

**Мансуров К.Т.**

# **Информатика для программиста**

**ВВЕДЕНИЕ В СПЕЦИАЛЬНОСТЬ**

ББК 32.97

М - 23

Утвержден в качестве учебника учебно-методическим советом ОшТУ.

Рецензенты:

доктор физико-математических наук, профессор Сопуев А.С.

кандидат педагогических наук, доцент Назаров М.М.

М-23 Мансуров К.Т. **Информатика для программиста. Введение в специальность.**

Учебник. Ошский Технологический Университет, г. Ош, 2002 г., 181 с.: ил.

ISBN 9967-21-148-2

Настоящий учебник предназначен для студентов младших курсов специальности "Программное обеспечение вычислительной техники и автоматизированных систем", а также специальности "Вычислительные машины, комплексы, системы и сети", хотя многие главы этой книги могут с успехом использовать студенты и других специальностей при изучении информатики.

Книга является введением в специальность для программистов. Она вводит читателя в круг тех идей, понятий, принципов и методов, на которых зиждется современное программирование и информационные технологии.

Приведены многочисленные задачи и их решения, а также сведения по численным методам, алгоритмах их реализации и пакетах программ.

Книга предназначена для студентов ВУЗов, специализирующихся в области программного обеспечения вычислительной техники.

М 2404000000-02

ББК 32.97

ISBN 9967-21-148-2

© Мансуров К.Т., 2002

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И КУЛЬТУРЫ  
КЫРГЫЗСКОЙ РЕСПУБЛИКИ

**ОШСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**

кафедра "Вычислительной техники"

**Мансуров К.Т.**

# Информатика для программиста

введение в специальность

Ош –2002 г.

## Предисловие

*Мы вступаем в этап, когда стоящие перед нами проблемы невозможно решить без применения компьютеров. Я не испытываю страха перед компьютерами. Меня страшит их отсутствие.*

Айзек Азимов

Настоящая книга возникла в результате переработки лекций, которые я читал на протяжении ряда лет студентам Ошского технологического университета.

Она предназначена в первую очередь для студентов младших курсов специальности "Программное обеспечение вычислительной техники и автоматизированных систем", а также специальности "Вычислительные машины, комплексы, системы и сети", хотя многие главы этой книги могут с успехом использовать студенты и других специальностей при изучении информатики.

Книга является "как бы" введением в специальность для программистов. Она вводит читателя в круг тех идей, понятий, принципов и методов, на которых зиждется современное программирование и информационные технологии.

Изучив эту книгу, читатель будет вполне подготовлен для усвоения более специальных дисциплин по избранной специальности, в частности таких как:

- алгоритмические языки и программирование;
- технология программирования;
- системное программирование;
- структуры данных;
- системы управления базами данных;
- операционные системы;
- электроника и системотехника;
- организация ЭВМ и комплексов и др.

Автор помнит, как в свое время он очень нуждался в подобной книге. Представьте себе первокурсника, семнадцатилетнего юношу (ах, где моя молодость!), который хочет узнать "сразу обо – всём". Как работает компьютер? Что у него "внутри" происходит, когда он вычисляет? Почему компьютер никогда не ошибается? Как "он понимает", что нужно делать? И много других почему! Хотелось иметь книгу, в которой просто, доходчиво и понятно, почти "на пальцах" объяснялись бы подобного рода вопросы, излагались бы основные идеи и приемы разработки программ, приводились бы примеры и т.д. Конечно, истинное и глубокое понимание придет к читателю на старших курсах, после прочтения уймы книг, изучения вышеупомянутых дисциплин,

но все – таки хотелось бы уже сейчас иметь некоторое представление обо всем этом.

В течение многих лет я вынашивал идею написания подобной книги и наконец "созрел"! Не знаю, удалось ли в полной мере автору реализовать свой замысел (об этом судить читателю!), но основная цель книги заключается в том, чтобы помочь пытливому (именно пытливому!) студенту усвоить азы своей будущей профессии.

Конечно, объять необъятное невозможно, поэтому материал книги подвергся тщательному отбору и здесь представлен тот необходимый минимум, который должен освоить студент. Это та база, опираясь на которую можно в дальнейшем расширять и углублять свои знания, слушая лекции и используя специальную литературу.

К сожалению, современные молодые люди очень мало читают, довольствуясь лишь посещением занятий. На мой взгляд это тенденция и очень опасная тенденция! Но стоп! Боюсь показаться в глазах читателя эдаким брюзжащим стариком, который вечно ругает молодежь (вот в наше время ...!). Как говорится, это философский вопрос, а моя задача в другом. Ограничусь просто советом, читайте больше!

Например, на формирование моих взглядов как программиста огромное влияние оказали труды таких ученых как Лебедев В.Н., Глушков В.М, Ершов А.П. Из зарубежных авторов сильнейшее влияние на меня оказали Дейкстра Э., Донован Дж., Мэдник С., Вирт Н. и, конечно же, Кнут Д. Не будучи знакомым с их книгами, я не состоялся бы как специалист! Так что читайте, читайте и ещё раз читайте, не жалейте на это ни сил, ни времени, ни средств!

Теперь немного о том, как и в какой последовательности читать эту книгу. Студентам специальности ПОВТАС и ВМКСС рекомендую прочитать всю книгу, от начала до конца. Студентам других специальностей ("чайникам") главу I, главу II (§2.1, §2.2) можно пропустить, поскольку они носят теоретический и немного специальный характер.

Материал главы, посвященной текстовому процессору Word – 97, носит более практический характер и разбит на отдельные уроки. В начале каждого урока даются краткие теоретические сведения, необходимые для работы, а также многочисленные примеры. В конце идет практическая часть урока, где студент должен выполнить одно или несколько заданий. На каждый урок отводится в среднем 2 - 4 часа аудиторных занятий в компьютерном классе.

Можно и нужно читать эту книгу непосредственно сидя за компьютером!

Небольшое замечание по поводу работы с книгой за компьютером – всюду в тексте, когда говорится щелкните клавишей мыши, подразумевается левая клавиша. Когда речь идет о правой клавише, то это особо оговаривается в тексте.

В заключение хочу выразить благодарность моим коллегам, преподавателям кафедры "Вычислительная техника" ОшТУ, с которыми я неоднократно обсуждал содержание книги, в частности Джуманову Д.А., Игамбердиеву Р, Худайбердиеву Д., которые проверили все программы этой книги на ком-

пьютере.

Спасибо моим детям, которые безропотно уступали мне компьютер, когда я работал над этой книгой.

Спасибо моей жене, которая мужественно терпела мои ночные бдения за компьютером, вместо того, чтобы заниматься более "полезным" (с её точки зрения) делом!

Спасибо моим многочисленным студентам, которые были "подопытными кроликами" во время моих "методических экспериментов" на лекциях.

Мансуров К.Т.

## Введение

*Кончай разговоры, пошли к лошадям.*  
Английская поговорка

Становление информатики как научной дисциплины относится к 60-годам прошлого столетия. В самом общем смысле под информатикой понимают фундаментальную естественную науку, изучающую процессы передачи, накопления и обработки информации.

Термин "Informatique" был введен французами на рубеже 60-70-х годов.

Но еще раньше американцами был введен в употребление термин "Computer Science" для обозначения науки о преобразовании информации, базирующейся на вычислительной технике.

В настоящее время термины "Informatique" и "Computer Science" употребляются в эквивалентном смысле.

В нашей стране для обозначения новой научной дисциплины применялись термины "Вычислительные науки" или "Вычислительное дело", использование же термина "Информатика" сдерживалось употреблением его в области, связанной с документалистикой. Новое толкование этого термина относится к 1976г., когда появился русский перевод книги Ф. Бауэра и Г.Гооза "Информатика".

Известно широкое определение информатики, данное Международным конгрессом в Японии в 1978г.: "Понятие информатики охватывает области, связанные с разработкой, созданием, использованием и материально-техническим обслуживанием систем обработки информации, включая машины, оборудование, математическое обеспечение, организационные аспекты, а также комплекс промышленного, коммерческого, административного, социального и политического воздействия". В этом определении можно выделить то главное, что и составляет основу современного содержания информатики, а именно компьютеры и компьютерную обработку информации.

Исходя из такого толкования ядра информатики можно сказать, что его содержание определяют три неразрывно связанные между собой части: алгоритмические, программные и технические средства. Элементы этих составных частей информатики и рассматриваются в настоящем учебнике.

### **Информация и информатика.**

Информация об окружающем мире необходима человеку так же, как воздух, вода, тепло. Представьте себе, что Вы оказались в замкнутом пространстве без света и звука, с абсолютно гладкими поверхностями. Вы почувст-

вуете себя очень неуютно. Отсутствие информации о времени, пространстве, окружающей действительности противоестественно для человека. Человек видит, слышит, осязает, чувствует запахи. Органы чувств служат человеку для получения информации.

Информация – это сообщение о состоянии и свойствах объекта, явления, процесса. Информация преобразуется уникальным устройством- человеческим мозгом. Ребенок, впервые увидевший огонь, не боится его. Но обжегшись, запоминает на всю жизнь – с огнем играть нельзя. Значит, мозг человека не только преобразует информацию, но и запоминает – хранит ее. Человечество за тысячелетия своего существования накопило огромное количество информации. Мозг человека не в состоянии хранить такой объем ее и без искажения передавать другим людям. Поэтому для хранения и накопления информации с незапамятных времен использовались природные средства: рисунки на стенах пещер, скалах. Носители информации непрерывно совершенствовались, так постепенно появились: пергамент, папирус, береста, бумага, перфорационные носители информации ( перфокарты, перфоленты), фото пленка, магнитные носители ( ленты, карты, диски и т.д.).

Первобытный человек, создав первые примитивные орудия труда, положил начало эпохе механизмов, увеличивающих физическую силу человека. В середине XX столетия были созданы первые электронные вычислительные машины, предназначенные для увеличения интеллектуальной мощи человека. Появление ЭВМ не случайно. Лавинообразно растут информационные потоки. Особенно это характерно для промышленности, управления и науки. Общая сумма человеческих знаний в XVIII веке удваивалась каждые 50 лет, к 1950г. –каждые 10 лет, к 1970г. – 5 лет, к 1985г. –каждые 2-3 года. До XX века основным предметом труда были материальные объекты, экономическая мощь государства измерялась его материальными ресурсами, теперь основным предметом труда становится информация. Человек не в состоянии без электронных помощников навести порядок в информационном хозяйстве.

Взросшая роль информации и уровень развития средств обработки информации привели к появлению понятия "информационные ресурсы". Информационные ресурсы – информация, используемая на производстве, в технике, в управлении обществом, специально организованная и обрабатываемая на компьютере. Процесс компьютеризации общества привел к появлению нового термина – пользователь. Пользователем будем называть человека любой профессии, который непосредственно взаимодействует с компьютером.

### **Формы и виды представления информации**

Выделяют две формы представления информации – непрерывную и дискретную. Непрерывная форма – это величина, характеризующая процесс, не

имеющий перерывов или промежутков, например: температура человека, скорость перемещения автомобиля за определенное время на участке пути без остановок.

Дискретная форма представления информации – это последовательность символов, характеризующая прерывистую, изменяющуюся величину. Например, рабочий за первый час смены обработал на станке 40 деталей, за второй – 45, за третий 38 и т.д.

Часто возникает необходимость представить информацию в форме отличной от привычной. Так, для передачи информации на расстояние изобретен телеграфный код Морзе, в которой буквы и цифры закодированы с помощью коротких и длинных импульсов (точка, тире). Почтовый индекс – закодированный адрес.

Электронные элементы, применяемые в компьютерах, имеют два состояния – есть импульс или нет импульса, поэтому вся информация для компьютеров кодируется в двоичной системе счисления. Любой символ (цифра, буква, знак) получает кодированное обозначение с помощью цифр 1 и 0, составляющих основу двоичной системы счисления.

Рассмотрим основные единицы измерения информации. Наименьшей единицей информации является бит (bit), принимающий только значение 1 или 0. Более крупная единица информации – байт (byte). Он состоит из восьми бит. Байт – основная единица количества информации, используется для образования еще более крупных единиц информации.

# Глава I. Теоретические основы информатики

*Записи были сделаны... римскими цифрами, отнюдь не арабскими, которые считались легкомысленным нововведением, неприличным в деловой книге.*

Неизвестный автор, 1494 г.

## §1.1. Системы счисления

**Система счисления** – совокупность правил и приемов для записи чисел с помощью различных знаков или символов.

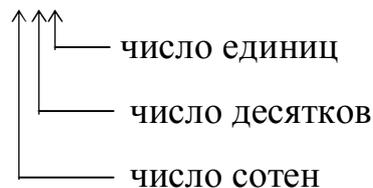
Системы счисления делятся на позиционные и непозиционные.

**Непозиционная система счисления** – система, для которой значение символа не зависит от его положения в числе. Примером непозиционной системы счисления может служить римская система счисления. Используется набор знаков I, V, X, L, C, D, M, ...

Например: числа IV и VI. Значение символа V одно и то же и равно пяти.

**Позиционная система счисления** – система, которая использует фиксированный набор знаков (символов) для записи числа и значение этого знака (символа) зависит от его положения в числе.

Например 654 это не одно и то же, что 645



Любая позиционная система счисления характеризуется основанием или базисом.

**Базисом (основанием)**  $q$  позиционной системы счисления является количество знаков или символов используемых для записи числа в данной системе счисления.

Например, в десятичной системе счисления  $q=10$ , используются знаки или цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Для любой позиционной системы счисления имеет место равенство:

$$A_{(q)} = a_n q^n + a_{n-1} q^{n-1} + \dots + a_1 q^1 + a_0 q^0 + a_{-1} q^{-1} + \dots + a_{-m} q^{-m} \quad (1.1.1)$$

где  $A_{(q)}$  - произвольное число в  $q$ -ичной системе счисления,  $q$  - базис данной системы счисления,  $a_i$  - знаки или символы данной системы счисления.

Для десятичной системы имеем:

$$q=10, \quad a_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$A_{(10)} = a_n 10^n + a_{n-1} 10^{n-1} + \dots + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + \dots + a_{-m} 10^{-m} \quad (1.1.2)$$

или в сокращенной записи:

$$A_{(10)} = \sum_{i=-m}^n a_i 10^i \quad (1.1.3)$$

На практике используют условную, сокращенную запись числа:

$$A_{(q)} = a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-m} \quad (1.1.4)$$

Пример:

$$A_{(10)} = 654 = 6 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0$$

$$A_{(10)} = 358,124 = 3 \cdot 10^2 + 5 \cdot 10^1 + 8 \cdot 10^0 + 1 \cdot 10^{-1} + 2 \cdot 10^{-2} + 4 \cdot 10^{-3}$$

Существуют и другие системы счисления:

### 1. Двоичная система счисления

$$q=2, \quad a_i = \{0,1\}$$

$$A_{(2)} = \sum_{i=-m}^n a_i 10^i \quad (1.1.5)$$

Правила образования чисел как в десятичной системе: вес  $k$ -й позиции числа равен  $q^k$ . Для двоичной системы  $2^k$ .

$$0_{(10)} = 0_{(2)}$$

$$1_{(10)} = 1_{(2)}$$

$$2_{(10)} = 10_{(2)}$$

$$3_{(10)} = 11_{(2)}$$

$$4_{(10)} = 100_{(2)}$$

$$\text{Число } 14_{(10)} = 1110_{(2)} = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 4 + 2 + 0 = 14_{(10)}$$

### 2. Восьмеричная система счисления

$$q=8, \quad a_i = \{0,1,2,3,4,5,6,7\}$$

$$A_{(10)} = 118,375_{(10)} = 166,3_{(8)} = 1 \cdot 8^2 + 6 \cdot 8^1 + 6 \cdot 8^0 + 3 \cdot 8^{-1} = 64 + 48 + 6 + \frac{3}{8} = 118,375_{(10)}$$

### 3. 16 – ричная система счисления

$$q=16, \quad a_i = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$$

$$100_{(10)} = 64_{(16)} = 6 \cdot 16^1 + 4 \cdot 16^0 = 100_{(10)}$$

Разрядом называется позиция одной цифры в числе. Вес разряда  $p_i$  в позиционной системе счисления это отношение  $p_i = \frac{q^i}{q^0} = q^i$ , где  $i$  - номер разряда справа налево.

Если разряд  $p_k$  имеет вес  $q^k$ , то следующий старший разряд (слева) имеет вес  $p_{k+1} = q^{k+1}$ , а следующий младший разряд имеет вес  $p_{k-1} = q^{k-1}$ .

Такая взаимосвязь между разрядами приводит к необходимости передачи информации между разрядами. Если в текущем разряде накопилось единиц больше или равное  $q$ , должна происходить передача единицы в соседний старший разряд. Этот процесс называется переносом. Передача единицы из старшего разряда в младший называется заемом.

Длиной числа называется количество разрядов необходимым для записи числа. В техническом аспекте длина числа интерпретируется как длина разрядной сетки процессора. В большинстве компьютеров длина разрядной сетки фиксирована. Если длина разрядной сетки задана, то это ограничивает максимальное и минимальное по модулю число, представимое в компьютере.

Пусть длина разрядной сетки равна  $n$  ( $n > 0$ , целое). Тогда

$$A_{(q)\max} = q^n - 1; A_{(q)\min} = -(q^n - 1) \quad (1.1.6)$$

Отсюда диапазон представления чисел в заданной системе счисления и для заданной разрядной сетке это интервал числовой оси, заключенный между  $A_{(q)\max}$  и  $A_{(q)\min}$ , т.е.

$$A_{(q)\min} \leq \text{ДП} \leq A_{(q)\max} \quad (1.1.7)$$

**Таблица соответствия десятичных чисел в других системах счисления**

Десятичные числа	Их эквиваленты в других системах		
	$q=2$	$q=8$	$q=16$
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	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

### Перевод чисел из одной системы счисления в другую.

Пусть необходимо перевести число в системе счисления с основанием  $q$  в число в системе счисления с основанием  $r$ .

$$A_{(q)} = a_n q^n + a_{n-1} q^{n-1} + \dots + a_1 q^1 + a_0 q^0 + a_{-1} q^{-1} + \dots + a_{-m} q^{-m} = \\ = b_k r^k + \dots + b_1 r^1 + b_0 r^0 + b_{-1} r^{-1} + \dots + b_{-s} r^{-s} = A_{(r)} \quad (1.1.8)$$

Задачу перевода числа из системы счисления  $q$  в систему счисления  $r$  можно представить в общем виде как задачу определения коэффициентов нового ряда  $b_j$

$$0 \leq a_i \leq q - 1, \quad i \in [-m, n] \\ 0 \leq b_j \leq r - 1, \quad j \in [-s, k] \quad (1.1.9)$$

Например,

$$56_{(10)} = 5 \cdot 10^1 + 6 \cdot 10^0 = 111000_{(2)} = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 32 + 16 + 8 = 56_{(10)}$$

#### Правило перевода целых чисел

Для перевода целого числа в системе счисления с основанием  $q$  в число в системе с основанием  $r$  необходимо последовательно делить это число и получаемые затем целые части частных на основание  $r$  новой системы счисления до тех пор, пока очередная полученная целая часть частного не станет меньше  $r$ . Эта вновь полученная целая часть определит старшую цифру числа в новой системе счисления, а все предыдущие остатки – младшие цифры вновь полученного числа. Пример: перевести в двоичную систему счисления число  $X_{(10)} = 98_{(10)}$

$$\begin{array}{r} 98 \overline{) 2} \\ 98 \overline{) 49} \overline{) 2} \\ \hline 0 \quad 48 \overline{) 24} \overline{) 2} \\ \hline \quad 1 \quad 24 \overline{) 12} \overline{) 2} \\ \hline \quad \quad 0 \quad 12 \overline{) 6} \overline{) 2} \\ \hline \quad \quad \quad 0 \quad 6 \overline{) 3} \overline{) 2} \\ \hline \quad \quad \quad \quad 0 \quad 2 \overline{) 1} \\ \hline \quad \quad \quad \quad \quad 1 \end{array}$$

Получим  $X_{(10)} = 98_{(10)} = 1100010_{(2)} = X_{(2)}$

#### Правило перевода правильных дробей.

Для перевода правильной дроби из одной системы счисления в другую надо её и получающиеся затем дробные части произведений умножать на основание новой системы счисления до тех пор, пока в новой дроби не будет нужного количества цифр, которая определяется требуемой точностью представления дроби. Правильная дробь в новой системе счисления записывается из целых частей произведений, получающихся при последовательном умножении, причем первая целая часть будет старшей цифрой новой дроби.

Пример:  $X_{(10)} = 0,375$  перевести в двоичную систему с 4-мя знаками по-

сле запятой.

$$\begin{aligned} & 0,375 \\ & \times \frac{2}{0,750} \\ & \times \frac{2}{1,500} \\ & \times \frac{2}{1,000} \\ & \times \frac{2}{0,000} \end{aligned}$$

$$X_{(2)} = 0,0110_{(2)} = 0,375_{(10)}$$

### **Правило перевода неправильных дробей.**

При переводе неправильных дробей отдельно переводят целую и дробную части числа по правилам, приведенными выше.

$$X_{(10)} = 98,375_{(10)} = 1100010,0110_{(2)}$$

### **Представление чисел в памяти компьютера**

Существуют две формы представления чисел: естественная и нормальная.

Естественной называется такая форма записи числа, которая в неявном виде реализует формулу:

$$X = \sum_{i=-m}^n a_i q^i \quad (1.1.10)$$

При этом число записывают только при помощи набора цифр  $a_i$ , без явного указания их весов и знаков сложения между ними.

Отсчет весов ведется от запятой, которая фиксируется между целой и дробной частью числа. Запись числа имеет вид:

$$a_n a_{n-1} \dots a_1 a_0, a_{-1} \dots a_{-m} \quad (125 = 1 \cdot 10^2 + 2 \cdot 10^1 + 5 \cdot 10^0)$$

Это формой мы обычно пользуемся.

Нормальной называется такая форма записи числа, которая в неявном виде реализует формулу:

$$X = \left( \sum_{i=-m}^n a_i q^{i-p} \right) q^p \quad (1.1.11)$$

когда число представляется в виде произведения правильной дроби и некоторой целой степени основания системы счисления.

При этом правильная дробь называется мантиссой, а показатель степени – порядком. Число 125 в нормальной форме представляется в виде:

$$0,125 \cdot 10^3$$

Так как в естественной форме положение запятой в числе строго зафиксирована между целой и дробной частью, то часто числа в такой форме назы-

вают числами с фиксированной запятой. Следует отметить, что во многих западных странах для отделения целой и дробной частей числа используют точку вместо запятой. Поэтому часто числа в такой форме называют ещё числами с фиксированной точкой.

В дальнейшем мы будем пользоваться в основном термином "с фиксированной точкой" как наиболее распространенной и часто используемой в литературе.

В определении нормальной формы не наложено никаких ограничений на величину мантиссы, лишь бы она была правильной дробью. Поэтому положение запятой может меняться при изменении порядка. Поэтому числа в нормальной форме называют числами с плавающей точкой или с плавающей запятой.

Недостатки чисел с плавающей точкой:

1. Сложная конструкция аппаратуры процессора для реализации различных операций над числами из-за неоднозначности записи чисел и необходимости вследствие этого нормализации числа.
2. Ещё большее усложнение аппаратуры, во – первых из – за сложности структуры числа, состоящей из двух частей: мантиссы и порядка и, во – вторых вследствие того, что приходится производить действия с обоими частями числа.

Представление с фиксированной точкой не имеет указанных недостатков, но к сожалению у этой формы есть свои недостатки:

1. Малый диапазон чисел, представимых в компьютере. Пусть имеется некий гипотетический компьютер с 9-ти разрядной сеткой для записи десятичных чисел. Пусть также задано, что компьютер может работать в двух режимах: с фиксированной и с плавающей точкой, причем на мантиссу отведено 5 разрядов и на дробную часть отведено тоже 5 разрядов.

Форма с фиксированной точкой:

1	2	3	4	5	6	7	8	9
знак	целая часть			дробная часть				

Форма с плавающей точкой:

1	2	3	4	5	6	7	8	9
знак	мантисса					знак	порядок	
мантиссы					порядка			

Знак числа имеет двоичную природу, так как может принимать лишь два значения (+) и (-). Знак (+) принято изображать цифрой 0, а знак (-) цифрой 1

Режим с фиксированной точкой.	Режим с плавающей точкой.
$+ X _{\max} = +999,99999$	$+ X _{\max} = +0,99999 \cdot 10^{+99}$
$+ X _{\min} = +000,00000$	$+ X _{\min} = +0,10000 \cdot 10^{-99}$
$- X _{\min} = -000,00000$	$- X _{\min} = -0,10000 \cdot 10^{-99}$
$- X _{\max} = -999,99999$	$- X _{\max} = -0,99999 \cdot 10^{+99}$

Отсюда мы можем получить оценку диапазонов представления чисел в обеих формах:

$$\begin{aligned} -10^{+3} < X_{\phi.m.} < +10^{+3} \\ -10^{+99} < X_{n.m.} < +10^{+99} \end{aligned} \quad (1.1.12)$$

## 2. Меньшая точность.

Рассмотрим для двоичной системы. Абсолютная погрешность для чисел с фиксированной точкой составляет:

$$\Delta_{\phi.m.} = \frac{1}{2} 2^{-n} = const \quad (1.1.13)$$

Абсолютная погрешность для чисел с плавающей точкой составляет:

$$\Delta_{n.m.} = \frac{1}{2} 2^{-n} 2^p \quad (1.1.14)$$

$\Delta_{n.m.}$  зависит от  $p$  и следовательно меняется от

$$|\Delta_{n.m.}|_{\min} = \frac{1}{2} 2^{-n} 2^{-|P_{\max}|} \quad (1.1.15)$$

до

$$|\Delta_{n.m.}|_{\max} = \frac{1}{2} 2^{-n} 2^{+|P_{\max}|} \quad (1.1.16)$$

Относительная погрешность для чисел с фиксированной точкой определяется следующим образом:

$$\delta_{\phi.m.\min} = \frac{\Delta_{\phi.m.}}{|X_{\phi.m.\max}|} \quad (1.1.17)$$

$$\delta_{\phi.m.\max} = \frac{\Delta_{\phi.m.}}{|X_{\phi.m.\min}|} \quad (1.1.18)$$

где

$$\begin{aligned} |X_{\phi.m.\max}| &= \overbrace{0,11 \dots 1}^n = 1 - 2^{-n} \\ |X_{\phi.m.\min}| &= 0,00 \dots 1 = 2^{-n} \end{aligned} \quad (1.1.19)$$

Теперь можно получить оценку относительной погрешности для чисел с фиксированной точкой:

$$\delta_{\phi.m.\min} = \frac{\frac{1}{2} 2^{-n}}{1 - 2^{-n}} \cong 2^{-(n+1)} \quad (1.1.20)$$

если  $1 \gg 2^{-n}$

$$\delta_{\phi.m.\max} = \frac{\frac{1}{2} 2^{-n}}{2^{-n}} = \frac{1}{2} = 50\% \quad (1.1.21)$$

То есть максимальная относительная погрешность может достигать 50%, что конечно во многих случаях неприемлемо!

Рассмотрим теперь относительную погрешность для чисел с плавающей точкой:

$$\delta_{n.m.} = \frac{\Delta_{n.m.}}{|X_{n.m.}|}$$

$$\delta_{n.m.\min} = \frac{|\Delta_{n.m.\max}|}{|X_{n.m.\max}|} = \frac{\frac{1}{2} 2^{-n} 2^{+|P_{\max}|}}{(1 - 2^{-n}) 2^{+|P_{\max}|}} \cong 2^{-(n+1)} \quad (1.1.22)$$

при  $1 \gg 2^{-n}$

$$\delta_{n.m.\max} = \frac{|\Delta_{n.m.\min}|}{|X_{n.m.\min}|} = \frac{\frac{1}{2} 2^{-n} 2^{-|P_{\max}|}}{\frac{1}{2} 2^{-|P_{\max}|}} = 2^{-n} \quad (1.1.23)$$

Отсюда видно, что относительная погрешность  $\delta_{n.m.}$  не зависит от величины  $p$ , а зависит от  $n$ .

Главный вывод в том, что величина ошибки для чисел с плавающей точкой изменяется в очень малом диапазоне и очень мала ( $2^{-n} \ll 1$ ), если числа нормализованы.

## §1.2. Кодирование чисел

*Писатель пользуется известными привилегиями, в благодетельности которых, надеюсь нет никаких оснований сомневаться. Так, встретив у меня непонятное место, читатель должен предположить, что под ним кроется нечто весьма полезное и глубокомысленное.*

Джонатан Свифт

Вопрос о кодировании чисел возникает по той причине, что в компьютер либо нельзя, либо нерационально вводить числа в том виде, каком они изображаются человеком – вычислителем на бумаге.

Во – первых, нужно кодировать знак числа. Во – вторых, приходится иногда и само число кодировать. В частности, при выполнении операций с фиксированной точкой все числа приводятся к виду правильных дробей.

Простейший машинный код, называемый прямым, получается в том случае, когда знак двоичной дроби кодируется 0 (+) и 1 (-), а цифровая часть остается без изменений.

Более строго прямой код можно определить следующим образом:

Прямым кодом отрицательной двоичной дроби называется её обычное изображение в естественной форме у которого в знаковом разряде проставляется единица. Прямой код положительной двоичной дроби совпадает с её обычным изображением в естественной форме. Аналитически:

$$[X]_{np} = \begin{cases} X, & X \geq 0 \\ 1 - X, & X < 0 \end{cases} \quad (1.2.1)$$

$$X_1 = +0,110011_{(2)} \quad [X_1]_{np} = 0,110011$$

$$X_2 = -0,110011_{(2)} \quad [X_2]_{np} = 1,110011$$

Так как операция сложения значительно проще реализуется, чем операция вычитания, то возникает вопрос: нельзя ли каким либо способом свести алгебраическое сложение к арифметическому, т.е. попросту говоря заменить операцию вычитания сложением.

Идея замены операции вычитания сложением основана на применении некоторых вспомогательных положительных чисел, однозначно связанных с исходными отрицательными числами. Эти вспомогательные числа находятся как дополнения (по абсолютной величине) заданных отрицательных чисел до некоторого  $X_{gp}$ :

$$X_{gp} = |X_{max}| + 1_{m.p.} = 10^n \quad (1.2.2)$$

где  $1_{m.p.}$  означает единицу младшего разряда.

Пусть имеются двухразрядные десятичные числа.

$$0 \leq X_{тек} \leq X_{max} = 99_{(10)}$$

Здесь  $X_{cp} = X_{max} + 1 = 99 + 1 = 100 = 10^2$

Пусть нужно алгебраически сложить

$$X_1 = 84 \text{ и } X_2 = -63$$

$X_2$  согласно вышесказанному, мы заменим его дополнением (по абсолютной величине) до  $X_{cp}$  (до ста), т.е.

$$[X_2]_{дон} = X_2 + X_{cp} = X_2 + 100 = -63 + 100 = 37$$

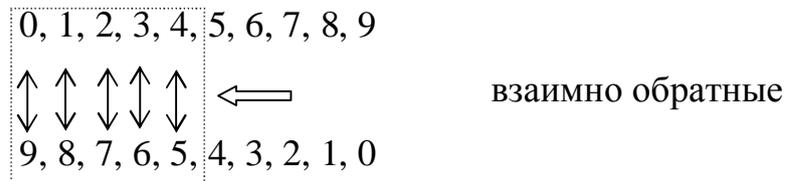
Далее складываем  $X_1$  и  $[X_2]_{дон}$

$$X_1 + [X_2]_{дон} = 84 + 37 = 121$$

└────────── отбрасывается

Прямое вычитание  $84 - 63 = 21$  показывает правильность наших действий.

Нахождение  $[X_2]_{дон}$  производят без вычитания. Выпишем цифры от 0 до 9 сначала в прямом порядке, затем в обратном:



Отсюда видно, что замена в некотором отрицательном числе всех цифр на взаимно обратные равносильно сложению исходного числа с  $|X|_{max}$ .

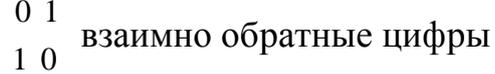
Правило нахождения дополнения отрицательных чисел без использования операции вычитания:

для получения дополнения отрицательного числа следует все цифры исходного числа заменить на взаимно обратные и к полученному результату добавить единицу младшего разряда.

Действительно:

$$[X_2]_{дон} = X_2 + X_{cp} = X_2 + |X_{max}| + 1 = -63 + 99 + 1 = 36 + 1 = 37$$

Аналогично и вместе с тем проще для двоичной системы счисления.



Дополнение здесь ищется до двух.

Дополнительным кодом отрицательной двоичной дроби называется её дополнение по модулю до двух.

Правило:

Для получения дополнительного кода отрицательной двоичной дроби нужно в знаковом разряде записать 1, а во всех других разрядах цифры заменить на взаимно обратные, т.е.  $0 \rightarrow 1$ ,  $1 \rightarrow 0$ , после чего к младшему разряду числа добавляется 1.

Дополнительный код положительной двоичной дроби совпадает с её

прямым кодом. Аналитически это можно выразить следующим образом:

$$[X]_{дон} = \begin{cases} X, & X \geq 0 \\ X + 10_{(2)}, & X < 0 \end{cases} \quad (1.2.3)$$

$$\begin{array}{ll} X_1 = +0,110011 & [X_1]_{дон} = 0,110011 \\ X_2 = -0,110011 & [X_2]_{дон} = 1,001100 \\ & + \quad 1 \\ & \hline & 1,001101 \end{array}$$

Переход от дополнительного кода отрицательного числа к прямому коду производится следующим образом:

Заменяют все цифры числа, за исключением знакового на взаимно обратные. После чего добавляют единицу к младшему разряду.

$$\begin{array}{l} [X]_{дон} = 1,001101 \\ [X]_{пр} = 1,110010 \\ + \quad 1 \\ \hline 1,110011 \end{array}$$

### Обратный код чисел.

Обратным кодом отрицательной двоичной дроби называется её дополнение по модулю до двух без единицы младшего разряда.

Он получается по правилу: в знаковом разряде числа проставляется 1, а во всех остальных разрядах цифры заменяются на взаимно обратные.

Обратный код положительной двоичной дроби совпадает с прямым.

Аналитически:

$$[X]_{обр} = \begin{cases} X, & X \geq 0 \\ X + 10_{(2)} - 10_{(2)}^{-n}, & X < 0 \end{cases} \quad (1.2.4)$$

$$\begin{array}{ll} X_1 = +0,110011 & [X_1]_{обр} = 0,110011 \\ X_2 = -0,110011 & [X_2]_{обр} = 1,001100 \end{array}$$

### Алгебраическое сложение чисел

Алгебраическое сложение всегда производится либо в дополнительном, либо в обратном коде. Сумма, естественно, также получается в одном из этих кодов соответственно. Правильный знак суммы получается автоматически.

В случае возникновения единицы переноса из знакового разряда суммы, её нужно отбросить при производстве сложения в дополнительном коде и прибавить к младшему разряду суммы при сложении в обратном коде.

Необходимо учитывать, что в поразрядном сложении участвуют не две, а три цифры. Третьим слагаемым будет единица переноса.

**Таблица сложения**

Текущий разряд первого слагаемого	Текущий разряд второго слагаемого	Цифра переноса из предыдущего разряда	Результат (текущий разряд и цифра переноса)
0	0	0	00
0	0	1	01
0	1	0	01
1	0	0	01
1	0	1	10
1	1	0	10
0	1	1	10
1	1	1	11

**Сложение с фиксированной точкой**

При алгебраическом сложении возможны 4 различных случая:

1.  $X_1 > 0; X_2 > 0; X_1 + X_2 = X_3 > 0$

$X_1 = +0,1100$

$X_2 = +0,0010$

$[X_1]_{np} = 0,1100; [X_1]_{don} = [X_1]_{np} \quad 0,1100$

$[X_2]_{np} = 0,0010; [X_2]_{don} = [X_2]_{np} \quad \begin{array}{l} 0,0010 \\ 0,1110 \end{array}$

$X_3 = +0,1110$

2.  $X_1 > 0; X_2 < 0; X_1 + X_2 = X_3 > 0$

$X_1 = +0,1100$

$X_2 = -0,0010$

$[X_1]_{np} = 0,1100; [X_1]_{don} = [X_1]_{np} \quad \begin{array}{cc} \text{в доп. коде} & \text{в обр. коде} \end{array}$

$[X_2]_{np} = 1,0010; [X_2]_{don} = 1,1110 \quad \begin{array}{cc} 0,1100 & 0,1100 \end{array}$

$[X_2]_{обр} = 1,1101 \quad \begin{array}{cc} 1,1110 & 1,1101 \\ 1|0,1010 & 1|0,1001 \end{array}$

$\begin{array}{c} 1 \\ \hline 0,1010 \end{array}$

$X_3 = +0,1010$

$$3. \quad X_1 < 0; \quad X_2 > 0; \quad X_1 + X_2 = X_3 < 0$$

$$X_1 = -0,1100$$

$$X_2 = +0,0010$$

$$[X_1]_{np} = 1,1100; \quad [X_1]_{обр} = 1,0011 \quad \begin{array}{cc} \text{в доп. коде} & \text{в обр. коде} \\ 1,0011 & 1,0100 \end{array}$$

$$[X_1]_{дон} = 1,0100 \quad \begin{array}{cc} 0,0010 & 0,0010 \\ \hline 1,0101 & 1,0110 \end{array}$$

$$[X_2]_{np} = 0,0010 = [X_2]_{обр} = [X_2]_{дон}$$

$$[X_3]_{обр} = 1,0101 \quad [X_3]_{дон} = 1,0110$$

$$[X_3]_{np} = 1,1010 \quad [X_3]_{обр} = 1,1001$$

$$\frac{1}{1,1010}$$

$$X_3 = -0,1010$$

$$4. \quad X_1 < 0; \quad X_2 < 0; \quad X_1 + X_2 = X_3 < 0$$

$$X_1 = -0,1100$$

$$X_2 = -0,0010$$

$$[X_1]_{np} = 1,1100; \quad [X_1]_{обр} = 1,0011 \quad \begin{array}{cc} \text{в доп. коде} & \text{в обр. коде} \\ 1,0100 & 1,0011 \end{array}$$

$$[X_1]_{дон} = 1,0100 \quad \begin{array}{cc} 0,1110 & 1,1101 \\ \hline 1|1,0010 & 1|1,0000 \end{array}$$

$$\frac{1}{1,0001}$$

$$[X_2]_{np} = 1,0010 \quad [X_2]_{обр} = 1,1101$$

$$[X_2]_{дон} = 1,1110$$

$$[X_3]_{обр} = 1,0001 \quad [X_3]_{дон} = 1,0010$$

$$[X_3]_{np} = 1,1110 \quad [X_3]_{обр} = 1,1101$$

$$\frac{1}{1,1110}$$

$$X_3 = -0,1110$$

При сложении любых положительных чисел, сумма которых  $> 1$ , результат их машинного суммирования будет иметь вид отрицательного числа, хотя никаких ошибок не было. В этом случае происходит переполнение разрядной сетки.

Необходимо, чтобы компьютер научился фиксировать такие ситуации. Для этого используются так называемые модифицированные коды. Модифицированные коды отличаются от обыкновенных тем, что в них знак числа изображаются двумя разрядами.

Знак  $+$  кодируется 00, а знак  $-$  кодируется 11. Комбинации 01 и 10 являются запрещенными.

Пусть имеются  $[X_1]_{np} = 0,10101$  и  $[X_2]_{np} = 1,10101$ . Перевести их в модифицированный обратный код.

$$[X_1]_{обр}^{mod} = 00,10101$$

$$[X_2]_{обр}^{mod} = 11,01010$$

Пусть требуется сложить числа:

$$X_1 = +0,1100$$

$$X_2 = +0,1010$$

Переведем их в модифицированный дополнительный код и выполним требуемую операцию:

$$[X_1]_{дон}^{mod} = 00,1100 \quad 00,1100$$

$$[X_2]_{дон}^{mod} = 00,1010 \quad \begin{array}{r} 00,1010 \\ \hline 01,0110 \end{array}$$

Произошло переполнение разрядной сетки (здесь мы предполагаем, что длина разрядной сетки равно 4).

Пусть теперь необходимо сложить числа:

$$X_1 = +0,1100$$

$$X_2 = -0,1010$$

Имеем:

$$[X_1]_{дон}^{mod} = 00,1100 \quad 00,1100$$

$$[X_2]_{дон}^{mod} = 11,1110 \quad \begin{array}{r} 11,1110 \\ \hline 1 \mid 00,1010 \end{array}$$

$$X_3 = +0,1010$$

т.е. знак результата получился автоматически.

### Сложение чисел с плавающей точкой.

Правило:

- 1) Уравниваются порядки слагаемых. Меньший порядок увеличивается до большего, а мантисса этого числа сдвигается вправо на соответствующее количество разрядов.
- 2) Производится преобразование мантисс в один из модифицированных кодов, дополнительный или обратный.
- 3) Производится сложение мантисс по правилам сложения чисел с фиксированной точкой.
- 4) Полученная сумма мантисс переводится в прямой код. К ней приписывается общий порядок слагаемых, производится нормализация суммы (в случае необходимости) и округление мантиссы.

Округление производится следующим образом: пусть используется  $n$  разрядов. В результате сложения мантисс получилось предположим  $n+k$  разрядов. Прежде чем отбросить последние  $k$  разрядов, к  $n+1$  разряду добавляется 1 (напомним, что речь идет о двоичном сложении), затем разряды с  $n+1$  до  $n+k$  отбрасываются.

Пример:

$$\begin{array}{r} 0,01011 \\ \hline 0,0110|0 \end{array} \qquad \begin{array}{r} 0,01010 \\ \hline 0,0101|0 \end{array}$$

Для лучшего уяснения рассмотрим операцию сложения с плавающей точкой сначала для десятичных чисел:

$$\begin{aligned} X_1 &= 0,125 \cdot 10^3 \\ X_2 &= 0,625 \cdot 10^2 = 0,0625 \cdot 10^3 \\ &\begin{array}{r} 0,125 \\ + 0,0625 \\ \hline 0,1875 \end{array} \\ &0,1875 \cdot 10^3 \cong 0,188 \cdot 10^3 \end{aligned}$$

Пример нормализации:

$$\begin{aligned} &0,0256 \cdot 10^3 = 0,256 \cdot 10^2 \\ &0,625 \cdot 10^3 \\ &+ \\ &\underline{0,600 \cdot 10^3} \\ &1,225 \cdot 10^3 \\ &1,225 \cdot 10^3 = 0,1225 \cdot 10^4 \end{aligned}$$

Теперь рассмотрим в двоичной системе счисления:

$$[X_1]_{np} = 0\ 1100\ 0\ 0100 = 0\ 001100\ 0\ 110$$

$$[X_2]_{np} = 0\ 0010\ 0\ 110$$

$$[m_1]_{дон}^{mod} = 00\ 001100 \quad (m - \text{мантисса})$$

$$[m_2]_{дон}^{mod} = 00\ 001000$$

$$[m_1]_{дон}^{mod} + [m_2]_{дон}^{mod} = 00\ 001100$$

$$\underline{00\ 001000}$$

$$00\ 010100$$

$$[m_3]_{дон}^{mod} = 00\ 010100 = [m_3]_{np}^{mod}$$

нормализация :

$$[X_3]_{np} = 0\ 010100\ 0\ 0110 = 0\ 10100\ 0\ 0101$$

округление :

0,10100

$\frac{1}{0,10101}$

$[X_3]_{np} = 0\ 1010\ 0\ 0101$

### Умножение чисел с фиксированной точкой

Умножение чисел с фиксированной точкой производится в два этапа:

- 1) определяется знак произведения при помощи сложения по модулю 2 (mod 2). Суммирование по mod 2 производится согласно правилам двоичной арифметики, но без учета единицы переноса. Правомерность такого определения знака доказывается с помощью таблицы:

$(+) \cdot (+) = (+)$	$0 \oplus 0 = 0$
$(+) \cdot (-) = (-)$	$0 \oplus 1 = 1$
$(-) \cdot (+) = (-)$	$1 \oplus 0 = 1$
$(-) \cdot (-) = (+)$	$1 \oplus 1 = 0$

- 2) производится перемножение модулей сомножителей, затем в случае необходимости округление и к полученному результату приписывается знак.

