводятся с клавиатуры.

Решение. На рис. 9 представлен разветвляющийся алгоритм, где в зависимости от условия выполнится одна из веток. В блок-схеме появился новый блок 3, который проверяет условие задачи. Остальные блоки знакомы из линейного алгоритма.

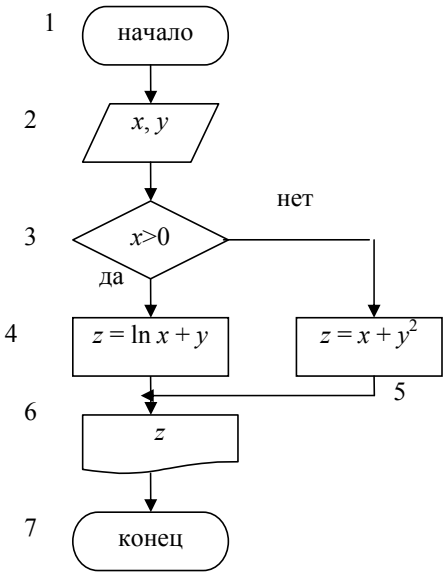


Рис. 9. Алгоритм ветвления

Пример. Найти максимальное значение из трёх различных целых чисел, введенных с клавиатуры. Составить блок-схему решения задачи.

Решение. Данный алгоритм предполагает проверку условия. Для этого выбирается любая из трёх переменных и сравнивается с другими двумя. Если она больше, то поиск максимального числа окончен. Если условие не выполняется, то сравниваются две остав-

шиеся переменные. Одна из них будет максимальной. Блок-схема к этой задаче представлена на рис. 10.

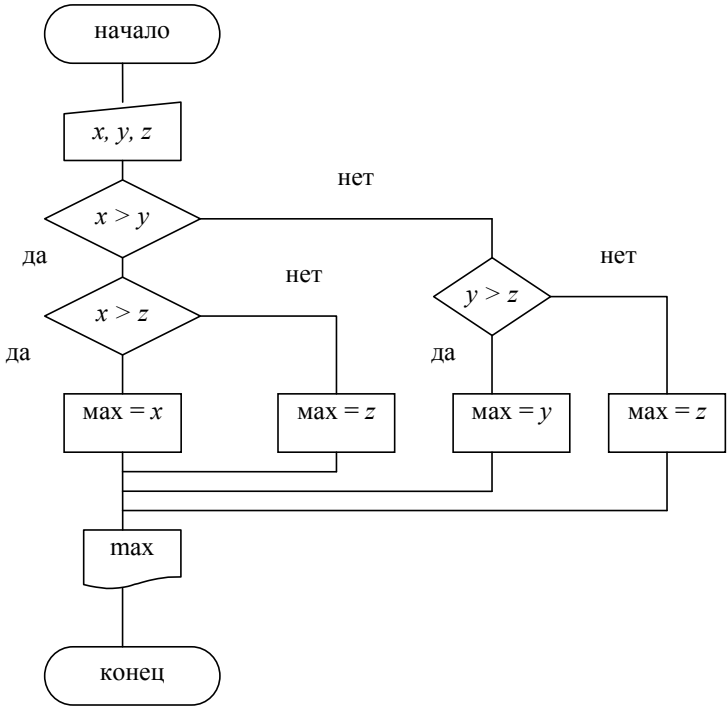


Рис. 10. Блок-схема поиска максимума

Циклические алгоритмы

Циклический алгоритм предусматривает повторение одной операции или нескольких операций в зависимости от условия задачи.

Выделяют два типа циклических алгоритмов:

1) с заданным количеством циклов или со счётчиком циклов;
2) с неизвестным количеством циклов.

Пример. В цикле вычислить значение функции $z = x * y$ при условии, что одна из переменных «*x*» меняется в каждом цикле на единицу, а другая переменная «*y*» не меняется и может быть любым целым числом. В результате выполнения цикла при начальном значении переменной $x = 1$ можно получить таблицу умножения.

Количество циклов может быть любым. Составить блок-схему решения задачи.

Решение. В примере количество циклов задаётся. Соответственно, выбирается алгоритм циклов первого типа. Алгоритм этой задачи приводится на рис. 11.

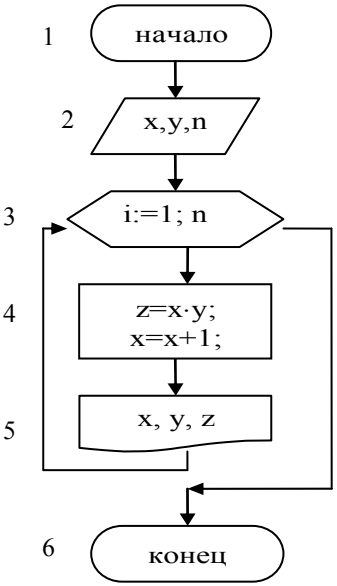


Рис. 11. Циклический алгоритм со счётчиком циклов

Во втором блоке вводятся количество циклов ***n*** и любые целые числа ***x***, *y*.

В блок-схеме появился новый блок 3, в котором переменная ***i*** указывает количество циклов, после каждого цикла увеличиваясь на единицу, пока счётчик не будет равен $i = n$. При $i = n$ будет выполнен последний цикл.

В третьем блоке указывается диапазон изменения счётчика цикла (от $i = 1$ до $i = n$).

В четвёртом блоке изменяются значения переменных: *z*, *x*.

В пятом блоке выводится результат. Четвёртый и пятый блоки повторяются в каждом цикле.

Этот тип циклических алгоритмов предпочтителен, если дано количеством циклов.

Если количество циклов неизвестно, то блок-схемы циклических алгоритмов могут быть представлены в виде рис. 12, 13.

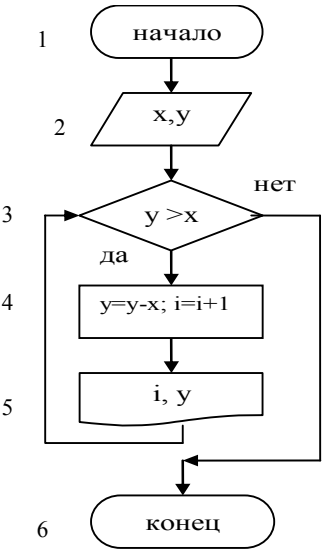


Рис. 12. Блок-схема циклического алгоритма с предусловием

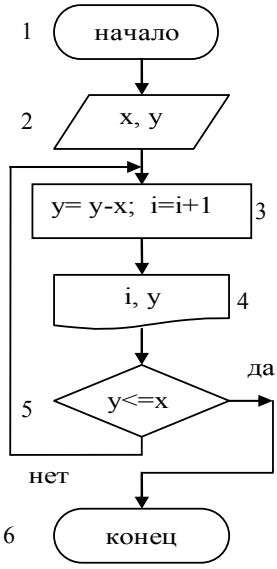


Рис. 13. Алгоритм цикла с постусловием

Пример. Вычислить $y = y - x$ пока $y > x$, если $y = 30$, $x = 4$. Подсчитать количество выполненных циклов, конечное значение переменной y. В цикле вывести значение переменной y, количество выполненных циклов. Составить блок-схему решения задачи.

Решение. В примере количество циклов неизвестно. Соответственно, выбирается алгоритм циклов второго типа. Алгоритм этой задачи приводится на рис. 12.

Условие проверяется на входе в цикл. В теле цикла выполняется два блока:

1) $y = y - x$; $i = i + 1$;

2) вывод значений переменных i, y.

Цикл выполняется до тех пор, пока выполняется условие $y > x$. При условии равенства этих переменных $y = x$ или $y < x$ цикл заканчивается.

Алгоритм, представленный на рис. 12, называется *циклический алгоритм с предусловием*, так как условие проверяется в начале цикла или на входе в цикл. При этом условии цикл выполняется.

Во втором блоке вводятся $y = 30$, $x = 4$.

В третьем блоке проверяется условие $\boldsymbol{y > x}$ на входе в цикл. Если условие выполняется, то переход к блоку 4, иначе на блок 6.

В четвёртом блоке вычисляется значение переменной **y**, подсчитывается количество выполненных циклов $i = i + 1$.

В пятом блоке выводится результат:

- значение переменной ***y***;
- количество выполненных циклов *i*.

Пример. Составить блок-схему примера (рис. 12), проверяя условие выхода из цикла. В этом примере условие задачи не меняется, и результат выведется тот же, но блок-схема будет другой.

Решение. В этом случае проверяется условие на выход из цикла: $y <= x$. При этом условии цикл не выполняется. Условие в блок-схеме следует перенести в конец цикла, после вывода на печать. Цикл выполняется до тех пор, пока выполняется условие $y > x$.

Алгоритм, если условие перенести в конец цикла, называется *алгоритмом цикла с постусловием*. Алгоритм этой задачи приводится на рис. 13.

Во втором блоке вводятся $y = 30$, $x = 4$.

В третьем блоке вычисляется значение переменной **y**, подсчитывается количество выполненных циклов $i = i + 1$.

В четвёртом блоке выводится результат:

- значение переменной ***y***;
- количество выполненных циклов *i*.

В пятом блоке проверяется условие $\boldsymbol{y <= x}$ на выход из цикла. Если условие выполняется, то переход к блоку 6, иначе на блок 3 и цикл повторяется.

5.3. Основные конструкции языка Turbo Pascal

Алгоритмический язык Паскаль был разработан в 1971 году швейцарским математиком Н. Виртом. Язык получил название в честь французского математика и философа Блеза Паскаля (1623–1662). С момента создания и до сегодняшних дней язык играет особую роль в практическом программировании. Автор реализовал в языке принцип структурного программирования.

Паскаль стал первым языком, с которым знакомится большинство будущих программистов. Существует много версий языка Паскаль.

В 80-е годы на основе Паскаля был разработан Turbo Pascal. Turbo – это торговая марка разработчика фирмы Borland.

Turbo Pascal – это система программирования, которая представляет собой единство двух самостоятельных составляющих:

1) компилятора языка программирования Паскаль;
2) инструментальной программной оболочки, способствующей повышению эффективности создания программ.

Таким образом, компилятором реализуется язык программирования Turbo-Pascal, а разнообразные сервисные услуги обеспечиваются инструментальной программной оболочкой.

Под конструкцией любого алгоритмического языка высокого уровня понимают все его составляющие: алфавит, данные, стандартные функции и процедуры, операторы.

Алфавит:

1) латинский шрифт;
2) русский шрифт;
3) цифры (0÷9);
4) символы:

а) знаки арифметических операций (+ – * /), нет возведения в степень;

б) знаки логических отношений (<, >, <= вместо ≤, >= вместо ≥, <> вместо ≠);

в) разделители (, . ; :);

г) прочие символы.

Данные и их типы

Данные могут быть разделены:

1) на константы – *const*;

2) переменные – *var*.

Константам и переменным даётся имя – идентификатор. С другой стороны, в зависимости от вида данных (число, текст, символ и т. д.) в Паскале имеет значение тип данных.

Понятие типа – одно из фундаментальных понятий Turbo Pascal. Паскаль – это типизированный язык, который характеризуется разветвленной структурой типов данных, построен на основе строгого соблюдения типов. Язык Turbo Pascal предоставляет большие возможности создания сложных типов, однако все они строятся на основе элементарных (стандартных) типов.

Для начала можно ограничиться стандартными типами данных (4 типа). Соответственно, можно выделить следующие данные: числовые, символьные, логические. Числовые данные подразделяются на целые и вещественные:

1) INTEGER – целочисленные данные, во внутреннем представлении занимают два байта; диапазон возможных значений от -32768 до +32767;

2) REAL – вещественные данные, занимают 6 байт; диапазон возможных значений модуля от 2.9E-39 до 1.7E+38; точность представления данных – 11...15 значащих цифр. Вещественные данные в Паскале могут записываться в двух форматах:

а) с фиксированной точкой (число 34,5 в Паскале запишется 34.5);

б) с плавающей запятой (число 34,5 в Паскале можно записать 0.345E2 или 3.45E1, где символ **E** называется десятичной экспонентой, означает число 10, а после записывается степень этого числа);

3) CHAR – символьные данные, занимает 1 байт;

4) BOOLEAN – логический тип, занимает 1 байт и имеет два значения: FALSE (ложь) и TRUE (истина).

Стандартные функции

Стандартные функции подразделяются на числовые, символьные, строковые и т. д. Числовые стандартные функции представлены в табл. 14.

Таблица 14

№ п/п	Запись на Паскале	Запись в математике	Тип результата	Примечание
1	sin (*x*)	sin *x*	вещественный	*x* – угол в радианах
2	cos (*x*)	cos *x*	вещественный	*x* – угол в радианах
3	arctan (*x*)	arctg *x*	вещественный	*x* – в радианах
4	exp (*x*)	e^x	вещественный	e = 2,7182... – основание натурального логарифма
5	ln (*x*)	ln *x*	вещественный	
6	sqr (*x*)	x^2	зависит от типа *x*	Квадрат числа *x*
7	sqrt (*x*)		вещественный	Корень квадратный
8	abs (*x*)	$\|x\|$	вещественный	Модуль числа *x*
9	trunc (*x*)	$\sqrt{x}$	целый	Целая часть (*x*)
10	int (*x*)		вещественный	Целая часть (*x*)
11	frac (*x*)		вещественный	Дробная часть (*x*)
12	round (*x*)		целый	Округление (*x*)
13	odd (*x*)		целый	Если *x* – нечётное, то функция true
14	pi		вещественный	π = 3,1415...

Примечания

1. После имени стандартной функции в скобках записывается аргумент, который может быть:

а) константой: например cos (1.3);

б) переменной: например cos (*x*);

в) арифметическим выражением: например cos (*x*+*y*);

г) стандартной функцией: например cos (ln (*x*)).

2. Аргумент тригонометрической функции должен быть задан в радианах. Если он задан в градусах, то его следует перевести в радианы по формуле

$$x = \frac{x \cdot pi}{180}.$$

3. Логарифмические функции:

$$\log_a x = \frac{\ln x}{\ln a}.$$

4. Обратные тригонометрические функции:

$$\arcsin x = \operatorname{arctg}\left(\frac{x}{\sqrt{1-x^2}}\right); \quad \arccos x = \operatorname{arctg}\left(\frac{\sqrt{1-x^2}}{x}\right);$$

$$\operatorname{arcctg}(x) = \operatorname{arctg}\left(\frac{1}{x}\right).$$

5. Гиперболические функции:

$$\text{ch } x = \frac{e^x + e^{-x}}{2};\ \ \text{sh } x = \frac{e^x + e^{-x}}{2};\ \ \text{th } x = \frac{\text{sh}x}{\text{ch}x};\ \ \text{cth } x = \frac{\text{ch}x}{\text{sh}x}.$$

6. Возведение в степень:

$$x^a = e^{a\ln x};\ \ x^a = \exp(a * \ln(x))$$

7. Тригонометрические функции:

- $\text{tg } x = \sin x / \cos x$;
- $\text{ctg } x = \cos x / \sin x$.

Арифметические, логические, символьные выражения

Арифметические выражения

Пример арифметического выражения:

$$\frac{\ln^2 X - \cos A^2 + \sqrt{Y - X^2}}{\text{ch}x}.$$

В Турбо Паскале есть все 4 арифметические операции над числовыми переменными:

- «+» сложение;
- «–» вычитание;
- «*» умножение;
- «/» деление вещественное.

Для данных типа INTEGER в Турбо Паскале есть еще операции деления:

- MOD – получение остатка от целочисленного деления;
- DIV – частное от целочисленного деления.