Умножение производится по обычным правилам арифметики согласно двоичной таблице умножения. Пример:

$$[X_1]_{np} = 0,1010$$

$$[X_2]_{np} = 1,1101$$

1. определяем знак произведения

$$0 \oplus 1 = 1$$

2. выполняем умножение

$$\begin{array}{r} 0,1010 \\ 0,1101 \\ \hline 1010 \\ 0000 \\ 1010 \\ 1010 \\ \hline 0,10000010 \end{array}$$

$$[X_3]_{np} = 1,1000$$

## Умножение чисел с плавающей точкой

$$X_1 = m_1 \cdot 10^{P_1}$$

$$X_2 = m_2 \cdot 10^{P_2}$$

$$X_3 = X_1 \cdot X_2 = (m_1 \cdot m_2) \cdot 10^{P_1+P_2}$$

Отсюда 4 этапа для умножения чисел с плавающей точкой:

1. определение знака произведения путем сложения знаковых разрядов по mod 2.
2. перемножение модулей мантисс сомножителей по правилам умножения чисел с фиксированной точкой.
3. сложение порядков по правилам алгебраического сложения чисел с фиксированной точкой.
4. округление полученного результата и приписывание соответствующего знака результату.

Пример:

$$[X_1]_{np} = 1\ 1010\ 0\ 101$$

$$[X_2]_{np} = 0\ 1011\ 1\ 011$$

$$1) \quad 1 \oplus 0 = 1$$

$$\begin{array}{r}
 2) \qquad \qquad \qquad 0,1010 \\
 \qquad \qquad \qquad \underline{0,1011} \\
 \qquad \qquad \qquad 1010 \\
 \qquad \qquad \qquad 1010 \\
 \qquad \qquad \qquad 0000 \\
 \qquad \qquad \qquad \underline{1010} \\
 \qquad \qquad \qquad 0,01101110
 \end{array}$$

$$3) [P_1]_{np}^{mod} = 00,101$$

$$[P_2]_{np}^{mod} = 11,011$$

$$[P_2]_{дон}^{mod} = 11,101$$

$$\begin{array}{r}
 00\ 101 \\
 \underline{11\ 101} \\
 1 \mid 00\ 010
 \end{array}$$

$$[P_3]_{np} = 0\ 010$$

$$4) [X_3]_{np} = 1,01101110\ 0\ 010$$

*округление:*

$$[X_3]_{np} = 1,0111\ 0\ 010$$



$$\begin{array}{r} 110 \\ \hline 1000 \\ 110 \\ \hline 10 \end{array}$$

$$|m_3| = 0,111$$

$$3) [P_1]_{\partial\partial n}^{mod} = 00,110$$

$$[P_2]_{\partial\partial n}^{mod} = 11,101$$

$$00,110$$

$$11,101$$

$$\hline 1|00,011$$

$$[P_3]_{np} = 0,011$$

$$[X_3]_{np} = 1 \ 111 \ 0 \ 011$$

### §1.3. Логические основы вычислительной техники

Как известно, логика – это наука о законах и формах мышления. Математическая логика занимается изучением возможностей применения формальных математических методов для решения логических задач. В вычислительной технике используется главным образом начальный раздел математической логики – исчисление высказываний или алгебра логики. Иногда её ещё называют булевой алгеброй по имени английского математика Джорджа Буля, который ещё в XIX веке разработал основные положения исчисления высказываний.

Однако сам Буль не видел путей практической реализации своих исследований. Но уже в 40-х годах XX века алгебра логики нашла широкое применение при проектировании и синтезе сложных переключательных схем в автоматических телефонных станциях, а затем и в вычислительной технике.

Начальным понятием алгебры логики является высказывание. Под высказыванием понимается любое утверждение, которое оценивается только с точки зрения его истинности или ложности. То есть высказывание может быть либо истинным, либо ложным.

В соответствии с такой двоичной природой высказываний условились называть их логическими двоичными переменными. Любое высказывание можно обозначить символом  $x$  и считать, что  $x=1$ , если высказывание истинно,  $x=0$ , если высказывание ложно.

Таким образом, логическая (булевая) переменная – это такая величина  $x$ , которая может принимать только два значения (0 или 1):

$$x=\{0, 1\}$$

Высказывание абсолютно истинно, если соответствующая ей логическая величина принимает значение  $x=1$  при любых условиях.

Высказывание абсолютно ложно, если соответствующая ей логическая переменная принимает значение 0 при любых условиях.

Сложным называется высказывание, значение истинности которого зависит от значения истинности других высказываний. Следовательно, любое сложное высказывание можно считать функцией некоторых двоичных аргументов – простых высказываний, входящих в его состав.

Логическая функция (функция алгебры логики) – функция  $f(x_1, x_2, \dots, x_n)$  принимающая значение 0 или 1 на наборе логических переменных  $x_1, x_2, \dots, x_n$ . Причем  $x_1, x_2, \dots, x_n$  могут быть в свою очередь логическими функциями. Отсюда следует, что можно построить функцию любой наперед заданной сложности, пользуясь принципом суперпозиции.

Ограниченный набор функций, обеспечивающий на основе суперпозиции построение функции любой наперед заданной сложности, называется функционально полной системой функций.

Из множества функционально полных систем рассмотрим один, получивший наибольшее распространение и вследствие этого названного основ-

НЫМ.

В его состав входят всего 3 функции:

I. функция отрицания (функция НЕ, инверсия).

Отрицанием называется такая функция  $y$ , которая истинна тогда и только тогда, когда ложен аргумент  $x$  и наоборот,  $y$  ложна тогда и только тогда, когда аргумент  $x$  истинен.

В алгебре логики любые функции удобно изображать в виде таблицы соответствия всех возможных комбинаций значений истинности входных переменных (аргументов) и выходной логической функции.

аргумент $x$	0	1
значение функции $y$	1	0

При помощи логико – математической символики функция отрицания записывается в виде:

$$y = \bar{x}, \neg x, \text{NOT}$$

Эта функция обладает свойством  $\bar{\bar{x}} = x$

II. Конъюнкция (логическое умножение, функция "И")

Конъюнкцией  $n$  высказываний называется такое сложное высказывание  $y$ , которое истинно тогда и только тогда, когда истинны все исходные высказывания; во всех остальных случаях  $y$  ложно. Таблица истинности для функции двух переменных:

$x_1$	0	0	1	1
$x_2$	0	1	0	1
$y$	0	0	0	1

Обозначения:

$$y = x_1 \cdot x_2$$

$$y = x_1 \& x_2$$

AND

Конъюнкция обладает 4-мя очевидными свойствами:

1.  $x \cdot x \cdots x = x$   
результат многократного логического умножения аргумента  $x$  на себя равен исходному.
2.  $x \cdot 1 = x$   
логическое произведение двух аргументов, один из которых абсолютно истинен, равно второму аргументу.
3.  $x \cdot 0 = 0$   
логическое произведение двух аргументов, один из которых абсолютно ложен, равно 0.
4.  $x \cdot \bar{x} = 0$

логическое произведение некоторого аргумента и его отрицания равно 0.

### III. дизъюнкция (логическое сложение, функция "ИЛИ")

Дизъюнкцией  $n$  высказываний называется такое сложное высказывание  $y$ , которое ложно тогда и только тогда, когда ложны все исходные высказывания. Если же истинно хотя бы одно исходное высказывание, функция  $y$  тоже истинно.

Таблица истинности для функции двух переменных:

$x_1$	0	0	1	1
$x_2$	0	1	0	1
$y$	0	1	1	1

Обозначения:

$$y = x_1 + x_2$$

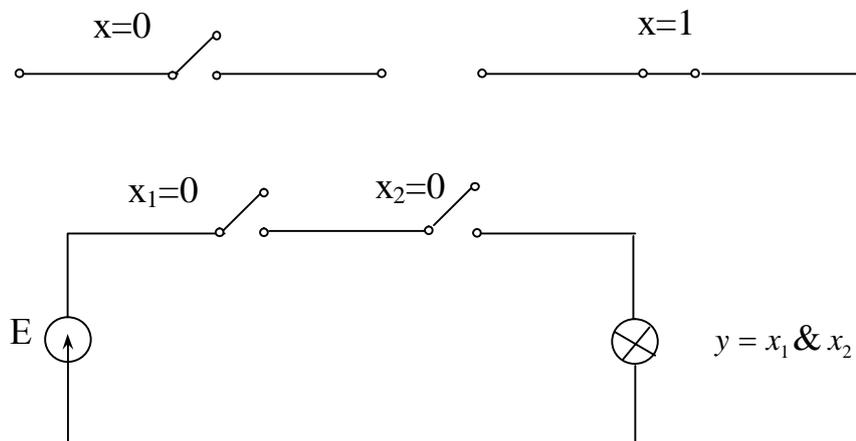
$$y = x_1 \vee x_2$$

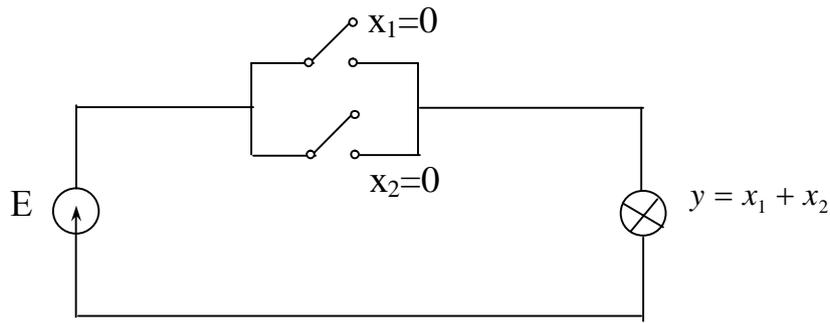
OR

Дизъюнкция обладает 4-мя очевидными свойствами:

1.  $x + x + \dots + x = x$   
результат многократного логического сложения аргумента  $x$  на себя равен исходному.
2.  $x + 1 = 1$   
логическое сложение двух аргументов, один из которых абсолютно истинен, равно 1.
3.  $x + 0 = x$   
логическое сложение двух аргументов, один из которых абсолютно ложен, равно второму аргументу.
4.  $x + \bar{x} = 1$   
логическое сложение некоторого аргумента и его отрицания равно 1.

Рассмотренные выше функции можно элементарным образом реализовать с помощью электрических схем.





Абсолютно истинную функцию называют константа 1, абсолютно ложную – константой 0.

Функция  $f(x) = x$ , т.е. функция повторяющая значение аргумента, называется тождественной.

Существуют и другие функции от двух логических переменных. Их всего 16. См. таблицу 1.2.

Функция  $f_9 = \overline{x_1 + x_2}$  называется функцией Пирса. Это функция, которая истинна тогда и только тогда, когда оба  $x_1$  и  $x_2$  ложны.

Функция  $f_{15} = \overline{x_1 \cdot x_2}$  называется функцией Шеффера. Это функция, которая ложна тогда и только тогда, когда оба  $x_1$  и  $x_2$  истинны.

$$f_9 = x_1 \downarrow x_2 - \text{стрелка Пирса}$$

$$f_{15} = x_1 | x_2 - \text{штрих Шеффера}$$

Табл. 2

Функция	Значения $x_1$ и $x_2$				Аналитическое представление функции
	00	01	10	11	
$f_1$	0	0	0	0	0
$f_2$	0	0	0	1	$x_1 \cdot x_2$
$f_3$	0	0	1	0	$x_1 \cdot \overline{x_2}$
$f_4$	0	0	1	1	$x_1$
$f_5$	0	1	0	0	$\overline{x_1} \cdot x_2$
$f_6$	0	1	0	1	$\overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2} = x_2$
$f_7$	0	1	1	0	$\overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2} = x_1 \oplus x_2$
$f_8$	0	1	1	1	$x_1 + x_2$
$f_9$	1	0	0	0	$\overline{x_1 + x_2}$
$f_{10}$	1	0	0	1	$x_1 \equiv x_2$
$f_{11}$	1	0	1	0	$\overline{x_1} \cdot \overline{x_2} + x_1 \cdot \overline{x_2}$
$f_{12}$	1	0	1	1	$\overline{x_1} \cdot \overline{x_2} + x_1 \cdot \overline{x_2} + x_1 \cdot x_2$
$f_{13}$	1	1	0	0	$\overline{x_1} \cdot \overline{x_2} + \overline{x_1} \cdot x_2 = \overline{x_1}$
$f_{14}$	1	1	0	1	$x_1 \rightarrow x_2$
$f_{15}$	1	1	1	0	$\overline{x_1 \cdot x_2}$
$f_{16}$	1	1	1	1	1

### Законы алгебры логики

В алгебре логики имеются 4 основных закона, регламентирующие порядок выполнения операций НЕ, И, ИЛИ в любом логическом выражении:

1. переместительный (коммутативный);
2. сочетательный (ассоциативный);
3. распределительный (дистрибутивный);
4. инверсии (правило Де Моргана).

Рассмотрим эти законы:

### **Переместительный закон**

Справедлив как для операции И, так и для операции ИЛИ.

$$1. x_1 + x_2 = x_2 + x_1$$

$$2. x_1 \cdot x_2 = x_2 \cdot x_1$$

### **Сочетательный закон**

Также справедлив как для операции И, так и для операции ИЛИ.

$$1. x_1 + x_2 + x_3 = x_1 + (x_2 + x_3) = x_2 + (x_1 + x_3) = x_3 + (x_1 + x_2)$$

$$2. x_1 \cdot x_2 \cdot x_3 = x_1 \cdot (x_2 \cdot x_3) = x_2 \cdot (x_1 \cdot x_3) = x_3 \cdot (x_1 \cdot x_2)$$

### **Распределительный закон**

1. Распределительный закон первого рода

Произведение суммы нескольких аргументов на какую – либо переменную равно сумме произведений.

$$(x_1 + x_2) \cdot x_3 = x_1 \cdot x_3 + x_2 \cdot x_3$$

2. Распределительный закон второго рода

Сумма произведения нескольких аргументов и какой – либо переменной равна произведению суммы каждого сомножителя и этой переменной.

$$x_2 \cdot x_3 + x_1 = (x_1 + x_2) \cdot (x_1 + x_3)$$

Этот закон выглядит несколько парадоксальным. Докажем его:

$$\begin{aligned} (x_1 + x_2) \cdot (x_1 + x_3) &= x_1 \cdot x_1 + x_2 \cdot x_1 + x_1 \cdot x_3 + x_2 \cdot x_3 = \\ &= x_1 + x_1 \cdot x_2 + x_1 \cdot x_3 + x_2 \cdot x_3 = x_1 \cdot (1 + x_2 + x_3) + x_2 \cdot x_3 = \\ &= x_1 \cdot 1 + x_2 \cdot x_3 = x_1 + x_2 \cdot x_3 = x_2 \cdot x_3 + x_1 \end{aligned}$$

### **Законы инверсии**

1. Отрицание логической суммы нескольких аргументов равно логическому произведению отрицаний этих же аргументов.

$$\overline{x_1 + x_2 + \dots + x_n} = \overline{x_1} \cdot \overline{x_2} \cdot \dots \cdot \overline{x_n}$$

2. Отрицание логического произведения нескольких аргументов равно логической сумме отрицаний этих же аргументов.

$$\overline{x_1 \cdot x_2 \cdot \dots \cdot x_n} = \overline{x_1} + \overline{x_2} + \dots + \overline{x_n}$$

### **Следствия из законов алгебры логики**

Логическим выражением называют всякую совокупность логических переменных и знаков операций, указывающих какие действия над этими переменными нужно выполнить. Будем считать отрицание логическим действием первой ступени (старшей логической операцией), конъюнкцию – логическим действием второй ступени, дизъюнкцию – логическим действием третьей ступени.

Тогда можно определить правило выполнения логических операций:

Если в логическом выражении встречаются только операции одной и той же ступени, то их нужно выполнять в том порядке, в каком они записаны (т.е. слева направо).

Если же в логическом выражении встречаются действия различных ступеней, то сначала выполняются действия первой ступени, затем второй и только после этого третьей.

Всякое отклонение от этого порядка должно регламентироваться скобками.

Пример. Вычислить значение логического выражения

$$y = x_1 \cdot x_2 + x_3 \cdot x_4 + x_5$$

при  $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1$

$$y = 1 \cdot 1 + 1 \cdot 1 + 1 = 1 + 1 \cdot 0 = 1 + 0 = 1$$

### Контрольные вопросы:

1. Формы представления информации.
2. Что такое система счисления?
3. В чем различие между позиционной и непозиционной системами счисления?
4. Перечислите наиболее распространенные системы счисления.
5. Правила перевода чисел из одной системы счисления в другую.
6. Формы представления чисел в памяти компьютера.
7. Дополнительный и обратный коды двоичных чисел.
8. Алгебраическое сложение двоичных чисел.
9. Умножение двоичных чисел.
10. Деление двоичных чисел.
11. Логические переменные. Логические функции.
12. Основная система логических функций ("И", "ИЛИ", "НЕ").
13. Законы алгебры логики.
14. Правило выполнения логических операций.

## Глава II. Технические средства вычислительной техники

### §2.1. Краткая история развития средств вычислительной техники

Многие математики и механики прошлого пытались сконструировать различные вычислительные устройства, приборы и машины для облегчения вычислений. Во многих странах с древнейших времен были известны счеты.

В 1617 году Джон Непер описал специальные палочки для быстрого умножения и деления на счетах.

В 1642 году Блез Паскаль изобрел суммирующую машину.

В 1694 году Лейбниц создал машину, которая могла не только складывать, но и умножать.

Идею построения вычислительной машины похожей на современную, впервые высказал ещё в XIX веке английский математик Чарльз Бэббидж. В 1833 году им был предложен проект аналитической машины, которая должна была выполнять элементарные математические операции, запоминать исходные, промежуточные и окончательные результаты. Его машина состояла из "склада" для хранения чисел и "мельницы" для выполнения действий над числами. Из-за невысокого уровня развития тогдашней техники, Ч. Бэббидж не смог собрать эту машину.

В 1937 г. американский ученый Айткен заново разработал принципы построения цифровых вычислительных машин с автоматическим ходом вычислений, не зная о том, что это давно было сделано Ч. Бэббиджем.

В 1944 г. вступила в строй первая релейно – механическая цифровая вычислительная машина "МАРК – I". Постройка следующей машины "МАРК – II" на реле улучшенных конструкций показала, что надежность релейных машин крайне низка.

В начале 40-х годов Клод Шеннон предложил для синтеза релейных схем использовать аппарат математической логики (булевой алгебры). Это дало возможность формальным способом строить экономичные схемы. Вследствие недостатков электромеханических реле конструкторы решают использовать электронные реле – триггеры. Это во много раз повысило надежность вычислительных машин.

Первая электронно – цифровая вычислительная машина была разработана в США в 1946 году и называлась "ЭНИАК". Она содержала 19000 электронных ламп. Скорость 5000 сложений в секунду.

Этот момент явился точкой отсчета пути по которому пошло развитие вычислительной техники.

Развитие вычислительной техники определялось развитием электроники, появлением новых элементов и принципов их действия, т.е. развитием элементной базы. В настоящее время можно выделить 4 поколения компьютеров. Причем компьютеры четвертого поколения отличаются от предшествующих не только элементной базой, но и структурой построения, а также способом общения с человеком и эта тенденция сохранится и в будущем.

Под поколением компьютеров понимаются все типы и модели компьютеров, разработанные, хотя и различными конструкторскими коллективами, но построенные на одних и тех же научных принципах и элементной базе. Появление каждого нового поколения определялось тем, что появлялись новые электронные элементы, технология изготовления которых принципиально отличалась от предыдущего поколения.

**Первое поколение** (1946 – середина 50-х годов).

Элементная база – электронные лампы. В ЭВМ "ЭНИАК" было 19000 ламп, из которых ежемесячно заменяли 5000.

В Советском Союзе первая вычислительная машина была создана в 1951 году и называлась МЭСМ.

**Второе поколение** (середина 50-х – середина 60-х годов)

Элементная база – полупроводниковые приборы (транзисторы, диоды). Значительно улучшились характеристики компьютеров (быстродействие, надежность, габариты, условия эксплуатации).

**Третье поколение** (середина 60-х – середина 70-х годов)

В качестве элементной базы используются интегральные схемы (ИС). ИС выполняет те же функции, что и аналогичная схема на полупроводниках, но при этом существенно меньше по размерам, увеличилась надежность и быстродействие. Более точно, ИС – это электронная схема, реализующая определенную функцию, технически выполненная на одном кристалле кремния площадью всего несколько квадратных миллиметров.

Первый компьютер на ИС – широко известная ИВМ/360.

**Четвертое поколение** (середина 70-х – по настоящее время)

Компьютеры создаются на основе БИС (больших интегральных схем) со степенью интеграции от сотен тысяч до нескольких миллионов элементов на одном кристалле.

В виде БИС удалось создать процессор компьютера, который получил название микропроцессор (МП). МП повлияли на структуру и конструкцию компьютеров. Размеры компьютеров значительно уменьшились, появились персональные компьютеры (ПК).

## ***§2.2. Логические элементы вычислительной техники, реализующие основные логические функции.***

Логическими элементами называются такие электронные устройства, в которых между входными и выходными сигналами реализуются те или иные логические операции. Логическим элементом называется устройство, реализующее только одну булеву функцию.

**Схема отрицания (элемент НЕ, инвертор)**

Как следует из определения функции НЕ, единичный сигнал на выходе схемы отрицания должен появиться при наличии на её входе нулевого сигнала и наоборот, на выходе схемы должен появиться нулевой сигнал, если на входе действует сигнал, соответствующий логической единице.

Функциональная схема имеет вид:

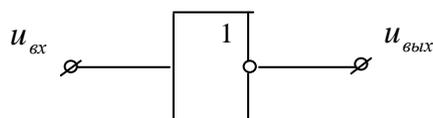


Рис. 2.1. Функциональная схема элемента НЕ

Слева вход, справа выход. На выходной стороне прямоугольника представляется  $\circ$  означающий инверсию. На практике чаще всего элемент НЕ реализуется при помощи однокаскадного усилителя постоянного тока, собранного на транзисторе. Схема электрическая принципиальная имеет вид:

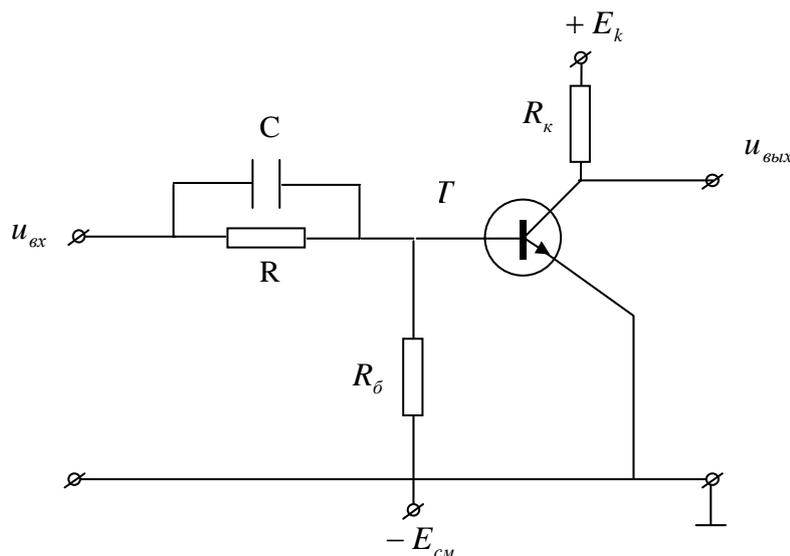


Рис. 2.2. Схема электрическая принципиальная

При этом используется свойство любого линейного усилительного каскада изменять у входного переменного сигнала фазу на  $180^0$ . В логических схемах все электронные приборы (транзистор в нашем случае) работают в ключевом режиме. При поступлении на вход схемы низкого уровня напряжения, транзистор (n-p-n типа) остается закрытым и на выходе будет  $u_{\text{вых}} = +E_k$  означающий "1". Если на вход поступает высокий уровень сигнала ("1"), то транзистор откроется и на выходе будет низкий уровень напряжения  $u_{\text{вых}} = E_k - I_k R_k$ , означающий логический "0".

О том, как работает транзистор и другие элементы этой схемы вы сможете узнать из соответствующих курсов физики, ТОЭ и электроники. Здесь за неимением места мы эти подробности опускаем. Для целей нашего курса достаточно уяснить, что между входным и выходным напряжениями схемы существует такая взаимосвязь, которая и реализует логическую функцию НЕ.

**Схема совпадения (схема "И")**

Единичный сигнал на выходе схемы должен появляться только тогда, когда на всех входах "1".

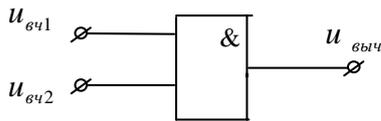


Рис. 2.3. Функциональная схема

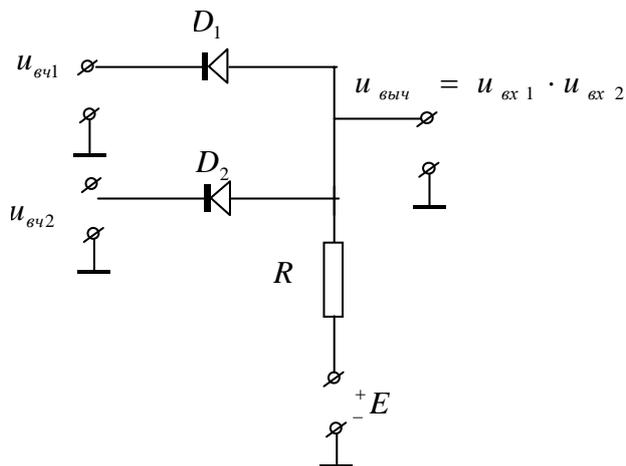


Рис. 2.4. Схема электрическая принципиальная

Электрическая схема реализована на полупроводниковых диодах, работающих в ключевом режиме (т.е. сопротивление открытого диода  $r_{д.о.} \rightarrow 0$ , а сопротивление закрытого диода  $r_{д.з.} \rightarrow \infty$ ). Резистор  $R$  выбирается из условия:

$$r_{д.о.} \ll R \ll r_{д.з.}$$

Выходное напряжение снимается с двух параллельно включенных диодов, сопротивления которых совместно с резистором  $R$  образует обыкновенный делитель напряжения. Возможны 3 случая:

1) Оба входных напряжения низкие

$$u_{вч1} = u_{вч2} = 0$$

и следовательно оба диода открыты. Тогда общее сопротивление диодов:

$$r = r_{д.о1} \parallel r_{д.о2} = 0$$

$$R \gg r$$

Практически все напряжение  $E$  падает на  $R$ , а на выходе низкий потенциал  $u_{вых} = 0$ .

2) Один  $u_{вч}$  низкий, а другой высокий. Пусть  $u_{вч1}$  низкий. Следовательно:

$$r_{д.о1} = 0, \quad r_{д.з2} \rightarrow \infty$$

$$r = r_{д.о1} \parallel r_{д.з2} = 0$$

$$R \gg r$$

Все напряжение  $E$  падает на  $R$ . На выходе низкий потенциал.

3) Оба входных сигнала высокие. Тогда:

$$r_{д.з1} \rightarrow \infty, \quad r_{д.з2} \rightarrow \infty$$

$$r \rightarrow \infty$$

$$R \ll r$$

Теперь все напряжение будет падать на  $r$  и на выходе будет высокий потенциал  $u_{вых} = E$ .

### Схема собирания (схема "ИЛИ")

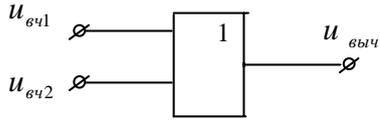


Рис. 2.5. Функциональная схема

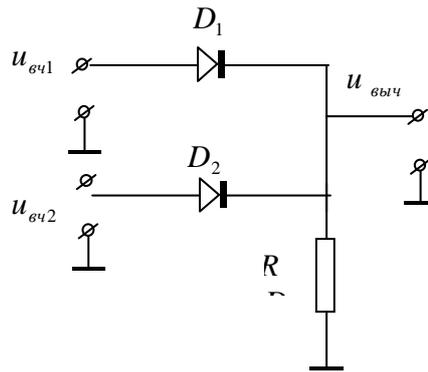


Рис. 2.6. Схема электрическая принципиальная

$$r_{д.о.} \ll R \ll r_{д.з.}$$

Анализ работы схемы производится аналогично предыдущей схеме.

### Схема И – НЕ

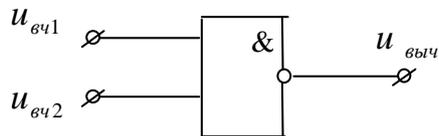


Рис. 2.7. Функциональная схема

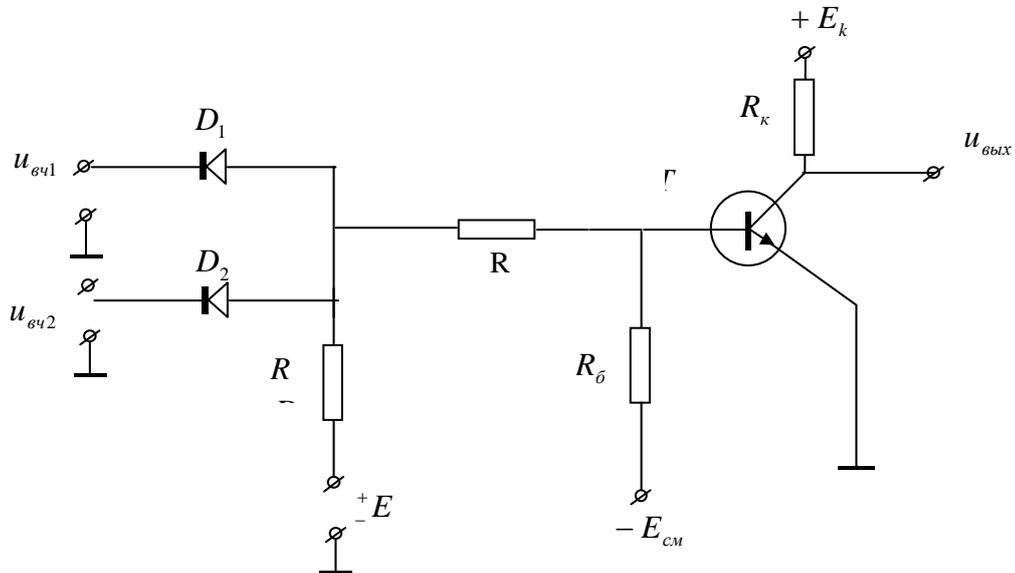


Рис. 2.8. Схема электрическая принципиальная

## Схема ИЛИ – НЕ

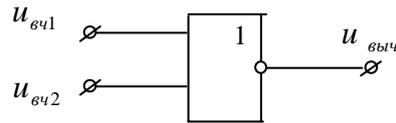


Рис. 2.9. Функциональная схема

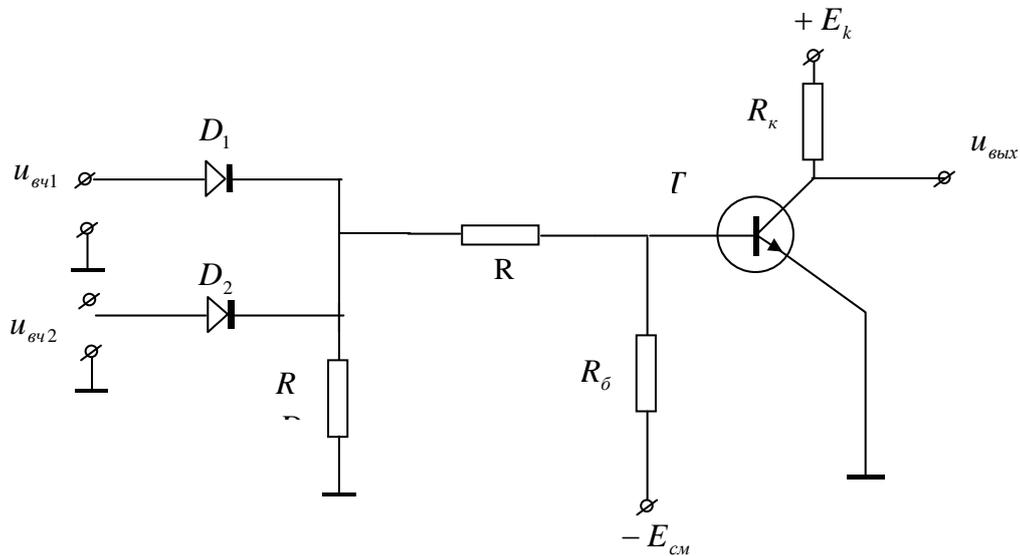


Рис. 2.10. Схема электрическая принципиальная

### Функциональные элементы

Функциональным элементом называется элемент реализующий несколько булевых функций.

Рассмотрим только один функциональный элемент – триггер. В переводе с английского слово триггер переводится как защелка. Триггером называется элементарный автомат имеющий два устойчивых состояния. Триггер используется для хранения двоичной информации. Рассмотрим структурную схему триггера на элементах ИЛИ – НЕ:

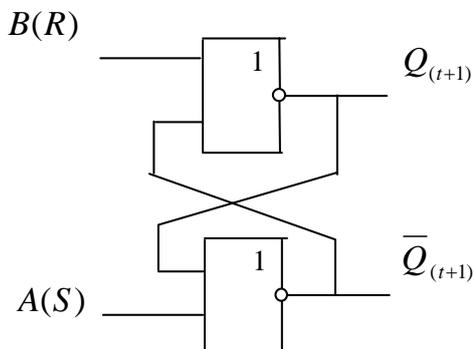


Рис. 2.11. Структурная схема триггера на элементах ИЛИ – НЕ

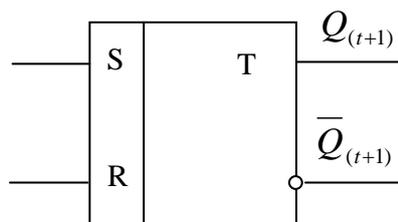


Рис. 2.12. Функциональная схема RS - триггера

**Таблица переходов RS – триггера**

$S_t$	$R_t$	$Q_t$	$Q_{t+1}$	$\bar{Q}_{t+1}$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	-	-
1	1	1	-	-

Принято считать, что триггер перешел в единичное состояние, если на выходе Q появился потенциальный сигнал низкого уровня. Соответственно за переход в нулевое состояние триггера принято считать появление на выходе Q сигнала высокого уровня. Для передачи единичного сигнала из триггера необходимо изменить его состояние с 1 на 0 и получить на выходе Q изменение потенциального сигнала с низкого на высокий.

### Типовые функциональные схемы

Типовые функциональные схемы строятся на основе логических и функциональных элементов. К ним относятся:

#### **Регистры.**

Регистром называется устройство, предназначенное для временного хранения двоичной информации. Каждый разряд регистра используется для ввода, вывода и хранения одного разряда двоичного числа. Рассмотрим, например, схему 3-х разрядного регистра:

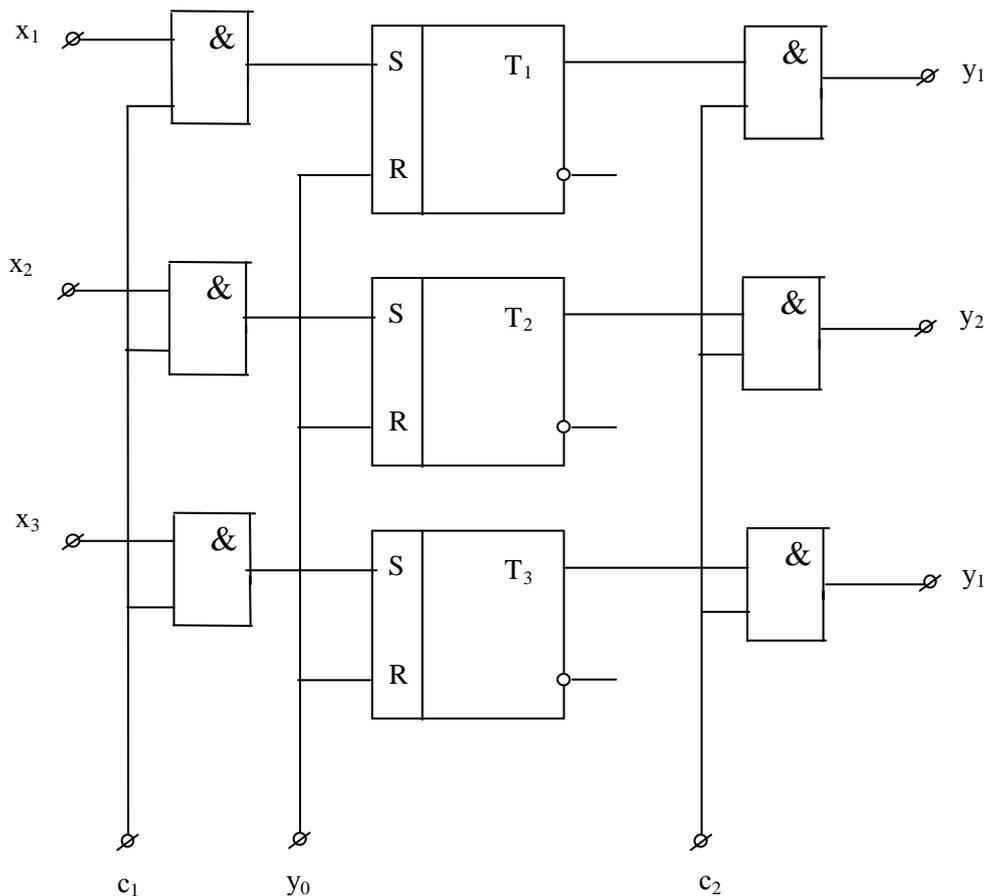


Рис. 2.13. Схема 3-х разрядного регистра

В начале работы все триггеры устанавливаются в 0 сигналом  $y_0$ . Конъюнкторы на входе пропускают двоичную информацию на входы S триггеров только при наличии синхроимпульса  $c_1$ . Для передачи двоичного числа из регистра на входы конъюнкторов необходимо на R входы триггеров подать импульс  $y_0$  который перебросят триггеры из 1→0. При переходе из 1→0 на входы конъюнкторов поступят сигналы передачи 1. При этом они пройдут в другие устройства только при наличии синхроимпульса  $c_2$ .

### Счетчики.

Назначением счетчиков является:

подсчет числа импульсов, поступающих на вход счетчика

временное хранение каждого состояния

преобразование непрерывного сигнала, заданного последовательностью импульсов в двоичный код

Счетчики бывают прямого счета (суммирующие), вычитающие, реверсивные.

### Дешифраторы

Дешифратором называется устройство, позволяющее преобразовать сочетание входных сигналов (входной код) в сигнал только на одном из выходов. С помощью дешифратора двоичный код числа преобразуется в сигнал, управляющий выбором соответствующего блока, схемы или устройства.

Схема дешифратора представляет собой матрицу логических элементов и содержит в общем случае  $n$  входов и  $2^n$  выходов. Каждой комбинации входных сигналов соответствует появление сигнала только на одном из выходов.

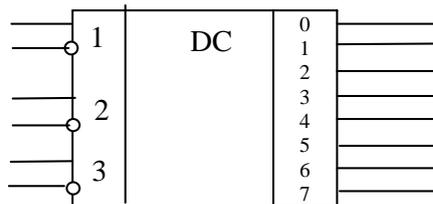


Рис. 2.14. Функциональная схема дешифратора

### §2.3. Принципы работы компьютера

*Компьютер компьютеру рознь!  
Один компьютер бывалоча все чего-то  
вычисляет, рисует, играет и ещё  
невесть что делает... А другой – бдит и  
студентов мучает!*

Из рассказов бывалого студента,  
прошедшего компьютерное тестирование

#### Что такое "компьютер"?

Слово компьютер произошло от английского compute – вычислять. Ранее в нашей стране применялся термин "электронно – вычислительная машина" (ЭВМ), что, на мой взгляд, более точно отражает суть дела. Дело в том, что в основе любого современного компьютера лежат электронные устройства, о которых мы говорили выше.

Автоматизация работ с информацией (данными) имеет свои особенности и отличия от автоматизации других видов работ. Для этого класса задач используют особые устройства, большинство из которых являются электронными приборами и устройствами. Совокупность устройств, предназначенных для автоматической или автоматизированной обработки данных, называют *вычислительной техникой*.

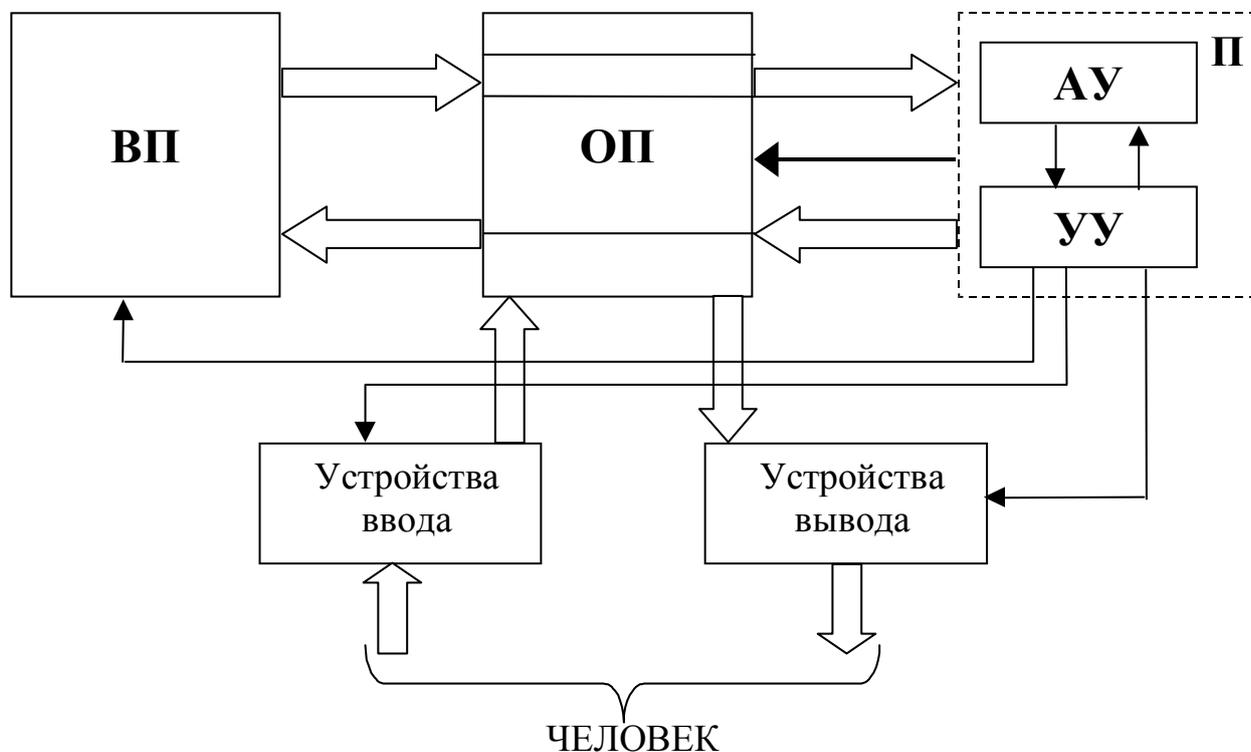
Конкретный набор взаимодействующих между собой устройств и программ называют *вычислительной системой*. Центральным устройством любой вычислительной системы является *компьютер*.