Пример. Найти частное A/Z. На Паскале частное A/Z имеет вид: A **div** Z.

Пример. Найти остаток от деления A/Z. На Паскале остаток от деления A/Z имеет вид:

A **mod** Z.

F:=17 DIV 5; деление нацело, ответ: F:=3;

R:=17 MOD 5; остаток от деления нацело, ответ: R:= 2.

Логические выражения

Примеры логических выражений:

(A>0) and (B>0) означает (A и B больше нуля);

(A>0) or (B>0) означает (A или B больше нуля).

В Турбо Паскале определены следующие логические операции из алгебры логики:

1) not (NOT) – логическое НЕ (логическое отрицание);
2) and (AND) – логическое И (конъюнкция, или логическое умножение);
3) or (OR) – логическое ИЛИ (дизъюнкция, или логическое сложение);
4) xor (XOR) – исключающее ИЛИ;
5) eqv (EQV) – эквивалентность;
6) IMP – импликация (если..., то...).

5.4. Структура программы на языке Паскаль

Структура программы на языке Turbo Pascal представлена в табл. 15.

Таблица 15

№ п/п	Структура программы на языке Turbo Pascal	Комментарий
1	PROGRAM Pr	Заголовок не обязателен
2	**Раздел описаний**	Не исполняемая часть программы
3	Begin	**Начало** раздела операторов
4	**Раздел операторов**	Исполняемая часть программы
5	END	**Конец** раздела операторов

Рассматривая структуру программы, выделяют два раздела в программе.

1. Раздел описаний

В разделе описаний задаётся описание констант ключевым словом **`const`**, переменные в этом разделе задаются ключевым словом **`var`**, описание нового типа переменных задаётся ключевым словом **`type`**.

2. Раздел операторов

Этот раздел является **исполняемой частью программы**. Чтобы отделить раздел описаний от раздела операторов между ними вставляется слово **`begin`,** которое означает **начало** исполняемой части программы. Раздел операторов заканчивается словом **`end.`,** обязательно в конце должна быть точка.

Пара (**begin...end.**) называется операторными скобками.

Такая структура обязательна для любой программы, что является следствием жесткого требования языка: любой нестандартный для языка Турбо Паскаль идентификатор, используемый в исполняемых операторах, должен быть предварительно описан в разделе описаний.

Описать идентификатор – значит указать тип связанного с ним объекта программы (константы или переменной).

5.5. Основные операторы Паскаля

Оператор присваивания

Пример. Представлен оператор присваивания:

```
R: =cos(x)+ln(y).
```

Оператор присваивания выполняется в два этапа:

1) первый этап – выполнение правой части, т. е. в примере вычисляется арифметическое выражение;
2) второй этап – присвоение результата левой части, т. е. в примере переменной R присваивается число, полученное при вычислении арифметического выражения.

Примечание. Недопустима запись оператора присваивания в виде `cos(x)+ln(y):=R.`

Операторы ввода

В Паскале нет специальных операторов ввода-вывода. Для обмена информацией в программах Паскаля используются специальные встроенные процедуры, которые не нуждаются в предварительном описании. Таким образом, все операторы ввода-вывода являются операторами обращения к встроенным процедурам ввода или вывода данных.

По операторам `READ`, `READLN` вызывается встроенная процедура ввода данных и программа останавливается в ожидании ввода.

Пример: `readln (x,y);`

Следует набрать на клавиатуре два числа через пробел и нажать клавишу «Ввод».

Операторы вывода

Основное назначение этих операторов – вывод результатов выполнения программы. Оператор вывода `WRITE` выводит строку на экран и оставляет курсор в конце выведенной строки. Если в программе несколько операторов `WRITE`, то вывод осуществляется в одну строку.

Оператор вывода `WRITELN` выводит в отдельную строку, после вывода результата осуществляет перевод строки и устанавливает курсор в начало следующей строки экрана. Пример записи оператора вывода переменных X,Y:

```
writeln (x,y);
```

Если в программе необходимо вывести текст на экран, следует этот текст заключить в апострофы. В частности, подсказка на экран для ввода данных записывается оператором:

```
writeln ('ввести X,Y,Z');
```

Пример записи оператора вывода переменной в формате с фиксированной точкой: writeln ('z=', z: 7: 3), где 7 – количество позиций под число **z**, 3 – количество позиций под дробную часть числа.

Комментарий

Комментарий в Турбо Паскале – это произвольная последовательность любых символов, обрамленная фигурными скобками. Комментарий разрешается вставлять в любое место программы, где по смыслу должен стоять пробел. В качестве ограничителей комментария допускается использование фигурных скобок «{» и «}», а также пары символов «(*» – слева от комментария и «*)» – справа от него:

{Это – комментарий}. (*Это тоже комментарий*).

Пример. Нужно написать программу линейного алгоритма (рис. 8), вычислить и вывести на экран значение функции: $z = (x - y)/x + y^2$.

Решение. Программа линейного алгоритма имеет вид:

```
PROGRAM PR1;
VAR
z, x, y:real;
BEGIN
writeln ('ввести x, y'); {На экран выводится подсказка – текст в скобках}
```

```
read (x, y); {Ввод с клавиатуры переменных x, y}
z:= (x-y)/x +y*y;
   writeln ('z=', z: 7: 3); {Вывод переменной z }
END.
```

В программе после слова «BEGIN» в фигурных скобках даются комментарии, поясняющие действия операторов.

В примере вывод переменной записан в формате с фиксированной точкой.

Примечание. Необходимо учесть последовательность действий при выполнении арифметического выражения с учётом приоритета арифметических действий:
1) вычисляются скобки;
2) операция возведения в степень;
3) операция деления;
4) операция сложения.

5.6. Операторы передачи управления

Назначение операторов передачи управления заключается в организации ветвлений в программе: условных или безусловных. С помощью этих операторов вычислительный процесс передается в указанную оператором точку программы по указанному в операторе условию либо без условия.

Оператор безусловного перехода

Действие оператора `GOTO` состоит в передаче управления соответствующему оператору. Структура оператора:

```
GOTO метка.
```

Метка в Турбо Паскале – это произвольный идентификатор, позволяющий именовать некоторый оператор программы и таким образом ссылаться на него. Метка располагается непосредственно перед помечаемым оператором и отделяется от него двоеточием. Перед тем как появиться в программе, метка должна быть задана в разделе описания. Описание меток состоит из зарезервированного слова LABEL (метка), за которым следует список меток.

Пример

```
LABEL 1; {в разделе описания};
goto 1; {в разделе операторов} {перейти на метку 1}
```

```
1: read(x,y); {строка с меткой 1 в разделе операторов}
```

При исполнении меток необходимо руководствоваться следующими правилами:

1) метка, на которую ссылается оператор GOTO, должна быть задана в разделе описаний и обязательно должна встретиться где-нибудь в разделе операторов программы;
2) метки, описанные в процедуре (функции), локализуются в ней, поэтому передача управления извне процедуры (функции) на метку внутри неё невозможна.

Однако в программировании **не рекомендуется использование оператора GOTO**, так как это затрудняет понимание программы, делает ее запутанной и сложной в отладке. Современная технология структурного программирования основана на принципе программирования без GOTO.

Операторы условного перехода

Структура условного оператора имеет следующий вид:

```
IF <условие> THEN <оператор 1> ELSE <оператор 2>;
```

где IF, THEN, ELSE – зарезервированные слова (если, то, иначе);

<условие> – произвольное выражение логического типа;

<оператор 1>, <оператор 2> – любые операторы языка Турбо Паскаль.

Условный оператор работает по следующему алгоритму. Вначале вычисляется условное выражение <условие>. Если результат есть TRUE (истина), то выполняется <оператор 1>, а <оператор 2> пропускается; если результат есть FALSE (ложь), наоборот, <оператор 1> пропускается, а выполняется <оператор 2>.

Поскольку любой из операторов <оператор 1> и <оператор 2> может быть любого типа, в том числе и условным, а в то же время не каждый из «вложенных» условных операторов может иметь часть ELSE <оператор 2>, то возникает неоднозначность трактовки условий. Эта неоднозначность в Турбо Паскале решается следующим образом: любая встретившаяся часть ELSE соответствует ближайшей к ней «сверху» части THEN условного оператора. Условный

оператор позволяет проверить некоторое условие и в зависимости от результатов поверки выполнить то или иное действие. Таким образом, условный оператор – это средство ветвления вычислительного процесса.

Операторы условного перехода подразделяются на простые и составные, короткие и полные.

Простой, короткий IF (если)

Структура оператора имеет вид:

```
IF (условие) THEN (оператор или метка);
```

Пример. Написать программу алгоритма ветвления.

Вычислить $y := \ln x$, если $x > 0$.

Решение. Программа алгоритма ветвления имеет вид:

```
Program PR2;
var y, x: real;
begin
writeln('ввести x');
Readln (x);
IF x > 0 THEN y: = ln(x); {простой, короткий IF}
writeln ('x=', x: 7:2, 'y=', y:7:2);
end.
```

Простой, полный IF

Пример. Написать программу алгоритма ветвления. Вычислить y = ln x, если $X > 0$, иначе $y = \cos x$. В примере рассматривается не только вариант «тогда», но и «иначе».

Решение. Программа алгоритма ветвления имеет вид:

```
Program PR3;
var x, y: real;
begin
writeln('ввести X');
Readln (X);
if x>0 THEN y:= ln (x) ELSE y:=cos(x); {простой,
полный IF}
Writeln (`x = `, x:6:2, `y = `, y:7:2)
end.
```

Если $X > 0$, тогда выполняется оператор за словом THEN (тогда), иначе выполняется оператор, следующий за словом ELSE (иначе).

Составной, короткий IF

Составной оператор – это последовательность произвольных операторов программ, заключенная в операторные скобки – зарезервированные слова BEGIN...END.

Составные операторы – важный инструмент Турбо Паскаля, дающий возможность писать программы по современной технологии структурного программирования (без перехода GOTO).

Язык Турбо Паскаль не накладывает никаких ограничений на характер операторов, входящих в составной оператор.

Пример. Вычислить $y = \ln x$, $z = y - 5x$, если $x > 0$.

Оператор условия запишется в виде:

```
IF x>0 then Begin y:=Ln(x); z:=y-5*x;
Writeln ('y=', y:7:2, 'z=', z:8:3);end;
```

Составной, полный IF

Пример. Вычислить $y = \ln x$, $z = y - 5x$, если $x > 0$. Вывод производить для каждого условия.

Решение. Оператор условия запишется в виде:

```
IF x>0 then Begin
Y:=ln (x);
Writeln (`x = `, x:6:2,`y =`, y:7:2);
End
Else begin
Y:=cos (x);
Writeln (`x = `, x:6:2,`y =`, y:7:2);
End;
```

В примере оператор условия составной, так как после слов then, else операторы заключены в операторные скобки.

Структурированный (вложенный) IF

Среди условных операторов можно выделить структурированный, который предполагает проверку условий путём вложения.

Структурированный, короткий, простой IF

В структурированном операторе содержится последовательная проверка вложенных условий.

Пример. Вычислить $r = \ln (x + y + z)$, если $x > 0$, $y > 0$, $z > 0$.

Решение. Структурированный оператор условия запишется в виде:

```
IF x>0 then
IF y>0 then
IF z>0 then
R:=LN(X+Y+Z);
```

Можно этот пример записать иначе коротким, простым оператором **IF** с помощью логического выражения:

```
IF (x>0) and (y>0) and (z>0) then R:=LN(X+Y+Z);
```

Пример. Вычислить r = $x + y + z$, если выполняется хотя бы одно из условий x>0, y>0, z>0.

Решение. Оператор условия запишется в виде:

```
IF (x>0) or (y>0) or (z>0) then R:=(x+y +z);
```

Структурированный, полный, простой IF

Пример. Вычислить:

r = ln ($x + y$ + z), если $x > 0$, $y > 0$, $z > 0$;

r= ln ($x + y$) + z, если $x > 0$, $y > 0$;

r= ln (x) + y + z, если $x > 0$, иначе $r = x + y + z$.

Решение. Оператор условия для трёх строк задачи запишется в виде:

```
IF x>0 then
IF y>0 then
IF z>0 then r:=ln (x+y+z)
else r:= ln (x+y)+z
else r:= ln (x)+y+z
else r:= x+y+z;
```

Вначале проверяются три условия. Если они выполняются, то вычисляется r = ln ($x + y$ + z). Иначе выполняются первые два условия, а последнее не выполняется и z ≤ 0 (первое слово else относится к последнему условию). В этом случае вычисляется r = ln ($x + y$) + z.

Если из двух условий выполняется только первое, то вычисляется r = ln $x + y$ + z (второе слово else относится ко второму условию) и в этом случае $y \le 0$. Последнее слово else относится к первому условию и в этом случае $x \le 0$. В этом случае вычисляется $r = x + y$ + z.

Пример. Написать программу разветвляющегося алгоритма (рис. 9).

При выполнении условия $x > 0$ вычисляется функция: $z = \ln x + y$, иначе, а именно, когда $x = 0$ или $x < 0$ вычисляется функция: $z = x + y^2$.

Решение. Программа алгоритма ветвления имеет вид:

```
PROGRAM PR4;
VAR
x, y, z:real;
BEGIN
Writeln ('ввести x, y'); {На экран выводится под-
сказка-текст в скобках}
Read (x, y); {Ввод с клавиатуры переменных x, y }
if x>0 then z:=ln (x)+ y
else z=x+y*y;
Writeln ('z=', z: 7: 3); {Вывод результата}
END.
```

Пример. Написать программу разветвляющегося алгоритма (рис. 10). Найти максимальное число из трех разных целых чисел.

Решение. Программа алгоритма ветвления, поиск максимального из трех разных целых чисел имеет вид:

```
program max;
var
x, y, z: integer;
begin
writeln ('введите x, y, z');
readln (x, y, z);
if x> y then
if x> z then   max:=x
else     max:=z
else
if y>z then max:=y
else     max:=z;
writeln ('max=', max);
readln;
end.
```

Оператор условия выделен курсивом.

Контрольные вопросы

1. Дать определение понятию переменной.
2. Дать определение понятию константа.
3. Числовые типы данных. Обозначения. Привести примеры.

4. Логический, символьный, строковый типы данных. Обозначения. Привести примеры.
5. Стандартные функции Pascal. Правила записи.
6. Арифметические выражения. Правила записи.
7. Логические выражения. Правила записи.
8. Оператор присваивания. Правила записи.
9. Оператор ввода. Привести примеры.
10. Оператор ввода. Привести примеры.
11. Оператор условного перехода. Полный, короткий, составной. Привести примеры.
12. Структурированный оператор условного перехода. Привести примеры.

5.7. Программирование. Циклы

Оператор цикла с параметрами

Счетный оператор цикла FOR имеет структуру:

```
FOR i:=a TO b DO <оператор>;
```

FOR, TO, DO – зарезервированные слова (для, до, выполнить);
i – переменная цикла типа INTEGER (счётчик циклов);
a – начальное значение счётчика циклов (INTEGER);
b – конечное значение счётчика циклов (INTEGER);
<оператор> – произвольный оператор Турбо Паскаля.

Шаг изменения параметра цикла равен единице.

Алгоритм выполнения оператора цикла с параметрами при выполнении оператора FOR:

1) счётчику циклов присваивается начальное значение $i := a$;
2) проверяется условие $i > b$ (счётчик циклов больше конечного значения);
3) если условие $i > b$ выполняется, то на пункт 7, иначе на пункт 4;
4) выполняется тело цикла;
5) счётчик увеличивается на единицу: $i := i + 1$;
6) переход на 2;
7) при выполнении условия **i > b** цикл заканчивается.

Пример. Найти сумму значений переменной цикла.

Решение. Фрагмент программы с оператором цикла запишется в виде:

```
For i:= 1 to 10 do s:=s+i;
Writeln('s=', s);
```

Счётный оператор цикла FOR может иметь такую структуру:

```
FOR i: = b DOWNTO a DO <оператор>;
```

Замена зарезервированного слова TO на DOWNTO означает, что шаг наращивания переменной цикла равен -1.

Пример. Найти сумму значений переменной цикла.

Фрагмент программы с оператором цикла запишется в виде:

```
For i:=10 downto 1 do s:=s+i;
Writeln('s=', s);
```

{Результат получится тот же, что и в примере с циклом `For.. to…do`}.

При работе с оператором `FOR` следует соблюдать ряд правил:

1) нельзя войти в цикл, минуя оператор FOR;
2) нельзя изменять параметры цикла (a, b) внутри цикла;
3) параметры цикла и переменная цикла должны быть целыми;
4) шагом цикла может быть единица или минус единица;
5) естественное окончание цикла осуществляется при условии i > b при шаге = 1;
6) из цикла можно выйти до естественного окончания цикла по условию.

Оператор цикла WHILE с предусловием

Структура оператора имеет вид:

```
WHILE <условие> DO <оператор>;
```

WHILE, DO – зарезервированные слова (WHILE – пока; DO – выполнить);

<условие> – выражение логического типа;

<оператор> – произвольный оператор Турбо Паскаля.

Если выражение <условие> имеет значение TRUE, то выполняется <оператор>, после чего вычисление выражения <условие> и его проверка повторяются. Если <условие> имеет значение FALSE, оператор WHILE прекращает свою работу.

Пример. Переписать фрагмент примера с циклом `While…do`, используя оператор цикла с предусловием.

Решение. Фрагмент программы с оператором цикла запишется в виде:

```
s:=0; i:=1;
while i<=10 do
Begin
s:=s+i;
i:=i+1;
End;
Writeln('s=', s);
```

В примере рассматривается составной оператор цикла, тело цикла заключено в операторные скобки.

Оператор цикла REPEAT...UNTIL с постусловием

Структура оператора имеет вид:

```
REPEAT <тело_цикла> UNTIL <условие>;
```

REPEAT, UNTIL – зарезервированные слова (повторять до тех пор, пока не будет выполнено условие);

<тело_цикла> – произвольная последовательность операторов Турбо Паскаля; <условие> – выражение логического типа.

Операторы, входящие в <тело_цикла>, выполняются хотя бы один раз, после чего вычисляется выражение <условие>: если его значение есть FALSE, операторы <тело_цикла> повторяются, в противном случае оператор REPEAT...UNTIL завершает свою работу.

Пример. Переписать фрагмент примера с циклом `While…do`, используя оператор цикла с постусловием.

Решение. Фрагмент программы с оператором цикла запишется в виде:

```
s:=0; i:=1;
repeat
s:=s+i;
i:=i+1;
Until i>10;
Writeln('s=',s);
```

В данном примере цикл выполняется, пока переменная $i \le 10$, при $i > 10$ цикл закончится.

Пример. Написать программу циклического алгоритма (рис. 11).

В цикле вычислить значение функции z = *x* y* при условии, что одна из переменных – ***x*** меняется в каждом цикле, а другая переменная – ***y*** не меняется и может быть любым целым числом.

Решение. Программа алгоритма цикла со счётчиком имеет вид:

```
PROGRAM PR5;
Var
x, y, z, i, n:integer;
begin
{циклический алгоритм}
writeln ('ввести x, y, количество циклов-n');
readln (x, y, n);
for i:=1 to n do {оператор цикла с параметрами}
begin
z:=x*y;
Writeln ('x= ', x, ' y= ', y, ' z= ', z);
x:=x+1;
End; {конец оператора цикла с параметрами}
End.
```

Если неизвестно количество циклов, то следует выбрать любой из двух операторов цикла: с предусловием или с постусловием.

Пример. Пока *y* > *x* вычислить *y* = *y* – *x*, если *y* = 30, *x* = 4. Вывести на экран количество циклов и значения переменной **y** в цикле. Алгоритм этой задачи представлен в виде блок-схемы (рис. 12).

Решение. Программа алгоритма цикла с предусловием имеет вид:

```
PROGRAM PR6;
Var
i, x, y: integer;
begin
{циклический алгоритм}
x:=4; y:=30; i:=0;
{оператор цикла с предусловием}
while y>x do
begin
y: = y - x;
i:=i+1;
readln ('i=', i, 'x= ', x, 'y= ', y);
end; {конец оператора}
 end.
```

В примере используется оператор цикла с предусловием, который работает при условии *y* > *x*. Условие проверяется при входе в цикл.

Пример. Составить программу для алгоритма, представленного на рис. 13, используя оператор цикла с постусловием. Цикл выполняется при условии $y > x$, но в конце оператора проверяется условие выхода из цикла ($y <= x$).

Решение. Программа алгоритма цикла с постусловием имеет вид:

```
PROGRAM PR7;
Var
i, x, y: integer;
begin
{циклический алгоритм}
x:=4; y:=30; i:=0;
{оператор цикла с постусловием}
repeat
y: = y – x;
i:=i+1;
readln ('i=', i, 'x= ', x, 'y= ', y);
until y<=x; {конец оператора}
 end.
```

5.8. Программирование. Массивы

Понятие и описание массивов

Массивом называется последовательность однотипных объектов, обозначаемая одним именем.

Массив характеризуется размером и размерностью. Размер массива – это количество элементов в нём. Размерность – это количество индексов в скобках.

Массив состоит из элементов. Чтобы выделить один из элементов массива, надо указать имя массива и номер элемента в нем. Номер элемента называется индексом, индекс указывается в квадратных скобах и может быть числом, переменной, выражением. Имя массива образуется по правилам образования имен переменных.

Пример. A [20], B [5,3].

Если для выделения элемента нужен 1 индекс, массив называется одномерным, два – двумерным и т. д.

Массивы относятся к структурированным типам данных. В программе массив можно описать двумя способами:

1) непосредственно в разделе описаний переменных:

var

имя_массива: **array [1..N] of** тип_элементов,

где *N* – максимально возможное количество элементов массива.

Пример

```
var
a, b: array [1..10] of real;
d, y: array [1..5,1..5] of integer;
```

2) объявлением типа – массива (удобно, когда требуется несколько одинаковых массивов).

Пример

```
type
mas= array [1..10] of real;
var
c,d:mas; {два массива типа mas}
```

Элементы массива могут быть любого типа, а индексы могут быть любого порядкового типа (например, типа **int, char**...). Но обычно используется тип – диапазон: 1..10, N..M.

Число элементов массива и его границы фиксируются при его описании и не могут быть изменены в процессе выполнения программы. Границы диапазона [N..M] могут быть заданы константами в разделе описания констант.

В ряде задач требуется ввести массив в виде константы, в котором записывается табличные значения (например, кривая намагничивания стали или начальное распределение температуры рассматриваемого объекта). Элементы такого массива-константы не могут быть изменены.

Пример

```
program PR8;
const
y: array [1..5] of integer=(7,1,5,3,9);
var
k: integer;
begin
for k:=1 to 5 do
```

```
writeln (y[k]);
end.
```

Ввод и вывод массивов

Ввод и вывод массивов осуществляется поэлементно. Часто это делают с помощью циклов (обычно используется цикл FOR).

Пример. Ввести с клавиатуры значения элементов одномерного массива вещественного типа, состоящего из 10 элементов. Вывести элементы массива на экран.

Решение

```
program PR9;
var
A: array[1..10] of integer;
k: integer;
begin
for k:=1 to 10 do
 readln (A[k]);
for k:=1 to 10 do
 writeln (A[k]);
end.
```

Двумерные массивы (матрицы) можно вводить по строкам или по столбцам.

Пример. Пусть требуется ввести массив A (3; 4):

$a_{1,1}\ a_{1,2}\ a_{1,3}\ a_{1,4}$
$a_{2,1}\ a_{2,2}\ a_{2,3}\ a_{2,4}$
$a_{3,1}\ a_{3,2}\ a_{3,3}\ a_{3,4}$

Решение

1) ввод и вывод по строкам:

```
program PR10;
var
a:array [1.. 3, 1.. 4] of integer;
i; j:integer;
begin
for i:=1 то 3
 for j:=1 то 4
 readln (a[i, j]);
for i:=1 то 3
 for j:=1 то 4
 writeln (a[i, j]);
end.
```

2) ввод по столбцам:

```
begin
for i:=1 to 4
 for j:=1 to 3
 readln (a[j, i]); {изменен порядок индексов}
```

Операции с массивами

Операции с массивом производятся только с *отдельными элементами массива*. С элементами массива можно делать все операции, которые разрешены для базового типа массива. Если массив числовой, то математические, если символьный или строковый, то, соответственно, операции с символьными или строковыми переменными.

Пример. Написать программу вычисления произведения положительных элементов одномерного массива целых чисел.

Решение

```
program Pr11;
var
a: array [1..100] of integer;
p, n, i: integer;
begin
p: =1;
writeln ('введите размер массива n<=100');
readln (n);
writeln ('ввод элементов массива');
for i:=1 to n do
begin
writeln('ввести a[ ', i, ']= ');
readln (a[i] );
end;
for i:=1 to n do
if a[i]>0 then p:= p * a[ i ];
writeln ('Произведение =', p);
readln;
end.
```

Двумерные массивы

Пример. Написать программу вычисления произведения положительных элементов второй строки двумерного массива целых чисел.

Решение

```
program Pr12;
var
```

```
a:array [1.. 10, 1.. 10] of integer;
p, i, j, n, m: integer;
begin
writeln(' Введите n<=10, m<=10 ');
readln (n,m);
writeln(' Введите элементы массива по строкам ');
for i:=1 to n do
for j:=1 to m do
readln(a[i, j]);
p:=1;
for j:=1 to m do
if a[2, j]>0 then p:= p * a[2, j];
writeln(' Произведение элементов p=', p);
readln;
end.
```

Контрольные вопросы

1. Оператор цикла с параметрами. Правила записи.
2. Оператор цикла с предусловием. Правила записи.
3. Оператор цикла с постусловием. Правила записи.
4. Описание одномерного массива.
5. Описание двумерного массива.
6. Ввод и вывод элементов одномерного массива.
7. Ввод и вывод элементов двумерного массива.

5.9. Практические работы

Алгоритмизация и программирование

Цель работы – научиться выполнять словесный алгоритм, представлять алгоритмы решений простейших задач в виде блок-схем и писать по ним программы.

Примечание

Студент должен выполнить задание в двух вариантах:

1) выполнить словесный алгоритм и записать его результат;
2) представить словесный алгоритм в виде блок-схемы и программы. Ввести программу, запустить её, получить результат.

Задание

Выполнить упражнение 1. Перед выполнением упражнения изучите материал по теме **5.1, 5.2**.

Упражнение 1. Линейный алгоритм

Задание

1. Выполнить словесный алгоритм. Записать результат.
2. Составить блок-схему и написать программу по заданному алгоритму.

Словесный алгоритм

В результате работы линейного алгоритма:

k:= 8;

m:= k + 2;

n:= k + m;

k:= n – 2 * k;

m:= k + n;

найти значение переменных: k, n, m.

Решение

1. Словесный алгоритм выполняется последовательно.

- Значение $k = 8$ подставляется в $m = k + 2 = 10$.
- Значение $k = 8$, $m = 10$ подставляется в $n = k + m = 18$.
- Вычисляется новое $k = n - 2 * k = 18 - 2 * 8 = 2$.
- Вычисляется новое m:= $k + n = 2 + 18 = 20$.

В результате работы линейного алгоритма значения переменных равны: $n = 18$, $k = 2$, $m = 20$.

2. Блок-схема алгоритма задачи представлена на рис. 14.

Программа алгоритма, представленного на рис. 14.

```
PROGRAM PR1;
VAR
k, m, n: integer;
BEGIN
Writeln ('ввести k'); {На экран выводится подсказка – текст в скобках}
Readln (k); {Ввод с клавиатуры переменной k}
m:= k + 2;
n:= k + m;
k:= n – 2 * k;
m:= k + n;
Writeln ('k=', k,' n=', n,' m=', m); {Вывод переменных k, n, m}
END.
```

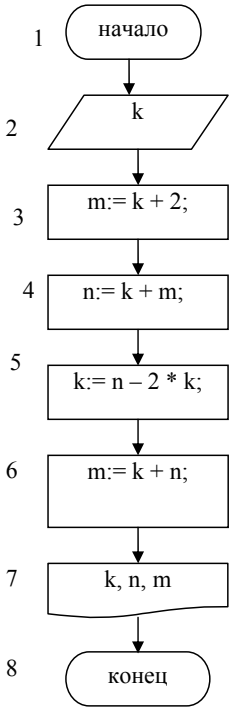


Рис. 14. Блок-схема линейного алгоритма

В фигурных скобках даются пояснения (комментарии) к операторам.

В блок-схеме, представленной на рис. 14, значение переменной ***k*** вводится с клавиатуры. Поэтому в программе этому блоку соответствует оператор ввода, что позволяет ввести с клавиатуры любое значение переменной ***k***.

Вывод. Алгоритм линейного типа, заданный в виде перечисления операций, может быть значительно сложнее. В результате вероятность ошибки словесного вычисления (задание 1) возрастает. Если представить алгоритм в виде блок-схемы, то чётко просматривается последовательность выполнения операций. Алгоритм можно усложнить за счёт ввода переменной ***k*** с клавиатуры.

Запись алгоритма в виде программы значительно упрощается, если следовать блок-схеме (рис. 14).

- Блоку 1 соответствует слово BEGIN (начало).

- Блоку 2 соответствует оператор ввода Readln (k).
- Блоки 3÷6 переписываются с рис. 14.
- Блоку 7 соответствует оператор вывода Writeln (‘k=’, k,’ n=’, n,’ m=’, m).
- Блоку 8 соответствует слово END (конец программы).

В результате выполнения программы линейного типа можно получить только по одному значению для каждой переменной. Если с клавиатуры ввести другое значение переменной ***k***, то оператор вывода выдаст следующий результат. Если необходимо вычислить таблицу значений при изменении переменной ***k***, то следует выбрать циклический алгоритм.

Упражнение 2. Разветвляющийся алгоритм

Задание

1. Выполнить словесный алгоритм. Записать результат.
2. Составить блок-схему и написать программу по алгоритму.

Словесный алгоритм

Задан фрагмент алгоритма:

если $W > R$, то $R = W + R$, иначе $W = R - W$.

В результате выполнения данного алгоритма с начальными значениями:

$W = -7$, $R = 55$

Решение

1. Для начальных значений: $W = -7$, $R = 55$ условие $W > R$ не выполняется. В этом случае выполняется вторая ветка $W = R - W = 55 + 7 = 62$.

В результате работы алгоритма значения переменных равны: $W = 62$, $R = 55$.

2. Блок-схема словесного алгоритма представлена на рис. 15.

На рис. 15 появился новый блок 3, в котором проверяется условие. Блок проверки условия образует ветвление по двум направлениям в алгоритме.