***Итак, компьютер – это электронный прибор, предназначенный для автоматизации создания, хранения, обработки и передачи данных.***

#### Простейшая структура компьютера

Принципы работы всех компьютеров одинаковы и носят название *принципов фон – Неймана*.

Общность принципов работы компьютеров и определяет общность структур компьютеров. Рассмотрим простейшую структуру компьютера лишь с целью уяснения основных принципов работы компьютера.



Здесь двойные стрелки означают передачу информации (данных), одинарные стрелки означают управляющие сигналы процессора. На рисунке процессор обозначен буквой П. Процессор – это устройство, которое управляет всеми остальными устройствами компьютера и выполняет арифметические и логические операции над данными, находящимися в памяти.

Ту часть процессора, которая предназначена для непосредственного выполнения операций называют *арифметическим устройством (АУ)*. Остальные функции процессора, как уже было сказано, – это управление работой различных устройств. Поэтому часть процессора, выполняющий эти функции, называют *устройством управления (УУ)*.

Память компьютера предназначена для хранения информации и команд программы. Информация, хранящаяся в памяти компьютера, представляет собой закодированные с помощью всего двух цифр 0 и 1 числа, буквы, символы, слова.

Команды, хранящиеся в памяти – это закодированные с помощью 0 и 1 инструкции компьютеру по выполнению определенных действий. Например, командой может служить предписание компьютеру сложить два числа, выдать сообщение на экран и т.п. Совокупность команд образует программу, которая и определяет последовательность действий компьютера.

Стрелка от ОП к П отражает два факта:

1) Процессор в своей работе руководствуется командами программы, выбираемых из памяти .

2) Аргументы выполняемых операций тоже выбираются из памяти.

Стрелка **П** → **ОП** отражает тот факт, что получаемые в процессоре результаты отправляются на запоминание в память.

Память компьютера состоит из отдельных ячеек – байтов, каждый из которых имеет свой номер, называемый адресом. Из байтов образуются более сложные структуры памяти, например, полуслово (два байта), слово (четыре байта), двойное слово (восемь байтов). Однако, внутри этих структур, компьютер может обращаться к отдельным байтам. Таким образом, минимальной адресуемой единицей памяти компьютера является байт.

Память компьютера делится на *основную (ОП)* и *внешнюю (ВП)*.

Основная память – это память высокого быстродействия, по скорости приближающаяся к скорости процессора, но ограниченной емкости. Конструктивно он выполнен в одном корпусе с процессором и является центральной частью компьютера. **ОП** состоит в свою очередь из *оперативной* и *постоянной* памяти. Оперативная память служит для хранения оперативной (нужной в данный момент) информации, предназначенной для решения данной задачи. При решении другой задачи в оперативной памяти будет храниться уже другая информация для другой задачи.

При отключении компьютера вся информация в оперативной памяти теряется.

Постоянная память (**ПЗУ** – постоянное запоминающее устройство) предназначена для хранения постоянной (но часто нужной) информации, которая не зависит от того, какая задача решается. Чаще всего в ней хранится служебная информация, необходимая процессору. При отключении компьютера информация в постоянной памяти сохраняется. Кроме того, информацию в постоянной памяти нельзя стирать, изменять, дополнять.

Внешняя память (**ВП**) предназначена для долговременного хранения информации независимо от того, работает компьютер или нет и характеризуется очень большим объемом. Информацию во внешней памяти можно изменять, дополнять, уничтожать и т.д.

Для ввода данных с внешних носителей в **ОП** служат устройства *ввода*. Результаты решения задачи запоминаются в памяти компьютера.

Для выдачи результатов, хранящихся в памяти служат устройства *вывода*.

### **Принципы фон - Неймана**

1. Принцип произвольного доступа к ячейкам памяти.

Компьютер на каждом такте своей работы может с одинаковым успехом обращаться к любым ячейкам памяти – как для чтения, так и для записи. Для обеспечения такого произвольного доступа к памяти каждой ячейке дается индивидуальное имя – эти имена и используются для указания нужных ячеек. Как правило, все ячейки пронумерованы от 0 до N-1, где N – общее число ячеек памяти и в качестве имени ячейки принимается ее порядковый номер, который называют адресом ячейки, а также адресом ее

содержимого (по аналогии с обычным почтовым адресом). Этот принцип обеспечивает большую свободу и простоту использования информации в памяти компьютера.

## 2. Принцип хранимой программы.

Компьютер выполняет программу, которая представлена в цифровой (двоичной) форме и хранится в той же памяти, что и данные, подлежащие переработке по этой программе. Перед началом решения задачи программа вводится (записывается) в отведенные для нее ячейки памяти и хранится там во время выполнения этой программы компьютером.

Принцип хранимой программы, прежде всего, делает компьютер универсальным в том смысле, что на одном и том же компьютере можно решать самые разные задачи, поскольку для решения каждой из них можно составить программу, ввести ее в память и заставить компьютер выполнить эту программу.

Факт представления программы в такой же форме, что и другие данные (например, числа) и хранения программы в памяти открывает целый ряд и других возможностей. Например, программа в процессе своего выполнения также может подвергаться переработке, что позволяет задавать в самой программе правила получения некоторых ее частей и тем самым получать весьма компактные программы. Более того, одна программа может являться результатом работы другой программы, что открывает принципиальную возможность автоматизировать процесс составления программы, поручив эту работу самой машине путем выполнения специальных программ.

## Машинные операции.

Каждый тип компьютера имеет свой набор машинных операций, т.е. элементарных действий (с точки зрения программиста) по преобразованию информации. Например, имеется операция сложения двух чисел. Причем сложение с фиксированной точкой и сложение с плавающей представляют собой две разные операции. Решение любой задачи должно быть сведено к выполнению последовательности операций из этого набора.

С точки зрения удобства использования желательно, чтобы набор операций был возможно шире и, чтобы он включал достаточно содержательные операции. С точки же зрения простоты реализации желательно, чтобы этот набор был невелик и, чтобы в него входили лишь простейшие операции, т.к. это позволяет сделать компьютер более простым, дешевым и надежным. Эти два требования противоречивы и на практике выбирается некоторый компромиссный вариант.

В каждой ячейке памяти хранится упорядоченная последовательность символов, которая называется *машинным словом* или просто словом.

Машинное слово может, например, представлять собой отдельное число, изображенное с достаточно высокой степенью точности, так что за одно обращение к памяти можно выбрать из нее сразу все число, являющийся одним из аргументов арифметической операции или запомнить результат вы-

полнения этой операции.

Машинное слово может представлять собой и отдельную команду программы. Команда – это предписание компьютеру выполнить одну операцию из ее набора операций. Компьютер предназначен для выполнения не какого-нибудь одного, а многих разных алгоритмов.

Это обстоятельство требует наличия специального языка, который был бы понятен машине. Такой язык называется *машинным языком*, запись алгоритма на машинном языке называется *программой*, а самостоятельная фраза языка, являющаяся некоторой операцией, называется *командой*. Таким образом, программа представляет собой некоторую последовательность команд, а выполнение компьютерной программы сводится к последовательному выполнению отдельных ее команд.

В общем случае команда должна содержать информацию в том, какую именно операцию нужно выполнить, информацию об ее операндах (аргументах и результате) и информацию о том, какую команду надо выполнить вслед за данной командой.

Информация об операндах и следующей команде, подлежащей выполнению, задается путем указания их адресов. На каждом типе компьютеров фиксируется определенный формат команд, позиции (разряды) машинного слова, представляющего собой команду, объединяются в определенные поля, и каждое поле служит для задания информации определенного назначения. Одно из полей, называемое полем кода операции, служит для указания вида операции.

Для этого все операции, входящие в набор машинных операций компьютера, нумеруются (кодируются) и в поле операции команды указывается номер (код) требуемой операции.

Например, команда может иметь следующий формат:

$\theta$	A1	A2	A3	A4
Поле кода операции	Поле первого адреса	Поле 2-го адреса	Поле 3-го адреса	Поле адреса следующей команды

Например:

0010	0100	0110	1000	1100
------	------	------	------	------

Такая команда называется *четырёхадресной*. Она обеспечивает принудительный порядок выполнения команд в компьютере, т.к. из памяти выбирается и выполняется та команда, адрес которой записан в четвертом поле адресной части команды. В настоящее время подавляющее распространение получили компьютеры с естественным порядком выполнения команд, при котором команды нумеруются при программировании и в этом порядке записываются в память, затем в этом же порядке выбираются из памяти и выполняются. При естественном порядке надобность в четвертом адресе

отпадает, и команда становится 3-х адресной.

$\theta$	A1	A2	A3
----------	----	----	----

Это структура наиболее точно отвечает формуле

$$Z=X \theta Y,$$

что представляет известные удобства при программировании. Но эта структура с точки зрения технической реализации не является оптимальной. Структуру команды можно значительно упростить, если результат операции оставлять в сумматоре АУ или заносить в специальную ячейку, например в А1.

Тогда надобность в 3-м адресе отпадает.

$\theta$	A1	A2
----------	----	----

И, наконец, структура команды становится наиболее простой, если одним из двух операндов считать результат предыдущей операций, зафиксированным в сумматоре.

$\theta$	A
----------	---

#### **§2.4. Классификация компьютеров**

Существуют различные системы классификации компьютеров. Рассмотрим некоторые из них.

##### **Классификация по назначению**

Классификация по назначению – одна из наиболее ранних систем классификации. Он связан с тем, как компьютер применяется. По этому принципу различают большие ЭВМ, мини – ЭВМ, микро – ЭВМ и персональные компьютеры, которые в свою очередь, подразделяются на массовые, деловые, портативные, развлекательные и рабочие станции.

##### **Большие ЭВМ**

Это самые мощные компьютеры. Их применяют для очень сложных расчетов, например в космических исследованиях, ядерной физике, в военных целях и др. За рубежом компьютеры этого класса называют мейнфреймами (mainframe). Для обслуживания таких компьютеров необходимо несколько десятков человек. На базе таких суперкомпьютеров создаются вычислительные центры, включающие в себя несколько отделов или групп.

##### **Мини – ЭВМ**

От больших ЭВМ компьютеры этой группы отличаются уменьшенными размерами и, соответственно, меньшей производительностью и стоимостью. Такие компьютеры используются крупными предприятиями, научными учреж-

дениями и некоторыми высшими учебными заведениями, сочетающими учебную деятельность с научной.

Мини – ЭВМ часто применяются для управления производственными процессами. Для обслуживания мини – ЭВМ также требуется создание вычислительного центра, хотя и не такой многочисленный, как для больших ЭВМ.

### **Микро – ЭВМ**

Компьютера данного класса доступны многим предприятиям и организациям. Для обслуживания этих компьютеров обычно не создаются вычислительные центры, т.к. для этого достаточно всего несколько человек. Микро – ЭВМ чаще всего используются как вспомогательные компьютеры для больших ЭВМ. Например, они используются для предварительной подготовки данных, перед тем как их будет обрабатывать большая ЭВМ.

### **Персональные компьютеры (ПК)**

Эта категория компьютеров получила особо бурное развитие в течение последних двадцати лет. Из названия видно, что такой компьютер предназначен для обслуживания одного рабочего места. Как правило, с персональным компьютером работает один человек. Несмотря на свои небольшие размеры и относительно невысокую стоимость, современные персональные компьютеры обладают немалой производительностью. Последние модели персональных компьютеров превосходят по производительности и особенно по своим возможностям большие ЭВМ 70-х годов, мини – ЭВМ 80-х годов и микро – ЭВМ первой половины 90-х годов 20 века.

Особенно широкую популярность персональные компьютеры приобрели после 1995 года в связи с бурным развитием Интернета. Возможностей персонального компьютера вполне достаточно для использования всемирной сети в качестве источника научной, справочной, учебной, культурной и развлекательной информации. Персональные компьютеры являются удобным средством автоматизации учебного процесса по любым дисциплинам, средством организации дистанционного (заочного) обучения и средством организации досуга.

До последнего времени персональные компьютеры условно рассматривали в двух категориях: бытовые ПК и профессиональные ПК. Бытовые модели имели, как правило, меньшую производительность, но в них были приняты особые меры для работы с цветной графикой и звуком, чего не требовалось для профессиональных моделей. В связи с достигнутым в последние годы резким удешевлением средств вычислительной техники, границы между профессиональными и бытовыми моделями практически стерлись.

Начиная с 1999 года в области персональных компьютеров действует международный классификационный стандарт – *спецификация PC99*. Он регламентирует принципы классификации персональных компьютеров и оговаривает минимальные и рекомендуемые требования к каждой категории персональных компьютеров. Новый стандарт устанавливает следующие категории

персональных компьютеров:

- Consumer Personal Computer (PC) (массовый ПК);
- Office PC (деловой ПК);
- Mobile PC (портативный ПК);
- Workstation PC (рабочая станция);
- Entertainment PC (развлекательный ПК).

Согласно спецификации PC99 большинство персональных компьютеров попадают в категорию массовых ПК. Для деловых ПК минимизированы требования к средствам воспроизведения графики, а к средствам работы со звуковыми данными требования вообще не предъявляются. Для портативных ПК обязательным является наличие средств для создания соединений удаленного доступа, то есть средств компьютерной связи. В категории рабочих станций повышены требования к устройствам хранения данных, а категории развлекательных ПК – требования к средствам воспроизведения графики и звука.

Однако жизнь не стоит на месте. Классификация PC99 уже устарела, и мы стоим на пороге новой классификации. Грани между указанными категориями персональных компьютеров также практически стерлись. Сейчас уже невозможно представить себе офисный компьютер без соответствующих мультимедийных средств, а на развлекательном ПК можно производить достаточно сложные вычисления и расчеты.

**Копилка** Под термином *мультимедиа* подразумевается сочетание нескольких видов данных в одном документе (текстовые, графические, музыкальные и видеоданные) или совокупность устройств для воспроизведения этих данных.

Персональные компьютеры получили столь широкое и повсеместное распространение, их применение настолько разнообразное, что сейчас уже не говорят персональные компьютеры, а говорят просто компьютеры. В дальнейшем мы тоже не будем подчеркивать "персональность" компьютеров и будем говорить просто о компьютерах.

### **Другие системы классификации компьютеров**

**Классификация по уровню специализации.** По уровню специализации компьютеры делят на *универсальные* и *специализированные*. На базе универсальных компьютеров можно собирать вычислительные системы произвольной конфигурации. Так, например, один и тот же компьютер можно использовать для работы с текстами, музыкой, графикой, фото и видеоматериалами. Специализированные компьютеры предназначены для решения конкретного круга задач. К таким компьютерам относятся, например, бортовые компьютеры автомобилей, судов, самолетов, космических аппаратов. Бортовые компьютеры решают только специальные задачи, на которые они запрограммированы на заводе изготовителе (например, оптимизацию расхода топлива двигателя в зависимости от конкретных условий движения или управление средствами ориентации и навигации и т.д.).

Специализированные компьютеры, ориентированные на работу с графикой, называют *графическими станциями*. Специализированные компьютеры, объединяющие компьютеры какого – нибудь предприятия в одну сеть, называют *файловыми серверами*. Компьютеры, обеспечивающие передачу информации между различными участками Интернета, называют *сетевыми серверами*.

Во многих случаях с задачами специализированных компьютеров могут успешно справляться и обычные универсальные компьютеры, и так часто на практике и происходит, хотя считается, что использование специализированных компьютеров все – таки эффективнее. Здесь опять таки идет тенденция к сближению обеих категорий компьютеров.

**Классификация по типоразмерам.** Различают *настольные (desktop)*, *портативные (notebook)*, *карманные (palmtop)*.

Настольные модели распространены наиболее широко. Они являются принадлежностью рабочего места. Эти модели отличаются простотой изменения конфигурации за счет несложного подключения дополнительных внешних устройств или установки дополнительных внутренних компонентов. Достаточные размеры корпуса в настольном исполнении позволяют выполнять большинство подобных работ без привлечения специалистов, а это позволяет настраивать компьютерную систему оптимально для решения именно тех задач, для которых она была приобретена.

Портативные модели удобны для транспортировки. Их используют бизнесмены, коммерсанты, руководители предприятий и организаций, проводящие много времени в командировках и переездах. Особая привлекательность портативных компьютеров связана с тем, что их можно использовать в качестве средства связи.

Карманные модели выполняют функции "интеллектуальных записных книжек". Они позволяют хранить оперативные данные и получать к ним быстрый доступ. Некоторые карманные модели имеют жестко встроенное программное обеспечение, что облегчает непосредственную работу, но снижает гибкость в выборе прикладных программ.

**Классификация по совместимости.** В мире существуют множество различных видов и типов компьютеров. Они выпускаются разными производителями, работают с разными программами. При этом очень важным вопросом становится *совместимость* различных компьютеров между собой. От совместимости зависит возможность переноса программ с одного компьютера на другой и возможность совместной работы разных типов компьютеров с одними и теми же данными.

*Аппаратная совместимость.* По аппаратной совместимости различают так называемые *аппаратные платформы*. В области персональных компьютеров на сегодняшний день наиболее широко распространены две аппаратные платформы – *IBM PC* и *Apple Macintosh*. Принадлежность компьютеров к одной платформе повышает совместимость между собой, принадлежность к разным платформам соответственно понижает совместимость, но, тем не менее, говорить о полной несовместимости этих двух платформ нельзя.

Кроме аппаратной совместимости существуют и другие виды совместимо-

сти: *совместимость на уровне операционной системы, программная совместимость, совместимость на уровне данных.*

## §2.5. *Аппаратное обеспечение*

В §2.3. мы на элементарном уровне рассмотрели принципы работы компьютера и основные его устройства. Рассмотрим теперь более подробно и на современном уровне устройства, из которых состоит компьютер.

Состав вычислительной системы называется *конфигурацией*. Аппаратные и программные средства вычислительной техники принято рассматривать отдельно. Соответственно, отдельно рассматривают *аппаратную конфигурацию* вычислительных систем и их *программную конфигурацию*.

К *аппаратному обеспечению* вычислительных систем относятся устройства и приборы, образующие аппаратную конфигурацию. Современные компьютеры и вычислительные комплексы имеют блочно – модульную конструкцию. Аппаратную конфигурацию, необходимую для выполнения конкретных видов работ, можно собирать из готовых узлов и блоков.

По способу расположения устройств относительно *центрального процессорного устройства* (ЦПУ – *Central Processing Unit, CPU*) различают внутренние и внешние устройства. Внешними, как правило, являются большинство устройств ввода/вывода данных (их также часто называют *периферийными устройствами*).

Согласование между отдельными узлами и блоками выполняют с помощью переходных аппаратно – логических устройств, называемых *аппаратными интерфейсами*. Стандарты на аппаратные интерфейсы в вычислительной технике называют *протоколами*.

**КОПИ/КА** Таким образом, *протокол – это совокупность технических условий, которые должны быть обеспечены разработчиками устройств для успешного согласования их работы с другими устройствами.*

Многочисленные интерфейсы, присутствующие в архитектуре любой вычислительной системы, можно условно разбить на две большие группы: *последовательные* и *параллельные*. Через последовательный интерфейс данные передаются последовательно, бит за битом, а через параллельный – одновременно группами битов. Количество битов, участвующих в одной посылке, определяется разрядностью интерфейса. Например, восьмиразрядные параллельные интерфейсы передают 8 бит (один байт) за один цикл.

Параллельные интерфейсы обычно имеют более сложное устройство, чем последовательные, но обеспечивают более высокую производительность. Их применяют там, где важна скорость передачи данных. Производительность параллельных интерфейсов измеряют *байтами в секунду* (байт/с; Кбайт/с; Мбайт/с).

В последовательных интерфейсах, как правило, не надо синхронизировать работу передающего и принимающего устройства (поэтому их часто назы-

вают *асинхронными интерфейсами*), но пропускная способность их меньше и коэффициент полезного действия ниже, так как из-за отсутствия синхронизации полезные данные предваряются и завершаются посылками служебных данных. Т.е. на один байт полезных данных могут приходиться 1 – 3 служебных бита (состав и структуру посылки определяет конкретный протокол).

Поскольку обмен данными через последовательные устройства производится не байтами, а битами, их производительность измеряют битами в секунду (бит/с; Кбит/с; Мбит/с). Несмотря на кажущую простоту перевода единиц измерения скорости последовательной передачи в единицы измерения скорости параллельной передачи путем механического деления на 8, такой пересчет не выполняют, поскольку он некорректен из-за наличия служебных данных.

Последовательные интерфейсы применяют для подключения "медленных" устройств (устройств печати, устройств ввода/вывода знаковой и сигнальной информации, например клавиатуры, контрольных датчиков, малопроизводительных устройств связи и т.п.).

## **Базовая аппаратная конфигурация**

Компьютер – универсальная техническая система. Его *конфигурацию* (состав оборудования) можно гибко изменять по мере необходимости. Тем не менее, существует понятие *базовой конфигурации*, которую считают типовой. В таком комплекте компьютер обычно поставляется. В настоящее время в базовой конфигурации рассматривают четыре устройства:

- системный блок;
- монитор;
- клавиатура;
- мышь.

## **Системный блок**

Системный блок представляет собой основной узел, внутри которого установлены наиболее важные компоненты. Устройства, находящиеся в системном блоке, называются *внутренними*, а устройства, подключаемые к нему снаружи, называются *внешними*.

По внешнему виду системные блоки отличаются формой корпуса. Корпуса выпускаются в горизонтальном (*desktop*) и вертикальном (*tower*) исполнении. Корпуса, имеющие вертикальное исполнение, различают по габаритам: *полноразмерный (big tower)*, *среднеразмерный (midi tower)* и *малоразмерный (mini tower)*. Среди корпусов, имеющих горизонтальное исполнение, выделяют *плоские* и *особо плоские (slim)*.

Кроме формы, для корпуса важен параметр, называемый *форм – фактором*. В настоящее время в основном используются корпуса двух форм – факторов:

АТ и АТХ. Форм – фактор корпуса должен быть обязательно согласован с форм – фактором главной (системной) платы компьютера, так называемой *материнской платы*.

Корпуса компьютеров поставляются вместе с блоком питания и, таким образом, мощность блока питания также является одним из параметров корпуса. Для массовых моделей достаточной является мощность блока питания 200 – 250 Вт.

Одним из основных устройств, находящихся в системном блоке является материнская плата. На ней размещаются:

- *процессор* – основная микросхема, выполняющая математические и логические операции;
- *микروпроцессорный комплект (чипсет)* – набор микросхем, управляющих работой внутренних устройств компьютера и определяющая основные функциональные возможности материнской платы;
- *шины* – наборы проводников, по которым происходит обмен сигналами между устройствами компьютера;
- *оперативная память* – набор микросхем, предназначенных для временного хранения данных, когда компьютер включен;
- *ПЗУ (постоянное запоминающее устройство)* – микросхема, предназначенная для длительного хранения данных, в том числе и когда компьютер выключен;
- разъемы для подключения дополнительных устройств (*слоты*).

Рассмотрим эти устройства более подробно.

### **Оперативная память**

*Оперативная память (RAM – Random Access Memory)* – это массив кристаллических ячеек, способных хранить данные. Существуют много различных типов оперативной памяти, но с точки зрения физического принципа действия различают *динамическую память (DRAM)* и *статическую память (SRAM)*.

*Ячейки динамической памяти (DRAM)* можно представить в виде микроконденсаторов, способных накапливать заряд на своих обкладках. Это наиболее распространенный и экономически доступный тип памяти. Недостатки этого типа связаны, во-первых, с тем, что как при заряде, так и при разряде конденсаторов неизбежны переходные процессы, то есть запись данных происходит сравнительно медленно. Второй важный недостаток связан с тем, что заряды ячеек имеют свойство рассеиваться в пространстве, причем весьма быстро. Если оперативную память постоянно не "подзаряжать", утрата данных происходит через несколько сотых долей секунды. Для борьбы с этим явлением в компьютере происходит постоянная *регенерация (освежение, подзарядка)* ячеек оперативной памяти. Регенерация происходит несколько десятков раз в секунду и вызывает непроизводительный расход ресурсов компьютера.

*Ячейки статической памяти (SRAM)* можно представить как электронные микроэлементы – *триггеры*, состоящие из нескольких транзисторов. В триггере хранится не заряд, а состояние (включен/выключен), поэтому этот тип памяти обеспечивает более высокое быстродействие, хотя технологически он сложнее и, соответственно, дороже.

Микросхемы динамической памяти используют в качестве основной оперативной памяти компьютера. Микросхемы статической памяти используют в качестве вспомогательной памяти (так называемой *кэш-памяти*), предназначенной для оптимизации работы процессора.

Каждая ячейка памяти имеет свой адрес, который выражается числом. В настоящее время в процессорах Intel Pentium и некоторых других принята 32-разрядная адресация, а это означает, что всего независимых адресов может быть  $2^{32}$ . Таким образом, в современных компьютерах возможна непосредственная адресация к полю памяти размером  $2^{32}=4\ 294\ 967\ 296$  байт (4,3 Гбайт). Однако это отнюдь не означает, что именно столько оперативной памяти непременно должно быть к компьютеру. Предельный размер поля оперативной памяти, установленной в компьютере, определяется микропроцессорным комплектом (чипсетом) материнской платы и обычно составляет несколько сот Мбайт.

Одна адресуемая ячейка содержит восемь двоичных ячеек, в которых можно сохранить 8 бит, то есть один байт данных. Таким образом, адрес любой ячейки памяти можно выразить четырьмя байтами.

Представление о том, сколько оперативной памяти должно быть в типовом компьютере, непрерывно меняется. В середине 80-х годов поле памяти размером в 1 Мбайт казалось огромным, в начале 90-х достаточным считался объем 4 Мбайт, к середине 90-х годов он увеличился до 8 Мбайт, а затем и до 16 Мбайт. Сегодня типичным считается размер "оперативки" 64 Мбайт, но очень скоро эта величина будет превышена в несколько раз даже для моделей массового потребления.

Оперативная память в компьютере размещается на стандартных панельках, называемых *модулями*. Модули оперативной памяти вставляют в соответствующие разъемы на материнской плате. Конструктивно модули памяти имеют два исполнения – однорядные (*SIMM-модули*) и двухрядные (*DIMM-модули*). На компьютерах с процессором Pentium однорядные модули можно применять только парами (количество разъемов для их установки на материнской плате всегда четное), а DIMM-модули можно устанавливать по одному. Многие модели материнских плат имеют разъемы как того, так и другого типа, но комбинировать на одной плате модули разных типов нельзя.

Основными характеристиками модулей оперативной памяти являются объем памяти и время доступа. SIMM-модули поставляются объемами 4, 8, 16, 32 Мбайт, а DIMM-модули 16, 32, 64, 128 Мбайт и более. Время доступа показывает, сколько времени необходимо для обращения к ячейкам памяти – чем оно меньше, тем лучше. Время доступа измеряется в миллиардных долях секунды (*наносекундах, нс*). Типичное время доступа к оперативной памяти для SIMM-модулей – 50-70 нс. Для современных DIMM-модулей оно составляет

## Процессор

Процессор – основная микросхема компьютера, в которой и производятся все вычисления. Конструктивно процессор состоит из ячеек, похожих на ячейки оперативной памяти, но в этих ячейках данные могут не только храниться, но и изменяться. Внутренние ячейки процессора называются *регистрами*. Важно также отметить, что данные, попавшие в некоторые регистры, рассматриваются не как данные, а как команды, управляющие обработкой данных в других регистрах. Среди регистров процессора есть и такие, которые в зависимости от своего содержания способны модифицировать исполнение команд. Таким образом, управляя засылкой данных в разные регистры процессора, можно управлять обработкой данных. На этом и основано исполнение программ.

С остальными устройствами компьютера, и в первую очередь с оперативной памятью процессор связан несколькими группами проводников, называемых *шинами*. Основных шин три: *шина данных, адресная шина и командная шина*.

**Адресная шина.** У процессоров Intel Pentium адресная шина 32-разрядная, то есть состоит из 32 параллельных линий. В зависимости от того, есть напряжение на какой-то линии или нет, говорят, что на этой линии выставлена единица или ноль. Комбинация из 32 нулей и единиц образует 32-разрядный адрес, указывающих на одну из ячеек оперативной памяти. К ней и обращается процессор для копирования содержимого этой ячейки в один из своих регистров либо для записи содержимого регистра в эту ячейку.

**Шина данных.** По этой шине непосредственно и происходит процесс копирования данных из оперативной памяти в регистры процессора и обратно. В компьютерах с процессором Intel Pentium шина данных 64-разрядная, т.е. состоит из 64 линий, по которым за один раз (такт или цикл) на обработку поступает сразу 8 байтов.

**Шина команд.** Для того, чтобы процессор мог обрабатывать данные, ему нужны команды. Он должен знать, что следует сделать с теми байтами, которые хранятся в его регистрах. Эти команды в соответствии со вторым принципом фон Неймана поступают в процессор тоже из оперативной памяти, но не из тех областей, где хранятся данные, а оттуда, где хранятся программы. Команды тоже представлены в виде байтов. Самые простые команды укладываются в один байт, однако есть и такие, для которых нужно два, три и более байтов. В большинстве современных процессоров шина команд 32-разрядная (например, в процессоре Intel Pentium), хотя существуют 64-разрядные и даже 128-разрядные.

**Система команд процессора.** Совокупность всех возможных команд, которые может выполнить процессор над данными, образует так называемую *систему команд процессора*. Процессоры, относящиеся к одному семейству,

имеют одинаковые или близкие системы команд. Процессоры, относящиеся к разным семействам, различаются по системе команд и невзаимозаменяемы.

**Процессоры с расширенной и сокращенной системой команд.** Чем шире набор системных команд процессора, тем сложнее его архитектура, тем длиннее формальная запись команды (в байтах), тем выше средняя продолжительность исполнения одной команды, измеренная в тактах работы процессора. Так, например, система команд процессоров Intel Pentium в настоящее время насчитывает более тысячи команд. Такие процессоры называют *процессорами с расширенной системой команд – CISC-процессорами (CISC – Complex Instruction Set Computing)*.

В противоположность CISC-процессорам в середине 80-х годов появились процессоры архитектуры RISC с сокращенной системой команд (*RISC – Reduced Instruction Set Computing*). При такой архитектуре количество команд в системе намного меньше, и каждая из них выполняется намного быстрее. Таким образом, программы, состоящие из простейших команд, выполняются этими процессорами много быстрее. Обратная сторона сокращенного набора команд состоит в том, что сложные операции приходится эмулировать далеко не эффективной последовательностью простейших команд сокращенного набора.

В результате конкуренции между двумя подходами к архитектуре процессоров сложилось следующее распределение их сфер применения:

- CISC-процессоры используют в универсальных вычислительных системах;
- RISC-процессоры используют в специализированных вычислительных системах или устройствах, ориентированных на выполнение единообразных операций.

Для компьютеров платформы IBM PC долгое время выпускались только CISC-процессоры, к которым относятся и все процессоры семейства Intel Pentium. Однако в последнее время компания AMD приступила к выпуску процессоров семейства AMD-K6, в основе которых лежит внутреннее ядро, выполненное по RISC-архитектуре, и внешняя структура, выполненная по архитектуре CISC. Таким образом, сегодня появились процессоры, имеющие гибридную структуру.

**Совместимость процессоров.** Если два процессора имеют одинаковую систему команд, то они полностью совместимы на программном уровне. Это означает, что программа, написанная для одного процессора, может исполняться и другим процессором. Процессоры, имеющие разные системы команд, как правило, несовместимы или ограниченно совместимы на программном уровне.

Группы процессоров, имеющих ограниченную совместимость, рассматривают как *семейство процессоров*. Так, например, все процессоры Intel Pentium относятся к так называемому семейству x86. Родоначальником этого семейства был 16-разрядный процессор Intel 8086, на базе которого собиралась первая модель компьютера IBM PC. Впоследствии выпускались процессоры

Intel 80286, Intel 80386, Intel 80486, Intel Pentium 60, 66, 75, 90, 100, 133; несколько моделей процессоров Intel Pentium MMX, модели Intel Pentium Pro, Intel Pentium II, Intel Celeron, Intel Xeon, Intel Pentium III и другие. Все эти модели, и не только они, а также многие модели процессоров компаний AMD и Cyrix относятся к семейству x86 и обладают совместимостью по принципу "сверху вниз".

Принцип совместимости "сверху вниз" – это пример неполной совместимости, когда каждый новый процессор "понимает" все команды своих предшественников, но не наоборот. Это естественно, поскольку двадцать лет назад разработчики процессоров не могли предусмотреть систему команд, нужную для современных программ.

Благодаря такой совместимости на современном компьютере можно выполнять любые программы, созданные в последние десятилетия для любого из предшествующих компьютеров, принадлежащего той же аппаратной платформе.

## Монитор

Монитор – устройство визуального представления данных. Это не единственно возможное, но главное устройство вывода. Его основными потребительскими параметрами являются: размер и шаг маски экрана, максимальная частота регенерации изображения, класс защиты.

*Размер экрана монитора* измеряется между противоположными углами трубки кинескопа по диагонали. Единица измерения – дюймы. Стандартные размеры: 14"; 15"; 17"; 19"; 20"; 21". В настоящее время наиболее универсальными являются мониторы размером 15 и 17 дюймов, а для операций с графикой желательны мониторы размером 19 – 21 дюйм.

Изображение на экране монитора получается в результате облучения люминофорного покрытия остронаправленным пучком электронов, разогнанных в вакуумной колбе. Для получения цветного изображения люминофорное покрытие имеет точки или полосы трех типов, светящиеся красным, зеленым и синим цветом. Чтобы на экране все три луча сходились строго в одну точку и изображение было четким, перед люминофором ставят маску – панель с регулярно расположенными отверстиями или щелями. Часть мониторов оснащена маской из вертикальных проволочек, что усиливает яркость и насыщенность изображения. Чем меньше шаг между отверстиями или щелями (*шаг маски*), тем четче и точнее полученное изображение. Шаг маски измеряют в долях миллиметра. В настоящее время наиболее распространены мониторы с шагом маски 0,25 – 0,27 мм. Устаревшие мониторы могут иметь шаг до 0,43 мм, что негативно сказывается на органах зрения при работе с компьютером. Модели повышенной стоимости могут иметь шаг маски менее 0,25 мм.

*Частота регенерации (обновления)* изображения показывает, сколько раз в течении секунды монитор может полностью сменить изображение (поэтому

её также называют *частотой кадров*). Частоту регенерации измеряют в герцах (Гц). Чем она выше, тем четче и устойчивее изображение, тем меньше утомление глаз, тем больше времени можно работать на компьютере без перерыва. При частоте регенерации порядка 60 Гц мелкое мерцание изображения заметно невооруженным глазом. Сегодня такое значение считается недопустимым. Минимальным считают значение 75 Гц, нормативным – 85 Гц, а комфортным – 100 Гц и более.

*Класс защиты* монитора определяется стандартом, которому соответствует монитор с точки зрения техники безопасности. В настоящее время общепризнанными считаются следующие международные стандарты: *MPR-II*, *TCO-92*, *TCO-95*, *TCO-99* (приведены в хронологическом порядке). Стандарт MPR-II ограничил уровни электромагнитного излучения пределами, безопасными для человека. В стандарте TCO-92 эти нормы были сохранены, а в стандартах TCO-95 и TCO-99 ужесточены. Эргономические и экологические нормы впервые появились в стандарте TCO-95, а стандарт TCO-99 установил самые жесткие нормы по параметрам, определяющим качество изображения (яркость, контрастность, мерцание, антибликовые свойства покрытия).

## Клавиатура

Клавиатура – это клавишное устройство управления компьютером. Она служит для ввода *алфавитно – цифровых (знаковых)* данных, а также команд управления. Клавиатура и монитор обеспечивают простейший *интерфейс пользователя*. С помощью клавиатуры управляют компьютером, а с помощью монитора получают от него отклик.

Клавиатура относится к стандартным средствам персонального компьютера. Ее основные функции не нуждаются в поддержке специальными системными программами (драйверами). Необходимое программное обеспечение для начала работы с компьютером уже имеется в микросхеме ПЗУ в составе базовой системы ввода/вывода (*BIOS – Basic Input Output System*), и потому компьютер реагирует на нажатие клавиш сразу после включения компьютера. Принцип действия клавиатуры заключается в следующем:

1. При нажатии на клавишу (или комбинацию клавиш) специальная микросхема, встроенная в клавиатуру, выдает так называемый *скан-код*.
2. Скан-код поступает в микросхему, выполняющая функции *порта клавиатуры*.

<b>КОПИ/ЖА</b> Порты – специальные аппаратно – логические устройства, отвечающие за связь процессора с другими устройствами.
--

3. Порт клавиатуры выдает процессору прерывание с фиксированным номером. Для клавиатуры номер прерывания – 9 (Interrupt 9, Int 9).

4. Получив прерывание, процессор откладывает текущую работу и по номеру прерывания обращается в специальную область оперативной памяти, в которой находится так называемый *вектор прерываний*. Вектор прерываний – это список адресных данных с фиксированной длиной записи. Каждая запись содержит адрес программы, которая должна обслужить прерывание с номером, совпадающим с номером записи.
5. Определив адрес начала программы – обработчика возникшего прерывания, процессор переходит к ее исполнению. Простейшая программа обработки клавиатурного прерывания "защита" в микросхему ПЗУ, но программисты могут "подставить" вместо нее свою программу обработки данного прерывания, если изменят данные в векторе прерываний.
6. Программа-обработчик прерывания направляет процессор к порту клавиатуры, где он находит скан-код, загружает его в свои регистры, потом под управлением обработчика определяет, какой код символа соответствует данному скан-коду.
7. Далее обработчик прерываний направляет полученный код символа в небольшую область памяти, известную как *буфер клавиатуры* и прекращает свою работу, известив об этом процессор.
8. Процессор прекращает обработку данного прерывания и возвращается к отложенной задаче.
9. Введенный символ хранится в буфере клавиатуры до тех пор, пока его не заберет оттуда та программа, для которой он и предназначался, например, текстовый редактор.

## **Мышь**

Мышь – устройство управления манипуляторного типа. Перемещение мыши по плоской поверхности синхронизировано с перемещением графического объекта (*указателя мыши*) на экране монитора.

В отличие от клавиатуры, мышь не является стандартным органом управления и компьютер не имеет для нее выделенного порта. Для мыши нет и постоянного выделенного прерывания, а базовые средства ввода и вывода (VIOS) компьютера, размещенные в постоянном запоминающем устройстве (ПЗУ), не содержат программных средств для обработки прерываний мыши.

В связи с этим в первый момент после включения компьютера мышь не работает. Она нуждается в поддержке специальной системной программы – *драйвера мыши*.

Драйвер устанавливается либо при первом подключении мыши, либо при установке операционной системы компьютера. Хотя мышь и не имеет выделенного порта на материнской плате, для работы с ней используют один из стандартных портов, средства для работы с которыми имеются в составе VIOS. Драйвер мыши предназначен для интерпретации сигналов, поступающих

через порт. Кроме того, он обеспечивает механизм передачи информации о состоянии и положении мыши операционной системе и работающим программам.

Компьютером управляют перемещением мыши по плоскости и кратковременными нажатиями правой или левой кнопок (клавиш). Эти нажатия называются *щелчками (click)*. В отличие от клавиатуры мышь не может напрямую использоваться для ввода знаковой информации – ее принцип управления является событийным. Перемещения мыши и щелчки ее кнопок являются *событиями* с точки зрения ее программы – драйвера. Анализируя эти события, драйвер устанавливает, когда произошло событие и в каком месте экрана в этот момент находился указатель. Эти данные передаются в прикладную программу, с которой работает пользователь в данный момент. По ним программа может определить команду, которую имел в виду пользователь и приступить к ее исполнению.

Комбинация монитора и мыши обеспечивает наиболее современный тип интерфейса пользователя, который называется *графическим*. Пользователь наблюдает на экране графические *объекты* и *элементы управления*. С помощью мыши пользователь изменяет свойства объектов и приводит в действие элементы управления компьютером, а с помощью монитора получает отклик в графическом виде.

### **Контрольные вопросы:**

1. Краткая история развития средств вычислительной техники.
2. Логические элементы вычислительной техники.
3. Функциональные элементы и типовые функциональные схемы.
4. Принципы фон – Неймана.
5. Классификация компьютеров.
6. Базовая аппаратная конфигурация компьютеров.

## Глава III. Программное обеспечение компьютеров

*Вопрос: "Что такое слон?"*

*Ответ: "Это мышь с операционной системой"*

Программистский фольклор

### §3.1. Структура программного обеспечения

Программное обеспечение составляет вторую неотъемлемую компоненту современных компьютеров, без которого невозможно их функционирование.

Программное обеспечение – это совокупность программ, позволяющих организовать решение задач пользователя. По назначению программное обеспечение можно условно разбить на три класса:



Системное программное обеспечение организует процесс обработки информации в компьютере. По сути – это программное устройство управления компьютером. Основу системного программного обеспечения составляют операционные системы (ОС).

Основное назначение ОС – управление процессом обработки информации и организация связи пользователя с компьютером. ОС – является неотъемлемой частью компьютера, обеспечивая управление всеми аппаратными средствами, а человеку (пользователю) позволяет управлять работой компьютера с помощью соответствующего графического интерфейса и (или) команд.

Ядро ОС дополняется набором сервисных программ, которые служат разным целям. С их помощью производится например, начальная разметка дисков, установка параметров внешних устройств, выдача информации на печать, тестирование ОП и других устройств, стыковка с другими компьютерами (локальная сеть) и многое другое.

В состав системного программного обеспечения входят также средства диагностики и контроля. Эти программы обеспечивают автоматический поиск

неисправностей в компьютере. Средства диагностики и контроля могут входить в состав ОС, но могут существовать и автономно.

Система программирования (СП) позволяет разрабатывать программы на удобном для восприятия человеком языке, а не в машинных кодах. В СП входят также компиляторы – комплекс программ, обеспечивающих автоматический перевод программ написанных на символическом языке на машинный язык.

Системы программирования – это особая категория программных средств. С их помощью создаются другие программы, в том числе и прикладные программы. Следовательно, эта категория программных средств совершенно аналогична средствам производства в промышленности – таким как станки, инструменты, сырье и т.д. При этом роль сырья играет информация – текстовые и числовые данные, закодированные сообщения, графическая информация и др.

Прикладное программное обеспечение составляют те программы ради которых и существует компьютер. Это различные пакеты программ, предназначенные для пользователей различных категорий. Например, это могут быть различные текстовые редакторы, СУБД, табличные процессоры и интегрированные программные средства.

Прикладные программы составляют категорию программных средств, обращенных к пользователям компьютера – людям, которые не обязаны уметь программировать или знать устройство компьютера. Их цель заключается либо в том, чтобы с помощью компьютера решать свои повседневные задачи, включая профессиональные, либо учиться определенным навыкам (не обязательно относящихся к компьютеру), либо проводить свой досуг, играя в компьютерные игры.

Прикладные системы могут иметь общий характер, например, обеспечивать составление документов и их печать или хранение и выдачу различных справок.