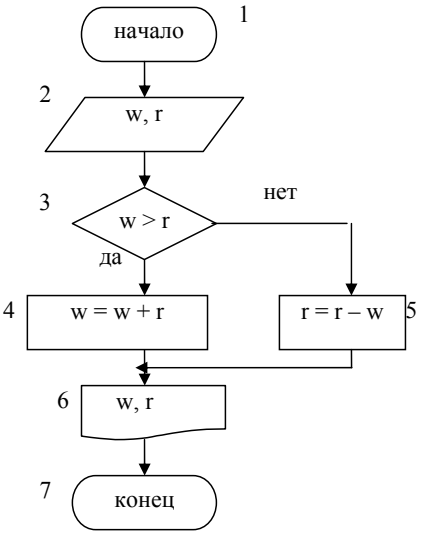


Рис. 15. Алгоритм ветвления

В блок-схеме видно, что в зависимости от условия $w > r$ выполняется одна из веток алгоритма. Затем выводится результат вычисления.

- Блоку 1 соответствует служебное слово BEGIN.
- Блоку 2 соответствует оператор ввода Readln (*w*, *r*).
- Блоку 3 соответствует оператор условия if $w > r$ then $w := w + r$ else $r := r - w$.
- Блоку 4 соответствует оператор присваивания $w = w + r$.
- Блоку 5 соответствует оператор присваивания $r = r - w$.
- Блоку 6 соответствует оператор вывода Writeln (' $w =$', *w*, ' $r =$', *r*).
- Блоку 7 соответствует служебное слово END.

Программа алгоритма ветвления, представленного на рис. 15.

```
PROGRAM PR2;
VAR
w, r: integer;
BEGIN
Writeln ('ввести w, r'); {На экран выводится подсказка – текст в скобках}
Readln (w, r); {Ввод с клавиатуры переменных w, r}
```

if $w > r$ then
$w := w + r$
else
$r := r - w$;
Writeln (' $w =$', w, ' $r =$', r); {Вывод результата}
END.

Упражнение 3. Алгоритмы. Циклы

Задание

1. Выполнить словесный алгоритм. Записать результат.
2. Составить блок-схему и написать программу по алгоритму.

Пример 1

Циклический алгоритм со счётчиком циклов задан в виде словесного описания.

Заданы начальные значения переменных:
$s := 0$; $d := 1$;
Начало цикла для i от 1 до 3;
$d := 2 * d$; $s := s + d$;
конец цикла; вывод d, s.

Решение

1. В алгоритме указан диапазон изменения счётчика ***i***, где видно, что должно быть выполнено три цикла.

- После выполнения первого цикла значения переменных равны $d = 2$, $s = 2$.
- Полученные значения подставляются во втором цикле.
- После выполнения второго цикла значения переменных равны $d = 4$, $s = 6$.
- Полученные значения во втором цикле подставляются при выполнении третьего цикла.
- В результате выполнения алгоритма значения переменных равны: $d = 8$, $s = 14$.

2. Блок-схема словесного алгоритма цикла со счётчиком представлена на рис. 16.

- Блоку 1 соответствует служебное слово BEGIN.
- Блоку 2 соответствует оператор ввода readln (n).
- Блоку 3 соответствуют операторы присваивания $s := 0$; $d := 1$.

- Блоку 4 соответствует оператор цикла со счётчиком for *i*: = 1 to *n* do.
- Блоку 5 соответствуют операторы присваивания *d*: = 2 * *d*; *s*: = *s* + *d*.
- Блоку 6 соответствует оператор вывода Writeln (‘*d* = ’, *d*, ‘*s* = ’, *s*).
- Блоку 7 соответствует служебное слово END.

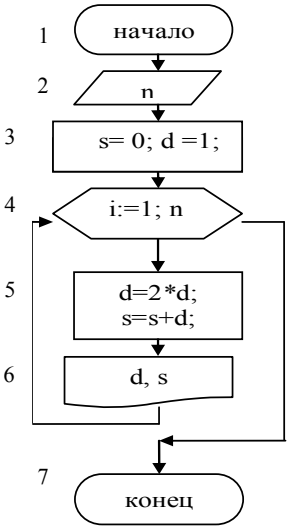


Рис. 16. Алгоритм цикла со счётчиком

Программа алгоритма цикла со счётчиком, представленного на рис. 16.

```
PROGRAM PR3;
Var
s, d, i, n: integer;
begin
writeln (‘ввести количество циклов – n’);
readln (n);
s: = 0; d: = 1;
for i: = 1 to n do {оператор цикла с параметрами}
begin
d: = 2 * d;
s: = s + d;
```

Writeln (‘*d* = ’, *d*, ‘ *s* = ’, *s*);

End; {конец оператора цикла}

End.

Пример 2

Циклический алгоритм с предусловием задан в виде словесного описания.

Заданы начальные значения переменных:

x: = 1; *y*: = 5;

Начало цикла. Пока *y* > *x* выполнить:

y: = *y* – *x*;

k = *k* + 1;

конец цикла.

Определить количество циклов ***k*** и значения переменной ***y*** после выхода из цикла.

Решение

1. Цикл выполняется до тех пор, пока выполняется условие *y* > *x*.

- Так как *y* = 5, *x* = 1, то условие *y* > *x* выполняется и значение ***y*** вычисляется по формуле *y* = *y* – *x*.
- В результате выполнения первого цикла *y* = 4.
- Во втором цикле условие *y* > *x* выполняется, после второго цикла значение *y* = 3.
- В третьем цикле условие *y* > *x* выполняется, после окончания третьего цикла значение *y* = 2.
- В четвертом цикле условие *y* > *x* выполняется, после выполнения цикла значение *y* = 1.
- При значениях *y* = 1, *x* = 1 условие *y* > *x* не выполняется, цикл не будет выполняться. Следовательно, цикл закончится и выполнится четыре цикла.

На выходе из цикла значения переменных будут равны: *k* = 4, *y* = 1, *x* = 1.

2. Программа алгоритма цикла с предусловием, представленного на рис. 12.

PROGRAM PR4;

Var

k, *x*, *y*: integer;

begin

```
writeln (‘ввести x, y’);
readln (x, y);
while y > x do          {оператор цикла с предусловием}
begin
y: = y – x;
k: = k + 1;
writeln (‘ k =’, k, ‘ y = ’, y);
end; {конец оператора цикла с предусловием}
end.
```

В программе до выполнения цикла не задано начальное значение k. По умолчанию оно равно нулю.

В примере используется оператор цикла с предусловием, который в данном примере выполняется при условии $y > x$. Условие проверяется при входе в цикл. В теле цикла счётчик задан в виде оператора присваивания $k: = k + 1$, который выдаёт количество выполненных циклов.

Пример 3

Циклический алгоритм примера 2 переписать, используя оператор цикла с постусловием. Результат будет тот же.

Программа алгоритма цикла с постусловием, представленного на рис. 13.

```
PROGRAM PR5;
Var
k, x, y: integer;
begin
writeln (‘ввести x, y, ’);
readln (x, y);
repeat {оператор цикла с постусловием}
y: = y – x;
k: = k + 1;
readln (‘ k =’, k, ‘ y = ’, y);
until y <= x; {конец оператора цикла с постусловием }
end.
```

Упражнение 4. Одномерные массивы

Пример 1. Требуется найти максимальный элемент одномерного массива и его номер по порядку следования в массиве. Представить алгоритм задачи в виде блок-схемы и написать по ней программу.

Решение

1. Алгоритм поиска: вводим переменную Max, в которую записываем первый элемент массива. Затем в цикле сравниваем каждый последующий элемент с Max. Если число, хранящееся в текущем элементе, больше хранящегося в Max, то число из текущего элемента записываем в Max.

2. Рекомендуется перед написанием программы составить таблицу идентификаторов.

№	Наименование переменной	Обозначения в программе
1	Имя массива	x
2	Размер массива	N
3	Индекс массива	i
4	Максимальный элемент	max
5	Номер максимального элемента	k

Программа поиска максимального элемента одномерного массива и его номера.

```
program PR6;
var
x: array[1..100] of integer;
k, max, n, i: integer;
Begin
Writeln (‘ввести количество элементов массива n’);
readln (n);
for i: = 1 to n do
readln (x[i]); {ввод элементов массива}
max: = x[1];
for i: = 1 to n do
if x[i] > max then
begin
max: = x[i];
k: = i;
```

```
end;
writeln(' max = ', max, ' k = ', k);
end.
```

Блок-схема алгоритма поиска максимального элемента одномерного массива и его номера представлена на рис. 17.

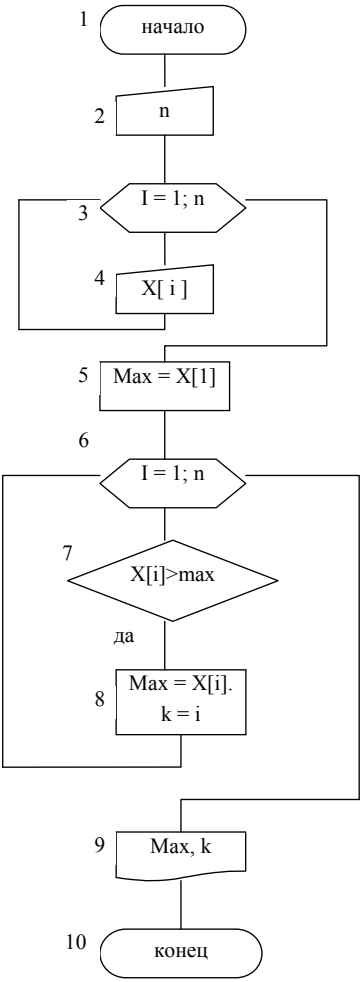


Рис. 17. Алгоритм поиска максимального элемента одномерного массива и его номера

Блок 2 – ввод количества элементов одномерного массива.

Блок 3 – начало цикла, в котором будут вводиться элементы одномерного массива.

Блок 4 – ввод элементов одномерного массива в цикле.

Блок 5 – значение первого элемента одномерного массива присваивается максимальному элементу.

Блок 6 – начало цикла, в блоке 7 которого проверяется условие максимального элемента одномерного массива, а в блоке 8 фиксируется значение и номер максимального элемента одномерного массива.

В блоке 9 – выводится максимальный элемент одномерного массива и его номер.

Двумерные массивы

Пример 2. Для двумерного массива, состоящего из *N* строк и *M* столбцов, найти сумму элементов 3 столбца.

Решение. Таблица идентификаторов.

№	Наименование переменной	Обозначения в программе
1	Имя массива	a
2	Количество строк в массиве	n
3	Количество столбцов в массиве	m
4	Индекс строки	i
5	Индекс столбца	j
6	Сумма элементов 3 столбца	s

Программа поиска суммы элементов 3 столбца двумерного массива:

```
program PR6;
var
a: array[1.. 10, 1..10] of integer;
s, i, j, n, m: integer;
begin
writeln ('ввести количество строк – n и столбцов – m');
readln (n, m);
for i: = l to n do
for j: = l to m do
begin
```

```
writeln (’ввести элемент массива a[ ’, i, ’, ’, j, ’ ] = ’);
readln (a[i, j],); {ввод элемента массива}
writeln (a[i, j]); {вывод элемента массива}
end;
s: = 0;
for i: = 1 to n do
s: = s + a[ i, 3]; {сумма элементов 3 столбца}
writeln(’s =’, s, );
end.
```

Глава 6. КОМПЬЮТЕРНЫЕ СЕТИ

Цель – ознакомить студентов с основами построения компьютерных сетей, их видами, принципами работы.

Учебные вопросы

1. Виды компьютерных сетей.
2. Характеристики компьютерных сетей.
3. Топология сети.
4. Назначение сети Интернет.
5. Службы сети Интернет.

Изучив данную тему, студент должен:

знать:

- основные виды и назначение компьютерных сетей;
- понятие топологии сети и их виды;
- службы сети Интернет и их функции;
- адресация в сети Интернет;
- назначение браузера;

уметь:

- просматривать и сохранять Web-страницы;
- использовать вычислительные системы в профессиональной деятельности.

6.1. Понятие компьютерных сетей

Компьютерной сетью называется совокупность взаимосвязанных через каналы передачи данных компьютеров, обеспечивающих пользователей средствами обмена информации и коллективного использования ресурсов сети.

Объединение компьютеров в сеть позволяет совместно использовать дорогостоящее оборудование: диски большой емкости, принтеры, основную память, иметь общие программные средства и данные. Сеть (network) – это группа компьютеров, соединённых между собой кабелем или какой-то другой средой передачи.

Основным назначением сети является обеспечение простого и удобного доступа пользователя к распределенным общесетевым ресурсам и организация их коллективного использования при надежной защите от несанкционированного доступа, а также обеспечения средств передачи данных между пользователями сети.

Общие принципы организации и функционирования компьютерных сетей

Компьютерные сети включают три составляющие: техническое, информационное и программное обеспечение.

Техническое обеспечение – это ЭВМ различных типов, средства связи, оборудование абонентских пунктов. Основные требования, которые предъявляются к сети, это универсальность, т. е. возможность выполнения практически неограниченного круга задач пользователей, и модульность, обеспечивающая возможность наращивания и изменения конфигурации сети.

Информационное обеспечение сети представляет собой единый информационный фонд, ориентированный на решаемые в сети задачи. В состав информационного обеспечения входят база знаний, банки данных и т. д.

Программное обеспечение сети предназначено для организации коллективного доступа к ее ресурсам, динамического распределения и перераспределения ресурсов сети с целью максимальной загрузки технических средств.

Основным компонентом программного обеспечения сети являются **сетевые операционные системы,** которые представляют собой комплекс управляющих и обслуживающих программ.

Основные характеристики компьютерных сетей

Для оценки качества коммуникационной сети можно использовать следующие характеристики:

1) скорость передачи данных по каналу связи;
2) пропускная способность канала связи;
3) достоверность передачи информации;
4) надежность канала связи и модемов.

Скорость передачи данных по каналу связи измеряется количеством битов информации, передаваемых за единицу времени (бит

в секунду). Скорость передачи данных зависит от типа и качества канала связи, типа используемых модемов.

Пропускная способность канала связи оценивается количеством знаков, передаваемых по каналу за единицу времени. Теоретическая пропускная способность определяется скоростью передачи данных. Единица измерения пропускной способности канала связи равна количеству знаков, передаваемых по каналу в секунду.

Достоверность передачи информации оценивают как отношение количества ошибочно переданных знаков к общему числу переданных знаков. Единица измерения достоверности передачи информации равна количеству ошибок на знак.

Надежность коммуникационной сети определяется либо долей времени исправного состояния в общем времени работы, либо средним временем безотказной работы. Единица измерения надежности равна среднему времени безотказной работы за час.

6.2. Классификация компьютерных сетей

Современные сети можно классифицировать по различным признакам.

По удаленности компьютеров

• **Локальные – LAN** (Local Area Network) – компьютеры расположены на расстоянии до нескольких километров и обычно соединены при помощи скоростных линий связи. Главная отличительная особенность локальных сетей – это единый, высокоскоростной канал передачи данных и малая вероятность возникновения ошибок в коммуникационном оборудовании.

• **Региональные сети – MAN** (Metropolitan Area Network) – объединяют пользователей области, города, небольших стран. В качестве каналов связи используются телефонные линии. Расстояние между узлами сети составляет от 10 до 1000 км.

• **Глобальные – WAN** (Wide Area Network) – включают другие глобальные, локальные сети, а также отдельно подключаемые к ней компьютеры.

По топологии (схема соединения компьютеров в сети)

• Общая шина – предполагает использование одного кабеля, к которому подключаются все компьютеры.

• Звезда – каждый компьютер через специальный сетевой адаптер подключается отдельным кабелем к центральному узлу.