Другие классы прикладных программ ориентированы на автоматизацию конкретных видов деятельности, например, обучение определенным предметам в школе, проектирование зданий и сооружений, анализ электрокардиограмм, проведение бухгалтерских расчетов и т.д. и т.п.

## §3.2. Основы Windows-95

### Графический интерфейс Windows – 95

Операционная система Windows – 95, разработанная фирмой Microsoft, обеспечивает большое количество возможностей и удобств для пользователей и программистов. Прежде всего, Windows – 95 обеспечивает удобный и наглядный графический интерфейс пользователя (так называемый стандарт GUI – Graphical User Interface), а также обеспечивает многозадачный режим работы, что означает возможность одновременно запускать и работать с несколькими программами. Каждая программа работает в своем собственном окне. Управление окнами в Windows – 95 стандартизовано. Это значительно облегчает изучение самых разных программных средств. Рассмотрим некоторые элементы графического интерфейса:

### Рабочий стол Windows – 95

На рисунке 1 изображен вид рабочего стола сразу после включения компьютера.

Рабочий стол можно ассоциировать с настоящим рабочим столом в Вашем

Папки и ярлыки

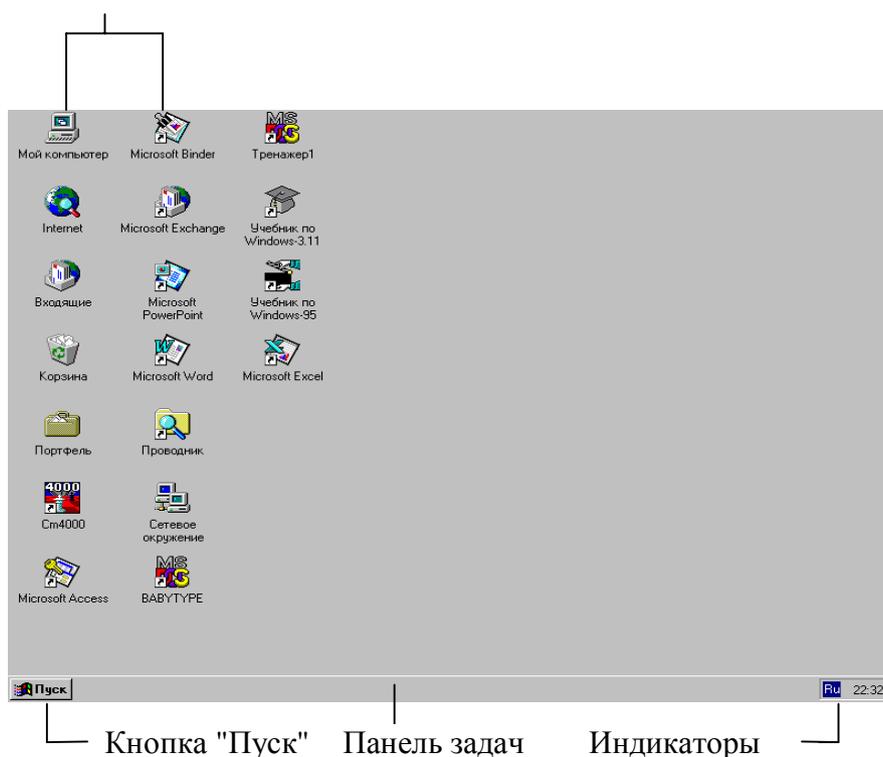


Рис. 1

офисе или дома. На нем можно расположить всё, что нужно Вам для работы. На рабочем столе расположены значки папок и значки ярлыков наиболее часто используемых программ. Количество значков на рабочем столе зависит от способа установки Windows и от Вашей настройки рабочего стола.

Значки ярлыков отличаются от значков папок наличием в левом нижнем углу небольшой стрелочки. (См. рис. 2). Ярлык относится к какому – либо приложению или документу. (Обратите внимание, что понятия программа и приложение означают одно и тоже). В дальнейшем мы будем использовать



Рис. 2

оба этих термина.

Чтобы запустить приложение или документ достаточно дважды щелкнуть по его ярлыку на рабочем столе, при этом, если вы открываете документ, то автоматически запускается приложение, которое обрабатывает этот документ.

Существует и другой способ запуска программ. Для этого надо воспользоваться кнопкой "Пуск". Если нажать мышью на эту кнопку, то появится **меню**.

Меню это такой механизм, который предоставляет пользователю возможность выбора дальнейших действий. Каждое меню состоит из отдельных пунктов (которые в свою очередь могут также быть меню – так называемые вложенные меню, говорят ещё подменю) или команд.

Итак, чтобы запустить приложение необходимо нажать кнопку "Пуск" и путешествуя по нужным пунктам меню и подменю добраться до нужной программы и нажать левую кнопку мыши или клавишу Enter.

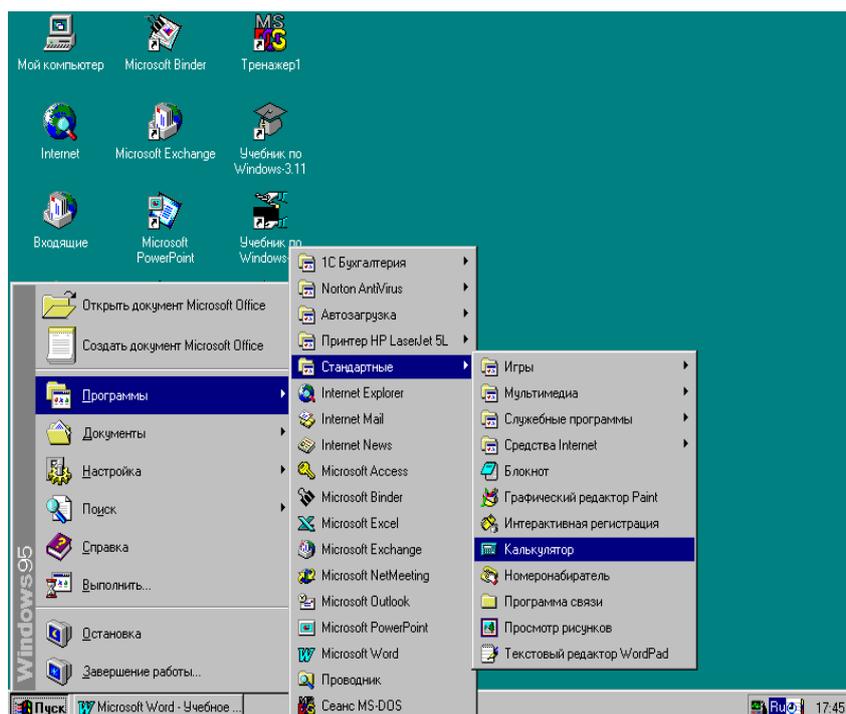


Рис. 3

На рисунке 3 показан пример запуска программы Калькулятор.

**Внимание!** При выключении компьютера всегда следует использовать кнопку "Пуск". Если выключить компьютер "просто так", то работа Windows завершится некорректно, могут быть потеряны некоторые данные и может нарушиться работоспособность Windows!

## Панель задач

Панель задач отображает значки работающих в данный момент приложений и позволяет быстро переключаться с одной задачи на другую. Для этого надо просто щелкнуть по соответствующему значку программы на панели задач. После закрытия приложения его значок также исчезает с панели задач.

## Индикаторы.

Индикаторы дают некоторую дополнительную информацию для пользователя. Например, в правом нижнем углу отображаются часы. Рядом слева находится индикатор используемого языка.

En (English) – означает, что в данный момент используется английский язык.

Ru (Russian) – означает, что в данный момент используется русский язык.

Для переключения с одного языка на другой используются клавиши Shift + Alt (знак + означает, что эти клавиши нужно нажимать одновременно). Если индикатор языка виден на экране, то можно просто щелкнуть по индикатору.

## Элементы окон Windows – 95

Как уже отмечалось, управление окнами в Windows – 95 стандартизовано. Практически все окна имеют одни и те же элементы управления. Рассмотрим их на рисунке 4.

Любое окно имеет *строку заголовка* в котором указывается имя программы и имя документа.

При нажатии кнопки "Свернуть" активное окно уменьшается до размеров значка и помещается на панель задач. Следует иметь в виду, что программа в этом случае продолжает работать, но в свернутом виде. При нажатии кнопки "Развернуть" окно разворачивается до максимально возможных размеров. При этом кнопка "Развернуть" заменяется на кнопку "Восстановить" 

Если щелкнуть по этой кнопке, то окно восстановит свой первоначальный размер. Чтобы закрыть окно, щелкните по кнопке "Закрыть".

Вторая строка окна называется *строкой меню*. Если щелкнуть по какому – либо пункту, то появится *подменю* – вертикальное меню в котором можно

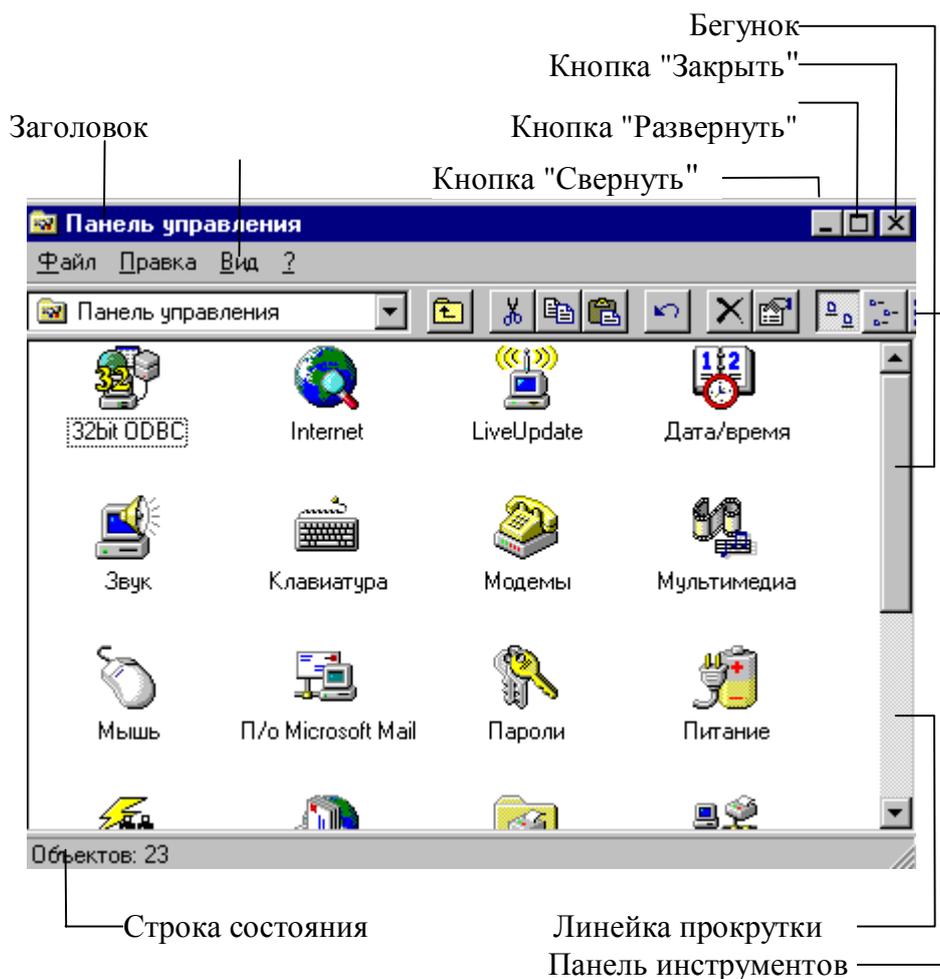


Рис. 4

выбрать соответствующие команды.

**Панель инструментов** позволяет выполнять некоторые действия быстрее, используя кнопки на панели и не обращаясь к пунктам меню.

Кнопки на панели инструментов дублируют некоторые, наиболее часто используемые команды меню.

**Линейка прокрутки** используется в том случае, когда вся информация не помещается в окне. Если нажать на кнопки  ,  то информация в окне будет сдвигаться вверх или вниз.

**Бегунок** показывает относительное положение в тексте. Например, если бегунок находится вверху линейки прокрутки, то вы находитесь ближе к началу документа, если внизу, то ближе к концу документа. Можно прокручивать текст и с помощью бегунка. Для этого надо нажать на бегунок мышью и, не отпуская двигать бегунок в нужном направлении. Линейки прокрутки бывают вертикальные и горизонтальные. Методы работы с горизонтальными линейками прокрутки абсолютно идентичны. В тех случаях, когда информация полностью помещается в окне, линейки прокрутки не выводятся.

**Строка состояния** отображает состояние различных элементов в данном окне. В дальнейшем мы познакомимся с соответствующими примерами, ра-

ботая в Word – 97.

## Меню

В окне каждой программы находится строка меню. Это так называемое горизонтальное меню. Каждый пункт этого меню состоит в свою очередь из соответствующих подменю – вертикальных меню. Для того чтобы открыть подменю надо щелкнуть по соответствующему пункту горизонтального меню. Каждый пункт меню содержит либо команды с помощью которых можно выполнить какие – то действия, либо режимы (опции), которые можно активизировать или дезактивизировать, либо содержат ещё дополнительные подменю (вложенные меню).

### Соглашения по обозначениям в меню

Таблица 1

Соглашения	Значение соглашения
Треугольник (►) после команды	Означает, что этот пункт содержит вложенное меню (подменю).
Многоточие (...) после команды	После выбора этого пункта появится диалоговое окно. Оно содержит необходимые для выполнения команды параметры, которые нужно выбрать или ввести.
Комбинация клавиш после команды	Эта комбинация клавиш является "быстрыми" клавишами для команды, т.е. для выполнения этой команды можно нажать на указанную комбинацию клавиш, не открывая меню.
Блеклый (или невидимый) пункт	В данный момент этот пункт (команду) нельзя использовать
Галочка (:) перед командой	Означает, что данная команда или режим активизирована. Повторное нажатие отключает команду или режим. Значок : при этом исчезает.

## Диалоговые окна

Для выполнения некоторых команд необходимо задать один или несколько параметров. Для этого используются *диалоговые окна*. Если после команды в меню стоит многоточие, то после её выбора открывается диалоговое окно.

Диалоговые окна содержат специальные области (называемые элементами

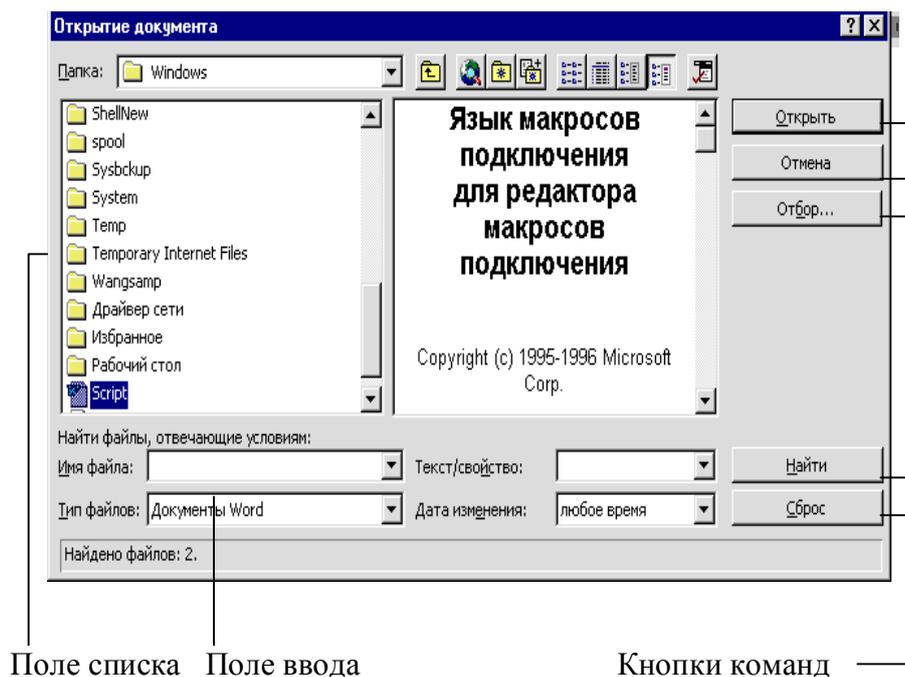


Рис. 5

управления или параметрами). Значения параметров необходимо выбрать из некоторого списка или ввести с клавиатуры.

### Типы параметров диалоговых окон.

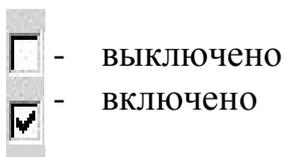
**Кнопки команд** – используются для выполнения действий. Все диалоговые окна обязательно имеют кнопки **ОК** и **Отмена**. При нажатии кнопки ОК диалог завершается и команда с выбранными вами параметрами выполняется. При нажатии кнопки Отмена окно диалога просто закрывается и команда не исполняется. Наличие остальных кнопок зависит от вида команды.

**Поле ввода** – используется для ввода значения параметра с клавиатуры.

**Поле списка** – требуемый параметр выбирается из списка. Выбор осуществляется простым щелчком по нужному элементу списка. При необходимости можно воспользоваться линейкой прокрутки.

**Раскрывающийся список** – используется в целях экономии места на экране. Чтобы раскрыть этот список нажмите на треугольник. В дальнейшем работа с этим списком полностью аналогична работе с обычным списком.

**Флажок** – это поле может иметь только два значения: включено/выключено и означает, что некоторый режим включен или выключен. Соглашения здесь таковы:



Чтобы включить или отключить флажок нужно просто щелкнуть по флажку. **Переключатель** – позволяет выбрать только один из возможных режимов. В этом его различие от флажка. Выбранный режим отмечается кружочком с точкой внутри.

**Вкладки** – некоторые команды используют очень большое количество различных параметров. В этом случае диалоговые окна оформляются в виде вкладок или подшивок. Чтобы выбрать некоторую группу параметров нужно нажать на требуемую вкладку. Рисунок 6.



Рис.6

## Файловая система Windows – 95

### Диски

Любая информация в компьютере хранится на диске. Диск вращается в устройстве подобно компакт – диску в проигрывателе, что позволяет записывать и считывать информацию с диска. Устройство в котором находится диск называется накопителем или дисководом. Диски бывают гибкие и жесткие, а также компакт – диски (CD – ROM). Жесткие диски находятся внутри компьютера в так называемом системном блоке.

Каждый диск имеет однобуквенное обозначение (имя). Дисковод для гибких дисков имеет обозначение А. Жесткий диск имеет обозначение С. И, нако-

нец, CD-ROM имеет обозначение D.

Большая часть информации в компьютере хранится на жестком диске. Гибкие диски обычно используются для копирования с жесткого диска (или, наоборот, на жесткий диск) информации, например, для перенесения информации с одного компьютера на другой. На жестком диске помещается столько информации, сколько помещается на сотнях (и даже на тысячах) гибких дисках. CD – ROM также имеет очень большой объем. Обычно на CD – ROM поставляется программное обеспечение, игры и т.д.

## **Файлы, папки, документы**

Файл – это область памяти на диске предназначенный для хранения информации. Информацией может быть программа, написанная на каком – либо языке программирования (например, Бейсик, Паскаль, Си), программа готовая к исполнению (в машинном коде), данные, необходимые для работы программы, текстовые документы. Кроме того, в файлах может содержаться: числовые данные, закодированная табличная, графическая и любая другая информация.

Каждый файл имеет *имя* и в зависимости от характера содержащейся в файле информации *тип*.

Имя файла – это последовательность букв, цифр и некоторых знаков, расположенных в произвольном порядке.

Тип (говорят ещё расширение имени файла) – это последовательность из трех букв, цифр или знаков. Между именем файла и типом всегда ставится точка. Например, Годовой отчет.doc

Здесь: Годовой отчет – имя файла, doc – тип файла.

В большинстве случаев пользователю не обязательно присваивать файлу тип. Приложение, в котором обрабатывается данный файл, автоматически присвоит ему нужный тип.

Основной единицей информации, которую обрабатывают приложения Windows – 95 является *документ*. Каждый документ хранится на диске в виде файла. Программы также хранятся на диске в виде файлов.

Без большой потери строгости можно считать, что файл и документ это одно и то же. Дело в том, что Windows и Microsoft Office создавались прежде всего для автоматизации учрежденческой деятельности. А в любом офисе или учреждении основная единица информации это документ. В дальнейшем мы будем употреблять оба этих понятия, не делая никаких различий между ними.

Обычно на диске хранятся сотни и тысячи документов. Для упорядочения их используют *папки*.

Папка – это группа документов, объединенных по какому – либо признаку. Папки можно ассоциировать с выдвижными ящичками стола, в каждом из которых хранятся документы.

Каждая папка должна иметь имя и может, в свою очередь, входить в другую

папку как единое целое (вложенные папки). Так образуется иерархическая файловая система Windows. На каждом диске имеется так называемый корневой каталог. В нем регистрируются файлы и папки 1-го уровня. В последующих регистрируются файлы и папки следующего 2-го уровня и т.д.

При такой сложной структуре файловой системы для нахождения нужного файла (документа) необходимо указать не только имя документа, но и всю цепочку вложенных папок. Такая цепочка называется *путем*. При записи пути каждый из вложенных папок отделяется от другого знаком \. Пример записи пути:

C:\Windows\Бухгалтерские документы\Отчеты\Годовой отчет.doc

Здесь C: - имя диска;

Windows – папка верхнего (первого) уровня;

Бухгалтерские документы – вложенная папка (второго уровня);

Отчеты – вложенная папка (третьего уровня);

Годовой отчет.doc – имя документа;

Из примера видно, что папки не имеют расширения имени.

## Проводник Windows – 95

С помощью Проводника можно просматривать содержимое дисков и управлять файлами. Под управлением файлами понимается выполнение различных операций над ними как целыми неделимыми объектами (копирование, перемещение, удаление, создание новых папок, удаление ненужных и пр.).

Чтобы открыть окно Проводника выполните одно из следующих действий:

- Если ярлык для Проводника создан и виден на Рабочем столе, щелкните по этому ярлыку;
- Щелкните по кнопке Пуск, выберите пункт Программы, а затем щелкните по пункту Проводник;
- Щелкните правой кнопкой по значку Мой компьютер, затем выберите пункт Проводник;
- Щелкните правой кнопкой по кнопке Пуск, затем выберите пункт Проводник;

Окно Проводника выглядит так, как показано на рисунке 7.

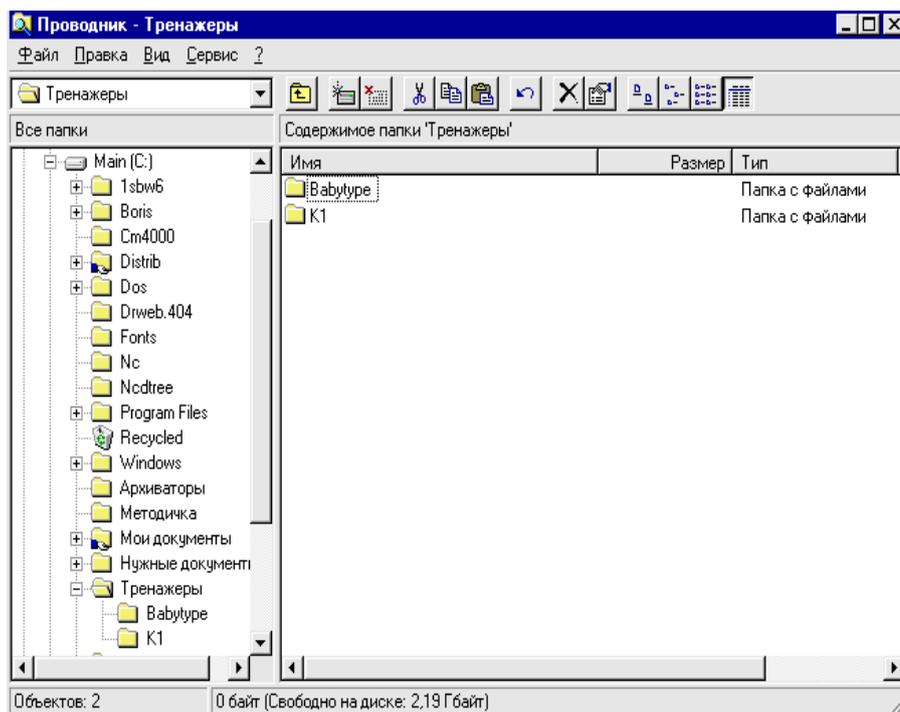


Рис. 7 Окно Проводника

Окно Проводника разбито на две части. Левое окно называется **Все папки**, а правое окно – **Содержимое папки**. В верхней части окна под строкой меню имеется раскрывающийся список, который называется **Переход к другой папке**. В той же строке находится панель инструментов Проводника.

### Выбор файла или папки

Чтобы выбрать папку или файл просто щелкните по имени файла или папки. Выделенный объект будет подсвечен. Если вы выделили папку в левом окне, то он откроется автоматически.

Чтобы открыть документ или папку в правом окне нужно дважды щелкнуть по имени документа или папки.

Чтобы перейти на один уровень вверх в иерархии папок щелкните по кнопке с изображением желтой папки с маленькой стрелкой вверх на панели инструментов.

### Создание новой папки

Документы близкие по смыслу и одинаковое предназначение удобнее хранить в одной папке (например, документы с отчетами). Для создания новой папки необходимо:

- Открыть папку, в которую нужно вложить новую папку.
- В окне *Содержимое папки* (правое окно) щелкнуть в свободном месте

правой кнопкой.

- Из появившегося контекстного меню выбрать пункт **Создать**, а затем пункт **Папку**. В списке файлов и папок появится новая папка с именем, присвоенным ей по умолчанию.
- Чтобы изменить имя папки, просто наберите его в строке ввода и нажмите Enter.

## Копирование и перемещение файлов и папок

Для выполнения этих действий:

- Выделите нужный файл или папку.
- Щелкните правой клавишей по имени выбранного файла или папки. Выберите команду **Копировать** (или, если хотите переместить, выберите команду **Вырезать**).
- Перейдите в ту папку, в которую хотите копировать или переместить. В свободном месте окна щелкните правой клавишей. Выберите команду **Вставить**

При копировании файл или папка остается на прежнем месте (копия отправляется на новое место). При перемещении файл или папка физически перемещается на новое место.

## Удаление файла или папки

Если Вам некоторые файлы и (или) папки не нужны, то можно их удалить, чтобы они не занимали лишнего места на диске.

Для удаления файла:

- Открыть папку, в которой находится удаляемый файл.
- Выделить нужный файл. Нажать на правую кнопку мыши
- Из появившегося контекстного меню выбрать пункт **Удалить**.
  - На экран выведется диалоговое окно **Подтверждение удаления файла**. Если вы уверены, что хотите действительно удалить этот файл, нажмите кнопку **Да**

Для удаления папки:

- Выделить нужную папку. Нажать правую кнопку мыши.
- Из контекстного меню выбрать пункт **Удалить**.

На экран выведется диалоговое окно **Подтверждение удаления папки**. Если вы уверены, что действительно хотите удалить эту папку –нажмите кнопку **Да**

#### **К Вашему сведению:**

1. При удалении папки удаляются и все вложенные в нее папки и файлы.
2. Все удаляемые папки и файлы помещаются в так называемую *корзину*.
3. Если Вы все же передумали, то ещё можете восстановить папки и файлы, взяв их обратно из корзины. Для этого на рабочем столе откройте корзину, дважды щелкнув по значку корзины. Выберите нужные папки и файлы. В меню **Файл** выберите пункт **Восстановить**.
4. Время от времени не забывайте очищать корзину от "мусора". Для этого в меню **Файл** выберите команду **Очистить**

## **Переименование файла или папки**

Чтобы переименовать файл или папку надо щелкнуть по имени правой клавишей и выбрать команду **Переименовать**

## **Выделение группы файлов или папок**

Чтобы выделить группу файлов или папок:

1. Если файлы или папки расположены подряд, то нажмите клавишу Shift и, не отпуская её, щелкните по имени первого файла или папки, затем последнего. После чего отпустите клавишу Shift.
2. Если файлы или папки расположены не подряд, то нужно нажать клавишу Ctrl и, не отпуская её щелкать по именам нужных файлов и папок.

С выделенной группой файлов и папок можно выполнять операции, так как будто это один файл. Однако это могут быть только операции копирования, перемещения и удаления.

## **Создание ярлыков**

Ярлыки – это одно из удобств, которые предоставляет Windows – 95. Они позволяют быстро запустить программу или быстро открыть документ простым двойным щелчком по ярлыку на рабочем столе, без долгих поисков нужного файла и копания в папках. Чтобы создать ярлык, надо щелкнуть по имени файла правой клавишей и из контекстного меню выбрать команду **Создать ярлык**.

Присвойте ярлыку имя. Для того чтобы переместить ярлык на рабочий стол нажмите на ярлык мышью и, не отпуская клавишу, перенесите ярлык на ра-

бочий стол, затем отпустите клавишу.

Если вы хотите, чтобы ярлык оставался и в той папке, где он был создан, то прежде чем переносить его на рабочий стол нажмите на клавишу Ctrl и, не отпуская его, нажмите мышью на ярлык и "тащите" его на рабочий стол.

Кстати, это ещё один способ копирования и перемещения – так называемый метод "перенести – оставить"

### **Контрольные вопросы:**

1. Перечислите и объясните значение элементов управления окнами в Windows – 95;
2. Где находится панель задач и что она отображает?
3. Что такое меню?
4. Как переключиться с русского языка на английский и наоборот?
5. Чем отличается флажок от переключателя?
6. Чем отличается сворачивание окна от её закрытия?
7. Что такое файл?
8. Что такое путь?
9. Как создать новую папку?
10. Чем отличается перемещение от копирования?

## Глава IV. Текстовый редактор Word – 97

### §4.1. Введение в Word – 97

Текстовый редактор Word – 97 на сегодняшний день является самой распространенной и лучшей программой обработки текстов. Word – 97 обладает широчайшими возможностями и позволяет реализовать полный цикл работ по подготовке документов, статей и книг любой сложности и структуры, включая ввод текста, его оформление различными шрифтами, использованием колонтитулов, контроля правописания, вставки в текст рисунков, созданием таблиц, оглавления и предметного указателя. Word – 97 позволяет внедрять в документ различные объекты, созданные в других приложениях Microsoft Office.

Все это делает текстовый редактор Word – 97 весьма привлекательным для пользователя.

### Запуск Word – 97

Существуют несколько способов запуска Word – 97:

- Если на Рабочем столе создан ярлык Word – 97, то достаточно дважды щелкнуть по его ярлыку.
- Нажмите кнопку "Пуск", выберите пункт **Программы**, найдите строку **Microsoft Word** и щелкните по нему.
- Нажмите кнопку "Пуск" правой клавишей, выберите пункт **Проводник**, откройте папку **Программы**, найдите строку **Microsoft Word** и щелкните по нему.

В дальнейшем вы познакомитесь и другими способами запуска Word.

### Элементы окна Word – 97

После запуска Word появится окно как показано на рис. 8. Поскольку Word является приложением Windows – 95, то все стандартные кнопки управления окном здесь также присутствуют. Здесь необходимо сделать замечание. Существуют два вида окон – окно приложения и окно документа. Управление ими совершенно идентично. Самые верхние кнопки "Свернуть", "Развернуть" и "Закреть" относятся к программе Word, а нижние к документу.

Таким образом, если вы хотите закрыть Word, то нажимаете на верхнюю кнопку "Закреть", а если вы хотите закрыть документ, но хотите продолжать работать в Word, то нажимаете на нижнюю кнопку "Закреть". Окно документа может быть открыто только внутри окна программы. В окне программы может быть открыто несколько окон документов, т.е. можно работать одновременно с несколькими документами.

Переключение между окнами документов производится с помощью клавиш

Ctrl+F6 или пунктом меню **Окно**.

Окно документа не может быть увеличено больше чем окно программы. Если нажать на кнопку "Развернуть", то окно документа увеличится до максимально возможного размера внутри окна программы. На рисунке 8 окно Word увеличено во весь экран, открыт один документ, окно которого тоже увеличено до максимально возможных размеров. Рабочим пространством документа является область ограниченная слева и сверху инструментальными линейками, а справа и снизу линейками прокрутки.

Все окна приложений имеют строку меню. Многие пункты меню в разных приложениях одинаковые, но есть различия, обусловленные спецификой каждой программы.

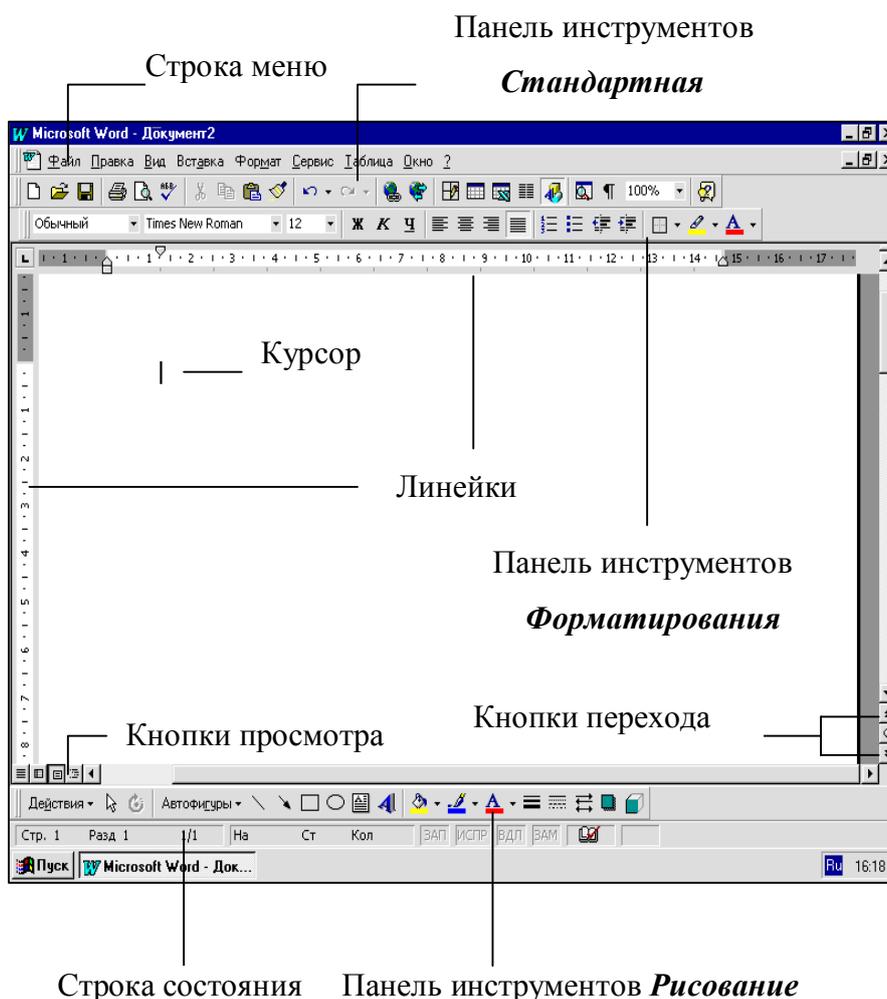


Рис. 8

## Панели инструментов Word

В Word имеется большое разнообразие различных панелей инструментов. Выше отмечалось, что панели инструментов облегчают работу, поскольку позволяют выполнять те или иные действия быстрее, используя инструментальные кнопки и не обращаясь к пунктам меню. Однако следует иметь в

виду, что с помощью панелей инструментов не все операции можно выполнить, а лишь только наиболее часто применяющиеся стандартные операции. Например, нажав на кнопку с изображением принтера, можно напечатать документ и в большинстве случаев этого бывает достаточно, но если вы желаете изменить параметры печати, изменить свойства принтера, то Вам придется действовать через меню. Тем не менее, панели инструментов значительно облегчают и ускоряют работу, а наличие большого количества инструментальных панелей с лихвой перекрывают "потребности" даже самых искусственных пользователей.

По умолчанию на экран выводятся две панели инструментов.

**Стандартная панель** содержит кнопки основных команд: создание нового документа, открытие уже существующего, сохранение документа, печать документа, проверка орфографии, различных манипуляций с фрагментами документа (копирование, вставка, вырезание), создание таблиц и т.д.

**Панель форматирования** позволяет, как ясно из самого названия панели, форматировать текст: изменять шрифты, размеры букв и символов, делать текст жирным или курсивом, выравнивать текст, изменять цвета символов и т.д.

**Панель рисования** позволяет выполнять несложные рисунки, вставлять автофигуры, делать надписи и пр.

**Линейки** позволяют изменять размеры полей, устанавливать отступы и табуляции.

Кнопки **перехода** позволяют переместиться на следующую или на предыдущую страницу документа.

**Строка состояния** содержит некоторую дополнительную информацию для пользователя, например, на какой странице документа сейчас мы находимся, сколько всего страниц в документе, координаты курсора, некоторые режимы работы.

## Кнопки просмотра

В Word используется пять режимов просмотра документов:

- **Обычный** – в этом режиме отображается текст, формат символов, формат абзацев, графические объекты. Однако, при этом многие элементы текста не отображаются, в том числе поля, колонтитулы и т.д. В этом режиме удобно делать правку документа.
- **Разметка страницы** - в этом режиме отображаются все объекты документа, видны поля, колонтитулы, картинки, кадры и пр. Предпочтительно работать именно в этом режиме.
- **Электронный документ** - этот режим позволяет упростить просмотр документа в целом. Режим включает средство, называемое *схемой* документа, позволяющее пользователю просматривать структуру документа в случаях, когда документ очень большой.
- **Структура** – в этом режиме документ отображается вместе с его струк-

турой, т.е. со структурой заголовков - иерархией заголовков (главы, параграфы, пункты, подпункты и т.д.). При этом можно развернуть какую – нибудь главу, а в остальных просмотреть только заголовки и т.п.

**Предварительный просмотр** – в этом режиме можно просмотреть документ в том виде, в котором он будет распечатан. Можно просматривать одновременно несколько страниц, охватить одним взглядом весь документ, увидеть, например, те места где "поехали" строчки, остались висячие строки и т.д.

## Справка Word – 97. Работа с "Помощником"

Умение пользоваться справочной системой совершенно необходимо для начинающего пользователя, особенно если у вас нет под рукой какого – нибудь руководства по Word .

Однако и достаточно квалифицированному пользователю рано или поздно придется более глубоко изучать различные возможности Word. И здесь справочная система Word окажет неоценимую услугу.

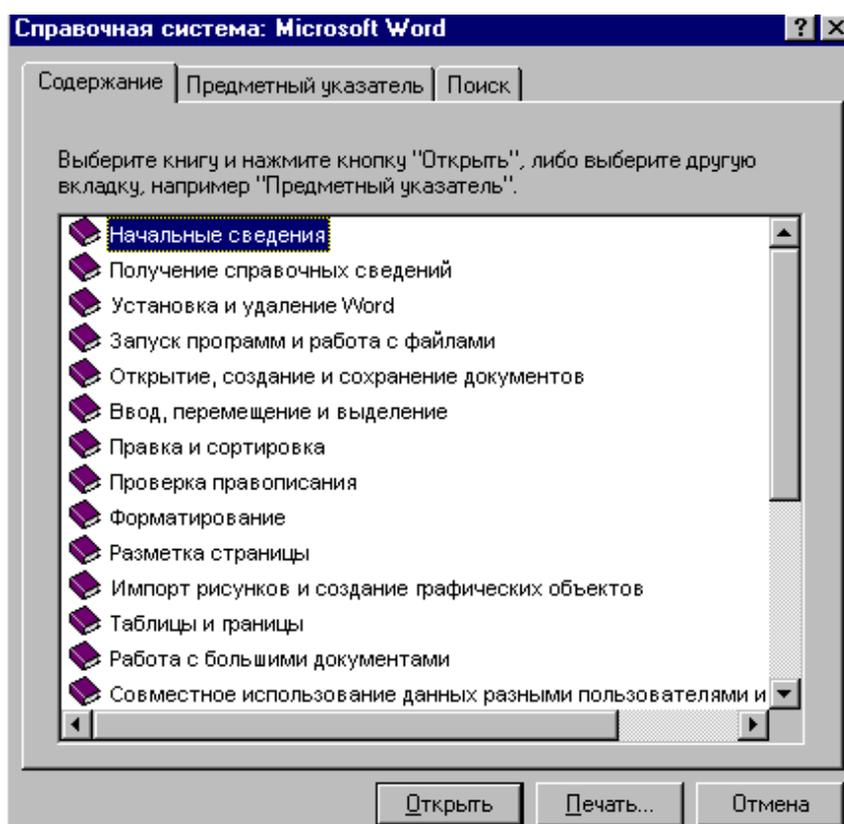


Рис.9

Для вызова справочной системы нужно нажать на клавишу F1 или на кнопку с изображением вопросительного знака в строке меню. (Подчеркиваю, что именно в строке меню. Дело в том, что на панели инструментов **Стандартная** имеется также кнопка с вопросительным знаком (там, правда, присутствует ещё изображение лампочки) – это вызов "Помощника", о котором мы

поговорим позже). При нажатии кнопки ? появится диалоговое окно. Выберите пункт **Вызов справки**. После вызова Справки его окно выглядит так, как на рис. 9.

На экране Справки выведутся темы (книги). Выбрав нужную тему, нажмите на кнопку **Открыть**. Внутри темы могут быть подтемы, оформленные также в виде книг, т.е. справочная система Word построена по иерархическому принципу. Раскрыв самую внутреннюю книгу вы увидите раздел справки. Рядом с разделом появится знак ?. Кнопка **Открыть** заменится на кнопку **Вывести**. Нажав на эту кнопку, вы можете получить справку по интересующему Вас вопросу.

В некоторых разделах справочника имеются кнопки демонстраций, воспользовавшись которыми можно просмотреть последовательность действий при выполнении той или иной операции.

Воспользовавшись кнопкой **Назад**, вы можете вернуться к изучению предыдущего раздела, а нажав на кнопку **Раздел** можно выйти на оглавление того раздела в котором находитесь. См. рис.10.

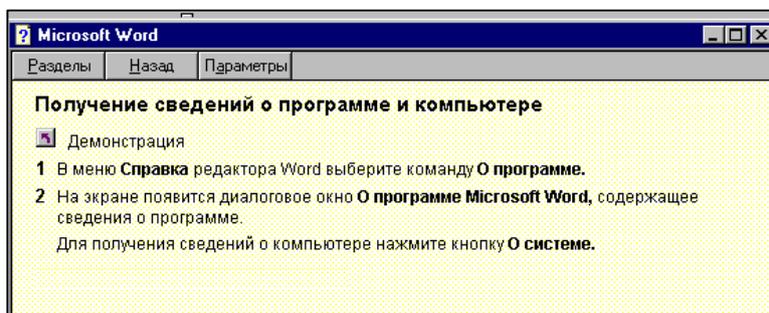


Рис. 10

Если вы знаете название конкретного раздела, по которому хотите получить справку, то можно воспользоваться **Предметным указателем**. В окне справки откройте вкладку **Предметный указатель**. В строке ввода введите название нужного раздела (например, слово *таблица*). Выберите нужный подраздел, нажмите на кнопку **Вывести**.

## Работа с Помощником

Помощник это новое средство появившееся в Word – 97. Помощник выводит полезные советы и подсказки, управляет контекстно- зависимой справочной информацией.

Вызвать Помощник можно нажав на кнопку с вопросительным знаком и лампочкой, расположенной на панели инструментов **Стандартная**. Кроме того, можно настроить Помощник, чтобы он вызывался при нажатии клавиши F1. После вызова Помощника экран Word принимает вид (Рис.11).