• Кольцо – данные передаются от одного компьютера другому по эстафете (по кругу).

По назначению и перечню предоставляемых услуг

• Общее использование файлов и принтеров – с помощью специальной ЭВМ (файл-сервер, принтер-сервер) организуется доступ пользователей к файлам и принтерам.

• Общее использование баз данных – с помощью специальной ЭВМ (сервер баз данных) организуется доступ пользователей к базе данных.

• Применение технологий Интернет – электронная почта, Всемирная паутина, телеконференции, видеоконференции, передача файлов через Интернет.

По способу организации взаимодействия

• **Одноранговые сети** – все компьютеры одноранговой сети равноправны, при этом любой пользователь сети может получить доступ к данным, хранящимся на любом компьютере. Главное достоинство одноранговых сетей – это простота установки и эксплуатации. Главный недостаток состоит в том, что в условиях одноранговых сетей затруднено решение вопросов защиты информации. Поэтому такой способ организации сети используется для сетей с небольшим количеством компьютеров и там, где вопрос защиты данных не является принципиальным.

• **Сети с выделенным сервером (иерархические сети)** – при установке сети заранее выделяются один или несколько **серверов** – компьютеров, управляющих обменом данных по сети и распределением ресурсов. Любой компьютер, имеющий доступ к услугам сервера, называют **клиентом сети** или **рабочей станцией**. Сам сервер может быть клиентом только сервера более высокого уровня иерархии. Иерархическая модель сети является наиболее предпочтительной, так как позволяет создать наиболее устойчивую структуру сети и более рационально распределить ресурсы. Также достоинством иерархической сети является более высокий уровень защиты данных.

По технологии использования сервера

• Сети с архитектурой **файл-сервер** – используется файловый сервер, на котором хранится большинство программ и данных. По требованию пользователя ему пересылаются необходимая программа и данные. Обработка информации выполняется на рабочей станции.

• Сети с архитектурой **клиент-сервер** – между приложением-клиентом и приложением-сервером осуществляется обмен данными. Хранение данных и их обработка производится на мощном сервере, который выполняет также контроль за доступом к ресурсам и данным. Рабочая станция получает только результаты запроса.

По скорости передачи информации компьютерные сети делятся на низко-, средне- и высокоскоростные.

По типу среды передачи сети разделяются на проводные (на коаксиальном кабеле, на витой паре, оптоволоконные) и беспроводные с передачей информации по радиоканалам или в инфракрасном диапазоне.

Контрольные вопросы

1. Понятие компьютерных сетей.
2. Общие принципы организации и функционирования компьютерных сетей.
3. Основные характеристики компьютерных сетей.
4. Классификация компьютерных сетей по удаленности.
5. Классификация компьютерных сетей по топологии.
6. Классификация компьютерных сетей по назначению и перечню предоставляемых услуг.
7. Классификация компьютерных сетей по способу организации взаимодействия.
8. Классификация компьютерных сетей по технологии использования сервера.

6.3. Протоколы

Перемещение информации между компьютерами различных схем является чрезвычайно сложной задачей. Международной организацией по стандартизации (ISO – International Standards

Organization) была разработана эталонная модель взаимодействия открытых систем (Open System Interconnections – OSI).

OSI быстро стала основной архитектурной моделью для передачи межкомпьютерных сообщений. Часто ее называют моделью архитектуры открытых систем.

Модель взаимодействия открытых систем состоит из семи уровней.

1. Физический уровень – самый нижний в модели. На физическом уровне осуществляются:

- представление информации в виде электрических или оптических сигналов;
- преобразование формы сигналов;
- выбор параметров физических сред передачи данных, передача потоков битов (нулей и единиц) по физической среде.

Он отвечает за кодирование данных, определяет скорость передачи физической информации, максимальные расстояния передачи информации.

2. Канальный уровень обеспечивает надежный транзит данных через физический канал, управление потоком данных в виде кадров, в которые упаковываются информационные пакеты, обнаруживает ошибки передачи. Выполняя эту задачу, канальный уровень решает вопросы физической адресации, топологии сети, упорядоченной доставки блоков данных и управления потоком информации.

3. Сетевой уровень отвечает за коммутацию пакетов, адресацию сообщений и перевод логических адресов в физические определяет маршрут от компьютера-отправителя к компьютеру-получателю. Традиционные протоколы сетевого уровня передают информацию вдоль этих маршрутов. *Таким образом, на этом уровне обеспечивается передача данных в сети, контроль нагрузки на сеть, маршрутизация пакетов.*

4. Транспортный уровень обеспечивает услуги по транспортировке данных, решает проблемы, связанные с отправкой и получением пакетов, обеспечивает доставку пакетов без ошибок и потерь, проверяет ошибки. *Таким образом, на этом уровне обеспечивается связь между конечными пунктами.*

5. Сеансовый уровень поддерживает и завершает сеанс связи, синхронизирует диалог между объектами представительного уров-

ня и управляет обменом информации между ними. *Таким образом, на этом уровне определяются тип связи, начало и окончание заданий, режим обмена запросами и ответами.*

6. Представительный уровень отвечает за преобразование протоколов, трансляцию данных, их шифрование, чтобы информация, посылаемая из прикладного уровня одной системы, была читаемой для прикладного уровня другой системы. *Таким образом, на этом уровне реализуются функции представления данных (кодирование, форматирование).*

7. Прикладной уровень управляет общим доступом к сети, потоком данных и обработкой ошибок, идентифицирует и устанавливает наличие предполагаемых партнеров для связи. *Таким образом, на этом уровне определяются и оформляются в блоки те данные, которые подлежат передаче по сети.* Это самый близкий к пользователю уровень OSI.

Чем выше уровень, тем более сложную задачу он решает. Достоинство семиуровневой модели OSI определяется возможностью изменений любого из уровней без изменений других.

Модель OSI устанавливает способы передачи данных по сети, определяет стандартные протоколы, используемые сетевым и программным обеспечением.

Протокол — это «язык», используемый для обмена данными при работе в сети. Чтобы различные компьютеры сети могли установить связь друг с другом, они должны «разговаривать» на одном языке, то есть использовать один и тот же протокол.

Граница между сеансовым и транспортным уровнями может быть представлена как граница между протоколами прикладного уровня и протоколами низших уровней. В то время как прикладной, представительный и сеансовый уровни заняты прикладными вопросами, четыре низших уровня решают проблемы транспортировки данных.

Существует множество протоколов, каждый из них выполняет различные задачи. Протоколы работают на разных уровнях OSI. Несколько протоколов могут работать совместно, так называемый стек (набор) протоколов. Уровни в стеке протокола соответствуют уровням модели.

На нижних уровнях используются два основных протокола: IP и TCP. Так как эти два протокола тесно взаимосвязаны, то их объединяют и считают базовым протоколом. Протоколы положены в основу алгоритма работы компьютерных программ, обслуживающих сервер. Все остальные многочисленные протоколы строятся на основе протоколов TCP/IP.

TCP/IP – это единый набор протоколов передачи данных, состоящий из многих программ (стек протоколов).

Термин «TCP/IP» обычно обозначает все, что связано с протоколами TCP и IP. Он охватывает целое семейство протоколов, прикладные программы и даже саму сеть. В состав семейства входят протоколы UDP, ARP, ICMP, TEL-NET, FTP и многие другие. TCP/IP – это технология межсетевого взаимодействия, технология internet.

TCP (Transmission Control Protocol) – протокол контроля передачи данных, он описывает, как большие массивы данных разбить на пакеты и собрать обратно.

Перед подачей в сеть данные разбиваются на пакеты. Пакет – это единица информации, передаваемая между устройствами сети как единое целое. На передающей стороне пакет проходит последовательно через все уровни системы сверху вниз (с прикладного уровня до физического). Затем он передаётся по сетевому кабелю на компьютер-получатель и опять проходит все уровни в обратном порядке.

Протокол обеспечивает надежность передачи данных. Интернет построен так, что пакеты следуют к месту назначения различными маршрутами и прибывают в конечную точку в другом порядке, нежели отправлялись, они могут теряться и дублироваться. Протокол TCP устраняет все возникающие проблемы и обеспечивает сборку всех пакетов в единое сообщение.

IP (Internet Protocol) – межсетевой протокол (протокол маршрутизации, транспортный протокол) на нижнем уровне описывает правила передачи небольших порций информации с одного компьютера на другой, управляет тем, как происходит передача информации.

IP называют протоколом адресации. Каждый пакет содержит адреса отправителя и получателя. Сервер, на который поступает данный пакет, сравнивает свой адрес с адресом получателя, указанным в пакете, и направляет пакет в нужную сторону.

Компьютеры локальной сети используют единый комплект протоколов для всех участников. Для организации обмена данными между компьютерами в ЛВС (локальной вычислительной сети) чаще всего используются стандартные протоколы (способ обмена сообщениями между ЭВМ по каналам связи), разработанные Международным институтом инженеров по электротехнике и радиоэлектронике IEEE (Institute of Electrical and Electronically Engineers).

6.4. Глобальная компьютерная сеть Internet

Интернет – это глобальная компьютерная сеть, объединяющая многие локальные, региональные и корпоративные сети и включающая десятки миллионов компьютеров. Реальное соединение серверов друг с другом напоминает паутину: один сервер может связываться с другим очень большим числом способов, и выход одного сервера из строя никак не повлияет на работу других. Паутинный способ соединения серверов в Интернете породил еще одно название этой сети – Всемирная паутина – WWW.

Структура и принципы работы Интернета

Отличительной особенностью сети Интернет является высокая надежность. Если выходят из строя некоторые линии связи или компьютеры, то сообщения могут быть переданы по другим линиям связи, так как всегда имеется несколько путей передачи информации. В качестве высокоскоростной магистрали передачи данных используются выделенные телефонные линии, оптоволоконные и спутниковые каналы связи.

Любая организация для подключения к Internet использует специальный компьютер, который называется **шлюзом** (gateway). На нем устанавливается обеспечение, осуществляющее обработку всех сообщений, проходящих через шлюз. Каждый шлюз имеет свой интернет-адрес. Если поступает сообщение, адресованное локальной сети, к которой подключен шлюз, то оно передается в эту локальную сеть. Если сообщение предназначено для другой сети, то оно передается следующему шлюзу. Каждый шлюз имеет информацию обо всех остальных шлюзах и сетях. Когда сообщение посылается из локальной сети через шлюз в Internet, то при этом выбирается

самый «быстрый» путь. Шлюзы обмениваются друг с другом информацией о маршрутизации и состоянии сети, используя специальный шлюзовой протокол.

У сети Интернет нет никакого единого центра, точно так же нет главной администрации или хозяина, хотя есть координирующие организации, которые выделяют IP-адреса и так называемые доменные адреса.

Пользователи Интернета (клиенты) подключаются к сети через компьютеры специальных организаций, которые называются поставщиками услуг Интернета (провайдеры или серверы). К сети могут быть подключены как отдельный компьютер, так и локальная сеть.

В отличие от своих клиентов серверы соединяются друг с другом (соединение «сервер-сервер») при помощи специальных выделенных каналов, им не нужно дозваниваться друг до друга, они постоянно на связи. Среди выделенных линий особую роль играют высокоскоростные линии. Именно на них ложится основная нагрузка по передаче информации. Серверы, соединенные друг с другом высокоскоростными выделенными линиями, образуют так называемый хребет Интернета. Каждый сервер не столько работает со своими клиентами, сколько передает информацию «чужих» клиентов к другим серверам. Компьютеры, подключенные к сети, называют узлами Интернета, или сайтами, от английского *site*, что переводится как *местонахождение*.

При подключении локальной сети предприятия к глобальной сети важную роль играет **понятие сетевой безопасности**. В частности, должен быть ограничен доступ в локальную сеть для посторонних лиц извне, а также ограничен выход за пределы локальной сети для сотрудников предприятия, не имеющих соответствующих прав.

Для обеспечения сетевой безопасности между локальной и глобальной сетью устанавливают брандмауэры. **Брандмауэром** может быть специальный компьютер или компьютерная программа, препятствующая несанкционированному перемещению данных между сетями.

Службы Интернета

К услугам сети Интернет прибегают сотни миллионов человек. Но сеть Интернет – это лишь средство связи компьютеров и локальных сетей между собой. Для хранения и передачи информации

по сети Интернет созданы специальные информационные службы, иногда называемые сервисами Интернет.

В простейшем понимании служба – это пара программ, взаимодействующих между собой согласно протоколам.

Этих служб несколько, наиболее распространенными являются следующие:

1. **Электронная почта (E-mail** – electronic mail) выполняет функции обычной почты. Она обеспечивает передачу сообщений из одного пункта в другой. Главным ее преимуществом является независимость от времени. Электронное письмо приходит сразу же после его отправления и хранится в почтовом ящике до получения адресатом.

2. **Телеконференции (UseNet)** разработаны как система обмена текстовой информацией (перемещения новостей) между компьютерами по всему миру. Она позволяет всем пользователям Internet участвовать в групповых дискуссиях, называемых телеконференциями, в которых обсуждаются всевозможные проблемы.

3. **Служба передачи файлов (FTP)**. Назначение FTP – обмен через Интернет отдельными файлами и целыми программами. Поиск и передача двоичных файлов.

4. **Терминальный режим (Telnet)** – служба удаленного управления компьютером.

5. **Служба IRC (Internet Relay Chat)** предназначена для прямого общения нескольких клиентов в режиме реального времени. Службу IRC называют чатом. В отличие от системы телеконференций в системе службы IRC общение происходит только в пределах одного канала, в работе которого принимают участие обычно лишь несколько человек.

6. **Служба ICQ –** одна из нескольких существующих в Интернете служб для мгновенного обмена сообщениями.

7. **Всемирная паутина (WWW).**

Адресация в Интернете

В Интернете используются два основных понятия: адрес и протокол. Каждому компьютеру, подключенному к Интернету, назначается уникальный сетевой IP-адрес. Хотя нет центра управления Интернетом, есть организации, занимающиеся проверкой и выдачей адресов.

Адрес в Интернете однозначно определяет местонахождение компьютера в сети. При пересылке информации протоколами TCP/IP используются присвоенные адреса. Адреса в Интернете могут быть представлены как последовательностью цифр, так и именем, построенным по специальным правилам. Для того чтобы серверам было легко ориентироваться в направлении пересылки пакетов, предусмотрен специальный способ адресации.

Каждый компьютер и каждый сервер сети имеют собственное имя-адрес, состоящее из четырех целых чисел от 0 до 255, разделенных точкой. Это числовой IP-адрес. Например: 217.89.14.35.

Начало адреса определяет часть Интернета, к которой подключен компьютер, а окончание определяет адрес компьютера в этой части сети.

Компьютеры при пересылке информации используют цифровые адреса, а пользователи в работе с Интернетом используют в основном имена, то есть доменную систему имен.

Доменная адресация. Служба имен доменов DNS

В Интернете используется доменная система имен компьютеров, которая включает принцип последовательных уточнений (уровни). Каждый уровень в такой системе называется доменом. Домены отделяются друг от друга точками. Домен верхнего уровня располагается в имени правее, а домен нижнего уровня – левее. В имени может быть любое число доменов, но чаще всего используются имена с количеством доменов от трех до пяти. Например, www.tltsu.ru.

В этом примере домен верхнего уровня *ru* указывает на то, что адрес относится к российской части Интернета (региональный адрес). Следующий уровень определяет организацию, к которой принадлежит данный адрес. В данном случае это Тольяттинский государственный университет. Интернет-адрес ТГУ соответственно *tltsu*. Все компьютеры, подключенные к Интернету в ТГУ, объединяются в группу, имеющую такой адрес. Кафедре информатики университета, допустим, выделен свой домен с именем *inf*. В результате полный Интернет-адрес имеет вид: www.inf.tltsu.ru.