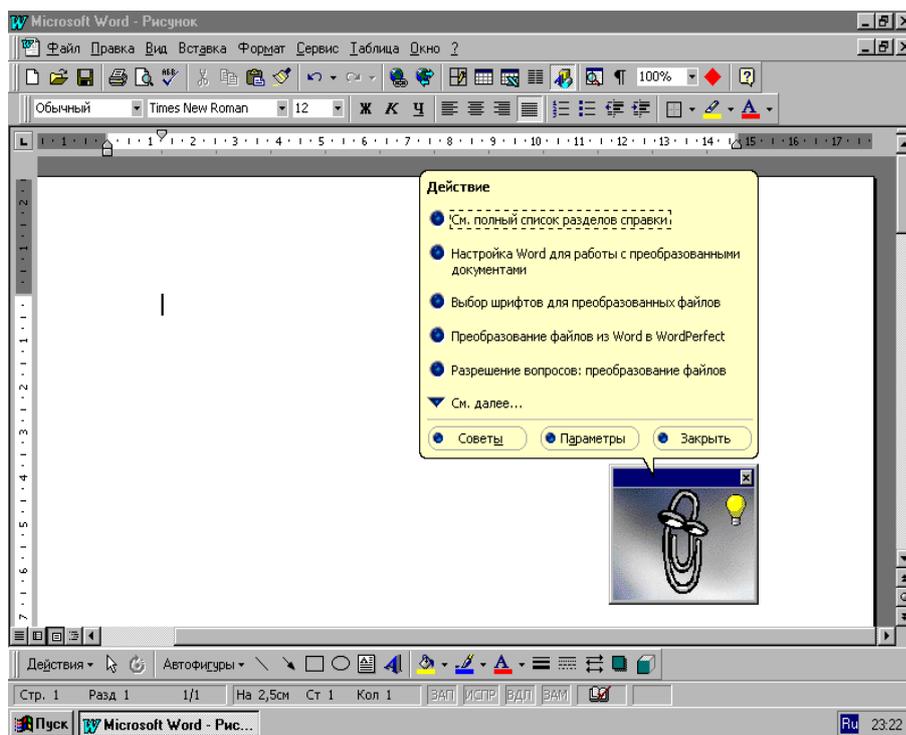


Рис. 11

У Помощника есть собственное меню. Если хотите увидеть его совет, щелкните по кнопке **Совет**. Если хотите настроить Помощник соответственно своим вкусам, нажмите на кнопку **Параметры**. Если хотите закрыть меню, нажмите на кнопку **Закреть**.

Кроме того, в меню под заголовком **Действие** Помощник выводит список тем, которые вы можете просмотреть. Для выхода на всю справочную систему Word нажмите на пункт **См. полный список разделов справки**.

Помощник отслеживает каждое Ваше действие и может выдать подсказку или совет именно по конкретному Вашему действию в настоящий момент. Если Помощник хочет дать Вам подсказку, то у него загорается лампочка. Достаточно щелкнуть по Помощнику, чтобы увидеть подсказку.

Чтобы отказаться от услуг Помощника нужно просто щелкнуть по кнопке "Закреть" в окне Помощника.

### Контрольные вопросы:

1. Как запустить Word?
2. Какие виды окон существуют?
3. Как переключиться из одного окна документа в другой?
4. Для чего используются панели инструментов?
5. Какие режимы просмотра документов существуют?
6. Как вызвать справочную систему Word?
7. Как настроить Помощник?

## §4.2 Практикум по Word – 97

### Урок 1

Содержание урока:

---

- Клавиатура компьютера.
  - Ввод текста документа.
  - Передвижение по тексту.
  - Сохранение вновь набранного документа.
- 

### Основные сведения, необходимые для работы

#### Клавиатура компьютера

Клавиатура современных компьютеров состоит из 101 или 102 клавиш. Они разбиты на нескольких групп (блоков) клавиш:

- Блок символов (блок пишущей машинки).
- Блок клавиш управления курсором.
- Цифровой блок.
- Функциональные клавиши.

#### Блок символов (блок пишущей машинки).

В этом блоке расположены все буквы, цифры, знаки и несколько так называемых служебных клавиш. Расположение клавиш в этом блоке обычно соответствует расположению клавиш на стандартной пишущей машинке. На каждой клавише нарисованы две буквы. Верхние буквы соответствуют буквам латинского алфавита, нижние буквы - буквам русского алфавита (кириллицы).

Клавиша **Shift** предназначена для перевода регистра, т.е. для перехода с ввода строчных (маленьких) букв на ввод заглавных.

Для ввода заглавной буквы надо нажать клавишу **Shift** и удерживая её нажатой нажать на нужную букву. Например, нажав клавиши **Shift** + б вы введете букву Б. (Знак + означает, что вы должны удерживать нажатой клавишу **Shift**).

Если нужно вводить некоторый фрагмент текста или даже весь текст только заглавными буквами, то удобнее воспользоваться клавишей **Caps Lock**. Нажав на нее, вы зафиксируете верхний регистр, при этом лампочка на клавише загорится (следует иметь в виду, что на верхний ряд клавиш этого блока, где находятся цифры и знаки, клавиша **Caps Lock** не действует).

Клавиша **Пробел (Space Bar)** используется для установки промежутков

(пробелов) между словами.

Клавиша **Enter** используется для ввода или для завершения текущего абзаца.

Клавиша **Tab** используется для установки табуляции.

Клавиша **Backspace** используется для удаления символа слева от курсора.

Группа клавиш **Ctrl, Shift, Alt** используются только в сочетании с другими клавишами, нажатие на эти клавиши без нажатия других клавиш не вызывает никаких действий (вероятно, вы это уже заметили, когда речь шла о клавише **Shift**). Использование этих сочетаний мы будем изучать по ходу дальнейшей работы.

Клавиша **Esc** используется для отмены какого – либо действия, выхода из режима, отключения меню и т.д.

## Клавиши управления курсором

Клавиша **↑** позволяет передвинуть курсор на одну строку вверх.

Клавиша **↓** позволяет передвинуть курсор на одну строку вниз

Клавиша **→** позволяет передвинуть курсор на один символ вправо.

Клавиша **←** позволяет передвинуть курсор на один символ влево.

Клавиша **Page Down** позволяет передвинуть курсор на одну страницу вниз (страница – часть текста, уместяющаяся в текущем окне).

Клавиша **Page Up** позволяет передвинуть курсор на одну страницу вверх.

Клавиша **Home** передвигает курсор к началу строки.

Клавиша **End** передвигает курсор на конец строки.

Клавиша **Insert** управляет режимом вставки/замены.

Клавиша **Delete** позволяет удалить символ, расположенный справа от курсора.

## Цифровой блок

Позволяет быстро вводить числа. Клавиши расположены как в калькуляторах. Клавиша Num Lock управлением режимом работы этого блока. Если клавиша нажата (лампочка горит), то можно вводить числа. Если лампочка не горит, то клавиши этого блока дублируют клавиши управления курсором.

## Функциональные клавиши

Это клавиши с именами F1 ÷ F12. Они используются для каких – то часто встречающихся действий или функций. Например, нажав на F1 можно вызвать справочную систему Word. Для каждой программы они имеют свое предназначение. Что касается Word, то мы будем изучать функциональные клавиши по мере возникновения необходимости.

Нам осталось познакомиться с тремя последними клавишами, расположенными справа от функциональных клавиш.

**Print Scrn** позволяет "снять" копию экрана. Например, все рисунки в этой книге сделаны с использованием этой клавиши.

**Scroll Lock** клавиша блокировки прокрутки текста. Сейчас нигде не используется. Даже если вы нажмете эту клавишу, Word спокойно будет прокручивать текст. Но, конечно, безо всякой надобности трогать эту клавишу не стоит!

### Практическая часть урока

Введите приведенный ниже текст в рамке. При наборе текста помните, что клавиша Enter используется для окончания абзаца и начала нового абзаца (абзац это фрагмент текста с отступом в первой строке). По умолчанию Word переносит слова целиком. Если вы хотите, чтобы слова переносились по частям, в соответствии с русской грамматикой, то установите режим переноса слов. Для этого откройте меню **Сервис**, пункт **Язык – Расстановка переносов**. В диалоговом окне установите флажок *Автоматическая расстановка переносов*.

Запомните также, что знаки препинания ставятся "впритык" к тексту, т.е. без пробелов, а вот после любого знака препинания обязательно надо ставить пробел. Иначе Word будет считать два слова, разделенных запятыми, за одно слово и давать ошибку, а также будет неправильно переносить слова, например:

"Мы надеемся, что ..." - неправильно

"Мы надеемся, что ..." – правильно

Биллу Гейтсу,  
президенту фирмы Microsoft,  
США.

Имеем честь сообщить Вам, что наша фирма приобрела продукцию Вашей фирмы – операционную систему Windows – 95 и пакет Microsoft Office - 97. Ваши программные средства завоевали заслуженную популярность во всем мире благодаря великолепной идее оконного интерфейса, простоте и удобству работы с ними, практически неисчерпаемым возможностям, заложенным в них.

Мы только что приступили к изучению Word. И это письмо – наш первый опыт работы с программой. Мы надеемся, что мы сможем освоить Word и применять его в повседневной работе.

Мы ждем от Вас и от Ваших прекрасных программистов новых, ещё более совершенных и мощных программных средств.

Наша фирма предлагает Вам своё сотрудничество в Вашем нелегком, но благородном труде. Мы хотим, чтобы вы присылали нам свои новые программы для их проверки и тестирования. Мы готовы стать их первыми испытателями, если хотите - стать первыми "подопытными кроликами" в деле проверки

Ваших новых программных средств. Поскольку мы в своей работе допускаем такие ошибки, такие "ляпсусы", которые Вам и Вашим программистам и не снились! А ведь хорошая программа должна быть защищена от любых мыслимых и немыслимых ошибок. И такие ошибки мы Вам гарантируем!

Разумеется, речь идет о взаимовыгодном сотрудничестве! Мы уверены, что наше сотрудничество принесет пользу и Вам и нам!

Мы думаем, что если Вы будете нам платить зарплату примерно долларов 700 в месяц каждому нашему сотруднику, то мы будем вполне довольны. Ну, а мне, как президенту фирмы, я думаю, вы будете платить больше эдак раз в десять – пятнадцать!

На днях наш сотрудник едет в США. У нас большая просьба к Вам.

Пожалуйста, поручите кому –нибудь из Ваших сотрудников встретить его в аэропорту, поскольку он в первый раз едет в Вашу страну и, к тому же, в английском языке он ни бум-бум! И ещё одна просьба. Забронируйте ему номер в отеле. Отель должен быть в центре города, обязательно пятизвездочный, окна с видом на море. В номере должны быть бар, телевизор со спутниковой антенной, сотовый телефон, факс. И, конечно же, компьютер!

Мы, конечно, можем заплатить. Но раз мы хотим сотрудничать, то не могли бы Вы сами заплатить за все это, так сказать в кредит, в счет наших будущих прибылей.

С уважением, Славик Недотепкин  
президент фирмы  
"Ош - Бамбук" Ltd

### **Передвижение по тексту документа**

Поупражняйтесь в передвижении по тексту с использованием клавиатуры. Используйте клавиши управления курсором. Кроме того, используйте следующие сочетания клавиш для быстрого передвижения по тексту документа. (Табл.2.) Постарайтесь запомнить эти клавиши. Они сэкономят Вам массу времени при работе в дальнейшем!

Таблица 2. Быстрые клавиши для передвижения по тексту

Сочетания клавиш	Действие
Ctrl +стрелка вправо	На начало следующего слова
Ctrl +стрелка влево	К началу текущего или предыдущего слова
Ctrl +стрелка вниз	К следующему абзацу
Ctrl +стрелка вверх	К предыдущему абзацу
Ctrl +End	Переход на конец документа
Ctrl +Home	Переход на начало документа

А теперь используйте мышь и линейки прокрутки. При прокрутке курсор остается на месте! Для установки курсора в нужное место щелкните по этому

месту или используйте клавиатуру.

## Сохранение вновь набранного документа

После того как вы набрали текст документа, его необходимо сохранить. Для этого откройте меню **Файл**, пункт **Сохранить как**. В диалоговом окне введите необходимые параметры. Рисунок 12.

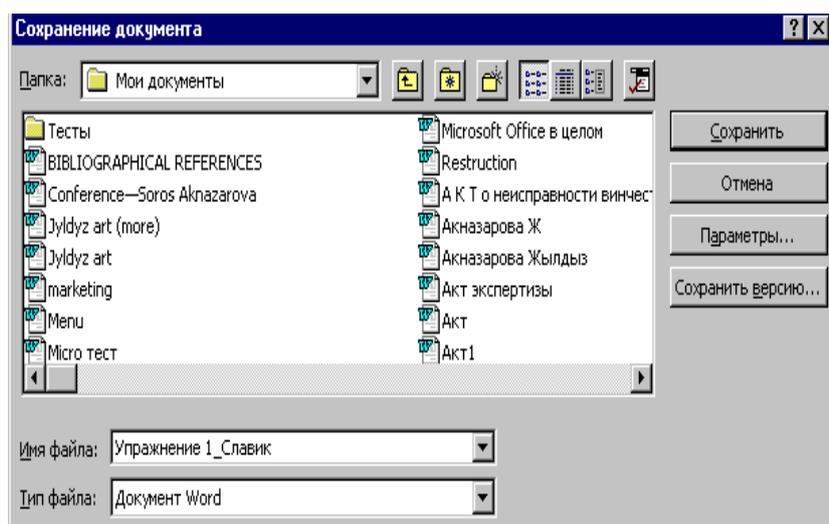


Рис. 12

Укажите папку, куда вы хотите сохранить документ. Обычно Word настроен на папку **Мои документы**, который создается в момент установки Word, но ничто не мешает Вам сохранить документ в любой другой папке.

Во избежания недоразумений, в процессе работы над упражнениями этой книги, лучше сохранять документы все - же в папке **Мои документы!**

В поле **Имя файла** введите имя файла. При записи имен файлов можно использовать любые символы, кроме следующих:

\* / \ < > ? " : |

Имейте в виду, что Word позволяет давать длинные имена файлам. Вы можете назвать свой документ, например вот так – *Мое первое письмо, адресованное самому богатому человеку в мире Биллу Гейтсу*.

Однако не слишком увлекайтесь. Помните, что краткость – сестра таланта! Но ещё хуже давать сокращенные имена, наподобие вот этого – *М.п.пис. адр. Б.Г.*

Когда вы будете работать "по – настоящему", а не в классе, у вас накопится довольно много документов и тогда отыскивать нужные документы станет для Вас сущим мучением. Многие, отчаявшись, предпочитают заново набрать документ. Вам своего времени не жаль?!

Поэтому старайтесь давать имена своим документам краткие, но понятные,

отражающие суть документа, например:  
*Первое письмо* или *Письмо Биллу Гейтсу*.

Мы с Вами договоримся при выполнении упражнений давать имена в следующем виде:

<*Упражнение*> <№>\_<Имя студента>

Например: *Упражнение 1\_Славик*

Как видите угловые скобки набирать не надо. Добавлять свое имя необходимо, поскольку это же упражнение будут выполнять и другие.

После того, как вы набрали имя документа, нажмите кнопку **Сохранить**.

### **Контрольные вопросы:**

1. Из каких блоков состоит клавиатура компьютера?
2. Для чего нужна клавиша **Caps Lock**?
3. Какой клавишей можно удалить символ, расположенный слева от курсора?
4. Какой клавишей можно удалить символ, расположенный справа от курсора?
5. Как быстро переместить курсор на конец строки?
6. Как быстро попасть в конец документа?
7. Перечислите действия, необходимые для сохранения нового документа.

## Урок 2

### Содержание урока:

---

- Открытие документа
  - Редактирование документа
  - Проверка орфографии и грамматики
  - Сохранение документа
- 

### Основные сведения, необходимые для работы

#### Открытие документа

Для того чтобы продолжить работу над ранее сохраненным документом, его необходимо открыть. Существует несколько способов открытия документа:

- Открыть меню **Файл**, выбрать пункт **Открыть**, в диалоговом окне открыть нужную папку, выбрать нужный файл, нажать кнопку **Открыть**.
- На стандартной панели инструментов нажать на кнопку **Открыть**. Далее как в предыдущем пункте. Рисунок 13.



Рис. 13

Все кнопки на панелях инструментов снабжены всплывающими подсказками, по которым можно узнать о назначении кнопки. Для того чтобы увидеть подсказку просто подержите некоторое время указатель мыши над этой кнопкой. На рисунке 13 изображена подсказка для кнопки **Открыть**.

Word автоматически запоминает список из нескольких недавно использовавшихся документов (максимально 9 документов). Если вы недавно работали с документом, то можно быстро открыть его следующим способом: откройте меню **Файл**, в списке недавно использовавшихся файлов выберите свой файл. Рисунок 14.

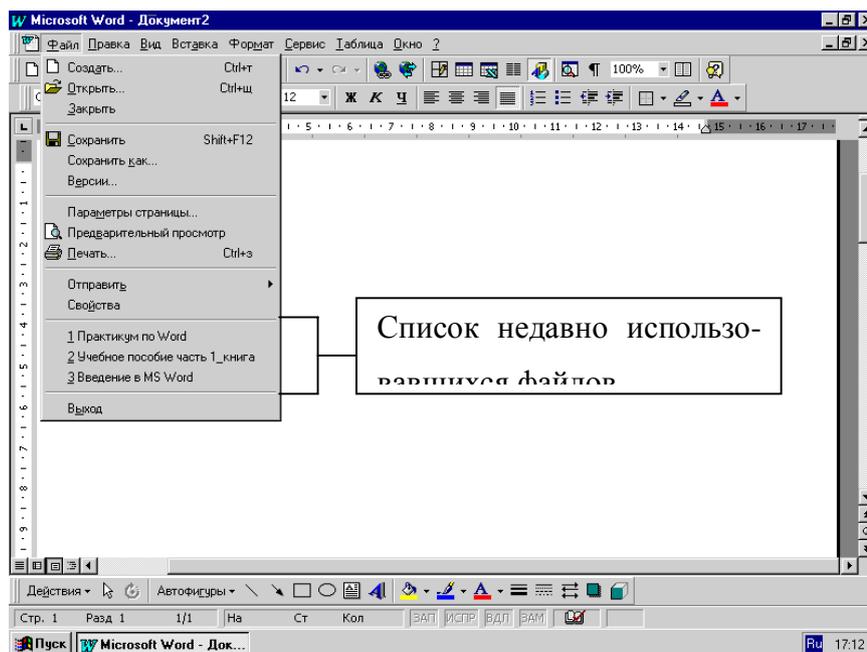


Рис. 14

## Редактирование текста

### Удаление небольших фрагментов текста

Для удаления одиночных символов используются клавиши **Backspace** и **Delete**. С помощью клавиши **Backspace** можно удалить символ, расположенный слева от курсора. С помощью клавиши **Delete** можно удалить символ справа от курсора.

Если требуется удалить несколько символов (например, целое слово), то лучше воспользоваться комбинациями клавиш **Ctrl+Backspace** и **Ctrl+Delete**. Установите курсор на удаляемое слово. Если курсор находится слева от слова, то нажмите **Ctrl+Backspace**, если курсор справа от слова, то нажмите **Ctrl+Delete**.

### Режим вставки/замены

В режиме **вставки** текст, расположенный справа от курсора при добавлении нового текста сдвигается. Это очень удобно, когда вы вставляете что-нибудь в середину текста.

Иногда бывает необходимо заменить часть текста другим. Можно, конечно, сначала удалить ненужный текст, а потом набирать новый, но "профессиональней" воспользоваться режимом **замены**. Для этого используется клавиша **Insert**. Клавиша работает как переключатель. Нажатие этой клавиши приводит к изменению режима. Если был установлен режим вставки, то установится режим замены и наоборот. О том, в каком режиме ввода вы находитесь,

указывает кнопка ЗАМ в строке состояния. Если кнопка блеклая, то включен режим вставки.

## Отмена изменений

Иногда бывает необходимо отменить последнее изменение. Например, вы случайно удалили не то слово, какое хотели. Для этого используется кнопка **Отменить** на стандартной панели инструментов. Найдите её самостоятельно, используя всплывающие подсказки. Для того чтобы отменить последнее изменение нажмите на эту кнопку. Word запоминает каждое Ваше действие. вы можете отменить предыдущие изменения, последовательно нажимая на кнопку **Отменить**, а также вернуть изменения, нажимая на кнопку **Вернуть**. Она расположена рядом, справа от кнопки **Отменить**. Пока вы не отменили хотя бы одно изменение, кнопка **Вернуть** не будет активизирована.

## Проверка правописания

При наборе текста могут появиться волнистые линии под словами, красного и зеленого цвета. Этим способом Word сигнализирует Вам о том, что вы, возможно, допустили ошибку в написании слова или в грамматике. Существуют два способа исправления ошибок. Первый - непосредственно во время набора текста. Этот способ наиболее простой и чаще всего используемый. Как только вы заметили ошибку, используя клавиши управления курсором, возвратитесь к этому слову и исправьте ошибку. Если вы не знаете, как правильно написать это слово, щелкните по нему правой клавишей. Word покажет Вам вариант написания данного слова (при условии, что это слово есть в словаре Word).

На рисунке 15 показано диалоговое окно, которое появится в этом случае.

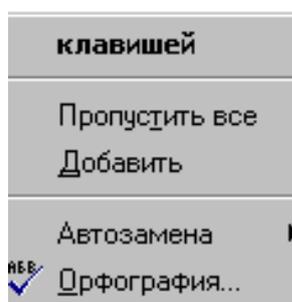


Рис. 15

При наборе слова *клавишей* была допущена ошибка. Word предлагает правильное написание этого слова. Вы можете согласиться с Word, щелкнув по слову или выбрать пункт **Пропустить все**. Если слово не содержалось в словаре Word, то выбрав пункт **Добавить**, можете внести его в словарь Word.

Если это ошибка грамматическая, то выйдет другое диалоговое окно. Рисунок 16.

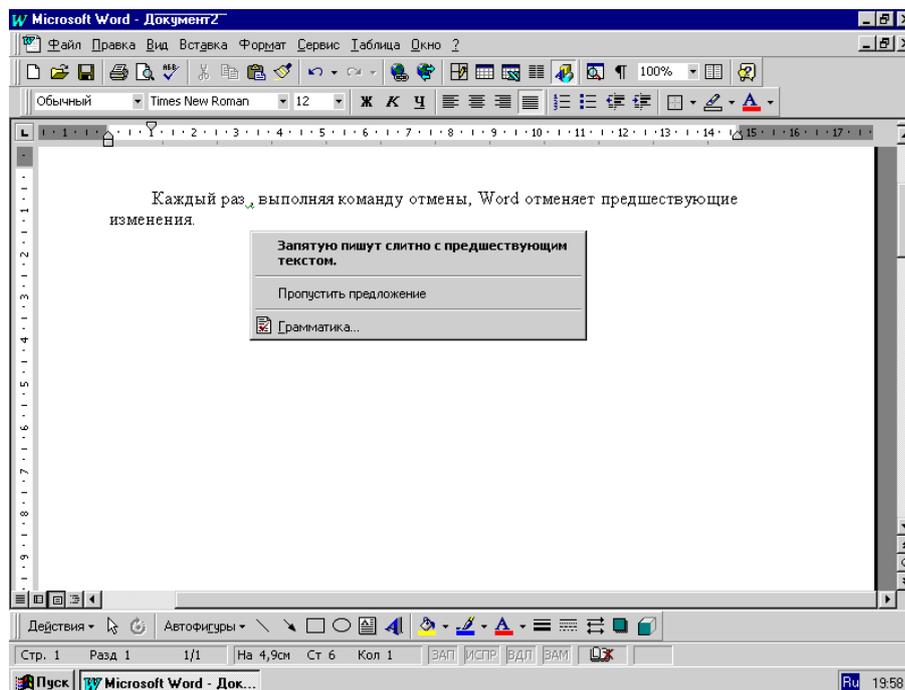


Рис. 16

Вы можете исправить предложение, расставив знаки пунктуации, так как предлагает Word. Однако Word не так силен в грамматике! Не всегда доверяйте Word по части грамматики.

Второй способ исправления ошибок заключается в запуске специального средства коррекции правописания после окончания набора текста. Для этого откройте меню **Сервис** и выберите пункт **Правописание** или нажмите клавишу F7. Появится диалоговое окно. Его вид зависит от того, какого рода ошибка, орфографическая или грамматическая.

Рассмотрим сначала случай орфографической ошибки. Вид окна диалога показан на рисунке 17.

Слово, в котором есть ошибка, будет выделено. В тексте стандартным цветом (черным), а в диалоговом окне красным. Дальнейшие действия зависят от вас:

- Если в списке, предложенном Word, имеется подходящее слово, то нажмите кнопку **Заменить**
- Если слово все же написано правильно, то это означает, что этого слова просто нет в словаре Word. Вы можете выбрать одну из кнопок: **Пропустить**, **Пропустить все**, **Добавить**. Такими словами часто являются имена и фамилии, термины, понятия и т.д. Если эти слова будут часто использоваться, то имеет смысл добавить их в словарь Word .

Кроме того, можно добавить эти слова в список автозамены. Отличие авто

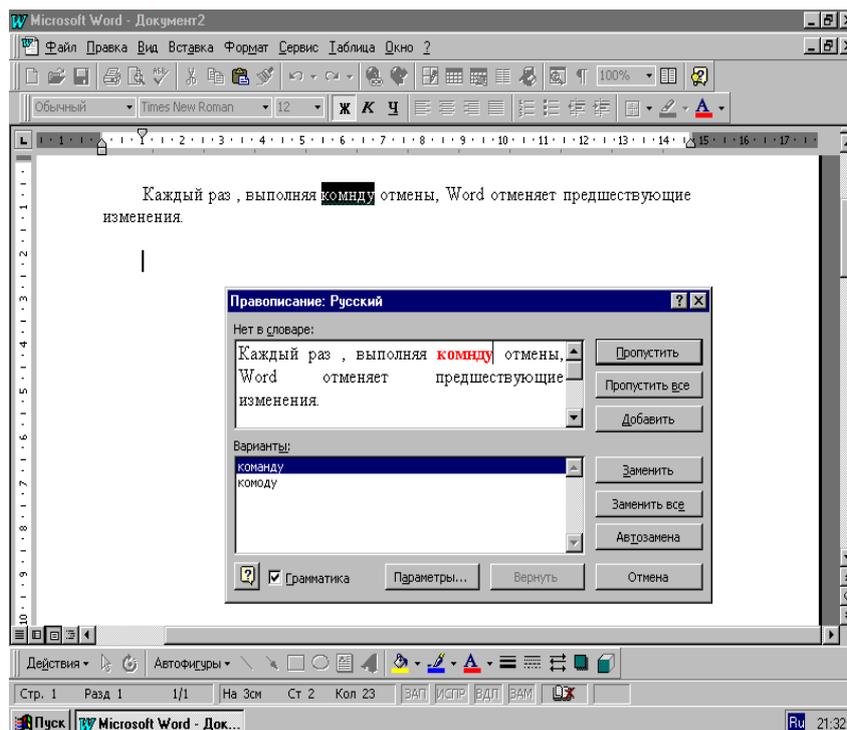


Рис. 17

замены от словаря в том что, введя их в список автозамены, можно назначить им комбинацию клавиш для быстрого их набора. Например, при наборе текста этой книги часто встречается слово **Word**. Чтобы каждый раз не набирать это слово (при этом приходится переключаться с русского на английский язык и обратно) автор ввел это слово в список автотекста и назначил этому слову букву ь.

- Если слово напечатано неправильно, но его нет в словаре, то исправить его можно щелкнув по слову непосредственно в документе. При этом кнопка **Пропустить** заменится на кнопку **Возобновить**. Щелкните по ней, чтобы продолжить проверку.

Теперь рассмотрим случай грамматической ошибки. Диалоговое окно принимает вид: (рисунок 18)

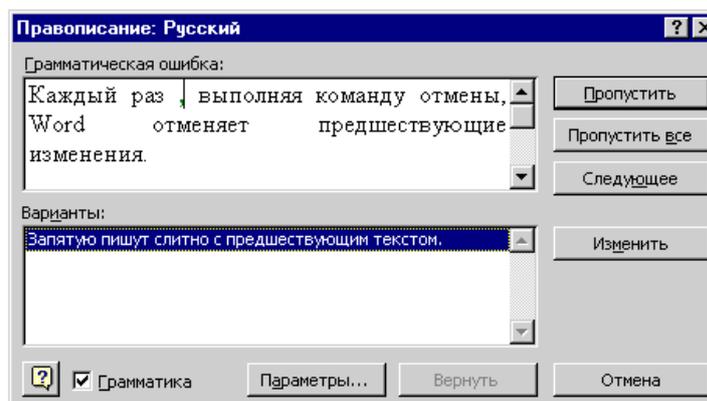


Рис. 18 14

В окне **Варианты** Word предлагает варианты изменения для исправления этой грамматической ошибки. Если вы согласны, нажмите кнопку **Изменить**. Если Word не знает, как изменить эту ошибку, то кнопка **Изменить** становится недоступной.

Кроме этого, вы можете выбрать следующие варианты действий:

- Выбрать кнопку Пропустить, чтобы оставить все как есть (потому что Word сам может ошибиться. Грамматика русского языка настолько сложна, что ещё не придумали алгоритма безошибочной проверки, а скорее всего, такого алгоритма просто не существует!)
- Выбрать кнопку Пропустить все, чтобы пропустить проверку в этом и других предложениях
- Выбрать кнопку Следующее для перехода к следующему предложению

### **Сохранение документа**

После внесенных изменений документ нужно снова сохранить! Как правило, сохраняют документ под тем же именем, которое было присвоено документу при первом сохранении. Для этого надо щелкнуть по кнопке **Сохранить** на стандартной панели инструментов. Найдите эту кнопку самостоятельно!

Иногда бывает необходимо сохранить документ под другим именем или в другую папку. Тогда нужно сохранять так, как будто документ сохраняется впервые (как описано в предыдущем уроке).

### **Практическая часть урока**

- Откройте свой документ
- Потренируйтесь в удалении небольших фрагментов текста (символов, слов).
- Используйте кнопки **Отменить** для отмены сделанных изменений и **Вернуть** для возврата изменений в текст.
- Добавьте несколько предложений в середину документа, используя режим вставки. Постарайтесь уяснить различие в режимах вставки и замены.
- При добавлении текста в документ у вас обязательно (на первых порах!) будут появляться ошибки во время ввода. Исправляйте их сразу, как только вы их заметите.
- Запустите систему автоматической проверки правописания. Поработайте с появляющимися во время проверки диалоговыми окнами. Если ошибок было мало, намеренно введите в текст ошибки и снова запустите корректор ошибок.
- Сохраните документ под тем же именем, что и раньше.

**Контрольные вопросы:**

1. Перечислите способы открытия документа.
2. Как быстро удалить целое слово?
3. Чем отличается режим вставки от режима замены?
4. Как запустить средство проверки орфографии и грамматики?

## Урок 3

Содержание урока:

---

- Выделение текста
  - Операции с выделенным текстом
  - Буфер обмена
  - Форматирование текста
- 

### Основные сведения, необходимые для работы

#### Выделение текста

В процессе работы бывает необходимо выделить некоторый фрагмент текста или весь текст. Word позволяет производить довольно большое количество операций с выделенным текстом.

Для того чтобы выделить текст надо установить указатель мыши на то место с которого вы хотите выделить текст, нажать на левую кнопку мыши и, не отпуская кнопку, передвигать мышь до нужного места текста, а затем отпустить кнопку мыши. Фрагмент текста выделится, как правило, черным цветом. Существуют несколько способов выделения типичных фрагментов текста (слова, предложения, строки, абзаца, всего текста):

**Таблица 3.**

Типичные фрагменты текста	Способ выделения
Слово	Дважды щелкнуть на слове
Предложение	Удерживая нажатой клавишу <b>Ctrl</b> , щелкнуть на предложении.
Строка	Установить указатель в левом поле страницы рядом со строкой, (указатель должен изменить вид на стрелку) и щелкнуть мышью.
Абзац	Дважды щелкнуть на левом поле возле абзаца.
Нескольких абзацев	Дважды щелкнуть на левом поле возле первого абзаца и, не отпуская кнопки мыши после второго щелчка, переместить указатель мыши к следующим абзацам.
Весь документ	Трижды щелкнуть на левом поле страницы, когда указатель мыши примет форму стрелки

Существуют способы выделения текста с помощью клавиатуры, о котором вы можете подробнее узнать из справочной системы Word.

Чтобы отменить выделение щелкните по любому месту текста вне области выделения или нажмите любую клавишу управления курсором.

С выделенным текстом можно выполнить следующие операции:

- Удалить
- Проверить орфографию
- Переместить, скопировать или вырезать
- Изменить стиль форматирования

Удаление производится нажатием клавиши Delete.

При перемещении используется так называемая техника **Drag and drop** (перетаски и оставь). Для этого выделите текст, установите указатель над выделенным фрагментом текста (при этом под указателем появится маленький серый квадратик) и, не отпуская кнопку мыши, "тащите" текст в нужное место. Этот метод перемещения удобен, когда вам видно на экране место, куда вы хотите переместить текст.

Чтобы скопировать текст, надо выделить его, нажать клавишу **Ctrl** и, не отпуская её, перетащить текст как описано в предыдущем абзаце.

Если вам нужно переместить (скопировать) текст далеко или в другой документ, применяется другой метод. При этом используется **буфер обмена**. Буфер обмена – специальная область памяти, куда можно поместить некоторый объект (текст, рисунок, диаграмма и пр.). Буфер доступен всем приложениям Windows – 95. Таким образом приложения Windows – 95 могут обмениваться данными между собой.

В буфере может находиться только один объект. Если поместить в буфер другой объект, предыдущий исчезает.

Для помещения объектов в буфер используются операции **вырезания** и **копирования**. При вырезании выделенный текст перемещается в буфер, т.е. в документе не остается. При копировании выделенный текст остается в документе, а в буфер помещается его точная копия.

На стандартной панели инструментов есть кнопки, позволяющие выполнить эти операции. Кнопки называются – **Вырезать**, **Копировать**, **Вставить**. Найдите их самостоятельно, используя всплывающие подсказки на кнопках. Кнопка **Вставить** позволяет вставить объект из буфера в нужное место документа.

## Форматирование текста

Под форматированием понимается выбор внешнего вида текста или документа. Можно выбрать размер и начертание символов, изменить шрифты, установить отступы и границы и т.д.

Форматирование позволяет сделать текст документа более удобочитаемым, улучшить его внешний вид и облегчить его восприятие.

Форматирование можно выполнить как с использованием меню, так и с использованием специальной инструментальной панели, которая так и называется – панель инструментов **Форматирование**.

Мы будем различать два вида форматирования:

- Форматирование текста – как сплошной последовательности символов.
- Форматирование документа – как последовательности абзацев.

В настоящем уроке мы разберем способы форматирования текста.

## Выбор способа оформления символов текста.

Существуют три основных варианта оформления вида символов текста: **Полужирный, Курсив, Подчеркнутый**.

Для установки нужного варианта:

1. Выделите нужный фрагмент текста.
2. На панели инструментов **Форматирование** нажмите одну из кнопок: **Ж**, **К**, **Ц**.

Можно выбрать способ оформления перед началом набора текста. Для этого просто нажмите одну из трех вышеперечисленных кнопок и набирайте текст.

## Изменение типа и размера шрифта

Для изменения типа и размеров шрифта используйте раскрывающиеся списки **Тип шрифта** и **Размер шрифта** на панели инструментов **Форматирование**. Рисунок 19.



Рис.19

Вы можете набрать текст любым шрифтом, а затем отформатировать его или сначала установить нужный шрифт, его размер, а затем набирать текст.

Размер стандартного машинописного текста соответствует обычно 12 размеру. Могут возникнуть вопросы – а что такое размер шрифта? Что означает число 12? Размер относительно чего? И вообще, что такое шрифт?

Начнем с последнего вопроса.

**Шрифт** – это набор символов (букв, цифр, знаков) одинакового стиля начертания. Размеры шрифтов измеряются в пунктах. Один пункт равен 1/72 дюйма. Сл-но, размер 12 означает размер символов данного шрифта, равного

12 пунктам.

Существует великое множество различных шрифтов, но не все шрифты содержат кириллицу, т.е. русские буквы. Вот как, например, выглядят начертание букв в слове "Шрифт" в различных шрифтах:

Шрифт – Times New Roman

Шрифт – Arial

–□\|] - Monotype Sorts

Шрифт – Tahoma и т.д.

Как уже отмечалось ранее, используя панель инструментов можно быстро отформатировать текст. Но, используя команды меню, можно выбрать гораздо больше режимов форматирования. Для этого:

1. Выделите нужный фрагмент текста.
2. Откройте меню **Формат**, выберите пункт **Шрифт**. Откроется диалоговое окно, изображенное на рисунке 20.

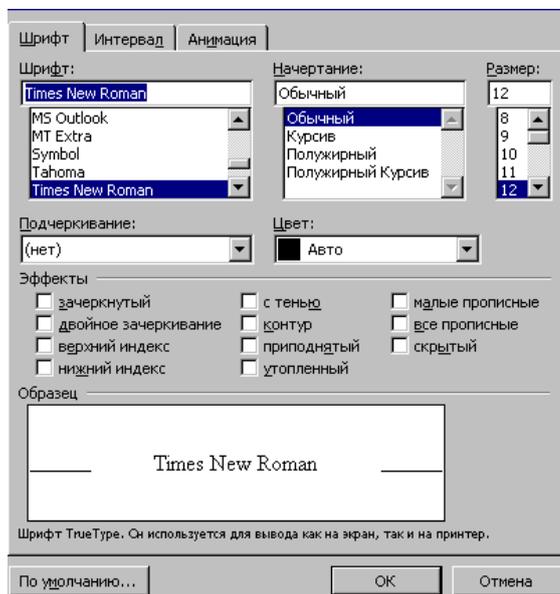


Рис. 20

Кроме уже упомянутых режимов форматирования текста, вы можете выбрать:

- Режим подчеркивания.
- Различные эффекты (например, с тенью, зачеркнутый и т.д.).
- Выбрать цвет символов.
- Изменить интервал между символами.
- Изменить смещение между символами.
- Установить эффекты анимации текста.

## Практическая часть урока

- Откройте свой документ.
- Потренируйтесь в методах выделения текста с помощью мыши.
- Откройте справочную систему Word, откройте вкладку **Содержание**, найдите раздел **Ввод перемещение и выделение**, откройте раздел **Выделение**, найдите тему **Выделение с помощью клавиатуры**, перепишите себе в тетрадь способы выделения текста с помощью клавиатуры и выучите их.
- Прodelайте различные операции с выделенным текстом (копирование, перемещение, удаление, вырезание). Если вы сделали что-то не так, широко пользуйтесь кнопками **Отменить** и **Вернуть**.
- Поработайте с буфером обмена. Постарайтесь уяснить разницу между удалением с помощью клавиши **Delete** и вырезанием!
- Отформатируйте текст письма. Используйте как панель инструментов, так и меню. Попробуйте все варианты форматирования текста.
- Сохраните свой документ с тем же именем.

### Контрольные вопросы:

1. Как сразу выделить целое предложение?
2. Как выделить целый абзац?
3. Какие операции можно выполнить с выделенным текстом?
4. Что это за техника перемещения Drag and drop?
5. Что такое буфер обмена?
6. Чем отличается операция вырезания от копирования?
7. Что значит форматировать документ?
8. Какие варианты оформления символов существуют?
9. В каких единицах измеряются размеры шрифтов?

## Урок 4

Содержание урока:

---

- Форматирование абзацев
  - Границы и заливка
- 

### Основные сведения, необходимые для работы

#### Форматирование абзацев

Абзац – это фрагмент текста, выражающая некоторую законченную мысль. (В принятой терминологии, заголовок считается отдельным абзацем). Всякий раз, когда вы нажимаете клавишу **Enter**, Word начинает новый абзац, при этом, в конце предыдущего абзаца он ставит специальный символ – маркер конца абзаца (или просто маркер абзаца). Маркер абзаца является непечатаемым символом и на экран, а также на печать не выводится. Однако его все – же можно увидеть, если щелкнуть по кнопке **Непечатаемые символы** на Стандартной панели инструментов. (Найдите его самостоятельно, используя подсказки на кнопках!).

Для форматирования одного абзаца не обязательно его выделять. Достаточно установить курсор в любом месте этого абзаца. Чтобы отформатировать сразу несколько абзацев, их необходимо выделить.

#### Отступы

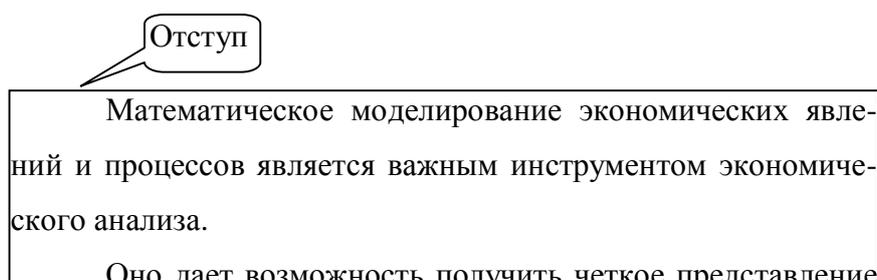


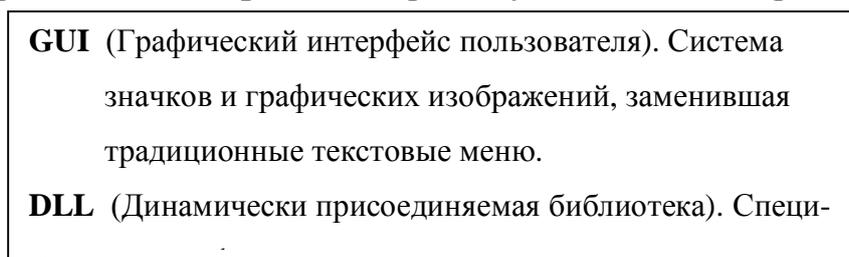
Рис. 21

Обычно абзац содержит отступ первой строки ("красную" строку). Рисунок 21.

Иногда бывает необходимо сдвигать все строки абзаца вправо, кроме первой строки (так называемый выступ). Рисунок 22.

Для того чтобы установить отступы первой строки абзаца нужно:

1. Открыть меню **Формат**, выбрать пункт **Абзац**, открыть вкладку **От-**



↑  
Выступ Рис. 22

**ступы и интервалы.**

2. Из раскрывающегося списка выбрать параметр **Отступ** для сдвига первой строки абзаца вправо (установка красной строки) или выбрать параметр **Выступ** для сдвига всех строк вправо, кроме первой.
3. В поле ввода **На** указать величину отступа. В окне **Образец** можно видеть как будет выглядеть абзац после форматирования.

Для того чтобы установить отступ всего абзаца нужно в той же вкладке **Отступы и интервалы** в полях ввода **Слева** и **Справа** указать нужные величины отступов.

Во многих случаях удобнее и быстрее для установки отступов пользоваться линейкой. Рисунок 23.

На линейке расположены маркеры отступов. Для установки отступов достаточно "ухватиться" за соответствующий маркер и перетащить его на нужное место. Верхний треугольник регулирует отступ первой строки (красную строку). При его движении другие маркеры остаются на месте. Нижний (левый) треугольник регулирует отступ вправо всех строк, кроме первого (выступ). При его движении вместе с ним будет двигаться и прямоугольник (маркер отступа слева всего абзаца). Если двигать прямоугольник, то будут пропорционально сдвигаться и маркер красной строки и маркер выступа, потому что это изменяет величину отступа всего абзаца. И, наконец, треугольник справа регулирует величину отступа всего абзаца от правой границы бумаги.

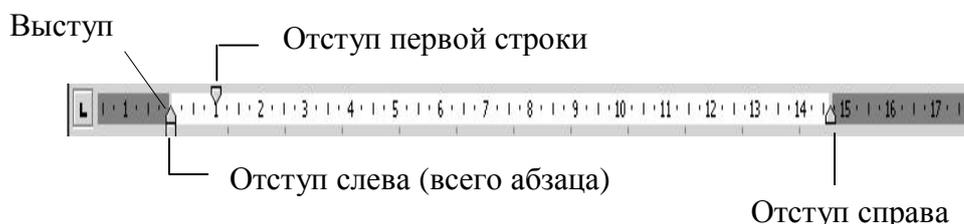


Рис.23

## Выравнивание

Существует 4 способа выравнивания краев абзаца. Рисунок 24.



Рис. 24

Чтобы выровнять края абзаца достаточно установить курсор на этот абзац и нажать на одну из кнопок выравнивания.

Чтобы выровнять несколько абзацев надо их выделить и нажать на одну из кнопок выравнивания.

## Интервалы

Для изменения межстрочного интервала необходимо:

1. Открыть меню **Формат**, выбрать пункт **Абзац**, во вкладке **Отступы и интервалы** открыть раскрывающийся список **Междустрочный**.
2. Выбрать нужное значение параметра.

Для установки интервала между абзацами введите нужные значения в поля ввода **Перед** и **После**.

## Предотвращение разрыва абзаца.

Если вы печатаете короткий абзац в конце страницы, то могут возникнуть несколько нежелательных ситуаций:

- Появление *висячей строки*. Это последняя строка абзаца, перешедшая на новую страницу.
- Появление *изолированной строки*. Это первая строка абзаца, оставшаяся на текущей странице.
- Желательно было бы, чтобы абзац поместился целиком на текущей странице (речь идет о коротких, в две – три строки абзацах), а Word разорвал этот абзац.
- Заголовок остался на одной странице, а следующий за ним абзац перешел на другую страницу (напоминаю, что заголовок считается отдельным абзацем!).

Чтобы не допускать таких ситуаций необходимо:

1. Установить курсор в текущем абзаце.
2. Открыть меню **Формат**, выбрать пункт **Абзац**, открыть вкладку **Положение на странице**.
3. Установить флажок **Запрет висячих строк**.
4. Установить флажок **Не разрывать абзац**.
5. Установить флажок **Не отрывать от следующего**.

И, наконец, ещё одна ситуация. Вы находитесь в середине страницы, но вам нужно начать новый абзац с новой страницы, например новую главу в книге. Многие, не раздумывая, начнут нажимать клавишу Enter. Здесь уместно сделать одно важное замечание.

Никогда не устанавливайте отступы и интервалы с помощью пробелов и клавиши Enter, т.к. в случае внесения в дальнейшем изменений в документ у Вас могут нарушиться установленные ранее режимы форматирования. Например, могут сдвинуться расположение заголовков, измениться расположение рисунков и т.д.

Итак, чтобы начать абзац с новой страницы надо в той же вкладке установить флажок **С новой страницы**.

## Границы и заливка

Под границей понимается создание рамки вокруг абзаца или разделительных линий между абзацами.

Заливка позволяет выделить абзац другим цветом.

На предыдущей странице вы как раз можете видеть пример применения границ и заливки.

Для установки границ и заливки сделайте следующее:

1. Установите курсор на нужном абзаце.
2. Нажмите кнопку **Границы** (рис.25) на панели инструментов **Форматирование**.
3. Выберите тип границы.
4. Нажмите кнопку **Выделение цветом** на той же панели инструментов (рис.25).
5. Выберите нужные параметры заливки.

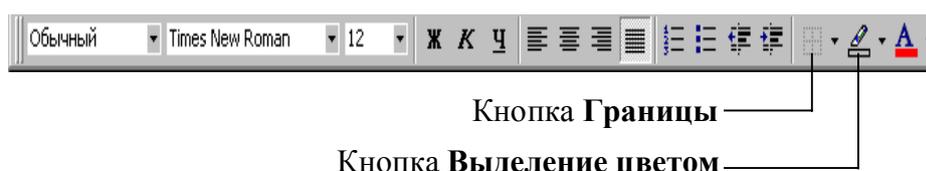


Рис. 25

## **Практическая часть урока**

- Откройте свой документ.
- Отформатируйте абзацы, используя все приведенные выше методы и способы.
- В нескольких абзацах установите границы и заливку.

## **Контрольные вопросы:**

1. Как начать новый абзац?
2. Как отформатировать сразу несколько абзацев?
3. Чем отличается отступ от выступа?
4. Как установить "красную" строку?
5. Как установить отступ слева всего абзаца?
6. Перечислите способы выравнивания краев абзаца.
7. Что такое висячая строка и как от нее избавиться?
8. Что такое изолированная строка и как от нее избавиться?
9. Что означает разрыв абзаца?

## Урок 5

### Содержание урока:

- Форматирование страниц
- Нумерация страниц
- Колонтитулы
- Списки

### Основные сведения, необходимые для работы

#### Форматирование страниц

Word позволяет применять функции форматирования к отдельным частям документа, например на одной странице печатать текст в три колонки, на другой напечатать конверт, а на третьей странице печатать текст вдоль длинной стороны бумаги.

Для этого документ нужно поделить на разделы. Каждый раздел может иметь различное форматирование страниц. Раздел отделяется от других разделов маркером разрыва раздела (или просто маркером раздела).

Так же как и маркер абзаца, маркер разрыва раздела не отображается в режиме разметки страницы. Чтобы отобразить на экране разрывы разделов нужно нажать на кнопку **Непечатные символы**. Рисунок 26.

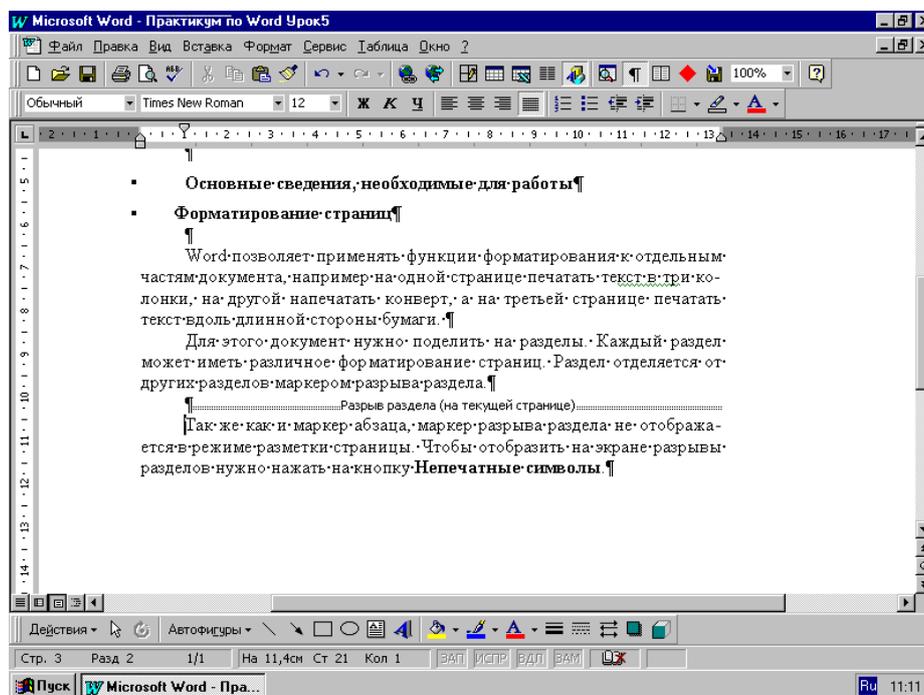


Рис. 26

Маркер разрыва раздела содержит всю информацию о форматировании

этого раздела. Как и в случае маркера абзаца, удаление маркера раздела повлечет за собой потерю этой информации, два соседних раздела соединятся вместе и приобретут одинаковый вид (в смысле форматирования).

Разрыв раздела может находиться посреди страницы, отделяя, например, текст в три колонки от обычного текста, или может определять конец страницы. Последний маркер абзаца содержит информацию о форматировании текущего раздела.

Форматирование страницы включает в себя следующие параметры:

- Разрыв страницы
- Разрыв раздела
- Поля
- Размер страницы
- Ориентация бумаги
- Нумерация страниц
- Колонтитулы

Рассмотрим эти параметры по порядку:

### **Разрыв страницы**

Разрыв страницы Word обычно ставит автоматически по достижению конца страницы. Но иногда требуется поставить разрыв страницы принудительно, например, когда нужно начать новую главу с новой страницы. Чтобы вставить разрыв страницы нажмите **Ctrl + Enter**.

### **Разрыв раздела**

Следует различать разрыв страницы от разрыва раздела. Внутри раздела может быть несколько разрывов страниц. Важно понять, что раздел представляет собой часть документа, отформатированный своим, отличным от других разделов, стилем. Чтобы установить разрыв раздела необходимо:

1. Установить курсор в начало строки, с которой будет начинаться новый раздел.
2. Открыть меню **Вставка**, выбрать пункт **Разрыв**, в диалоговом окне выбрать нужный тип разрыва.

### **Поля**

Поля определяют границы области печати текста документа, т.е. определяют основное доступное пространство для набора текста на странице. Однако существуют объекты, которые могут быть помещены на полях страницы, например колонтитулы.

Установку полей страницы можно выполнить используя меню или линейки. Рассмотрим оба способа:

## Установка полей страницы с помощью меню

1. Открыть меню **Файл**, выбрать пункт **Параметры страниц**
2. В диалоговом окне ввести значения полей **Верхнее, Нижнее, Левое, Правое**
3. В списке **Применить** выбрать параметр **Ко всему документу** или **До конца документа**. Если выбрать **Ко всему документу**, Word изменит поля всех страниц документа. Если выбрать **До конца документа**, Word вставит разрыв раздела и изменит поля всех страниц начиная с текущей и до конца документа. Предыдущие страницы останутся без изменений!

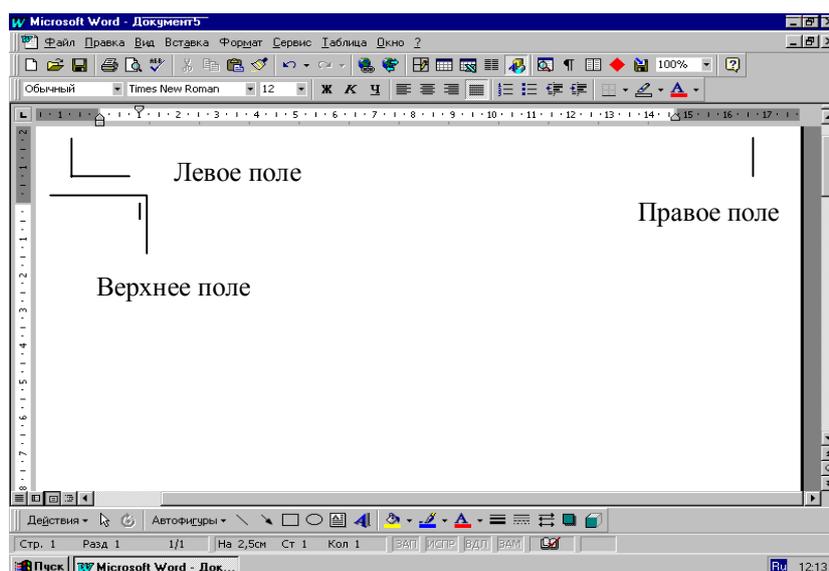


Рис. 27

## Установка полей страницы с помощью линейки

1. Подвести указатель мыши к границе поля.
2. Указатель должен принять вид двусторонней стрелки.
3. Перетащить границу поля в нужное положение.  
Если на экране не видно линеек, то откройте меню **Вид** и выберите команду **Линейка**.

## Размер страницы и её ориентация

Существует два вида ориентации бумаги - **книжная** и **альбомная**. (Не путать с половой ориентацией!)

Когда текст печатается по ширине страницы, ориентация называется книжной. Когда текст печатается вдоль длинной стороны страницы, ориентация называется альбомной.

Для изменения размера страницы и её ориентации:

1. Установите курсор в разделе, формат страницы которого вы решили изменить. Если документ состоит из одного раздела, изменения произойдут во всем документе.

2. Откройте меню **Файл**, выберите пункт **Параметры страницы**, откройте вкладку **Размер бумаги**.
3. Откройте раскрывающийся список **Размер бумаги** и выберите нужный.
4. Чтобы изменить ориентацию бумаги, установите нужный переключатель.
5. Выберите необходимое значение из раскрывающегося списка **Применить**.

## Нумерация страниц

Для автоматической нумерации страниц:

1. Откройте меню **Вставка**, выберите команду **Номера страниц**.
2. Из списка **Положение** выберите параметр **Вверху страницы** или **Внизу страницы**.
3. В списке **Выравнивание** установите способ горизонтального выравнивания номера страницы. Если вы выберете **Внутри**, то номер страницы будет располагаться у левого края нечетной страницы или у правого края четной, то есть внутри переплёта книги. Выбор параметра **Снаружи** приведет к обратному эффекту. Остальные параметры этого списка ясны без комментариев.
4. Если на первой странице номер не нужен, снимите флажок **Номер на первой странице**.
5. Чтобы изменить другие параметры номера страницы, перейдите в диалоговое окно **Формат** и установите требуемые параметры.

## Колонтитулы

Колонтитул – это текст и/или рисунок, который печатается внизу или вверху каждой страницы документа. В колонтитулах можно разместить номера страниц, названия глав, вообще любой объект.

Чтобы создать колонтитул:

1. Откройте меню **Вид**, выберите команду **Колонтитулы**. На экран выведется панель инструментов форматирования колонтитулов.
2. Введите в область колонтитула нужный текст и отформатируйте его.
3. Нажмите кнопку **Заккрыть**.

Используя панель инструментов форматирования колонтитулов, можно вставить в колонтитул один или несколько объектов. Можно вставить номера страниц, число страниц, дату, время, автотекст.

## Списки

Во многих документах встречаются списки или перечисления. Примеров перечисления несть числа в этой книге, причем почти на любой странице. Использование механизма перечисления позволяет структурировать документ. Бывают списки **нумерованные** и **маркированные**. Нумерованные списки предполагают некоторую последовательность и порядок.

### Для создания нумерованного списка:

1. Установите курсор для начала списка.
2. Нажмите на кнопку **Нумерация** на панели инструментов форматирования.
3. Напечатав элемент списка, нажмите клавишу **Enter**. Word автоматически добавит номер следующего элемента списка.
4. Напечатав последний элемент списка и нажав **Enter**, нажмите ещё раз кнопку **Нумерация**.

### Другой способ:

1. Установите курсор на новую строку.
2. Перед набором первого элемента списка наберите цифру **1** с точкой, затем нажмите клавишу **Пробел**.
3. Наберите первый элемент списка и нажмите **Enter**.
4. Word автоматически добавит номер следующего элемента списка.
5. Напечатав последний элемент списка и нажав **Enter**, нажмите клавишу **Backspace**.

Создание маркированного списка ничем не отличается от создания нумерованного списка, только нужно нажимать кнопку **Маркеры** на панели инструментов форматирования или:

1. Установите курсор на новую строку.
2. Перед набором первого элемента списка наберите символ **\***, а затем нажмите клавишу **Tab**.
3. Напечатайте элементы списка, нажимая после каждого клавишу **Enter**.
4. Напечатав последний элемент списка и нажав **Enter**, нажмите клавишу **Backspace**.

Можно изменить стиль маркера в маркированных списках или формат цифр в нумерованных списках. Для этого надо использовать меню:

1. Откройте меню **Формат**, выберите пункт **Список**, выберите понравившийся вам стиль
2. Если хотите изменить дополнительные параметры, нажмите кнопку **Изменить**.

### Практическая часть урока

- Наберите приведенный ниже текст в рамке.
- Используя материал уроков 1 – 5 полностью отредактируйте и отформатируйте документ, причем используйте все режимы форматирования: символов, абзацев, страниц. Создайте несколько разделов в документе. В каждом разделе примените разные способы форматирования, в том числе измените размеры страниц, ориентацию бумаги, измените поля, вставьте колонтитулы.

- Вставьте нумерацию страниц, причем двумя способами, используя колонтитулы и простой способ нумерации.

## **Классификация основных средств**

### *Введение*

С переходом в рыночные отношения предприятия перешли на самофинансирование и самоокупаемость. Это повышает роль учета и контроля за рациональным использованием всех ресурсов, в том числе и основных средств. Значение основных средств в общественном производстве определяется тем, какое место занимает орудия труда в развитии производительных сил и производственных отношений. С развитием технического прогресса усиливается зависимость общественного производства от состояния и уровня использования основных средств.

Орудия труда, являясь важным звеном в НТП, все более подчиняют себе не только сам процесс труда, но и весь процесс производства, его технологию и организацию. Качественное состояние основных средств, степень их развития и соответствия последним достижениям НТП определяет уровень производительных сил общества, способность его производственного аппарата решать поставленные задачи.

В процессе перестройки экономики обновление основных средств путем широкого внедрения передовой техники и новейших технологических процессов приобретает первостепенное значение. Новые методы хозяйствования, резко расширяя границы самостоятельности предприятий, призваны создавать условия для последовательного осуществления всего взаимосвязанного комплекса мер по интенсификации обновления основных средств.

На предприятиях источником интенсивного обновления средств труда в действующем производстве будут служить зарабатываемые трудовым коллективом средства, аккумулированные в фонде развития производства, науки и техники.

Механизация учета основных средств на базе ЭВМ вызывает необходимость переноса данных содержащихся в первичных документах, на машинные носители информации. Содержание дубликатов первичных документов требует затрат, которые оправданы лишь в том случае, когда последующая механизированная обработка этих данных обеспечивает достаточно большую экономию. Автоматизированная обработка первичных документов по учету движения основных средств обуславливает возможность отказаться от ведения инвентарных карточек, инвентарных описей, автоматически могут быть получены и сводные показатели движения основных средств. Что позволит заменить карточки учета движения основных средств единым показателем, характеризующим роль изменения месячных сумм затрат на законченный капитальный ремонт. С 1 января 1998 года в нашей стране был введен кыргызский стандарт бухгалтерского учета, разработанный на основе международного стандарта. КСБУ-8, который называется "Учет основных средств и прочих долгосрочных активов", разработанный на основе МСБУ-4 и МСБУ-16. Целью данного стандарта является определение бухгалтерской методики учета основных средств и прочих долгосрочных активов, находящихся во владении и контролируемых субъектами.

В постановке правильного учета и для получения достоверной информации об объектах основных средств определенное значение имеет их классификация по различным признакам. Классификация используется в целях обеспечения единообразия группировки основных средств в учете и отчетности всеми организациями независимо от их отраслевой принадлежности и организационно- правовой формы.

Группировка основных средств по их видам состоит из материальных основных фондов и нематериальных основных фондов, которые в бухгалтерском учете принято обозначать соответственно как основные средства и нематериальные активы, и учитывать отдельно.

Объектом классификации материальных основных фондов (основных средств) является объект со всеми приспособлениями и принадлежностями

или отдельно конструктивно обособленный предмет, предназначенный для выполнения определенных, самостоятельных функций или же обособленный комплекс конструктивно - сочлененных предметов, представляющих собой единое целое, предназначенный для выполнения определенной работы. Комплекс конструктивно - сочлененных предметов - это один или несколько предметов одного или разного назначения, имеющих общие приспособления и принадлежности, общее управление, смонтированных на одном фундаменте, в результате чего каждый входящий в комплекс предмет может выполнять свои функции только в составе комплекса, а не самостоятельно.

Материальные основные фонды (основные средства) сгруппированы в разделах:

**"Здания" (кроме жилых)**

**"Сооружения"**

"Жилища"

**"Машины и оборудования"**

**"Средства транспортные"**

**"Инвентарь производственный и хозяйственный"**

**"Скот рабочий, продуктивный, и племенной"**

**"Насаждения многолетние"**

**"Прочие основные фонды".**

К разделу **"Здания" (кроме жилых)** относятся здания, представляющие собой архитектурно- строительные объекты, назначением которых является создание условий для труда, социально- культурного обслуживания населения и хранения материальных ценностей. Здания имеют в качестве основных конструктивных частей стены и крышу.

Объектом классификации является каждое отдельно стоящее здание. Если здания прилегают друг к другу и имеют общую стену, но каждое представляет собой самостоятельное конструктивное целое, их считают отдельными объектами. Наружные пристройки к зданию, имеющие самостоятельное хозяйственное значение, отдельно стоящие здания котельных, а также надвор-

ные постройки (склады, гаражи, ограждения, сараи, заборы и др.) является самостоятельными объектами.

Помещения, встроенные в здания и предназначенные для магазинов, столовых, парикмахерских, ателье, пунктов проката, предметов культурно-бытового назначения и хозяйственного обихода, детских садов, яслей, отделений связи, банков и других организаций, назначение которых иное, чем основное назначение здания, входят в состав основного здания.

В состав зданий входят коммуникации внутри зданий, необходимые для их эксплуатации, как-то:

**система отопления**, включая котельную установку для отопления (если последняя находится в самом здании);

**внутренняя сеть водопровода, газопровода и канализации** со всеми устройствами;

**внутренняя сеть силовой и осветительной электропроводки** со всей осветительной арматурой;

**внутренние телефонные, сигнализационные сети;**

**вентиляционные устройства** обще санитарного назначения; подъемники и лифты.

Встроенные котельные установки, включая их оборудование, по принадлежности также относятся к зданиям. Основные фонды отдельно стоящих котельных относят к соответствующим разделам: "**Здания**" (кроме жилых), "**Машины и оборудование**" и др.

Водо -, газо - и теплопроводные устройства, а также устройства канализации включают в состав зданий, начиная от вводного вентиля или тройника у зданий или от ближайшего смотрового колодца, в зависимости от места присоединения подводящего трубопровода.

Проводку электрического освещения и внутренние телефонные и сигнализационные сети включают в состав зданий, начиная от вводного ящика или котельных концевых муфт или проходных втулок.

Фундамент под всякого рода объектами, не являющимися строениями, кот-

лами, генераторами, станками, машинами, аппаратами и прочими объектами, расположенными внутри здания – не входят в состав здания, кроме фундаментов крупногабаритного оборудования, сооруженные одновременно со строительством здания, входят в состав здания. Они, как и ряд специализированных инженерно-строительных сооружений, являются неотъемлемыми составными частями самих зданий. В этой связи в наименование группировок ряда специализированных зданий, предназначенных для осуществления конкретных видов деятельности, введено словосочетание "специализированные здания".

- После форматирования сохраните свой документ.

**Контрольные вопросы:**

1. Что такое раздел?
2. Как поставить разрыв страницы?
3. Чем отличается разрыв раздела от разрыва страницы?
4. Какие виды ориентации бумаги вы знаете?
5. Что означает книжная ориентация бумаги?
6. Что нужно сделать для автоматической нумерации страниц?
7. Что нужно сделать, чтобы номер на первой странице документа не печатался?
8. Что такое колонтитул?

## Урок 6

Содержание урока:

- 
- Работа с таблицами
- 

### Основные сведения, необходимые для работы

Таблицы позволяют разместить данные в удобном, наглядном и структурированном виде. Каждая таблица состоит из **строк** и **столбцов**. Пересечение строки и столбца образует **ячейку**. Информацию (текст или графику) вводят в ячейки. Рисунок 28.

Ячейка А1			Строка 1
	Ячейка В2		Строка 2
Столбец А	Столбец В	Столбец С	

Рис. 28

Из рисунка видно, что столбцы именуются буквами латинского алфавита, а строки нумеруются целыми числами. Имя ячейки образуется из имени столбца и номера строки.

Чтобы создать таблицу необходимо:

1. Установить курсор в том месте, где должна быть таблица (её левый верхний угол).
2. Открыть меню **Таблица**, выбрать команду **Добавить таблицу**.
3. В диалоговом окне указать количество строк и столбцов в будущей таблице.
4. Нажать кнопку **ОК**.

При вводе текста в ячейку способы набора текста, редактирования и форматирования абсолютно совпадают с теми, что при работе с обычным документом. Текст можно выравнивать, использовать отступы, менять шрифты и т.д.

### Передвижение по таблице

Для того, чтобы установить курсор в требуемую ячейку просто щелкните по этой ячейке.

Для передвижения по таблице с использованием клавиатуры применяйте следующие комбинации клавиш:

**Таблица 3**

Комбинации клавиш	Действие
Tab	Переход к следующей ячейке
Shift +Tab	Переход к предыдущей ячейке
Alt +Home	Переход к первой ячейке в строке
Alt +End	Переход к последней ячейке в строке
Alt +Page Up	Переход к первой ячейке в столбце
Alt +Page Down	Переход к последней ячейке в столбце
↑	Переход на ячейку вверх
↓	Переход на ячейку вниз

### Изменение ширины столбца

Проще всего для изменения ширины столбцов воспользоваться линейкой. См. рисунок 29.

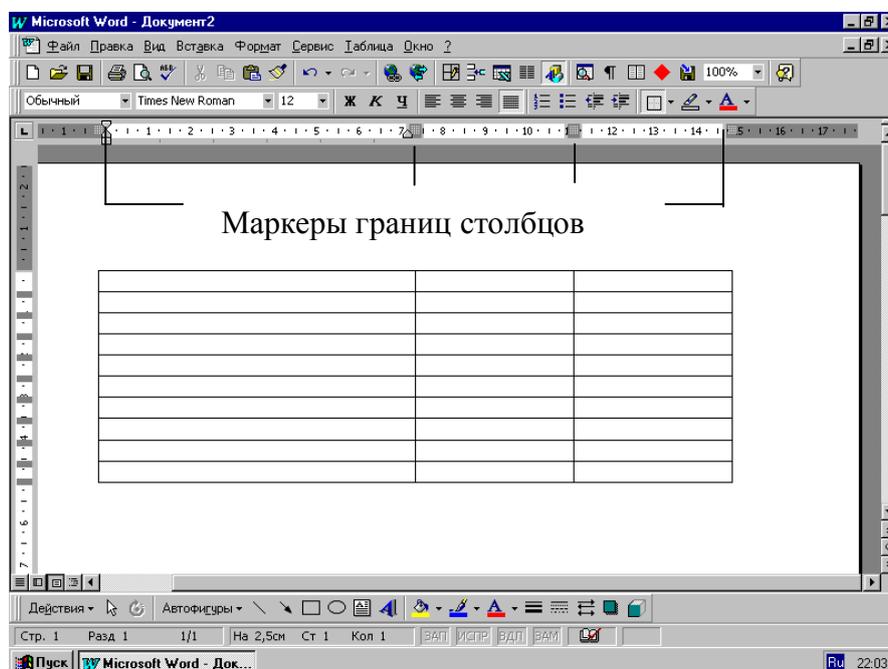


Рис. 29

При создании таблицы на линейке отображаются маркеры границ столбцов. Для того, чтобы изменить границы столбцов нужно подвести указатель мыши на нужную границу столбца. Указатель должен измениться на двустороннюю стрелку. Тогда нужно просто перетащить маркер границы столбца на нужное место. При этом будет отображаться пунктирной линией новая граница столбца. Как только вы отпустите кнопку мыши, граница столбца передвинется на новое место.

Этот способ наиболее употребителен и удобен, но при этом способе

нельзя точно установить одинаковую ширину у нескольких столбцов. Если необходимо установить одинаковую ширину у нескольких столбцов, то сначала надо их выделить. Затем откройте меню **Формат**, выберите пункт **Выровнять ширину столбцов**.

В качестве тренировки создайте таблицу, изображенную на рисунке 30.

<b>Ведомость заработной платы за</b>		<b>2002 г.</b>	
<b>№</b>	<b>Фамилия</b>	<b>Сумма к выдаче</b>	<b>Подпись</b>
1.	Иванов И.И.	500,00	
2.	Петров П.П.	545,50	
3.	Сидоров С.С.	435,00	
4.	Недотепкин С.Н.	2000,00	

Рис. 30

### **Добавление строк и столбцов**

Не всегда удается заранее точно предусмотреть количество строк или столбцов в таблице. Иногда приходится добавлять строки и столбцы в таблицу.

Чтобы добавить строку в таблицу:

1. Установите курсор в нужную строку таблицы.
2. В меню **Таблица** выберите команду **Добавить строки**
3. Новая строка добавится выше той строки, где стоял курсор.
4. Чтобы добавить несколько строк в начало или середину таблицы, выделите столько строк, сколько хотите добавить. Причем выделение начинайте с той строки, откуда хотите вставить новые строки. Далее как в п.2.
5. Если хотите вставить новые строки в конец таблицы, то установите курсор "впритык" после последней строки таблицы.
6. Выберите команду **Добавить строки** в меню **Таблица**.
7. В диалоговом окне укажите количество добавляемых новых строк.

Чтобы добавить столбец в таблицу:

1. Выделите столбец. Для выделения столбца щелкните по верхней границе столбца, когда курсор примет вид маленькой черной стрелочки, направленной вниз.
2. В меню **Таблица** выберите команду **Добавить столбец**. Столбец вставится слева от того столбца, который вы выделили.
3. Для того чтобы добавить столбец после последнего столбца таблицы, щелкните мышью сразу после правого верхнего угла самой верхней ячейки последнего столбца "впритык". Появятся черные квадратики вдоль последнего столбца.
4. Далее как в пункте 2.

## Выделение ячеек

Как уже отмечалось, содержимое ячейки можно форматировать любыми уже изученными вами методами. Но если таблица содержит очень много ячеек, форматировать каждую ячейку по отдельности становится неудобно. Word позволяет форматировать сразу несколько ячеек или даже всю таблицу. Для этого нужно научиться выделять группу ячеек. См. таблицу 4.

**Таблица 4**

Объект выделения	Необходимые действия
Ячейка	Щелкнуть мышью чуть правее левой границы ячейки. Указатель должен иметь форму стрелки, направленной вправо.
Несколько ячеек	Нажать левую кнопку мыши и, не отпуская её, перемещать над нужными ячейками. Или установить курсор в первую выделяемую ячейку, нажать клавишу <b>Shift</b> и, не отпуская её, щелкнуть по последней выделяемой ячейке.
Строка	Щелкнуть рядом с левой границей строки. Указатель должен иметь форму стрелки, направленной вправо. Или воспользоваться командой <b>Выделить строку</b> в меню <b>Таблица</b> .
Столбец	Щелкнуть по верхней границе столбца. Указатель должен иметь форму маленькой черной стрелочки, направленной вниз. Или воспользоваться командой <b>Выделить столбец</b> в меню <b>Таблица</b> .
Вся таблица	Можно воспользоваться методом выделения нескольких ячеек, выделив при этом все ячейки таблицы, но проще всего воспользоваться командой <b>Выделить таблицу</b> в меню <b>Таблица</b> .

## Объединение и разбиение ячеек

При создании сложных таблиц часто приходится объединять или разбивать ячейки.

Чтобы объединить несколько ячеек в одну необходимо:

1. Выделить нужные ячейки
2. Открыть меню **Таблица**, выбрать команду **Объединить ячейки**. Рисунок 31 и 32.

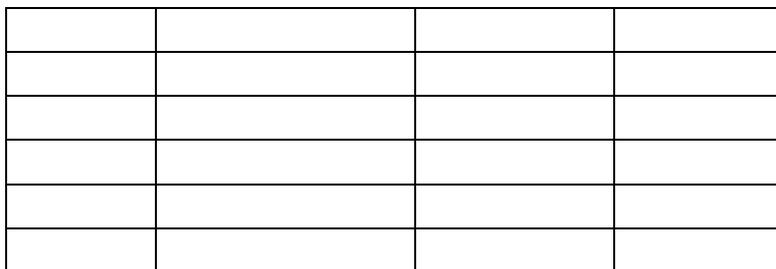



Рис. 31 Таблица до объединения ячеек первой строки

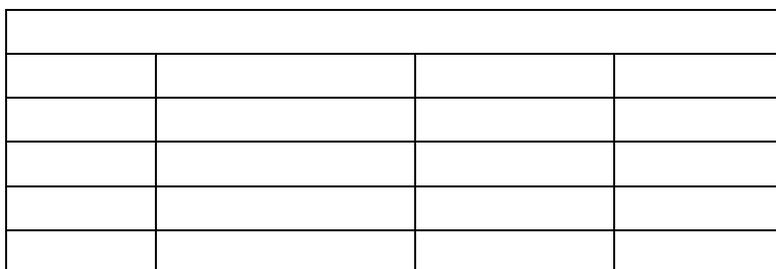



Рис. 32 Таблица после объединения ячеек

Чтобы разбить ячейку (ячейки) на несколько ячеек:

1. Установить курсор на нужной ячейке, если разбивается одна ячейка или выделить группу ячеек.
2. Открыть меню **Таблица**, выбрать команду **Разбить ячейки**.

На рисунке 33 показана таблица после разбиения столбцов В и D. Причем столбец В разбит на три столбца, а столбец D на два столбца.

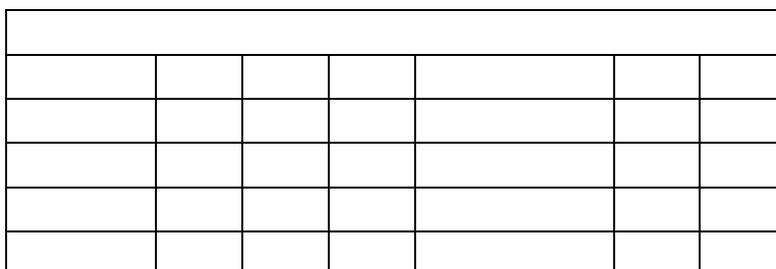



Рис. 33

Кроме указанных действий, можно менять высоту строк, ширину столбцов, высоту и ширину ячейки.

**Использование панели инструментов для создания**

## и форматирования таблицы

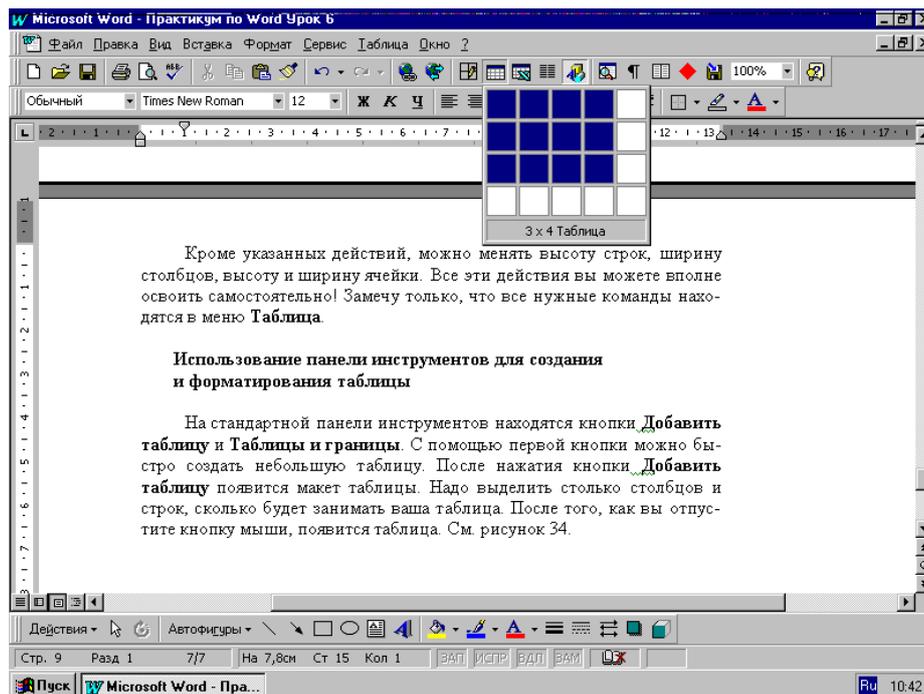


Рис. 34.

На стандартной панели инструментов находятся кнопки **Добавить таблицу** и **Таблицы и границы**. С помощью первой кнопки можно быстро создать небольшую таблицу. После нажатия кнопки **Добавить таблицу** появится макет таблицы. Надо выделить столько столбцов и строк, сколько будет занимать ваша таблица. После того, как вы отпустите кнопку мыши, появится таблица. См. рисунок 34.

Кнопка **Таблицы и границы** выводит дополнительную панель **Форматирование таблицы**. В ней продублированы практически все команды меню **Таблица** и добавлены ещё несколько кнопок для установки типов линий в таблице, их толщины, цветов границ, цветов заливки и т.д. Каждая кнопка этой панели снабжена всплывающими подсказками. Попробуйте отформатировать вашу тренировочную таблицу (рис.30) так как показано на рисунке 35.

Ведомость заработной платы за			2002 г.
№	Фамилия	Сумма к выдаче	Подпись
1.	Иванов И.И.	500,00	
2.	Петров П.П.	545,50	
3.	Сидоров С.С.	435,00	
4.	Недотепкин С.Н.	2000,00	

Рис. 35

## Создание сложных таблиц

Существует два способа создания таблиц сложной структуры. Первый способ использует методы объединения и разбиения ячеек, а также использует методы изменения границ рамки. С объединением и разбиением ячеек вы уже познакомились. Рассмотрим методы изменения границ рамки. На панели форматирования таблицы имеется кнопка **Внешние границы**. (Кстати, точно такая же кнопка имеется и на панели форматирования). Рисунок 36.

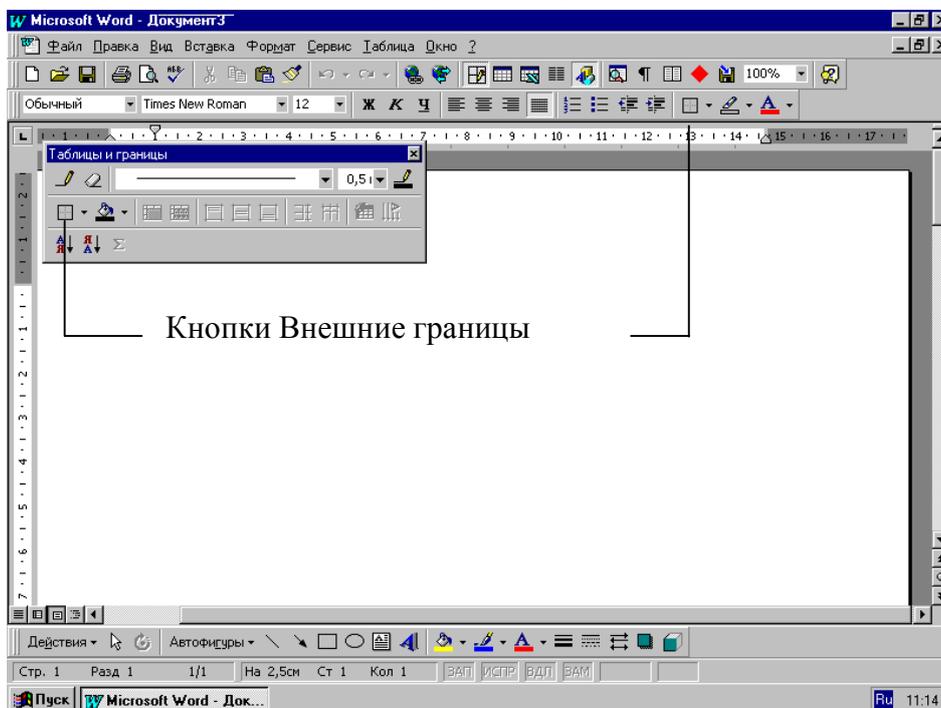


Рис. 36

Установив курсор в нужную ячейку и воспользовавшись кнопкой **Внешние границы**, вы можете добавить линии границ или убрать ненужные линии. Таким образом можно создать таблицу сложной структуры. На рисунке 37 показан образец таблицы сложной структуры. Попробуйте воспроизвести эту таблицу в своем документе.


Рис. 37

Второй способ создания таблиц сложной структуры заключается в рисовании с помощью специального **карандаша и ластика**. Однако, я не рекомендую рисовать всю таблицу. Сначала создайте простую таблицу, а затем пририсуйте необходимые линии и границы.

Эта возможность полезна при составлении таблиц с неравными по размеру ячейками. А теперь попробуйте создать таблицу рисунка 37 с использованием карандаша и ластика.

### Автоформат таблицы

Существует возможность автоматического форматирования таблицы. (Имеется в виду вся таблица, т.е. её внешний вид. О форматировании содержимого ячеек здесь речь не идет).

Автоформат таблицы можно провести как при создании (в этом случае создавать таблицу нужно через меню), так и после создания и заполнения таблицы данными.

При создании таблицы в диалоговом окне создания вы можете открыть вкладку автоформат. Если таблица уже создана, то откройте меню **Таблица** и выберите команду **Автоформат**. Откроется диалоговое окно. Рисунок 38. В списке **Форматы** можно выбрать вид таблицы. В окне **Образец** вы можете видеть как будет выглядеть таблица. Кроме того, можно задать ряд параметров форматирования.

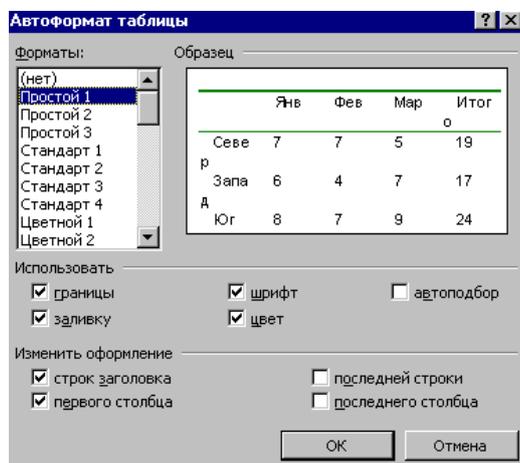


Рис. 38

### Сортировка и вычисления в таблицах

Существует возможность сортировать информацию в таблице. Сортировка может производиться либо по возрастанию, либо по убыванию для чисел и по алфавиту для текстовой информации.

Для сортировки необходимо:

1. Выделить строки подлежащие сортировке
2. Открыть меню **Таблица**, выбрать команду **Сортировка** или на пане-

ли инструментов форматирования таблицы нажать на кнопку **Сортировка по возрастанию**, либо на кнопку **Сортировка по убыванию**.

Отсортируем, например таблицу рисунка 30 по алфавиту фамилий (рисунок 39).

<b>Ведомость заработной платы за</b>			<b>2002 г.</b>
№	Фамилия	Сумма к выдаче	Подпись
1.	Иванов И.И.	500,00	
4.	Недотепкин С.Н.	2000,00	
2.	Петров П.П.	545,50	
3.	Сидоров С.С.	435,00	

Рис. 39

В таблицах можно производить вычисления. Конечно, в Word можно производить лишь некоторые, самые простые вычисления, например подсчет суммы чисел в строке или столбце (итоги), округление чисел, нахождение максимального или минимального и т. д.

Для выполнения сложных табличных расчетов существует специальная программа в составе Microsoft Office, которая называется Excel 97.

Здесь мы рассмотрим самую распространенную задачу – подсчет итоговых сумм.

Для подсчета суммы значений по строкам или столбцам:

1. Установить курсор на пустой ячейке справа или под ячейками, итоговую сумму которых надо рассчитать.
2. Щелкнуть по кнопке **Автосумма** на панели инструментов форматирования таблицы.

Подсчитаем, например, итоговую сумму выплат по ведомости (рис. 39). Для этого:

1. Сначала надо в таблицу добавить пустую строку. Как это сделать вы уже знаете. Если забыли, повторите соответствующий раздел этого урока.
2. Подведите курсор в пустую ячейку, расположенную в столбце **Сумма к выдаче**.
3. Нажмите кнопку **Таблицы и границы** в стандартной панели инструментов.
4. Нажмите в появившейся панели форматирования таблицы кнопку **Автосумма**.