Доменная система образования адресов во всем Интернете содержит только один такой адрес.

Ранее отмечалось, что каждый компьютер имеет числовой IP-адрес, состоящий из четырех целых чисел от 0 до 255. Пользователю сети неудобно работать с числовым представлением IP-адреса, зато доменное имя запоминается легко. С другой стороны, автоматическая работа серверов сети организована с использованием четырехзначного числового адреса. Благодаря IP-адресу промежуточные серверы могут осуществлять передачу запросов и ответов в нужном направлении, не зная, где находится отправитель и получатель. Поэтому необходимо преобразование доменных имен в цифровую форму IP-адреса. Этим занимаются серверы службы имен доменов DNS (Domain Name Servise).

Запрос на получение одной из страниц сервера www.xyz.com сначала обрабатывается сервером DNS и далее направляется по IP-адресу, а не по доменному имени. Таким образом, существуют две разные формы записи адреса одного и того же сетевого компьютера:

1) числовой IP-адрес;
2) доменное имя.

Адрес URL. Протокол передачи данных

Каждый файл одного локального компьютера обладает уникальным полным именем, в которое входит собственное имя файла, включая расширение имени, и путь доступа к файлу, начиная от имени устройства, на котором он хранится. Уникальное имя файла во Всемирной сети значительно расширяется. Адрес любого файла во всемирном масштабе определяется унифицированным указателем ресурса.

При работе в Интернете чаще всего используются не просто доменные адреса, а универсальные указатели ресурсов, называемые URL – (Universal Resource Locator).

URL – это адрес любого ресурса в Интернете вместе с указанием протокола, с помощью которого следует к нему обращаться, программы для запуска на сервере и конкретного файла для обращения на сервер.

Унифицированный указатель на ресурс (сервер, сайт, страница, каталог, файл). URL в общем случае может указывать тип и место расположения ресурса.

Адрес URL состоит из трех частей/

1. Указание службы, которая осуществляет доступ к данному ресурсу (обычно обозначается именем прикладного протокола, соответствующего данной службе).

Например, для службы WWW прикладным является протокол HTTP- протокол передачи гипертекста. С учетом этого адрес URL будет начинаться следующей записью: http://...

2. Указание доменного имени компьютера (сервера), на котором хранится данный ресурс. Дополняя доменное имя компьютера, адрес URL уточняется и будет иметь вид: http://www.tltsu.ru.

Данную строку можно записать без ссылки на службу www: http://tltsu.ru.

3. Указание полного пути доступа к файлу на данном компьютере. Добавив его, адрес URL имеет вид: http://www.tltsu.ru/stud/aprel.doc, где запись: stud/aprel.doc – указывает спецификацию файла (папку и имя документа).

При записи URL-адреса следует соблюдать регистр символов. В Интернете строчные и прописные символы в именах каталогов и файлов считаются разными.

Общий формат URL, учитывая три составляющих, имеет вид:

<протокол>://<сервер><локальный адрес>,

где <протокол> – обозначает конкретный протокол передачи данных одной из служб Интернета: http, ftp, telnet и т. п.;
<сервер> – доменный адрес сервера информационной службы;

<локальный адрес> – путь к странице.

Гипертекстовая связь между множеством документов, хранящихся на физических серверах Интернета, является основой существования логического пространства WWW. Такая связь не могла бы существовать, если бы каждый документ в этом пространстве не обладал своим уникальным адресом.

Если путь к конкретной странице не указан, подразумевается начальная страница сайта или Web-сервера.

Например, адрес компьютера, на котором расположен WWW-сервер поисковой системы Rambler, имеет вид: http://www.rambler.ru.

По этому адресу в программах просмотра загружается стартовая страница системы Rambler, а Web-страница, описывающая

поисковый язык системы, имеет URL http://www.rambler.ru/new/help.html.

В этом адресе *rambler* – имя «мелкого» домена, а *ru* – имя корневого домена. Пользователи узлов (компьютеров сети Интернет), входящих в состав WWW, общаются между собой на основе протокола HTTP.

HTTP (Hyper Text Transfer Protocol – протокол передачи гипертекста). Этот протокол задает правила общения между программой просмотра Web-страниц и WWW-сервером, которые укладываются в схему «запрос – ответ». Указывая доменный адрес сервера и вид протокола (HTTP), запрашивается определенная услуга: найти на сервере в нужном месте нужный HTML-документ.

В простейшем случае программа просмотра Web-страниц требует некий документ, и сервер его выдает. Таким образом, чтобы просмотреть нужную Web-страницу, следует в адресном поле программы просмотра Web-страниц написать требуемый адрес и нажать на клавишу <Enter>.

Однако если адрес Web-документа не известен, необходимо обратиться к поисковым системам, которые представляют собой специализированные Web-узлы. **Электронная почта** работает по протоколам SMTP-Simple Mail Transfer Protocol (простой протокол пересылки почты) и POP3-Post Offise Protocol (протокол почтового офиса). Данные два протокола являются стандартными протоколами. Интернета, построенными на основе TCP/IP.

SMTP – протокол, определяющий правила отправки почтовых сообщений по Интернету.

POP3 –протокол для получения сообщений. В соответствии с ним почта принимается сервером и накапливается на нем. Программа – почтовый клиент – периодически проверяет почту на сервере и загружает сообщения на локальный компьютер.

Таким образом, почту отправляют с помощью протокола SMTP, а принимают с помощью POP3. В процессе создания учетной записи почты вводятся названия как сервера SMTP, так и сервера POP3.

Адрес электронной почты в общем случае выглядит примерно так: kmga@hotmail.com.

Адрес электронной почты состоит из двух частей, разделенных символом @. Справа от символа находится интернет-адрес компьютера, на котором располагается почтовое отделение абонента. Этот адрес формируется так же, как и любое другое доменное имя в Интернете. Слева от символа расположено имя абонента.

Слово, которое стоит в левой части адреса (до знака @), является именем адресата. То, что находится справа от знака @, является доменным именем компьютера, на котором хранятся сообщения.

Формат адреса электронной почты должен иметь вид:

bмя пользователя @ адрес компьютера.

Интернет является крупнейшим хранилищем файлов в мире. Сервис FTP – File Transfer Protocol (протокол передачи файлов) позволяет передавать и получать файлы.

FTP (File Transfer Protocol) – протокол передачи файлов, это протокол сети для работы с любыми типами файлов: текстовыми и бинарными, являющийся примером системы с архитектурой «клиент-сервер».

Протокол обеспечивает способ перемещения файлов между двумя компьютерами и позволяет абоненту сети Internet получить в свое распоряжение множество файлов. Пользователь получает доступ к различным файлам и программам, хранящимся на компьютерах, подключенных к сети.

Программа, реализующая этот протокол, позволяет установить связь с одним из множества FTP-серверов в Internet. FTP-сервер – это компьютер, на котором содержатся файлы, предназначенные для открытого доступа.

Программа FTP-клиент не только реализует протокол передачи данных, но и поддерживает набор команд, которые используются для просмотра каталога FTP-сервера, поиска файлов и управления перемещением данных: поиск и передача документов с помощью гипертекстовых ссылок (WWW, или Всемирная паутина); разговор в реальном режиме времени через Интернет (CHAT).

Служба ICQ. Для пользования этой службой надо зарегистрироваться на ее центральном сервере: http: // www. isq. com/.

WWW и HTML

World Wide Web (WWW) популярная служба Интернет. **WWW** – это единое информационное пространство, состоящее из сотен миллионов взаимосвязанных электронных документов. Вся информация в этой службе хранится на WWW-серверах (Web-серверах) в виде гипертекстовых документов. Отдельные документы, составляющие пространство Web, называют Web-страницами. Группы тематически объединенных Web-страниц называют Web-узлами, или просто сайтом.

Web-страницы пишутся на языке HTML (Hyper Text Markup Language) и могут содержать информацию различного вида: текст, рисунки, аудио и видео, что делает эту информацию чрезвычайно привлекательной для пользователей.

Язык HTML – язык разметки гипертекста, который содержит набор команд. Такие команды называются тегами. Правила записи тегов содержатся в спецификации особого языка разметки, близкого к языкам программирования. Таким образом, Web-документ представляет собой обычный текстовый документ, размеченный тегами HTML.

Гиперссылки в HTML-документах могут указывать как на другую часть этого документа, так и на другой документ, расположенный на любом сервере сети Интернет. Это позволяет легко отыскивать требуемую информацию, переходя посредством гиперссылок от документа к документу. Для поиска информации в сети Интернет используются специальные ***поисковые серверы***.

Браузеры – программы просмотра Web-страниц

Для работы с системой WWW необходимо установить на компьютере специальную программу просмотра Web-документов, называемую WWW-браузер. Это прикладная программа, которая взаимодействует с системой WWW, получает затребованные документы, интерпретирует данные и отображает содержание документов на экране. Основная задача программы-браузера (англ. *browse* [*brauz*] – пролистать, проглядеть, просмотреть) – открыть по указанному адресу Web-страницу.

Браузер выполняет отображение документа на экране, руководствуясь командами, которые автор документа внедрил в его текст. Современные браузеры располагают значительно более ши-

рокими возможностями и позволяют работать не только со службой WWW, но и с электронной почтой, телеконференциями и другими службами Интернет.

Они позволяют отображать на экране любые документы, созданные в любой операционной среде и на любом компьютере с конфигурацией, которая обеспечивает работу в сети.

С операционной системой Windows поставляется встроенный браузер Internet Explorer (IE). Ссылка на Internet Explorer обычно находится в пункте *Программы* главного меню Windows. Запускается и завершается программа любым из стандартных для Windows способов.

Контрольные вопросы

1. Перечислите и дайте краткую характеристику уровням модели взаимодействия открытых систем.
2. Структура и принципы работы Интернета.
3. Службы Интернета.
4. Правила адресации в Интернете.
5. Доменная адресация.
6. Адрес URL.

КОНТРОЛЬНАЯ РАБОТА

Темы

1. Системы счисления.
2. Алгебра логики.
3. Алгоритмизация и программирование.

1. Рекомендации по выполнению контрольной работы

Номер варианта контрольной работы выбирается из списка группы в алфавитном порядке. Номер студента в списке группы соответствует номеру варианта задания.

Контрольную работу следует оформить в текстовом редакторе Microsoft Word. Таким образом студент демонстрирует умение работать в текстовом редакторе. В процессе выполнения заданий контрольной работы студент может получить консультацию у преподавателя. График консультаций – в деканате заочного отделения.

Контрольная работа должна содержать:

1) титульный лист (рис. 18);
2) индивидуальное задание студента;
3) решение задач индивидуального задания студента.

Тольяттинский государственный
университет

КОНТРОЛЬНАЯ РАБОТА

по дисциплине «ИНФОРМАТИКА»

Выполнил

студент группы... Петров П.П.

Проверил
преподаватель Иванов И.И.

Тольятти 201..

Рис. 18. Пример оформления титульного листа контрольной работы

Раздел 3 контрольной работы должен включать:

1) подробное описание решения задач контрольной работы студента;

2) ссылки на теоретический материал, формулы.

2. Задания контрольной работы

Задание 1. Переведите числа из одной системы счисления в другую.

1. Ознакомьтесь с примером выполнения задания (раздел 1.3 учебно-методического пособия).
2. Выполните задание в соответствии с номером варианта.

Вариант	Задание
1	1. Выполнить действия в двоичной системе счисления: • 1100 – 0111 • 11001 * 111 • 1111 + 1011 2. Перевести из одной системы счисления в другую: • число 42 из десятичной системы счисления в двоичную • число 100111 из двоичной системы счисления в десятичную • число FA2 из шестнадцатеричной системы счисления в пятеричную
2	1. Выполнить действия в двоичной системе счисления: • 1111 – 1101 • 1001 * 110 • 10111 + 111 2. Перевести из одной системы счисления в другую: • число 39 из десятичной системы счисления в двоичную • число 11001100 из двоичной системы счисления в десятичную • число DBC из шестнадцатеричной системы счисления в троичную
3	1. Выполнить действия в двоичной системе счисления: • 1011 + 10111 • 10111 – 1111 • 1100101 * 101 2. Перевести из одной системы счисления в другую: • число 89 из десятичной системы счисления в двоичную • число 10100101 из двоичной системы счисления в десятичную • число 110111 из двоичной системы счисления в шестеричную
4	1. Выполнить действия в двоичной системе счисления: • 10011 + 1101 • 11000 – 1101 • 1011 * 111 2. Перевести из одной системы счисления в другую: • число 73 из десятичной системы счисления в двоичную • число 1110011 из двоичной системы счисления в десятичную • число 111101 из двоичной системы счисления в семеричную

Вариант	Задание
5	1. Выполнить действия в двоичной системе счисления: • 11001 – 10101 • 10011 * 111 • 11001 + 100011 2. Перевести из одной системы счисления в другую: • число 64 из десятичной системы счисления в двоичную • число 111100101 из двоичной системы счисления в десятичную • число 100101 из двоичной системы счисления в пятеричную
6	1. Выполнить действия в двоичной системе счисления: •11011 + 11111 • 11101 – 1101 • 1011 * 1001 2. Перевести из одной системы счисления в другую: • число 71 из десятичной системы счисления в двоичную • число 101100101 из двоичной системы счисления в десятичную • число 111101 из двоичной системы счисления в шестеричную
7	1. Выполнить действия в двоичной системе счисления: • 10111 – 1101 • 10011 * 1010 • 11011 + 11111 2. Перевести из одной системы счисления в другую: • число 85 из десятичной системы счисления в двоичную • число 110101 из двоичной системы счисления в десятичную • число 63D из шестнадцатеричной системы счисления в семеричную
8	1. Выполнить действия в двоичной системе счисления: • 110011 – 10001 • 101101 * 1101 • 10011001 + 1101 2. Перевести из одной системы счисления в другую: • число 42 из десятичной системы счисления в двоичную • число 111100001 из двоичной системы счисления в десятичную • число 4F9 из шестнадцатеричной системы счисления в шестеричную
9	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 110101 – 11011 • 1011 * 1001 2. Перевести из одной системы счисления в другую: • число 69 из десятичной системы счисления в двоичную • число 111100011 из двоичной системы счисления в десятичную • число 110101 из двоичной системы счисления в троичную
10	1. Выполнить действия в двоичной системе счисления: • 110101 + 10011 • 1001101 – 110011 • 1011 * 1001

Вариант	Задание
	2. Перевести из одной системы счисления в другую: • число 75 из десятичной системы счисления в двоичную • число 111100111 из двоичной системы счисления в десятичную • число 423 из пятеричной системы счисления в семеричную
11	1. Выполнить действия в двоичной системе счисления: • 1100101 + 100011 • 100111 – 11011 • 1011 * 1001 2. Перевести из одной системы счисления в другую: • число 312 из десятичной системы счисления в двоичную • число 111100110 из двоичной системы счисления в десятичную • число 621 из семеричной системы счисления в пятеричную
12	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 11111 – 10111 • 1011 * 1001 2. Перевести из одной системы счисления в другую: • число 251 из десятичной системы счисления в двоичную • число 111100100 из двоичной системы счисления в десятичную • число 3FD из шестнадцатеричной системы счисления в четверичную
13	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 110101 – 11101 • 1011 * 1001 2. Перевести из одной системы счисления в другую: • число 103 из десятичной системы счисления в двоичную • число 1111000001 из двоичной системы счисления в десятичную • число 565 из семеричной системы счисления в троичную
14	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 111101 – 1101 • 1011 * 1001 2. Перевести из одной системы счисления в другую: • число 149 из десятичной системы счисления в двоичную • число 110110111 из двоичной системы счисления в десятичную • число 332 из четверичной системы счисления в шестеричную
15	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 110111 – 1001 • 1011 * 1001 2. Перевести из одной системы счисления в другую: • число 107 из десятичной системы счисления в двоичную • число 111010111 из двоичной системы счисления в десятичную • число 110101 из двоичной системы счисления в семеричную