Ведомость заработной платы за \_\_\_\_\_ 2002 г.

№	Фамилия	Сумма к вы- даче	Подпись
1.	Иванов И.И.	500,00	
4.	Недотепкин С.Н.	2000,00	
2.	Петров П.П.	545,50	
3.	Сидоров С.С.	435,00	
	<b>Итого:</b>	3480,5	

Если вы хотите произвести другие математические операции, кроме автосуммирования, то:

1. Установите курсор на пустой ячейке справа или под ячейками, над которыми надо выполнить математические операции.
2. Откройте меню **Таблица**, выберите пункт **Формула**.
3. В диалоговом окне в раскрывающемся списке **Вставить функцию** выберите нужную вам функцию.
4. Если хотите, можете выбрать нужный вам формат чисел.
5. Нажмите **ОК**.

### Практическая часть урока

- Подготовьте накладную, образец которой приведен ниже.
- Отформатируйте содержимое ячеек.
- Отформатируйте таблицу в целом
- Выполните сортировку и подсчитайте итоги

### **Накладная № 27**

ОсОО **Каскад-Ош** ЛТД, р/с 4172000200116 в "Кайрат – банке",  
МФО 330601791, ИНН № 0120699810010

Кому: ОшТУ, факультет "Кибернетики и информационных технологий"

Тип поставки: по договору Дата поставки: 28 августа 2001 г.

№	Наименование	Кол- во	Цена за ед.	Сумма
1	Жесткий диск 3.2 Гб	1	6584,00	6584,00
2	CD – Золотая бухгалтерия	1	200,00	200,00
3	CD – Windows 95	1	200,00	200,00
4	Полное руководство по Win- dows-95	2	324,00	648,00
5	Microsoft Word 97 Справочник	1	170,00	170,00
6	Word 97 Русская версия	2	170,00	340,00
7	Excel 97 Шаг за шагом	1	170,00	170,00
8	Access 97 для пользователя	1	225,00	225,00
	<b>Итого:</b>			8537,00

Сумма прописью: Восемь тысяч пятьсот тридцать семь сом.

Руководитель \_\_\_\_\_ Отпустил \_\_\_\_\_

МП

Гл. бухгалтер \_\_\_\_\_ Получил \_\_\_\_\_

**Контрольные вопросы:**

1. Что такое ячейка таблицы?
2. Как перейти к следующей ячейке справа от текущей?
3. Как установить точную ширину столбцов?
4. Как добавить строки в конец таблицы?
5. Как сразу выделить строку таблицы?
6. Как сразу выделить столбец таблицы?
7. Как объединить несколько ячеек?
8. Как отсортировать таблицу?
9. Как подсчитать итоговую сумму по строкам?

## Урок 7

Содержание урока:

- 
- Работа с графикой
  - WordArt
  - Буквица
- 

### Основные сведения, необходимые для работы

Word позволяет вставить в документ различные графические объекты. Это может быть рисунок, фотография, слайд, видеоклипы и другие типы графических объектов.

### Вставка рисунка

1. Установить курсор в позицию вставки рисунка
2. Открыть меню **Вставка**, выбрать пункт **Рисунок**
3. Из всплывающего меню выбрать пункт **Картинки**
4. В диалоговом окне выбрать нужный раздел рисунков (все рисунки сгруппированы по разделам, например **Люди**, **Растения** и т.д.)
5. Выбрать рисунок из окна просмотра. Если хотите увеличить рисунок для того, чтобы лучше рассмотреть детали, установите флажок **Увеличить**.
6. Нажать кнопку **Вставить**.
7. После вставки, вокруг рисунка будут маленькие прямоугольники. Это маркеры границ рисунка. Они всегда появляются при вставке документа. Щелкните мышью в любое место вне этих маркеров.

#### **Внимание!**

Если на экране появится только рамка, а самого рисунка не видно, то откройте меню **Сервис**, выберите пункт **Параметры**, вкладку **Вид** и установите фла-

### Операции над рисунками

Ранее неоднократно отмечалось, что для того, чтобы выполнить какие – то действия над некоторыми объектами, их надо предварительно выделить. Так вот, появление маркеров границ рисунка и означает, что рисунок выделен. Чтобы выделить рисунок надо просто щелкнуть по нему. Чтобы отменить выделение щелкните по любому месту документа вне маркеров границ.

Над рисунками (как целыми, неделимыми объектами) можно выполнить следующие операции:

- перемещение
- изменение размеров

- копирование (через буфер обмена)
- вырезание (в буфер обмена)
- удаление

Для перемещения рисунка необходимо:

1. Выделить рисунок
2. Нажать кнопку мыши и, не отпуская, переместить рисунок на новое место.
3. Отменить выделение рисунка

Для изменения размеров рисунка необходимо:

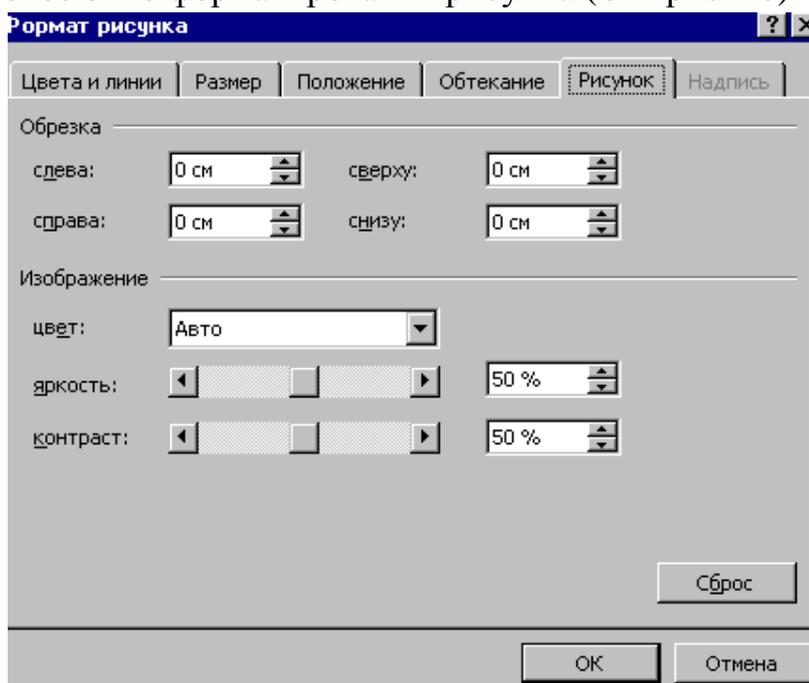
1. Выделить рисунок
2. Установить указатель мыши на один из маркеров границ. При этом указатель примет вид двойной стрелочки.
3. Переместить маркер в сторону увеличения или уменьшения размеров рисунка. Если вы хотите изменить размеры пропорционально, то используйте угловые маркеры. Если же вы хотите отдельно изменить высоту или ширину, то используйте центральные маркеры.

Остальные операции выполняются точно так же, как и операции над выделенным фрагментом текста.

### **Форматирование и настройка параметров рисунка**

Для форматирования рисунка:

1. Выделите рисунок
2. Щелкните правой клавишей, выберите пункт **Формат рисунка** или откройте меню **Формат** и выберите пункт **Рисунок**. Появится диалоговое окно форматирования рисунка (См. рис. 40).



3. Откройте вкладку **Цвета и линии**. Здесь вы можете добавить к рисунку рамку, цвет рамки, а также цвет фона рисунка, т.е. сделать заливку.
4. Во вкладке **Размер** можете изменить размеры рисунка, но проще изменять размеры с помощью мыши, так как мы это делали в предыдущем пункте.
5. Во вкладке **Положение** можно установить положение рисунка на странице (опять же удобнее это делать с помощью мыши путем перемещения), а также установить **привязку**.  
Под привязкой понимается установление связи рисунка с абзацем. Если позднее вы будете изменять документ, вставляя или удаляя абзацы, рисунок будет перемещаться вместе с абзацем, с которым установлена связь. Для этого надо установить флажок **Перемещать вместе с текстом**.  
Чтобы рисунок всегда находился на одной странице с абзацем, с которым он связан, установите флажок **Установить привязку**.

Рис. 40

6. Во вкладке **Обтекание** можно установить способы обтекания текстом рисунка. Рассмотрим эти способы:
  - Обтекание **Вокруг рамки**. Вокруг рисунка создается невидимая рамка и текст обтекает рисунок вокруг этой рамки.
  - Обтекание **По контуру**. Текст "обволакивает" рисунок. Можно регулировать плотность прилегания текста к рисунку.
  - **Сквозное** обтекание. Текст пропечатывается в области рисунка в свободных местах.
  - **Нет обтекания**. Текст налагается на рисунок. В этом случае, часть рисунка или текста может стать невидимой.
  - Обтекание **Сверху и снизу**. По бокам рисунка текст не печатается.
7. Во вкладке **Рисунок** можно регулировать яркость и контрастность, вырезать часть рисунка. Чтобы не путать с обычным вырезанием, когда вырезается весь рисунок и помещается в буфер, применяется термин **Обрезка**.

### **Панель настройки изображения**

Во многих случаях для форматирования и настройки параметров рисунка удобнее воспользоваться панелью **Настройка изображения**. Для того чтобы вывести эту панель щелкните правой клавишей по рисунку. Вместе с диалоговым окном выведется панель настройки изображения. В случае её отсутствия, в диалоговом окне выберите команду **Отобразить панель настройки изображения**. Рисунок 41.

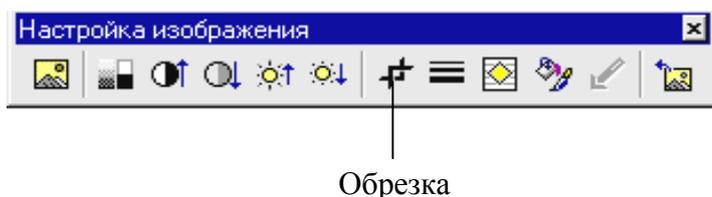


Рис. 41

В этой панели продублированы все команды форматирования рисунка в виде кнопок. Все кнопки снабжены всплывающими подсказками, поэтому предлагаю читателю самостоятельно разобраться в работе с этой панелью. Я продемонстрирую лишь метод обрезки рисунка. Рисунок 42.

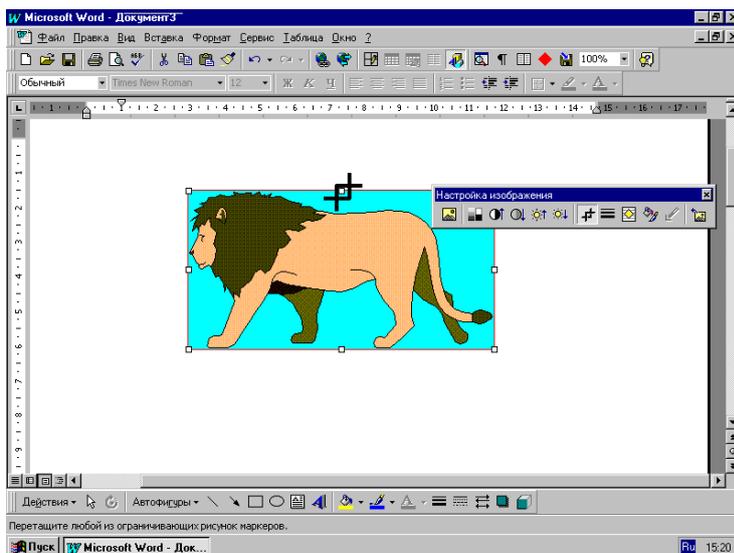


Рис. 42

Чтобы обрезать часть рисунка:

1. Выделите рисунок
2. Щелкните правой клавишей, чтобы вывести панель настройки изображения.
3. Нажмите на кнопку **Обрезать**
4. Когда будете перемещать указатель, на его "кончике" будет "висеть" значок обрезки.
5. Переместите этот значок на один из маркеров границы.
6. Нажмите на левую кнопку мыши и ведите до нужного места рисунка.
7. Отпустите кнопку мыши. Рисунок 43.

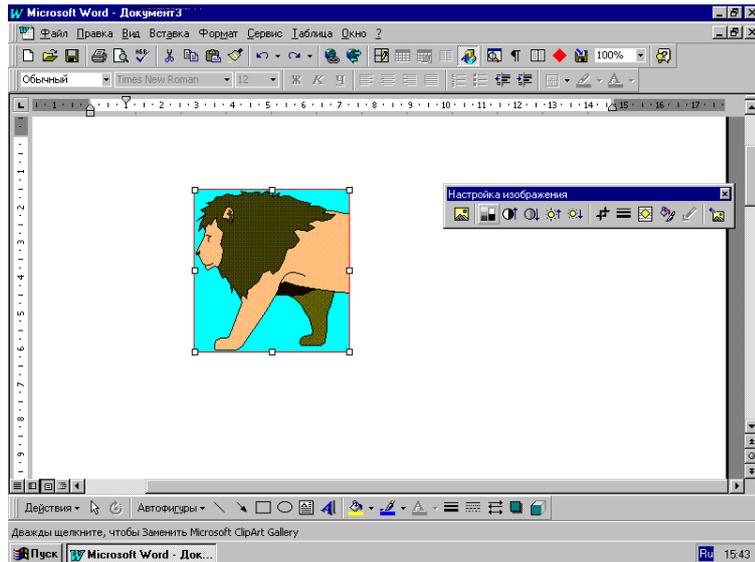


Рис. 43

### Добавление названия к рисунку

В документах желательно, чтобы каждый рисунок был снабжен надписью. Можно вставить стандартное название, предлагаемое Word – **Рисунок 1**, **Рисунок 2** и т.д., а можно и своё собственное.

Для добавления названия к рисунку:

1. Выделить рисунок
2. Открыть меню **Вставка**, выбрать пункт **Название**. Появится диалоговое окно. См. рисунок 44.

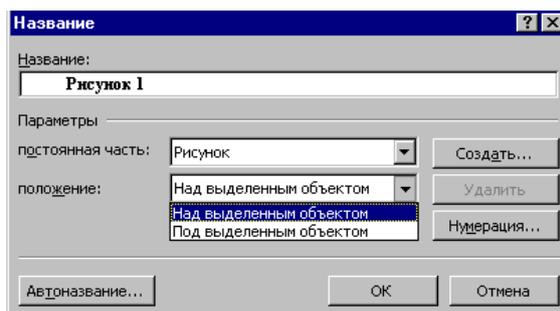


Рис. 44

3. Если вас устраивает стандартное название Рисунок, то нажмите кнопку **ОК**.
4. Если нет, то в поле ввода **Название** введите текст названия.
5. Выберите местоположение названия (Над выделенным объектом/Под выделенным объектом).
6. Если у вас много однотипных рисунков, то вы можете создать посто-

- янную часть названия, нажав кнопку **Создать**.
7. Вы можете так же ввести нумерацию рисунков.

### Работа с WordArt

WordArt является отдельным графическим приложением. Он позволяет преобразовывать текст в графику. Чтобы запустить WordArt:

1. Откройте меню **Вставка**, выберите пункт **Рисунок**, из всплывающего меню выберите **Объект WordArt..**
2. Появится окно **Коллекция WordArt**. Рисунок 45.



Рис. 45

3. Выберите эффект преобразования текста и нажмите **ОК**
4. Появится еще одно диалоговое окно. Рисунок 46.

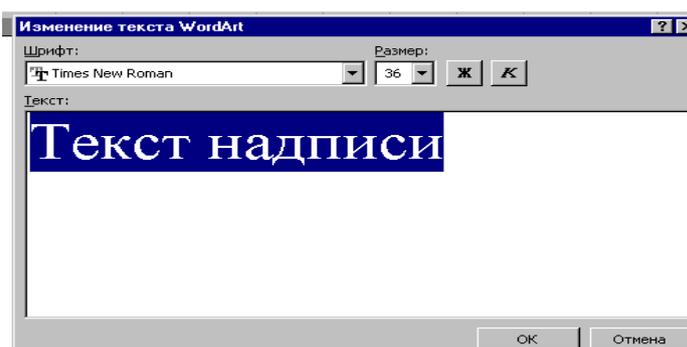


Рис. 46

5. Наберите свой текст в окне **Текст**
6. Выберите тип шрифта и его размер.
7. Выберите тип оформления символов шрифта (**Полужирный** или **Курсив**).
8. Нажмите **ОК**.

Объект WordArt вставится в документ. Одновременно появится панель инст-

рументов форматирования объекта WordArt.

Все кнопки панели, как обычно, снабжены всплывающими подсказками. Используя эти кнопки, вы можете сделать следующие действия:

- Щелкнув по кнопке **Формат объекта WordArt**, отформатировать объект WordArt. Диалоговое окно, смысл действий по форматированию абсолютно идентичен форматированию рисунков.
- Изменить форму расположения текста. Нажмите на кнопку **Форма WordArt**.
- Повернуть текст на некоторый угол. Нажмите на кнопку **Свободное вращение**.
- Вертикально расположить текст. Нажмите на кнопку **Вертикальный текст WordArt**.
- Установить одинаковую высоту символов. Нажмите на кнопку **Выровнять буквы WordArt по высоте**.
- Выровнять текст. Нажмите на кнопку **Выравнивание WordArt**.
- Установить межсимвольный интервал. Нажмите на кнопку **Межсимвольный интервал WordArt**.
- Изменить текст. Нажмите на кнопку **Изменить текст**.
- Добавить новый объект WordArt. Нажмите на кнопку **Добавить объект WordArt**.

## Буквица

**П**ри помощи буквицы можно увеличить первую букву или слово абзаца. Такие эффекты используются обычно в начале главы при написании художественной литературы.

Для создания буквицы:

1. Установите курсор в абзаце, в котором хотите создать буквицу.
2. Откройте меню **Формат**, выберите команду **Буквица**.
3. В диалоговом окне выберите значение параметра **Положение**.
4. Выберите тип шрифта из раскрывающегося списка **Шрифт**.
5. В поле ввода **Высота в строках** введите соответствующее значение. Этот параметр указывает на то, сколько места будет занимать высота буквицы.
6. В поле ввода **Расстояние до текста** введите необходимое значение. Это параметр определяет величину отступа буквицы от текста.

## Практическая часть урока

- Вставьте какой –нибудь рисунок в свой документ.
- Выполните все операции по форматированию и настройке параметров рисунка.

- Попробуйте обрезать часть рисунка. (Если при выполнении операций по форматированию и настройке, вы сделали что-то не так, как хотелось, вы можете вернуть первоначальный вид рисунка и размер, нажав на кнопку **Сброс** в панели настройки изображения).
- Вставьте название к рисунку.
- Добавьте в документ титульный лист с использованием Word Art.
- Добавьте в документ в нескольких абзацах буквицу.
- Сохраните свой документ под тем же именем.

### **Контрольные вопросы:**

1. Как выделить рисунок?
2. Какие операции допустимы над рисунками?
3. Что означает привязка рисунка?
4. Назовите и объясните способы обтекания рисунка текстом.
5. Как ввести нумерацию рисунков?
6. Что такое буквица?

## Глава V. Основы программирования

### §5.1. Понятие алгоритма.

Компьютер - это устройство для решения задач. Не обязательно задачи чисто математического характера. Это могут быть и шахматные задачи, и задачи управления станками или ракетами, и задачи планирования производства, и задачи информационно- справочного обслуживания. Чтобы решить какую - либо задачу на компьютере необходимо сначала придумать как ее вообще решить, т.е. придумать алгоритм ее решения.

Что такое алгоритм?

Само по себе слово "алгоритм" (algorithm) очень интересно: на первый взгляд может показаться, будто кто-то намеревался написать слово "логарифм" (logarithm), но перепутал порядок первых четырех букв. Имеется старая форма этого слова "algorism", что означает правила выполнения арифметических действий с использованием арабских цифр. В средние века абакисты считали на счетах, алгоритмики использовали зачатки математической символики (арабские цифры и операции +, -, \*, / по правилам предложенным аль- Хорезми).

И слово algorism произошло от имени этого ученого Abu Za' far Mohammed ibn Musa al- Khowarizmi означающее буквально: "Отец Джафара, Магомет, сын Мусы, уроженец Хорезма". Вышеназванный уроженец Хорезма написал знаменитую книгу: "Kitab al jabr W'al- muqabala" (правила действий над числами). Название книги дало начало другому слову "алгебра", хотя книга было совсем не алгебраической. Постепенно форма и значения слово "algorism" искажилось на "algorithm".

Итак, что понимается под алгоритмом?

Алгоритм - это строгая и четкая, конечная система правил, которая определяет последовательность действий над некоторыми объектами и после конечного числа шагов приводит к достижению поставленной цели.

Из определения алгоритма следует, что он должен удовлетворять следующим требованиям:

1) конечность (финитность)

Алгоритм всегда должен заканчиваться после конечного числа шагов. Процедуру обладающую всеми характеристиками алгоритма, за исключением конечности, вызывают вычислительным методом.

2) определенность (детерминированность)

Каждый шаг алгоритма должен быть строго определен. Действия, которые необходимо произвести, должны быть строго и недвусмысленно определены в каждом возможном случае, так чтобы если дать алгоритм нескольким людям, то они действуя по этому алгоритму получали один и тот же результат. Поскольку обычный язык полон двусмысленностей, то чтобы преодолеть это затруднение, для описания алгоритмов разработаны формально определенные языки программирования, или машинные языки, в которых каждое утверждение имеет абсолютно точный смысл.

Запись алгоритма на языке программирования называется программой.

3) Алгоритм должен иметь некоторое число входных данных, т.е. величин, объектов заданных ему до начала работы. Эти данные берутся из некоего конкретного множества объектов.

4) Алгоритм имеет одну или несколько выходных величин, т.е. величин, имеющих вполне определенное отношение ко входным данным.

5) Эффективность

От алгоритма требуют, чтобы он был эффективным. Это означает, что все операции, которые необходимо произвести в алгоритме, должны быть достаточно простыми, чтобы их в принципе можно было выполнить точно и за конечный отрезок времени с помощью карандаша и бумаги.

Попытаемся сравнить алгоритм с рецептом в кулинарной книге. Рецепты обладают свойством финитности, имеют входные данные (соль, мука, яйца и пр.) и выход (званный обед и т.п.), но отличаются печально известным отсутствием определенности. Возьмем например инструкцию: "Добавьте щепотку соли". Она не вполне определена, т.к. неясно сколько означает щепотка. Инструкции типа "Слегка потрясите смесь, пока она не станет комковатой", "подогрейте коньяк в маленькой кастрюльке" являются, возможно, вполне достаточными для опытного повара, но алгоритм должен быть четко определен, настолько чтобы даже компьютер смог следовать инструкции.

Следует отметить, что для практических целей ограничение "финитности" на самом деле не является жестким: используемый алгоритм должен иметь не просто конечное, а предельно конечное, разумное число шагов. Например, в принципе имеется алгоритм, определяющий, является ли начальное положение в игре в шахматы форсированно выигранным для белых или нет. Но для выполнения этого алгоритма требуется фантастически огромный промежуток времени. Пусть имеется компьютер, обладающий быстродействием 100 млн. операций в секунду. Тогда этот компьютер будет выполнять алгоритм в течение  $10^{23}$  лет. Для сравнения: период времени с начала возникновения жизни на земле и до наших дней меньше  $10^{23}$  лет.

### Пример алгоритма.

Алгоритм Евклида нахождения наибольшего общего делителя двух целых чисел, т.е. наибольшее целое число, которое делит нацело заданные числа.

1. Рассмотреть А как первое число и В как второе число. Перейти к п.2.
  2. Сравнить первое и второе числа. Если они равны, то перейти к п.5. Если нет, то перейти к п.3.
  3. Если первое число меньше второго, то переставить их местами. Перейти к п.4.
  4. Вычесть из первого числа второе и рассмотреть полученную разность как новое первое число. Перейти к п.2.
  5. Рассмотреть первое число как результат.
- Стоп.

Этот набор правил является алгоритмом, т.к. следуя ему любой человек умеющий вычитать, может получить наибольший общий делитель для любой пары чисел.

$$\begin{array}{r}
 544 \text{ и } 119 \\
 A = 544 \quad B = 119 \\
 - \quad 544 \\
 \hline
 119 \\
 \hline
 425
 \end{array}$$

$$\begin{array}{r}
 A = 425 \quad B = 119 \\
 - \quad 425 \\
 \hline
 119 \\
 \hline
 306
 \end{array}$$

$$\begin{array}{r}
 A = 306 \quad B = 119 \\
 - \quad 306 \\
 \hline
 119 \\
 \hline
 187
 \end{array}$$

$$\begin{array}{r}
 A = 187 \quad B = 119 \\
 - \quad 187 \\
 \hline
 119 \\
 \hline
 68
 \end{array}$$

$$A = 68 \quad B = 119$$

$$A < B$$

Меняем местами

$$\begin{array}{r}
 A = 119 \quad B = 68 \\
 - \quad 119 \\
 \hline
 68 \\
 \hline
 51
 \end{array}$$

$$A = 51 \quad B = 68$$

$$A < B$$

$$\begin{array}{r}
 A = 68 \quad B = 51 \\
 - \quad 68 \\
 \hline
 51 \\
 \hline
 17
 \end{array}$$

$$A = 17 \quad B = 51$$

$$A < B$$

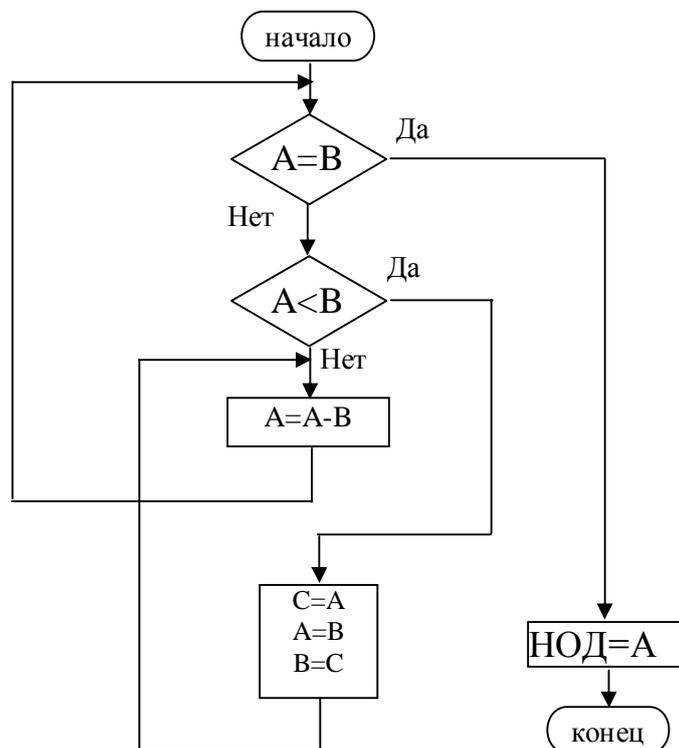
$$\begin{array}{r}
 A = 51 \quad B = 17 \\
 - \quad 51 \\
 \hline
 17 \\
 \hline
 34
 \end{array}$$

$$\begin{array}{r}
 A = 34 \quad B = 17 \\
 - 34 \\
 \hline
 17 \\
 \hline
 17 \\
 A=B=17=\text{НОД}
 \end{array}$$

Данный способ записи алгоритмов возможен, но неудобен. Во-первых, нет наглядности, во-вторых, "многословен". Одним из способов записи алгоритма являются блок-схемы. В них используют математическую символику для обозначения действий, но разрешается записывать и словами. Блок-схемой называется такое графическое изображение структуры алгоритма, в котором каждый этап или шаг процесса переработки данных представляется в виде прямоугольника, ромба, овала или другой геометрической фигуры, называемой блоком.

Эти фигуры соединяются между собой линиями со стрелками, отображающими последовательность выполнения алгоритма. Внутри каждой фигуры разрешается писать произвольный текст, в котором на понятном человеку языке сообщаются о нужных вычислениях в соответствующей части программы.

Приняты определенные стандарты графических обозначений. Так, прямоугольник, обозначает вычислительные действия в результате которых изменяются значения данных. Ромбом обозначают этап разветвления алгоритма. Выбор одного из двух возможных направлений дальнейшего счета производится в зависимости от выполнения условия, записанного в ромбе. Овалом обозначают начало и конец алгоритма. Запишем алгоритм Евклида в виде блок-схемы:



Итак, чтобы решить какую либо задачу, нужно придумать алгоритм ее решения. Затем следует представить этот алгоритм в таком виде, чтобы данный компьютер мог его выполнить. Для этого нужно, во- первых, разбить алгоритм на элементарные операции которые умеет выполнять компьютер, и, во- вторых, записать каждую такую операцию на языке понятном компьютеру. Такая запись алгоритма на языке компьютера называется программой. Процесс разработки программы называется программированием. А человек, выполняющий эту работу- программистом.

Программирование – это научная дисциплина. Если бы процессы программирования разных задач не имели между собой ничего общего, то программирование, как таковое, не было бы научной дисциплиной. Но дело обстоит не так. Существуют общие методы, которые позволяют, постепенно расчлняя задачи на подзадачи, сводить их решение, в конечном счете к некоторым элементарным операциям (чаще всего к элементарным арифметическим операциям), подобно тому, как разбирая совершенно непохожие сложные механизмы, мы обнаруживаем, что они состоят из одинаковых деталей и узлов, только по разному соединенных (напр. подшипники, болты, гайки и т.д.). Из этого, конечно, не следует что процесс программирования не содержит в себе элементов творчества. Составление программы, такой же творческий процесс, что и разработка сложного механизма из заданных наборов деталей.

Что же касается компьютеров, то, при всем их разнообразии, с точки зрения разработки алгоритмов все они очень похожи. Алгоритм можно составить один для всех типов машин. Основное назначение алгоритмов заключается в их фактическом выполнении тем или иным исполнителем. Т.е. алгоритм составляется для того, чтобы он был выполнен для решения какой либо задачи. Основное назначение компьютера состоит в фактическом исполнении алгоритмов, разработанных человеком. Таким образом, компьютер выступает как инструмент, приспособление человека для исполнения алгоритма.

Важно понять, что компьютер сам по себе ничего не делает. Он лишь слепо выполняет то, что укажет ему человек. Поэтому, когда говорят, что компьютер управляет станками, сочиняет музыку, играет шахматы «как бы самостоятельно», то это неверно!

Это человек придумывает алгоритм, т.е. способ управления станками, сочинения музыки, игры в шахматы, а компьютер лишь исполняет этот алгоритм. Таким образом, компьютер это помощник, инструмент человека для решения различных задач, но инструмент очень мощный, разноплановый. Это настоящий "пожиратель" алгоритмов, причем ненасытный. С помощью компьютера человек может решать самые разнообразные задачи. При этом человек выполняет наиболее сложную, творческую часть работы – составляет, придумывает, "творит" алгоритм решения задач, а рутинную не творческую часть работы, связанную с большими и трудоемкими вычислениями поручает выполнить компьютеру.

## §5.2. Введение в язык Турбо – Паскаль

Язык Паскаль, изобретенный в начале 70-х годов 20 века Н. Виртом и названный в честь французского математика и философа Блеза Паскаля, является одним из наиболее распространенных языков программирования. От других языков он выгодно отличается возможностью более ясно и логично записывать программы.

Здесь мы рассмотрим лишь основные конструкции и возможности языка. Более подробно изучение языка Паскаль предусмотрено в специальном курсе "Алгоритмические языки и программирование", читаемого на факультете кибернетики и информационных технологий Ошского технологического университета для студентов специальности "Программное обеспечение вычислительной техники и автоматизированных систем".

Для первого знакомства с языком достаточно тех сведений, которые изложены в этой книге.

Программа на языке Паскаль состоит из двух частей: описание действий, которые должны быть выполнены и описание данных, над которыми они выполняются. В тексте программы описание данных предшествует описанию действий. В этом выражается общее правило языка – каждый встречающийся в программе объект должен быть предварительно описан.

Описание данных состоит из описания переменных. Операторами называются действия над данными. В общем виде любая Паскаль – программа имеет вид:

заголовок программы  
раздел описания переменных  
раздел операторов

Заголовок программы имеет вид:

PROGRAM имя программы;

Здесь слово PROGRAM – это так называемое ключевое (или служебное или еще говорят зарезервированное) слово. Оно должно записываться именно так, а не иначе. Допускается использовать как строчные, так и прописные буквы. Записи PROGRAM, Program, ProgRam – разрешены и означают одно и то же.

Так начинаются все программы, написанные на языке Паскаль. Здесь нечего понимать, просто так принято разработчиком языка.

Имя программы – это любая последовательность букв, цифр и некоторых знаков, начинающихся с буквы. Такие последовательности называются идентификаторами. Идентификатор не должен совпадать ни с одним из ключевых слов. Из знаков допускаются использовать при записи идентификаторов знак \_ (подчеркивание), знаки \$, @. В идентификаторе не должно быть (.) – точки, (,) – запятой, ( ) – пробелов, знаков операций.

Примеры правильных идентификаторов:

x1  
Summa  
VOLVO  
Select\_screen\_color

Примеры неправильных идентификаторов:

3x начинается с цифры  
JAN.3 внутри идентификатора есть точка  
VOL VO есть пробел

### Переменные. Стандартные типы.

Каждая переменная имеет имя и тип. Имя переменной – это произвольный идентификатор. В дальнейшем будем говорить "переменная x", вместо "переменная с именем x"

Тип переменной определяет множество её возможных (допустимых) значений.

В Паскале существуют следующие стандартные типы переменных: `integer` (целый), `real` (вещественный), `boolean` (логический), `char` (символьный), `string` (строковый).

Значениями переменных целого типа являются целые (и только!) числа.

Примеры целых чисел:

25        +150        -200    10000

Операции над целыми числами таковы:

+ (сложение), - (вычитание), \* (умножение), `div` (деление нацело), `mod` (остаток от деления двух целых чисел).

Значениями переменных вещественного типа являются вещественные числа.

Определены следующие операции над вещественными числами:

+ (сложение), - (вычитание), \* (умножение), / (деление).

Запись вещественных чисел похожа на общепринятую, только вместо запятой используется точка и вместо степени 10 используется буква E.

Пример.

Общепринятая	на Паскале
5,30	5.30
-1,0	-1.0
41000	41000 или 4.1E4
-0,0073	-0.73E-2

Значениями переменных логического типа является `true` (истина), `false` (ложь). Определены операции: `not` (не), `and` (и), `or` (или).

Значения переменных символьного типа – символы из таблицы ASCII, о котором речь пойдет позже.

Значения переменных строкового типа – цепочка символов. При записи констант символьного и строкового типа используют одиночные кавычки.

Пример.

'A' - это символ A  
'Это цепочка символов'

### Операции отношения

Существуют следующие операции отношения:

= равно, <> не равно, < меньше, > больше, <= меньше или равно, >= больше или равно.

Результатом этих операций являются логические значения true или false.

Следует отметить, что над символьными переменными определены лишь операции = и <>.

### Раздел описаний переменных

Этот раздел имеет вид:

var описание 1; описание 2; ...; описание n;

var – ключевое слово (от английского variable – переменная)

описание имеет вид:

переменная 1, переменная 2, ..., переменная m: тип;

переменная 1, переменная 2, ..., переменная k: тип;

.....

переменная 1, переменная 2, ..., переменная s: тип;

тип – одно из ключевых слов: integer, real, boolean, char, string

Пример раздела описаний:

```
var a,b,c,x,y: real;  
    i,j,k,m,n: integer;  
    FLAG: boolean;  
    symbol: char;
```

### Выражения. Порядок выполнения операций.

Совокупность переменных и констант, соединенных знаками операций и скобками, называется выражением.

Пример.

$$(a+b)/c$$
$$((n+q1)*dx+(i+j)*dy)/(x1-2)*(y1-2)$$

Правила выполнения операций в Паскале:

1. Умножение и деление выполняются раньше, чем сложение и вычитание. Говорят также, что умножение и деление имеют более высокий приоритет, чем сложение и вычитание.

2. Операции, имеющие одинаковый приоритет выполняются слева направо.

Умножение и деление имеют одинаковый приоритет, сложение и вычитание имеют также одинаковый приоритет.

Исходя из этих правил выражение

$$4/8*5$$

будет вычисляться следующим образом:

Сначала будет вычислено  $4/8 (=0.5)$ , а затем результат будет умножен на 5. Получится  $0.5*5=2.5$

Всякое отклонение от этих правил должно регламентироваться скобками, т.к. действия над переменными стоящими в скобках выполняются в первую очередь.

### Константы

В Паскале есть возможность присвоить константе имя, при этом в последующем тексте программы всюду вместо этой константы можно использовать её имя. Все определения констант перечисляются в специальном разделе – разделе описания констант, имеющем вид:

```
const
    имя 1=значение 1;
    имя 2=значение 2;
    .....
    имя n=значение n;
```

#### Пример.

```
const
    r=1.87E+5;
    g=981E-2;
    atmosph=760;
    pi=3.14159;
```

Давая имена константам, мы делаем программу более понятной. Запись  $2*pi*r$  гораздо понятнее и информативнее, нежели запись  $2*3.14159*1.87E+5$

Кроме того, при внесении изменений в программу нам будет достаточно изменить только значение константы.

### Основные операторы Паскаля

Основная часть программы на Паскале – раздел операторов. Он начинается ключевым словом `begin` и заканчивается ключевым словом `end`, за которым следует точка. Рассмотрим основные операторы:

#### Оператор присваивания

Элементарное действие над переменной – изменение её значения. Для этого применяется оператор присваивания, имеющий вид:

имя переменной:= выражение;

В нем переменная и выражение должны быть одного типа.

Пример.

Пусть  $x$  – переменная целого типа.

$x := x + 1;$

В левой части оператора  $x$  обозначает переменную, а в правой части – число, являющаяся её текущим значением. Выполнение этого оператора приводит к увеличению значения переменной  $x$  на единицу.

### Операторы ввода/вывода

Во время исполнения программы она обменивается информацией с "внешним миром". Например, она может выдавать информацию на экран или получать информацию с клавиатуры. Для этого используются операторы ввода и вывода.

Оператор вывода имеет вид:

`write(выражение);`    или

`writeln(выражение);`

В результате выполнения этого оператора значение соответствующего выражения будет выведено на экран. Выражение может быть любым из указанных выше типов.

Пример.

`write(2+2);`    будет выведено на экран 4

`write(x=y);`    будет выведено true или false в зависимости от значений  $x$ ,  $y$

Оператор `writeln` отличается от оператора `write` тем, что выведет значение выражения с начала новой строки, а оператор `write` с той позиции строки, где находится курсор.

В операторе вывода можно указывать несколько выражений, разделяя их запятыми, а также любой текст, заключенный в одинарные кавычки.

Пусть значение  $A=5$ . Тогда при выполнении оператора

`writeln('Значение A=', A);`

будет выведено на экран с новой строки

Значение  $A=5$

Оператор ввода имеет вид:

`read(имя переменной 1, имя переменной 2,..., имя переменной n);`

`readln(имя переменной 1, имя переменной 2,..., имя переменной n);`

В результате его выполнения переменной (переменным) присваивается счи-

танное с клавиатуры значение. Вводимое значение должно записываться при вводе так, как описана переменная, т.е. если например, переменная целого типа, значит вводимое число должно быть целым. При этом, если используется `read`, то значение вводится в то место на экране, где в данный момент находится курсор, если же используется `readln`, то с новой строки и с первой колонки экрана.

Пример программы с использованием операторов ввода/вывода.

```
Program summa;
var result, A, B: integer;
begin
read(A,B);
result:=A+B;
writeln('Сейчас появится результат');
writeln('1-е введенное число=',A);
writeln('2-е введенное число=',B);
writeln('Сумма двух чисел=',result);
end.
```

### **Операции с целыми числами**

До сих пор мы рассматривали лишь одну операцию с целыми числами – сложение. Естественно в Паскале разрешены и другие операции. Рассмотрим программу, показывающую все возможные операции с целыми числами:

Пример.

```
Program operation;
var result, A, B, C: integer;
begin
  read(A,B);
  writeln('A=',A,'B=',B);
  C:=A+B;
  writeln('Демонстрация сложения, C=',C);
  C:=A*B;
  writeln('Демонстрация умножения, C=',C);
  C:=A div B;
  writeln('Демонстрация деления нацело, C=',C);
  C:=A mod B;
  writeln('Остаток от деления, C=',C);
  C:=A-B;
  writeln('Демонстрация вычитания, C=',C);
end.
```

Из этой программы видно, что знак умножения \* (а не x). Тем самым избегают возможности спутать его с буквой x.

Далее с целыми числами определены операции:

`div` (от английского `divide` – делить) – деление нацело.

Пусть  $A=17$ ,  $B=3$ , отсюда  $17:3=5*3+2$  и  $A \text{ div } B$  дает результат 5

`mod` (от английского `modulo` – определить остаток) – определение остатка от деления нацело.

$A \text{ mod } B$  дает результат 2

### Стандартные функции с целыми переменными (аргументами)

Рассмотрим программу:

```
Program func;  
var a, b, c: integer;  
begin  
  a:=-2;  
  b:=abs(a);  
  writeln('abs(-2)=' ,b);  
  c:=sqr(b);  
  writeln('sqr(b)=' ,c);  
  c:=sqr(b+b);  
  writeln('sqr(b+b)=' ,c);  
  end.
```

Оператор `b:=abs(a);` присваивает переменной `b` абсолютное значение числа `a`.

Под абсолютным значением числа понимается значение этого числа, если отбросить знак.

`abs(-2)=2`

`abs(-10)=10`

`abs(5)=5`

Оператор `c:=sqr(b)`- присваивает переменной `c` квадрат числа `b`, т.е.  $c = b^2$ , `sqr` (от английского `square` – квадрат). Число в скобках называется аргументом функции. В качестве аргумента может быть выражение, например,

$\sqrt{b^2 - 4ac}$  запишется на Паскале следующим образом:

`sqr(sqr(b) - 4 * a * c);`

### Операции с вещественными числами (тип `Real`).

С вещественными числами можно выполнять различные операции. Все возможные операции иллюстрируются следующей программой:

```
Program Real_Ex;  
var a, b, c: real;  
begin  
  a:=17.3;  
  b:=3.4;
```

```

c:=a*b;
writeln('a*b=',c);
c:=a/b;
writeln('a/b=',c);
c:=a+b;
writeln('a+b=',c);
c:=a-b;
writeln('a-b=',c);
end.

```

### Одновременное использование вещественных и целых чисел.

В программе могут встречаться переменные разных типов:

```

Program Int_Real;
var n, k: integer;
    a, b: real;
begin
  a:=3.6;
  n:=4;
  b:=n;
  writeln('b=', b);
  n:=trunc(a);
  writeln('trunc(3.6)=' ,n);
  k:=round(a);
  writeln('round(3.6)=' ,k);
end.

```

В программе мы видим запись `b:=n;` где вещественной переменной `b` присваивается значение целой переменной `n`. Кроме того, в таких записях как `b:=n+4.6;` или `b:=3*7.2+n;` встречаются вещественные и целые числа, стоящие в правой части выражения. Такие записи разрешены. И наоборот, присвоение вещественных значений целым переменным просто запрещены. Для этого используются стандартные функции `trunc` и `round`. С помощью функции `trunc` производится преобразование вещественного числа в целое путем отбрасывания всех цифр, стоящих после десятичной точки (`truncate` – усекать), `round` – позволяет преобразовать вещественное число в целое путем округления (`round` – округлять).