Вариант	Задание
16	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 11001101 – 1111 • 1011 * 1001 2. Перевести из одной системы счисления в другую: • число 143 из десятичной системы счисления в двоичную • число 111110111 из двоичной системы счисления в десятичную • число 201 из семеричной системы счисления в троичную
17	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 111101 – 1011 • 1011 * 101 2. Перевести из одной системы счисления в другую: • число 65 из десятичной системы счисления в двоичную • число 111111111 из двоичной системы счисления в десятичную • число 813 из девятеричной системы счисления в четверичную
18	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 11001101 – 1101 • 1011 * 111 2. Перевести из одной системы счисления в другую: • число 98 из десятичной системы счисления в двоичную • число 111110101 из двоичной системы счисления в десятичную • число 110101 из двоичной системы счисления в семеричную
19	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 11001101 – 11101 • 111 * 101 2. Перевести из одной системы счисления в другую: • число 155 из десятичной системы счисления в двоичную • число 111100010 из двоичной системы счисления в десятичную • число 442 из шестеричной системы счисления в пятеричную
20	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 11001101 – 10011 • 1011 * 111 2. Перевести из одной системы счисления в другую: • число 91 из десятичной системы счисления в двоичную • число 110100101 из двоичной системы счисления в десятичную • число 648 из девятеричной системы счисления в пятеричную
21	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 11001101 – 11011 • 1111 * 1001

Вариант	Задание
	2. Перевести из одной системы счисления в другую: • число 88 из десятичной системы счисления в двоичную • число 100111101 из двоичной системы счисления в десятичную • число 435 из семеричной системы счисления в девятеричную
22	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 101101 – 10110 • 1011 * 101 2. Перевести из одной системы счисления в другую: • число 115 из десятичной системы счисления в двоичную • число 1010101 из двоичной системы счисления в десятичную • число 306 из семеричной системы счисления в троичную
23	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 11001101 – 110101 • 1111 * 1001 2. Перевести из одной системы счисления в другую: • число 106 из десятичной системы счисления в двоичную • число 110101101 из двоичной системы счисления в десятичную • число 110101 из двоичной системы счисления в шестеричную
24	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 110001 – 10101 • 111 * 1001 2. Перевести из одной системы счисления в другую: • число 59 из десятичной системы счисления в двоичную • число 1101101 из двоичной системы счисления в десятичную • число 340 из семеричной системы счисления в пятеричную
25	1. Выполнить действия в двоичной системе счисления: • 11100101 + 100011 • 110011 – 10011 • 1011 * 101 2. Перевести из одной системы счисления в другую: • число 157 из десятичной системы счисления в двоичную • число 1001101011 из двоичной системы счисления в шестнадцатеричную • число 142 из девятеричной системы счисления в семеричную
26	1. Выполнить действия в двоичной системе счисления: • 10101101 + 100011 • 110101 – 10111 • 1111 * 101 2. Перевести из одной системы счисления в другую: • число 233 из десятичной системы счисления в двоичную • число 110100111 из двоичной системы счисления в десятичную • число 110101 из двоичной системы счисления в пятеричную

Вариант	Задание
27	1. Выполнить действия в двоичной системе счисления: • 10000001 + 111011 • 110101 – 11101 • 11101 * 11001 2. Перевести из одной системы счисления в другую: • число 131 из десятичной системы счисления в двоичную • число 110101101 из двоичной системы счисления в десятичную • число 351 из семеричной системы счисления в пятеричную
28	1. Выполнить действия в двоичной системе счисления: • 11010101 + 110011 • 11101 – 1011 • 111 * 10001 2. Перевести из одной системы счисления в другую: • число 253 из десятичной системы счисления в двоичную • число 1101101101 из двоичной системы счисления в десятичную • число 404 из шестеричной системы счисления в семеричную
29	1. Выполнить действия в двоичной системе счисления: • 11100101 – 100011 • 11001101 + 1111 • 10011 * 1101 2. Перевести из одной системы счисления в другую: • число 141 из десятичной системы счисления в двоичную • число 110101111 из двоичной системы счисления в десятичную • число 511 из семеричной системы счисления в пятеричную
30	1. Выполнить действия в двоичной системе счисления: • 1100 * 0111 • 11001 + 1011 • 111101 – 10011 2. Перевести из одной системы счисления в другую: • число 147 из десятичной системы счисления в двоичную • число 110100101 из двоичной системы счисления в десятичную • число 1101010 из двоичной системы счисления в девятеричную

Задание 2. Алгебра логики

1. Ознакомиться с примером выполнения задания (раздел 1.5 учебно-методического пособия).
2. Выполнить задания в соответствии с номером варианта.

Составить таблицу истинности

Вариант 1	Вариант 2	Вариант 3
$(x+\overline{y})\,\vert\,(x\sim yz)$	$((x\vee\overline{y})z)\to((x\sim z)+y)$	$((x\sim z)+y)\cdot(x\,\vert\,yz)$
Вариант 4	Вариант 5	Вариант 6
$((x\vee\overline{y})z)\to(x+y)$	$\overline{x}\to(z\sim(y+x\overline{z}))$	$(x\vee\overline{y})z\to((x\downarrow y)\,\vert\,z)$

Вариант 7	Вариант 8	Вариант 9
$\bar{x} \to (\bar{z} \sim (y + xz))$	$((xy) \mid z) \to (\bar{x} \sim y)$	$(xz \to y) \mid (xy + xz)$
Вариант 10	Вариант 11	Вариант 12
$(x + (yz)) \mid (xy)$	$(x \sim (\bar{y} + z)) \mid (xy)$	$((x \downarrow \bar{z}) \to y) \mid (xy \sim xz)$
Вариант 13	Вариант 14	Вариант 15
$(\bar{x} \vee y) \downarrow \bar{z} \to ((x + y) \mid z)$	$(x + y\bar{z}) \to (z \sim (y \downarrow (x \vee \bar{z})))$	$(\bar{y}(x \mid \bar{z})) \cdot ((\bar{x} \sim y) \mid yz)$
Вариант 16	Вариант 17	Вариант 18
$((\bar{x} \vee z) + \bar{y}) \mid (x \downarrow yz)$	$(x \sim \bar{y}) \mid (x \downarrow (\bar{y}z + \bar{x}))$	$(x\bar{y}) \mid (x \sim yz)$
Вариант 19	Вариант 20	Вариант 21
$((x \downarrow \bar{y})z) \to ((x \sim z) + y)$	$((\bar{x} \sim z) + y) \cdot (x \vee yz)$	$((x \vee \bar{y})z) \to (x \sim y)$
Вариант 22	Вариант 23	Вариант 24
$\bar{x} \to (z \sim (y + (x \downarrow \bar{z})))$	$(x + \bar{y})\bar{z} \to ((x \downarrow y) \mid z)$	$(\bar{x} \vee z) \to (\bar{z} \sim (\bar{y} + x))$
Вариант 25	Вариант 26	Вариант 27
$((x \sim y) \mid \bar{z}) \to (\bar{x} + y)$	$(xz \to y) \mid ((\bar{x} \sim y) + x\bar{z})$	$((x \vee \bar{z}) + (y + z)) \mid (x\bar{y})$
Вариант 28		
$((x + \bar{y}) \sim z) \mid ((x \sim z) + \bar{y})$		

Задание 3. Алгоритмизация и программирование

1. Ознакомиться с примером выполнения задания (раздел 1.5 учебно-методического пособия).
2. Выполнить задания в соответствии с номером варианта.

Задание 3.1. **Линейный алгоритм**

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму
1	d: = 22; w: = d – 5; r: = w * 3; d: = r + 2*d; w: = d + r; Вывод переменных: d, w, r
2	y: = 2; x: = 5 + y; r: = x + y; y: = y + 3 * r; x: = y – r; Вывод переменных: x, r, y
3	a: = 5; b: = 3 * a; d: = a + b; a: = d * 4; b: = 7 + b; Вывод переменных: a, b, d
4	x: = 75; y: = x + 15; b: = 2 * y; y: = b/3; x: = y/5; Вывод переменных: b, x, y
5	x = 33; z: = x + 7; x: = 5 + z; r: = x + z; z = r/5; Вывод переменных: x, z, r.
6	x = 12; y: = x + 3; x: = 2 * y; t: = x + y; y = t/y; Вывод переменных: x, y, t
7	k = 15; s: = k + 5; m: = 10 + s; k = k/5; s: = k + m; Вывод переменных: k, s, m
8	x = 34; y: = x – 4; x: = 2 * y; b: = x + y; y = b/y; Вывод переменных: b, x, y

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму
9	x: = -32; c: = x + 50; y: = 2 * c – x; x: = y/4; y: = c/(x + 1); Вывод переменных: x, y, c
10	z: = 2; b: = z * 3; y: = b * 5; z: = z * y; y: = y – b; Вывод переменных: b, z, y
11	x: = 2; c: = x * 5; y: = x * c; x: = x + 3 * y; c: = x – c; Вывод переменных: x, y, c
12	x: = 15; m: = x/5; y: = x + m; x: = 2 * x + y; m: = x/m; Вывод переменных: x, y, m
13	x: = 3; n: = x+7; y: = x * n; x: = x + 3 * y; n: = y + x; Вывод переменных: x, y, n
14	x: = -1; a: = x + 3; y: = x + a; x: = 2 + y; a: = y * x; Вывод переменных: x, y, a
15	x: = 48; s: = x/6; y: = x – s; x: = x – y/4; s: = s + x; Вывод переменных: x, y, s
16	x: = 10; k: = x + 5; y: = x + k; x: = 3 * y; k: = x/k; Вывод переменных: x, y, k
17	x: = 35; c: = x – 10; y: = x + c; x: = x + 2 * y; y: = y + x; Вывод переменных: x, y, c
18	x: = -14; t: = x + 34; y: = x + t; x: = 2 * y; t: = t/5; Вывод переменных: x, y, t
19	x: = 18; b: = x/3; y: = x + 5 * b; x: = y – x; b: = y/ b; Вывод переменных: x, y, b
20	x: = 4; d: = x * 2; y: = x + d; x: = x + y; d: = y * d; Вывод переменных: x, y, d
21	x: = 11; m: = x * 4; y: = x + m; x: = 2 * y; m: = m + y; Вывод переменных: x, y, m
22	x: = 8; n: = x/2; y: = x + 5 * n; x: = 4 * x + y; n: = y/n; Вывод переменных: x, y, n
23	z: = 2; c: = z + 8; y: = z * c; z: = z + y; c: = c * z; Вывод переменных: z, y, c
24	x: = 1; m: = x + 3; y: = x + m; x: = 5 * y; y: = x – y; Вывод переменных: x, y, m
25	x: = 10; a: = x + 5; y: = 2 * x + a; x: = x + 3 * y; a = y – a; Вывод переменных: x, y, a
26	x: = 3; c: = x – 1; y: = x + c; x: = x+3 * y; y: = y + x; Вывод переменных: x, y, c
27	k: = 5; m: = k * 2; n: = k + m; k: = n+2 * m; m: = k + n; Вывод переменных: k, m, n
28	a: = 15; b: = 2 * a; d: = a + b; a: = a+5; b: = d + b; Вывод переменных: a, b, d
29	x: = 11; c: = x + 9; y: = x + c; x: = 3 * y; c = y + x; Вывод переменных: x, y, c
30	x: = 27; c: = x/3; y: = x – c; x: = x + 2 * y; y: = x – y; Вывод переменных: x, y, c

Задании 3.2. **Алгоритмы ветвления**

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму
1	Задан фрагмент алгоритма: если x < z, то z = z + x, иначе x = 3 * z. В результате выполнения данного алгоритма с начальными значениями x = 55, z = 11. На экран будет выведено: x z
2	Задан фрагмент алгоритма: если a < b, то c = b – a, иначе c = 3 * (a – b), d = d + 10, В результате выполнения данного алгоритма с начальными значениями a = 20, b = 10, d = 0. На экран будет выведено: c d
3	В результате работы алгоритма: k: = 30; b: = 4; если k < b, то k = k – b, иначе b = b + k. На экран будет выведено: k b
4	В результате работы алгоритма: c: = 7; d: = 5; если c > d, то c = d * c, иначе d = d + c. На экран будет выведено: c d
5	В результате работы алгоритма: k: = 20; b: = 45; если k > b, то k = k + 3 * b, иначе b = k + 2 * b. На экран будет выведено: k b
6	В результате работы алгоритма: m: = 30; n: = 5; если m > n, то n = n – m, иначе m = m * n. На экран будет выведено: m n
7	В результате работы алгоритма: max: = -2; x: = -10; если x > max, то y = max, иначе (y = x + 25; x = max). На экран будет выведено: x y
8	В результате работы алгоритма: min: = 1; y: = 5; z: = 12; если z < min, то y = min + z, иначе (y = z – y; z = z + min). На экран будет выведено: z y
9	В результате работы алгоритма: n: = 1; m: = 5; если m > n, то r: = m + n, иначе r: = m * n; На экран будет выведено: r
10	В результате работы алгоритма: max: = 5; y: = 2; x: = 7; если x > max, то (x = max – y; y: = x – max), иначе (y: = x; x: = max). На экран будет выведено: z y
11	В результате работы алгоритма: min: = -8; d: = -9; c: = -10; если c > min, то d: = min – d, иначе (c: = min – c; d: = min + c). На экран будет выведено: c d
12	В результате работы алгоритма: max: = -1; y: = 3; x: = -0,1; если x > max, то y: = max, иначе (y = x + y; x = max). На экран будет выведено: z y
13	В результате работы алгоритма: max: = -0,01; y: = 4; x: = -0,1; если x > max, то max: = y, иначе (y: = x; x: = max). На экран будет выведено: z y

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму
14	В результате работы алгоритма: max: = -11; x: = -15; если x > max, то y: = max, иначе (y: = x + 25; x: = 9 – max;) На экран будет выведено: x y
15	В результате работы алгоритма: min: = 5; y: = 3; z: = 7; если z < min, то min = z, иначе (y: = z + y; z = min). На экран будет выведено: z y
16	Задан фрагмент алгоритма: если a < b, то a = a/b, иначе b = a * b, В результате выполнения данного алгоритма с начальными значениями: a = -45, b = -15, переменные примут значения: a b
17	Задан фрагмент алгоритма: если a < b, то c = b – a, иначе d = 5 * (a – b). В результате выполнения данного алгоритма с начальными значениями: a = 25, b = 50 переменные примут значения: c d
18	В результате работы алгоритма: max: = -15; x: = -30; если x > max, то y: = max + x, иначе (y: = x + 50; x: = max – x;), На экран будет выведено: x y
19	В результате работы алгоритма: min: = 10; y: = -25; z: = 20; если z < min, то min = z + y, иначе (y = z – y; z = z + min). На экран будет выведено: z y
20	В результате работы алгоритма: a: = 15; b: = 17; если b < a, то a: = b – 7, иначе b: = a + 5; На экран будет выведено: На экран будет выведено: a b
21	В результате работы алгоритма: x: = 5; y: = 7; если x > y, то x: = x – 2, иначе y: = x + y; На экран будет выведено: x y
22	В результате работы алгоритма: x: = 10; y: = 30; если x > y, то x: = x + 15, иначе y: = y – x; На экран будет выведено: x y
23	Задан фрагмент алгоритма: если a < b, то c = b-a, иначе (c = 2 * (a – b), d = d + 1). В результате выполнения данного алгоритма с начальными значениями: a = 8, b = 3, d = 0. На экран будет выведено: c d
24	В результате работы алгоритма: max: = -1; x: = -10; если x > max, то y: = max, иначе (y: = x + 15; x: = 10 – max) На экран будет выведено: x y
25	Задан фрагмент алгоритма: если x < z, то z = z – x, иначе x = 3 * z; В результате выполнения данного алгоритма с начальными значениями: x = 5, z = 9. На экран будет выведено: x z
26	В результате работы алгоритма: a: = 55; b: = 27; если b < a, то a: = b – 7, иначе b: = a + 10. На экран будет выведено: a b

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму
27	В результате работы алгоритма: x: = -17; y: = -11; если x > y, то x = x + 30, иначе y = y + 14. На экран будет выведено: x y
28	Задан фрагмент алгоритма: если a < b, то c = b – a, иначе d = 3 * (a – b). В результате выполнения данного алгоритма с начальными значениями: a = 10, b = 5. На экран будет выведено: c d
29	В результате работы алгоритма x = -20; y = -10; если x > y, то x = x + 15, иначе y = y + 25. На экран будет выведено: x y
30	Задан фрагмент алгоритма: если a < b, то a = b + a, иначе b = a * b, В результате выполнения данного алгоритма с начальными значениями: a = 4, b = 5. На экран будет выведено: a b

Задание 3.3. **Алгоритмы. Циклы**

Переписать программу задания 3.3 с оператором цикла с постусловием.

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму

1	s: = 0; k: = 1;
	Начало цикла
	для i от 1 до 5 выполнить
	k: = k * 2; s: = s + k;
	конец цикла; Вывод k, s

2	m: = 15; n: = 75; k: = 0; Начало цикла **пока** n> = m выполнить n: = n – m; k: = k + 1; конец цикла; Вывод n, k
3	s: = 0; k: = 1; Начало цикла **для** i от 1 до 6 выполнить k:=k * 2; s: = s + i; конец цикла; Вывод k, s
4	b: = 10; d: = 75; k: = 0; Начало цикла **пока** d > b выполнить d: = d – b; k: = k + 1; конец цикла. Вывод d, k

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму
5	s: = 0; p: = 1; Начало цикла **для** i от 1 до 4 выполнить p: = p * 3; s: = s + p; конец цикла; Вывод p, s
6	b: = 21; d: = 4; k: = 0; Начало цикла **пока** d < b выполнить d: = d + 2; k: = k + 1; конец цикла. Вывод d, k
7	k: = 0; p: = 1; Начало цикла **для** i от 1 до 6 выполнить k: = k + 3; p: = p * i; конец цикла; Вывод p, k
8	b: = 10; d: = 40; k: = 0; Начало цикла **пока** d > = b выполнить d: = d – 5; k: = k + 1; конец цикла; Вывод d, k
9	s: = 0; k: = 0; Начало цикла **для** i от 1 до 5 выполнить k: = k + 3; s: = s + k; конец цикла; Вывод k, s
10	b: = 7; d: = 65; k: = 0; Начало цикла **пока** d > b выполнить d: = d – b; k: = k + 1; конец цикла; Вывод d, k
11	s: = 0; k: = 0; Начало цикла **для** i от 1 до 6 выполнить k: = k + 2; s: = s + k; конец цикла; Вывод k, s
12	b: = 30; d: = 93; k: = 0; Начало цикла **пока** b < d выполнить b: = b + 5; k: = k + 1; конец цикла; Вывод b,k
13	x: = 10; p: = 1; Начало цикла **для** i от 1 до 5 выполнить x: = x + 5; p: = p*i; конец цикла. Вывод p, x
14	b: = 5; d: = 70; k: = 0; Начало цикла **пока** b < d выполнить d: = d – b; k: = k + 1; конец цикла; Вывод k, d

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму

15	y: = 3; s: = 0;
	Начало цикла
	для i от 1 до 7 выполнить
	y: = y + 5; s: = s + i;
	конец цикла: Вывод y, s

16	b: = 3; d: = 77; k: = 0; Начало цикла **пока** b < = d выполнить b: = b + 10; d: = d – b; k: = k + 1; конец цикла; Вывод d, k

17	s: = 0; k: = 1;
	Начало цикла
	для i от 1 до 3 выполнить
	k: = k * 2; s: = s + k;
	конец цикла; Вывод s, k

18	x: = 100; y: = 50; k:=0;
	Начало цикла
	пока y < x выполнить
	y: = y + 10; k: = k + 1;
	конец цикла; Вывод y, k

19	s: = 0; k: = 100; Начало цикла **для** i от 1 до 5 выполнить k: = k – 2 * i; s: = s + i; конец цикла; Вывод k, s
20	b: = 13; d: = 65; Начало цикла **пока** b< = d выполнить k: = k + 1; d: = d – b; конец цикла; Вывод k, d
21	s: = 0; k: = 7; Начало цикла **для** i от 1 до 3 выполнить k: = k – 2; s: = s + k; конец цикла; Вывод k, s

22	b: = 3; d: = 33;
	Начало цикла
	пока b < d выполнить
	b: = b + 10; k: = k + 1;
	конец цикла; Вывод k, b

Вариант	Задание 1. Выполнить словесный алгоритм. Записать результат 2. Составить блок-схему и написать программу по алгоритму
23	s: = 50; k: = 0; Начало цикла **для** i от 1 до 5 выполнить k: = k + 2; s: = s – k; конец цикла; Вывод k, s
24	b: = 4; d: = 9; Начало цикла **пока** d> = b выполнить d: = d – b; k: = k + 1; конец цикла; Вывод d, k
25	s: = 20; k: = 1; Начало цикла **для** i от 1 до 3 выполнить k: = k * 2; s: = s – k; конец цикла; Вывод k, s
26	b: = 11; d: = 45; Начало цикла **пока** d > b выполнить d: = d – b; k: = k + 1; конец цикла. Вывод k, d
27	Дано: k: = 5; p: = 1; Начало цикла **для** i от 1 до 4 выполнить k: = k + 2; p: = p * k; конец цикла; Вывод p, k
28	b: = 6; d: = 55; Начало цикла **пока** d > = b выполнить d: = d – 5; k: = k + 1; конец цикла. Вывод p, k
29	k: = 34; s: = 0; Начало цикла **для** i от 1 до3 выполнить k: = k – 8; s: = s + k; конец цикла; Вывод k, s
30	b: = 10; d: = 40; Начало цикла **пока** d >= b выполнить d: = d – b; k: = k + 1; конец цикла; Вывод d, k

Задание 3.4. **Одномерные массивы**

Вариант	Задание
1	Найти среднеарифметическое положительных элементов
2	Найти произведение чётных элементов
3	Поменять местами первый и последний элемент
4	Найти среднеарифметическое нечётных элементов
5	Найти среднеарифметическое отрицательных элементов
6	Переписать положительные элементы в другой массив
7	Найти количество чётных элементов
8	Найти сумму положительных элементов
9	Найти сумму нечётных элементов
10	Если последний элемент массива нечётный, то увеличить его вдвое, иначе уменьшить его на единицу
11	Поменять местами последний и предпоследний элементы
12	Найти сумму нечётных элементов
13	Найти сумму элементов с нечётными номерами
14	Переписать чётные элементы в другой одномерный массив
15	Если третий элементов нечётный, то увеличить его вдвое, иначе уменьшить его на единицу.
16	Найти произведение чётных элементов
17	Найти номер первого отрицательного элемента
18	Если второй элемент массива чётный, то увеличить его на 10, иначе заменить его на последний элемент
19	Подсчитать количество чётных элементов
20	Переписать положительные элементы в другой массив
21	Подсчитать сумму нечётных элементов
22	Если третий элемент массива чётный, то поменять его местами с первым, иначе приравнять его к последнему элементу
23	Если второй элемент нечётный, то увеличить его на 5, иначе прибавить к нему последний элемент
24	Найти количество положительных элементов
25	Если последний элемент отрицательный, то взять модуль этого элемента, иначе прибавить к нему второй элемент

Вариант	Задание
26	Найти произведение элементов с чётными номерами
27	Если третий элемент чётный, то увеличить его втрое, иначе прибавить к нему первый элемент
28	Найти количество отрицательных элементов
29	Подсчитать среднеарифметическое нечётных элементов
30	Если первый элемент положительный, то поменять местами с последним, иначе прибавить к нему третий элемент

Глоссарий

Алгебра логики – изучает высказывания, рассматриваемые со стороны их логических значений (истинности или ложности), и логические операции над ними.

Алгоритм – последовательность арифметических, логических и прочих операций, необходимых для выполнения на ЭВМ.

Байт – основная единица измерения информации в ЭВМ.

Бит – минимальная двоичная единица количества информации (то, что необходимо для различения двух равновероятных сообщений).

Блок-схема – графическое представление алгоритма.

Ветвление – алгоритм может пойти по одной из двух возможных ветвей. Происходит выбор одного из путей работы алгоритма.

Высказывание – имеющее смысл языковое выражение, относительно которого можно утверждать, что оно либо истинно, либо ложно.

Данные – информация, представленная в формализованном виде и предназначенная для обработки ее техническими средствами.

Интерпретатор (англ. *interpreter* – переводчик) – программа, которая переводит каждый оператор программы, записанной на алгоритмическом языке, на машинный язык и выполняет программу построчно, что позволяет сразу редактировать и исправлять ошибки.

Инструментальное программное обеспечение – предназначено для разработки новых программ и программных комплексов.

Информатика – наука, изучающая способы создания, хранения, обработки и передачи информации с помощью компьютера, а также принципы функционирования компьютеров и методы управления ими.

Информация – сведения, снимающие неопределенность об окружающем мире, которые являются объектом хранения, преобразования, передачи и использования.

Количеством информации – называют числовую характеристику сигнала, отражающую ту степень неопределенности (неполноту знаний), которая исчезает после получения сообщения в виде данного сигнала.

Компилятор (англ. *compiler* – составитель) – программа, которая читает всю программу, записанную на алгоритмическом языке, целиком, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется.

Линейный алгоритм – последовательное выполнение операций. В этом алгоритме не предусмотрены проверки условий или повторений.

Логическая операция – построение из высказываний (или из высказывания) нового высказывания.

Логическими связками – называются знаки логических операций.

Оперативная память (ОЗУ) – память, часть системы памяти ЭВМ, в которую процессор может обратиться за одну операцию. Энергозависимая память для кратковременного хранения программ и данных во время работы компьютера.

Операционная система, ОС (англ. *operating system*) – базовый комплекс компьютерных программ, обеспечивающий интерфейс с пользователем, управление аппаратными средствами компьютера, работу с файлами, ввод и вывод данных, а также выполнение прикладных программ и утилит.

Постоянное запоминающее устройство (ПЗУ) – энергонезависимая память, в которой хранятся программы для загрузки компьютера, библиотеки. В момент включения компьютера стартовый адрес указывает на ПЗУ для проверки исправности ПК и первоначальной загрузки.

Прикладное программное обеспечение (ППО) – комплекс программ для решения задач определённого класса конкретной предметной области.

Программное обеспечение (software) – совокупность программных систем, которые предназначены для обработки самой разнообразной информации с самыми различными целями.

Сведения – знания, выраженные в сигналах, сообщениях, известиях, уведомлениях и т. д.

Сигнал – любой процесс, несущий информацию.

Системное программное обеспечение – совокупность программ для обеспечения работы компьютера.

Система счисления – способ записи чисел с помощью заданного набора специальных знаков (цифр).

Сообщение – информация, представленная в определенной форме и предназначенная для передачи.

Таблица истинности – используется в алгебре логики для описания логических операций.

Транслятор – программа, которая преобразует исходную программу (написанную на одном из языков высокого уровня) в программу, состоящую из машинных команд.

Техническое обеспечение **(hardware)** – технические средства реализации информационных процессов.

Файл – наименование (имя) совокупности данных, в том числе документа на машиночитаемом носителе.

Файловая система – система хранения файлов и организации каталогов.

Циклический алгоритм – многократное повторение одной или нескольких операций в зависимости от условия задачи.

Энтропия – мера неопределенности информации.

Языки программирования – формальные языки, специально созданные для общения человека с вычислительной машиной.

Библиографический список

1. Акулов, О.А. Информатика: базовый курс : учебник для вузов / О.А. Акулов, Н.В. Медведев. – М. : Омега-Л, 2007. – 557 с.
2. Бройдо, В.Л. Вычислительные системы, сети и телекоммуникации : учебник для вузов / В.Л. Бройдо. – СПб. : Питер, 2006. – 703 с.
3. Интернет : самоучитель / А. Денисова [и др.]. – СПб. : Питер, 2004. – 368 с.
4. Информатика: базовый курс : учеб. пособие для вузов / под ред. С.В. Симонович [и др.]. – СПб. : Питер, 2005. – 639 с.
5. Коньков, К.А. Устройство и функционирование ОС Windows: практикум по курсу «Операционные системы» : учеб. пособие / К.А. Коньков. – М. : Интернет-университет информационных технологий : БИНОМ : Лаб. знаний, 2008. – 207 с.
6. Левин, В.И. История информационных технологий : учеб. пособие / В.И. Левин. – М. : Интернет-университет информационных технологий : БИНОМ : Лаб. знаний, 2007. – 335 с.
7. Информатика : учеб. пособие для вузов / А.В. Могилёв [и др.]. – М. : Академия, 2004. – 842 с.
8. Телекоммуникации. Руководство для начинающих / М. Мур [и др.]. – СПб. : БХВ-Петербург, 2005. – 624 с.
9. Олифер, В.Г. Компьютерные сети. Принципы, технологии, протоколы : учебник для вузов / В.Г. Олифер, Н.А. Олифер. – СПб. : Питер, 2004. – 864 с.
10. Панкратова, Л.П. Контроль знаний по информатике : тесты, контрольные задания, экзаменационные вопросы, компьютерные проекты / Л.П. Панкратова, Е.Н. Чулак. – СПб. : БХВ-Петербург, 2004. – 440 с.
11. Сырецкий, Г.А. Информатика: фундаментальный курс : учебник для вузов. Т. 1. Основы информационной и вычислительной техники / Г.А. Сырецкий. – СПб. : БХВ-Петербург, 2005. – 822 с.
12. Шапорев, С.Д. Информатика: теоретический курс и практические занятия : учебник для вузов / С.Д. Шапорев. – СПб. : БХВ-Петербург, 2008. – 469 с.

Содержание

Учебное издание

Панюкова Екатерина Владимировна
Егорова Эльвира Валентиновна

ИНФОРМАТИКА

Учебно-методическое пособие

Редактор *Е.Ю. Жданова*
Технический редактор *З.М. Малявина*
Компьютерная верстка: *Л.В. Сызганцева*
Дизайн обложки: *Г.В. Карасева*

Подписано в печать 20.12.2012. Формат 60×84/16.
Печать оперативная. Усл. п. л. 8,6.
Тираж 100 экз. Заказ № 1-85-11.

Издательство Тольяттинского государственного университета
445667, г. Тольятти, ул. Белорусская, 14