## Другие стандартные функции с вещественными аргументами

Функция	Запись на Паскале
$x^2$	<code>sqr(x)</code>
$\sqrt{x}$	<code>sqrt(x)</code>
$\sin x$	<code>sin(x)</code>
$\cos x$	<code>cos(x)</code>
$arctg x$	<code>arctan(x)</code>
$\ln x$	<code>ln(x)</code>
$e^x$	<code>exp(x)</code>

Еще раз напоминаю, что аргументом функции (т.е. то, что стоит в скобках) может быть выражение. Например,  
`s:=sin(a+b*2/5.2-sqr(a));`

## Булевы переменные и условные операторы

Булевы переменные (логические переменные) – это переменные имеющие только два значения `false` и `true`. Над булевыми переменными определены логические операции `not`, `and`, `or`.

Кроме того, определены следующие операции отношения:

`=` равно,

`<` меньше,

`<=` меньше или равно,

`>=` больше или равно

Результатом этих операций отношения являются логические `false` или `true`.

Рассмотрим следующее выражение:

`x>3`

В зависимости от значения `x` это выражение будет либо истинным (`true`), либо ложным (`false`).

Пример.

```
PROGRAM LOGICA;  
VAR X: INTEGER;  
    BOL: BOOLEAN;  
BEGIN  
    X:=4;  
    BOL:=X>3;  
    WRITELN('BOL=',BOL);  
    BOL:=X<3;
```

```
WRITELN( 'BOL=' ,BOL) ;  
END .
```

Допускается размещать справа и слева от знаков отношений арифметические выражения:  $X+6.5 < X+5 \rightarrow$  такие выражения называются логическими или булевыми выражениями.

Рассмотрим программу где используются логические функции:

```
PROGRAM LOG1 ;  
VAR X: INTEGER ;  
    L1, L2, RESULT: BOOLEAN ;  
BEGIN  
    X:=4 ;  
    L1:=X>3 ;  
    L2:=X<3 ;  
    WRITELN ( 'БУЛЕВАЯ ПЕР.L1=' ,L1 ) ;  
    WRITELN ( 'БУЛЕВАЯ ПЕР.L2=' ,L2 ) ;  
    RESULT: =L1 AND L2 ;  
    WRITELN('L1 AND L2 ПАВНО' ,RESULT) ;  
    RESULT : =L1 OR L2 ;  
    WRITELN('L1 OR L2 ПАВНО' ,RESULT) ;  
    RESULT: = NOT RESULT ;  
    WRITELN('NOT RESULT ПАВНО' ,RESULT) ;  
END .
```

### **Условные операторы.**

Условные операторы –это такие операторы , с помощью которых можно изменять последовательность выполнения операторов программы.

### **Оператор IF ....THEN**

Этот оператор имеет вид:

```
IF условие THEN оператор;
```

где оператор – любой оператор Паскаля;  
условие –логическое выражение.

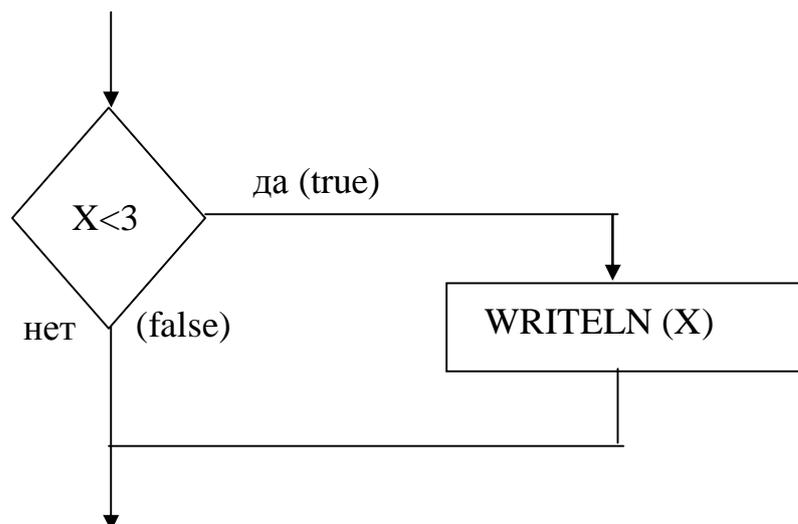
Если это условие выполняется (т.е. это условие дает значение TRUE), то будет выполнен оператор стоящий после слова THEN.

Рассмотрим пример:

```
IF X<3 THEN WRITELN (X) ;
```

Здесь записано : "Если  $X < 3$  , то вывести на экран значение X".

Оператор IF ... THEN можно представить в виде структурной схемы , с помощью которой показывается ход выполнения программы:



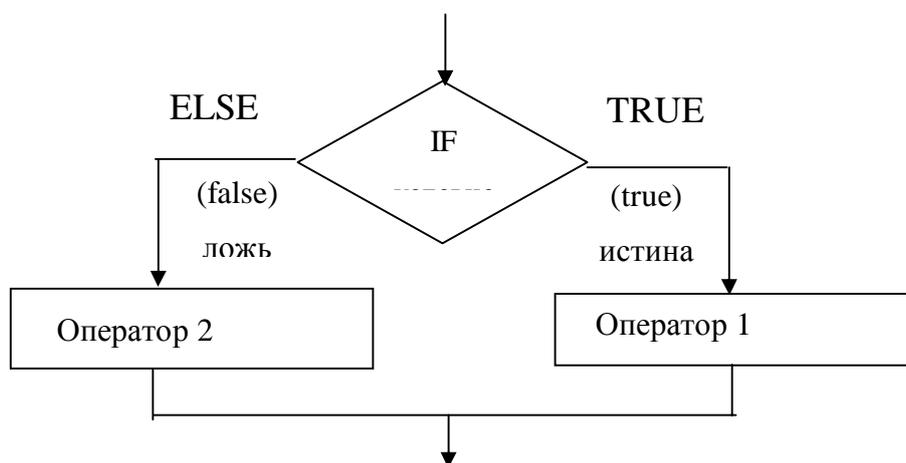
### Оператор IF ...THEN.....ELSE

Этот оператор записывается следующим образом:

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

В этом операторе будет выполняться либо оператор 1, либо оператор 2, что иллюстрирует структурная схема.

Обратите внимание, что перед ELSE ; не ставится



Из этой структурной схемы видно, что выбирается либо один, либо другой вариант продолжения программы.

Например:

```
IF X < 2 THEN X1 := 0
```

Пример. Вычислить:

$$y = \begin{cases} x, & x > 0 \\ 0, & x = 0 \\ x^2, & x < 0 \end{cases}$$

```
PROGRAM EX7; {Вариант 1}
VAR X: INTEGER;
BEGIN
  WRITELN( ' Введите значение X' )
  READLN( X );
  IF X>0 THEN Y:=X;
  IF X=0 THEN Y:=0;
  IF X<0 THEN Y:=SQR( X )
  WRITELN( ' X=' , X , ' Y=' , Y );
  READLN;
```

END.

```
PROGRAM EX8; {Вариант 2}
VAR X: INTEGER;
BEGIN
  WRITELN( ' Введите значение X' )
  READLN( X );
  IF X>0 THEN Y:=X
  ELSE IF X=0 THEN Y:=0
  ELSE Y:=SQR( X );
  WRITELN( X=' , X , ' Y=' , Y );
  READLN;
```

END;

Часто бывает необходимо, чтобы выполнялись или не выполнялись несколько операторов. Для этого эти операторы объединяются в блок, в котором перед первым оператором ставится слово BEGIN, а после последнего слово END.

Все операторы между BEGIN и END образуют так называемый составной оператор, а BEGIN и END как бы выполняют роль скобок. Их часто так и называют – операторные скобки.

Пример:

```
PROGRAM DEMO_IF;
VAR X: INTEGER;
BEGIN
  READ( X );
  IF X<2 THEN
    BEGIN
      WRITELN( ' ВЫПОЛНЯЕТСЯ ВЕТКА ПРОГРАММЫ ПО УСЛОВИЮ TRUE' );
```

```

WRITELN('X = ', X);
END;
ELSE WRITELN('X = ', X) ;
END.

```

Иллюстрация в общем виде :

```

BEGIN
S1;
S2;
S3;
S4;
S5;
END.

```

где S1, S2, S3, S4, S5- операторы Паскаля , в том числе и составные операторы, т. е. составные операторы могут быть вложены друг в друга .Схематично это выглядит так:

```

      BEGIN
      .....
        BEGIN
          .....
          ..... .
          .....
          .....
        END;
      END;

      BEGIN
      S1;
      S2;
      S3;
      BEGIN
      P1;
      P2;
      P3;
      END;
      S5;
      END;

```

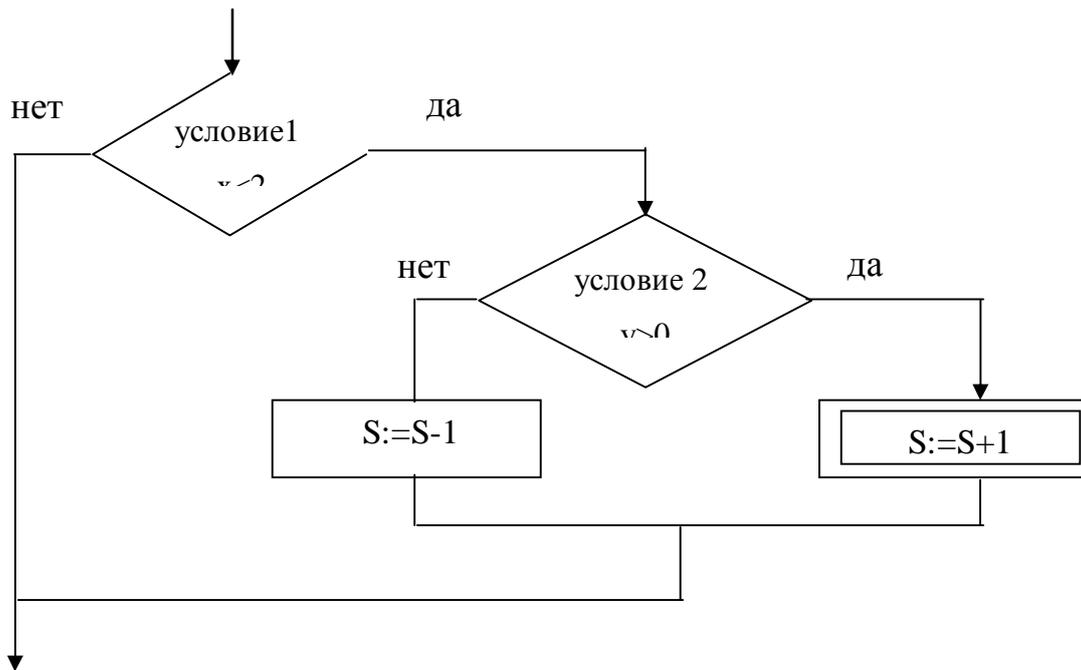
Здесь S4 составной оператор.

Условные операторы также могут быть вложены друг в друга.

```

IF X<2 THEN
IF Y>0 THEN S:=S+1
ELSE S:=S-1;

```



### Операторы цикла

Операторы цикла используются для организации многократного повторения выполнения одних и тех же операторов. В языке Паскаль существует три типа операторов цикла.

#### Оператор цикла с предусловием .

Этот оператор имеет вид:

```

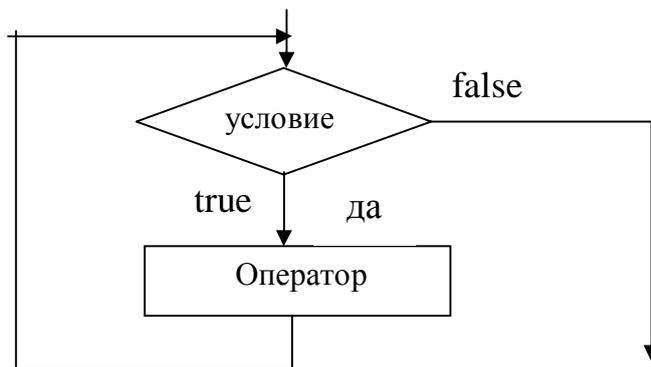
while  условие
do    оператор;

```

условие – булевское выражение ;  
 оператор – любой оператор Паскаля, в частности может быть и составным оператором. Слова `while` и `do` являются служебными словами. Выполняется этот оператор следующим образом: сначала вычисляется значение булевского выражения. Если это значение есть `true`, то выполняется оператор после слова `do` и снова происходит возврат к вычислению булевского выражения. Так повторяется пока булевское выражение имеет значение `true`. Как только значение булевского выражения станет `false`, то происходит выход из цикла, т. е. оператор после служебного слова `do` уже не выполняется, а будет выполняться следующий после оператора цикла оператор.

В данном операторе вычисление выражения происходит раньше, чем будет выполняться оператор после `do`, поэтому он и называется оператор цикла с предусловием. Может так случиться, что оператор после `do` не будет выполнен вообще, если значение условия с первого раза будет `false`.

## Структурная схема цикла с предусловием



## Оператор цикла с постусловием

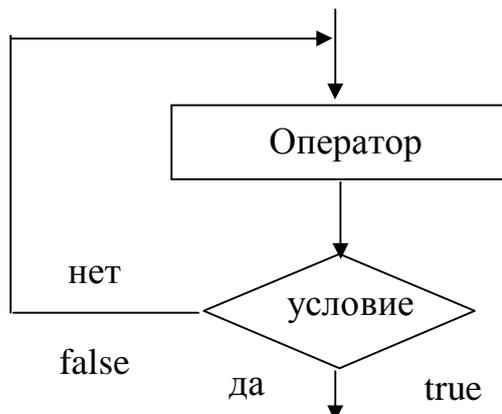
Этот оператор имеет вид:

```
repeat оператор  
until условие;
```

где оператор – любой оператор Паскаля, в том числе и составной.  
условие – булевское выражение.  
repeat и until-служебные слова.

Этот оператор выполняется следующим образом: сначала выполняется оператор следующий за служебным словом repeat, затем вычисляется значение булевского выражения (условия). Если значение условия false, то происходит возврат к выполнению оператора и после этого снова вычисляется значение булевского выражения. Так повторяется до тех пор, пока значение булевского выражения false. Как только условие станет true, выполнение оператора цикла прекращается.

Структурная схема оператора



В отличие от оператора цикла while- do здесь оператор будет вы-

полнен хотя бы один раз , независимо от значения условий.

**Предупреждение!** Чтобы рассмотренные выше операторы цикла выполнялись конечное число раз, при построении цикла необходимо предусмотреть, чтобы среди выполняемых операторов обязательно был оператор, который изменял бы значение условия, таким образом, чтобы когда-нибудь значение условия принимало бы false (для оператора while-do) или true (для оператора repeat-until).

В противном случае цикл будет повторяться бесконечное число раз и компьютер "зациклится". Ответственность за правильное применение этих операторов цикла несет на себе программист!

Пример: Вычислить  $S = \sum_{X=1}^{100} X$

```
PROGRAM SUM1; {Вариант 1}
VAR X, SUM : INTEGER;
BEGIN
  SUM:=0;
  X:=1;
  WHILE X<=100 DO
  BEGIN
    SUM:=SUM+X;
    X:=X+1;
  END;
  WRITELN( 'SUM=' ,SUM);
  READLN;
END.
```

```
PROGRAM SUM2; {Вариант 2}
VAR X, SUM : INTEGER;
BEGIN
  SUM:=0;
  X:=1;
  REPEAT
    SUM:=SUM+X;
    X:=X+1;
  UNTIL X>100
  WRITELN( 'SUM=' ,SUM);
  READLN;
END.
```

Пример: Вычислить функцию:

$$y = \begin{cases} x^2 + 1, & x > 0 \\ 0, & x = 0 \\ x^2 - 1, & x < 0 \end{cases} \quad \text{для } x \in [-10, 10] \text{ с шагом } 1$$

```

PROGRAM EX9; {Вариант 1}
VAR X,Y: INTEGER;
BEGIN
  X:=-10;
  WHILE X<=10 DO
  BEGIN
    IF X>0 THEN Y:=SQR(X)+1;
    IF X=0 THEN Y:=0;
    IF X<0 THEN Y:=SQR(X)-1;
    WRITELN('X=',X,' Y=',Y);
    X:=X+1;
  END;
READLN;
END.

```

```

PROGRAM EX10; {Вариант 2}
VAR X,Y: INTEGER;
BEGIN
  X:=-10;
  WHILE X<=10 DO
  BEGIN
    IF X>0 THEN Y:=SQR(X)+1;
    ELSE IF X=0 THEN Y:=0
    ELSE Y:=SQR(X)-1;
    WRITELN('X=',X,' Y=',Y);
    X:=X+1;
  END;
  READLN;
END.

```

В программах EX9 и EX10, как вы могли заметить, использовались составные операторы.

```

PROGRAM EX11; {Вариант 3 с использованием оператора цикла с постусловием}
VAR X,Y: INTEGER;
BEGIN
  X:=-10;
  REPEAT
  IF X>0 THEN Y:=SQR(X)+1;

```

```

IF X=0 THEN Y:=0;
IF X<0 THEN Y:=SQR(X)-1;
    WRITELN(X=' ,X, '    Y=' ,Y);
    X:=X+1;
    UNTIL X>10;
READLN;

```

END.

Заметим, что в операторе цикла с постусловием, если после слова REPEAT используется не один, а несколько операторов, то не обязательно использовать операторные скобки BEGIN и END, поскольку служебные слова REPEAT и UNTIL сами выполняют роль операторных скобок.

### **Оператор цикла с параметром.**

Иногда заранее точно известно, сколько раз должно быть выполнено определенное действие. Для задач такого типа в языке Паскаль, имеется оператор цикла с параметром. Этот оператор имеет вид:

```
FOR переменная := выражение 1 TO выражение 2 DO оператор;
```

где FOR, TO, DO – служебные слова; переменная- переменная целого типа, называемая индексом или параметром цикла.

Выражение 1, выражение 2 – арифметические выражения целого типа, т.е. значения выражений должны быть целыми;

оператор – простой или составной оператор.

Для того, чтобы оператор цикла выполнялся хотя бы один раз, значение выражения 1 должно быть меньше или равно значению выражения 2 (на практике значения выражения 1 всегда меньше значения выражения 2). Оператор работает следующим образом: вначале переменной (параметру цикла) присваивается значение выражения 1, затем сравнивается значение параметра цикла и значение выражения 2. Если параметр цикла меньше значения выражения 2, то выполняется оператор после слова do. Затем параметр цикла увеличивается на 1, после этого вновь сравнивается значение параметра цикла и выражение 2, если параметр цикла меньше, то вновь выполняется оператор после слова do. И так продолжается до тех пор, пока параметр цикла не станет больше выражения 2. Как только это происходит, оператор цикла заканчивается.

Пример:

```

PROGRAM EX12;
VAR X,Y:INTEGER;
BEGIN
    FOR X:=-10 TO 10 DO
        BEGIN
            IF X>0 THEN Y:=SQR(X)+1

```

```

        ELSE IF X=0 THEN Y:=0
        ELSE Y:=SQR(X)-1;
        WRITELN('X=' ,X, 'Y=' ,Y);
    END;
    READLN;
END.

```

### Второй вариант оператора цикла с параметром

Оператор цикла с параметром может быть записан и в таком виде:

FOR переменная:= выражение 1 DOWNTO выражение 2 DO оператор;

В этом варианте параметр цикла (переменная) после каждого повторения не увеличивается, а уменьшается на 1. Значение выражения 1 больше или равно (на практике всегда больше) значения выражения 2.

Оператор цикла заканчивается как только параметр цикла станет меньше выражения 2.

Пример:

```

PROGRAM EX13;
VAR X,Y:INTEGER;
BEGIN
    FOR X:=10 DOWNTO -10 DO
        BEGIN
            IF X>0 THEN Y:=SQR(X)+1
            ELSE IF X=0 THEN Y:=0
            ELSE Y:=SQR(X)-1;
            WRITELN(X=' ,X, ' Y=' ,Y);
        END;
        READLN;
    END.

```

Пример: Вычислить  $S = \sum_{x=1}^{10} x$ ,

```

PROGRAM EX14; {Вариант 1}
VAR X,Y:INTEGER;
BEGIN
    S:=0;
    FOR X:=1 TO 10 DO S:=S+X;
    WRITELN('X=' ,X, 'Y=' ,Y);
    READLN;
END.

```

```

PROGRAM EX15; {Вариант 2}
VAR X,Y:INTEGER;

```

```

BEGIN
  X:=1;S:=0;
  WHILE X<=10 DO
  BEGIN
    S:=S+X;
    X:=X+1;
  END;
  WRITELN('X=',X,'Y=',Y);
  READLN;
END.

```

Вычислить  $S = \frac{1}{x} + \frac{1}{x^2} + \frac{1}{x^3} + \dots + \frac{1}{x^n} + \dots$ ,  
 $x \in [1.5, 2]$  с шагом  $h=0.5$ , точность  $\varepsilon = 10^{-5}$

```

PROGRAM EX16;
VAR X,S,H,EPS:REAL;
T:REAL;
BEGIN
  X:=1.5; H:=0.5;
  EPS:=0.00001;
  WHILE X<=2 DO
  BEGIN
    S:=0;T:=1;
    REPEAT
      T:=T*1/X;
      S:=S+T;
    UNTIL T<=EPS;
    WRITELN('X=',X,'S=',S);
    X:=X+H;
  END;
  READLN;
END.

```

Вычислить  $\sin x$  используя его разложение в ряд Тейлора:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

$x \in [0, 1]$ ,  $h=0.2$ , точность  $\varepsilon = 10^{-3}$

Вывести также "точное" значение, вычисленное с помощью функции SIN(X)

```

PROGRAM EX20;
VAR X,Y,S,H,EPS,T:REAL;
N:INTEGER;
BEGIN
  X:=0; H:=0.2; EPS:=0.001;
  WHILE X<=1 DO

```

```

BEGIN
  S:=X; T:=X; N:=2;
  REPEAT
    T:=-T*(SQR(X)/N*(N+1));
    S:=S+T;
    N:=N+2;
  UNTIL ABS(T)<EPS;
  Y:=SIN(X);
  WRITELN('X=',X,' Приближенное значение синуса=',S,
' Точное значение синуса=',Y);
  X:=X+H;
END;
  READLN;
END.

```

### Символьные и строковые типы данных.

Список символов, т.е. список цифр, букв и других знаков содержит таблица ASCII. Номер, который символ имеет в таблице ASCII, называется кодом этого символа. Символ можно представить указав его в кавычках, а можно использовать значок #, за которым следует код символа. Например, буква 'A', в таблице ASCII имеет номер 65, т.е. его код равен 65, тогда можно указать # 65 и это будет означать 'A'. Для обозначения типа "символ" в Паскале используется зарезервированное слово CHAR. Для хранения переменной типа "символ" требуется один байт памяти.

Строкой называется, некоторая последовательность символов: 'AABB', 'ВАСЯ-ЭТО КОТ'.

Строка может состоять из одного символа, а может быть совсем пустой, т.е. состоять из пробелов. Максимальная длина строковой переменной – 255 байтов. Тип строковой переменной указывается зарезервированным словом string. При этом память для строковой переменной резервируется так, как если бы она содержала строку максимальной длины. Но если заранее точно известна длина строки, например 30 символов, то можно указать string [30].

Пример описания строк и символов.

```

VAR SYMBOL:CHAR;
    NAME: STRING[10];
    MESSAGE: STRING;

```

Переменная SYMBOL переменная типа символ (CHAR) – для него отводится 1 байт памяти.

Переменная NAME – типа STRING (строковая переменная), её отводится 10 байт памяти.

MESSAGE – также строковая переменная, её отводится 255 байт памяти, по-

сколько для нее не указана длина.

Какие операции допустимы для строк и символов? Только операция (сложения). При этом в результате получается строка.

Пример:

```
PROGRAM EX21;
VAR SYMBOL:CHAR;
    ANY,BODY: STRING;
BEGIN
    SYMBOL:='A';
    BODY:= 'Ш';
    ANY:='МАМА';
    BODY:=BODY+SYMBOL;
    {теперь BODY='ША'}
    ANY:=ANY+BODY;
    {ANY='МАМАША'}
    WRITELN(ANY);
    BODY:='+ПАПАША=ПРЕДКИ';
    ANY:=ANY+BODY;
    WRITELN(ANY);
    READLN;
END.
```

Пример:

```
PROGRAM EX22;
VAR ANSW,PASSW: STRING[4];
BEGIN
    PASSW:='BOSS';
    REPEAT
    WRITELN('Введите пароль');
    READLN(ANSW);
    IF PASSW<> ANSW THEN
    WRITELN('Вы не пользователь');
    ELSE
    WRITELN('Здравствуйте, мой друг!');
    UNTIL ANSW = PASSW;
    READLN;
END.
```

### **Строковые процедуры и функции.**

Функция Concat- выполняет конкатенацию последовательности строк, т. е. объединения нескольких строк.

Описание: `Concat(S1[,S2,...,Sn]:string):string;`

Имя функции `Concat`

`S1,S2,...,Sn` – формальные параметры, имеют тип `string`. Квадратные скобки указывают, что параметры `S2,...,SN` могут отсутствовать. Результат функции строка символов, т.е. имеет тип также `string`.

Пример:

```
PROGRAM EX23;  
VAR A,C: STRING;  
BEGIN  
  A:='КОЛЯ+ ОЛЯ';  
  C:='РАВНЯЕТСЯ ЛЮБОВЬ';  
  WRITELN(CONCAT(A,C)); READLN;  
END.
```

Функция `COPY` –возвращает подстроку строки.

Описание:

`COPY(S:STRING, INDEX, COUNT: INTEGER):STRING;`

`S`– исходная строка,

`INDEX`– номер символа, начиная с которого выделяется подстрока из `S`,

`COUNT`– количество символов в подстроке.

Пример:

```
PROGRAM EX24;  
VAR A,B: STRING;  
BEGIN  
  A:='head&shoulders';  
  B:=COPY(A,6,9);  
  WRITELN(B);READLN;  
END.
```

Процедура `DELETE`– удаляет из строки подстроку

Описание :

`DELETE(S:STRING, INDEX: INTEGER, COUNT: INTEGER);`

`S` – исходная строка, `INDEX` – номер символа, начиная с которого удаляется подстрока из `S`, `COUNT`– количество символов в подстроке.

Пример:

```
PROGRAM EX25;  
VAR A: STRING;  
BEGIN
```

```

A := '123XYZ' ;
DELETE(A, 4, 3) ;
WRITELN(A) ; READLN ;
END .

```

Процедура INSERT – добавляет подстроку в строку .

Описание:

```

INSERT(S1 : STRING, S2 : STRING, INDEX : INTEGER) ;

```

S2- исходная строка

S1- добавляемая строка

INDEX- номер символа строки S2, начиная с которого добавляется подстрока S1 .

Пример:

```

PROGRAM EX26 ;
VAR S : STRING ;
BEGIN
  S := '123678' ;
  INSERT('45', S, 4) ;
  WRITELN(S) ;
  READLN ;
END .

```

Функция LENGTH – возвращает динамическую длину строки.

Описание :

```

LENGTH(S : STRING) : INTEGER ;

```

Пример:

```

PROGRAM EX27 ;
VAR S : STRING ;
    I : INTEGER ;
BEGIN
  S := 'ABCDE' ;
  I := LENGTH(S) ;
  WRITELN(S, 'ДЛИНА СТРОКИ=' , I) ;
  S := '' ; I := LENGTH(S) ;
  WRITELN(S, 'ТЕПЕРЬ ДЛИНА СТРОКИ =' , I) ;
  READLN ;
END .

```

Функция POS- производит поиск подстроки в строке.

Описание :

POS(Substr, S: ISTRING): BYTE

Substr –подстрока, поиск которой производится в строке S. Функция POS возвращает номер символа, начиная с которого подстрока Substr найдена в S. Если такой подстроки нет, то возвращается 0.

Пример:

```
PROGRAM EX28;
VAR S: STRING;
    I: INTEGER;
BEGIN
    S:='МАМАША';
    I:=POS('ША',S);
    WRITELN(S, 'Номер символа, с которого начинается подстрока ША=', I);
READLN;
END.
```

Пример: Дана строка символов длиной 30. Определить, входит ли в эту строку подстрока 'ША'. Строку ввести с клавиатуры.

```
PROGRAM EX29;
VAR S: STRING[30]; Substr:STRING[2];
    I: INTEGER;
BEGIN
    WRITELN('Введите строку символов');
    READLN(S); Substr:='ША';
    I:=POS(Substr,S);
    IF I=0 THEN WRITELN('В данной строке', S,'подстрока',
    Substr,'отсутствует')
    ELSE
    WRITELN('подстрока',Substr,'входит в данную строку начиная с но
    READLN;
END.
```

Процедура Str – преобразует численное значение в его строковое представление.

Описание :

Srt(X[:width[:decimals]],s:string);

X- числовая переменная (типа integer или real)

width- длина строки S;decimals- количество знаков после запятой.

Пример:

```
PROGRAM EX30;
```

```

VAR S: STRING; X:REAL;
  BEGIN
X:=123.565;
str(X:6:2,S);
{теперь S='123.56'}
      WRITELN(S);
READLN;
END.

```

Процедура Val преобразует строковое значение в его численное представление.

Описание :

```
VAL(S:string,V,code:integer);
```

S- исходная строка символов

V- числовая переменная – результат преобразования S

code- целая переменная

Если невозможно преобразовать S в число, то после выполнения процедуры Val code≠0, если преобразование произошло удачно, то code=0.

Пример:

```

PROGRAM EX31;
VAR S: STRING;
    X:REAL;
    C :INTEGER;
BEGIN
  S:='234.12';
  VAL(S,X,C);
  WRITELN(X);
  READLN;
END.

```

### **Массивы.**

В некоторых случаях приходится сталкиваться с ситуацией, когда должно использоваться относительно много переменных одного типа.

Представим себе программу, которая позволяет вычислить определенные метеорологические данные и в которой всегда будет не меньше 365 переменных. Таким образом можно присваивать отдельным переменным метеоданные по отдельным дням всего года.

В принципе сейчас нет никакой проблемы, чтобы объявить тип REAL у 365 переменных, используя запись.

```
VAR DAY1, DAY2, DAY3 и т.д. DAY365:REAL;
```

Однако в Паскале нельзя писать и "т.д.". Нужно перечислить все перемен-

ные. Для этого нужна по крайней мере страница. Если теперь нужно подсчитать среднее арифметическое на один день, то нужно записать:

```
SRED := (DAY1+DAY2+и т.д. DAY365)
```

и опять как быть с "и т.д."?

Для таких случаев Паскаль предоставляет возможность введения большого числа переменных одного и того же типа, используя простые выражения.

Эта возможность в Паскале реализуется с помощью так называемых массивов. Используется служебное слово ARRAY, которая указывает на ряд переменных одного и того же типа и имеющих одно имя.

Например :

```
VAR DAY:ARRAY[1..365] OF REAL;
```

С помощью этого описания определяется массив DAY состоящий из 365 элементов (переменных), т.е. каждый элемент массива может использоваться как отдельная переменная. Можно записать:

```
DAY[1]:=1.25;
```

```
DAY[2]:=0.34;
```

Другими словами, каждый элемент массива определяется именем массива и номером элемента в квадратных скобках. Говорят еще индекс массива. Причем в качестве индекса можно использовать арифметическое выражение целого типа.

В описании массива после имени в квадратных скобках указывается минимальное значение индекса. В нашем случае массив DAY состоит из 365 элементов, нумерация идет от 1 до 365 т.е. шаг нумерации по умолчанию всегда равен 1.

При использовании массивов запись программы становится намного короче и наглядней.

Пусть например, в начале все элементы массива DAY должны быть приравнены 0.

Вместо того, чтобы писать

```
DAY[1]:=0;
```

```
DAY[2]:=0;
```

и т.д.

```
DAY[365]:=0;
```

мы можем поступить следующим образом:

```
VAR DAY:ARRAY[1..365] OF REAL;
```

```
K : INTEGER;
```

```
BEGIN
```

```
FOR K:=1 TO 365 DO
```

```
    DAY [K]:=0;
```

Пример. Пусть имеются 2 массива типа STRING. В одном массиве содержатся фамилии людей, а в другом номера их домашних телефонов. Написать программу, которая в цикле по номеру телефона выводит на экран фамилию

этого абонента.

```
PROGRAM EX32;
VAR NAME:ARRAY[1..50]OF STRING[30];
    TEL:ARRAY[1..50]OF STRING[7];
                                K : INTEGER;PHONE:STRING[7];
BEGIN
    WRITELN('Введите фамилию и номер телефона ');
    FOR K:=1 TO 50 DO
    READLN(NAME[K],TEL[K]);
    PHONE:=' ';
    WHILE PHONE<>'*****' DO
    BEGIN
        WRITELN('Введите номер телефона ');
        READLN(PHONE);
        K:=1;
        WHILE K<= 50 DO
        BEGIN
            IF TEL[K]=PHONE THEN
                WRITELN('Фамилия этого абонента',NAME[K]);
                K:=K+1;
            END;
        END;
    READLN;
END.
```

Недостаток этой программы в том, что если указанный номер телефона в массиве не существует, то никаких сообщений на этот счет не выводится на экран. Кроме того, после того как найден абонент, например для k=2, программа ещё 48 раз прокрутит цикл. Как модифицировать программу?

```
PROGRAM EX33;
VAR NAME:ARRAY[1..50]OF STRING[30];
    TEL:ARRAY[1..50]OF STRING[7];
    K : INTEGER;PHONE:STRING[7];
    FLAG : BOOLEAN;
BEGIN
    WRITELN('Введите фамилию и номер телефона ');
    FOR K:=1 TO 50 DO
    READLN(NAME[K],TEL[K]);
    PHONE:=' ';
    WHILE PHONE<>'*****' DO
    BEGIN
        WRITELN('Введите номер телефона ');
        READLN(PHONE);
        K:=1;
```

```

WHILE K<= 50 DO
BEGIN
  IF TEL[K]=PHONE THEN
  BEGIN
    FLAG:=TRUE;K:=51;
    WRITELN( 'Фамилия этого абонента', NAME[K] );
  END
  ELSE BEGIN  FLAG:=FALSE;K:=K+1;END;
END;
IF FLAG:=FALSE AND PHONE<>'*****' THEN
WRITELN( 'Абонента с таким номером телефона нет в
        справочнике' );
  END;
READLN;
END.

```

### **Модули в Турбо Паскале.**

Модули в Турбо Паскале содержат ряд полезных процедур, функций и констант, которые дают возможность использовать почти всю мощь компьютеров. Существуют следующие модули в Турбо Паскале:

SYSTEM- содержит стандартные функции и процедуры "классического" Паскаля, например, вычисления функции COS, SIN, LN, EXP и др. Модуль SYSTEM автоматически доступен всем программам. Фактически мы широко пользовались этим модулем.

DOS- содержит процедуры и функции, позволяющие использовать средства операционной системы MS-DOS.

CRT – набор процедур для работы с экраном, клавиатурой.

GRAPH – содержит обширный набор программ, который позволяет использовать графические возможности компьютера.

Для использования модуля достаточно в программе после строки заголовка вставить строку.

USES имя модуля;

Если используется несколько модулей то пишется так :

USES имя модуля 1 , имя модуля 2, ....., имя модуля N;

Модуль CRT

Модуль CRT, как уже говорилось, содержит ряд процедур и функции для работы с экраном в текстовом режиме, клавиатурой и динамиком. Рассмотрим экран компьютера в текстовом режиме. Обычно на нем размещается 25 строк текста и 80 символов в каждой строке. Для того, чтобы полностью владеть экраном, нужно знать координаты курсора, менять цвет символов, текста и фона. Для этого существуют следующие процедуры:

- GOTOXY( I , J ) –позиционирует курсор в I строку и J столбец

экрана.

- WRITE(S) - выводит строку S начиная с текущей позиции курсора.

- процедура Textcolor(COLOR) - устанавливает цвет выводимого текста.

Существуют следующие константы цветов:

Black - черный;

Blue - синий;

Green - зеленый;

Cyan - бирюзовый;

Red - красный;

Magenta - малиновый;

Brown - коричневый;

LightGray - светло-серый;

DarkGray - темно-серый;

LightBlue - ярко-голубой;

LightGreen - ярко-зеленый;

LightCyan - ярко-бирюзовый;

LightRed - ярко-красный;

LightMagenta - ярко-малиновый;

Yellow - желтый;

White - белый;

Blink - мерцание (мигание);

Для вывода на экран мигающего текста надо использовать процедуру TEXTCOLOR(COLOR+BLINK).

Например

```
TEXTCOLOR(BLUE+BLINK);
```

```
WRITELN("Это мигающий текст ")
```

- Процедура TEXTBACKGROUND(COLOR) - устанавливает цвет фона.

- Процедура CLRSCR - очищает экран, одновременно окрашивая его в цвет установленного фона.

- Процедура WINDOW(X1, Y1, X2, Y2) - определяет на экране окно где X1, Y1 - координаты левого верхнего угла, а X2, Y2 - координаты правого нижнего угла окна

Функция KEYPRESSED - возвращает значение TRUE, если клавиша нажата и FALSE - в противном случае.

Функция READKEY - считывает символ с клавиатуры.

Пример: Написать программу, которая очищает экран и устанавливает синий фон. Затем рисует окно красного цвета и выводит текст белыми буквами.

```
PROGRAM EX33;
```

```

USES CRT;
BEGIN
  TEXTBACKGROUND (BLUE);
  CLRSCR; TEXTBACKGROUND (RED);
  WINDOW (5, 5, 15, 65);
  CLRSCR; GOTOXY (5, 6); TEXTCOLOR (WHITE);
  WRITE (' ПРИВЕТ , ВАСЯ! ');
  READLN;
END.

```

Пример. В некоей компании по продаже компьютеров имеется 20 менеджеров, фамилии которых, хранятся в массиве NAME. В другом массиве имеются данные о количестве проданных компьютеров каждым менеджером за месяц. Написать программу, которая выводит таблицу, в которой следующие графы: фамилия, количество проданных компьютеров, сумма выручки, сумма премии. Итоговая строка должна содержать общее количество проданных компанией компьютеров за месяц, общую сумму выручки, сумму премиальных. Стоимость компьютера принять равной 1000\$, размер премии за каждый проданный компьютер 25\$

```

PROGRAM EX34;
USES CRT;
VAR NAME: ARRAY[1..20] OF STRING[30];
    KOL: ARRAY[1..20] OF INTEGER;
    SUM, COST, PREM, I, K: INTEGER;
    PROGPREM, SUM1: INTEGER;
BEGIN
  WRITELN(' Введите фамилию и количество реализованных
  компьютеров ');
  FOR K:=1 TO 20 DO
    READLN(NAME[K], KOL[K]);
    TEXTBACKGROUND (BLUE); CLRSCR;
    TEXTCOLOR (WHITE); GOTOXY (0, 15);
    WRITE (' СВЕДЕНИЯ ');
    WRITE (' О РЕАЛИЗАЦИИ КОМПЬЮТЕРОВ ');
    TEXTBACKGROUND (RED); GOTOXY (1, 12);
    WRITE (' За ЯНВАРЬ 2002 г. ');
    TEXTBACKGROUND (GREEN);
    WINDOW (2, 10, 24, 70); TEXTCOLOR (YELLOW);
    GOTOXY (2, 10);
    WRITE (' ФАМИЛИЯ ', ' КОЛИЧЕСТВО ', ' СУММА ВЫРУЧКИ ', ' СУММА
    ПРЕМИИ ');
    TEXTCOLOR (BLACK); GOTOXY (3, 10);
    COST:=1000; PROGPREM:= 25;
    FOR K:=1 TO 20 DO

```

```

BEGIN GOTOXY(K+2,10);
  SUM:=COST*KOL[K];
  PREM:=PROGPREM* KOL[K];
  WRITE(NAME[K],KOL[K],SUM,PREM);
END;
SUM:=0;SUM1:=0;I:=0;
FOR K:=1 TO 20 DO
  I:=I+KOL[K];
  SUM:=I*COST; {СУММА ВЫРУЧКИ}
  PREM:=I*PROGPREM;
  GOTOXY(23,10);
  TEXTCOLOR(YELLOW);
  WRITE('ИТОГО');
  WRITE(I,SUM,PREM);
  READLN;
END.

```

### Создание и обработка последовательных файлов

Файл – некоторый участок на диске, где хранятся данные. Каждому файлу присваивается уникальное имя, благодаря которому становится возможным обращаться к различным данным на диске. Файлы состоят из отдельных записей – совокупности данных, имеющих свои имена и тип, объединенных одним именем. Запись – комбинированный тип данных. Каждая запись состоит из отдельных полей, которые могут иметь разные типы.

Общий вид объявления записи:

```

TYPE
  <имя записи>=RECORD
    <имя1>:<тип1>;
    <имя2>:<тип2>;
    .....
    <имяN>,<имяN+1>:тип;
  END;
  VAR <имя переменной>:<имя записи>;

```

Пример. Пусть имеется запись следующей структуры:

Фамилия	Имя	Отчество	Группа	Год рождения
---------	-----	----------	--------	--------------

```

TYPE
  STUD=RECORD
    family,name,otch,group:string[20];
    birthday:integer;
  END;
  VAR  A,B:STUD;

```

Обращение к элементам записи выполняется через уточненное (составное) имя.

Пример:

```
A.family:='Иванов'; B.group:='ПОВТАС-1/02';
```

Чтобы не удлинять слишком текст программы, в Паскале предусмотрен оператор присоединения WITH. Это дает возможность определить кусок программы, внутри которого можно просто указывать требуемые поля записи, не указывая каждый раз при этом имя самой записи.

Синтаксис оператора:

```
WITH <список имен записей> DO
begin
... {в этом блоке при обращении к полям записи не обязательно каждый
раз указывать имя записи}
...
end;
```

Пример:

```
TYPE
  STUD=RECORD
    family,name,otch,group:string[20];
END;
VAR  A,B:STUD;
begin
  with B do
    begin
      family:='Иванов';
      group:='ПОВТАС-1/02';
    end;
end.
```

### **Файловые типы**

В программе для работы с файлами вводятся файловые переменные. Переменные файловых типов необходимы тем программам, которым требуется читать данные с диска или записывать данные на диск. Для описания таких переменных используется зарезервированное слово FILE. Синтаксис описания файловой переменной:

```
VAR <имя файловой переменной>: FILE of <тип компонентов>;
```

Пример.

```
VAR NAME: FILE OF STRING[30];  
KOL: FILE OF INTEGER;
```

Файл NAME состоит из последовательности строк длиной 30 символов, файл KOL из целых чисел.

При работе с файлами используются следующие процедуры:

- Для связи файловой переменной с файлом во внешней памяти используется процедура:

```
assign(f, fname);
```

где *f* - файловая переменная, *fname* - переменная типа *string*, содержащая имя файла. Процедура ASSIGN – присваивает имя внешнего файла на диске файловой переменной.

Общий вид имени файла:

```
<диск>:\<имя каталога>\<имя подкаталога>\...\<имя файла>
```

Если опущены <диск>, <имя подкаталога>, то принимается текущий каталог и текущее логическое устройство. Только после выполнения ASSIGN можно обращаться к другим процедурам по обработке файлов.

```
ASSIGN(NAME, 'NAME.DAT')
```

```
ASSIGN(KOL, 'C:\TP\WORK\KOL.DAT')
```

- Процедура REWRITE – создает и открывает новый файл.  
Описание:

```
REWRITE(f)
```

*f* – является файловой переменной. Перед использованием REWRITE переменная *f* должна быть связана с дисковым файлом с помощью процедуры ASSIGN. Если файл с таким именем уже существует, то он удаляется, а на его месте создается новый пустой файл.

- Процедура RESET – открывает существующий файл для чтения или записи. Описание:

RESET(*f*) – где *f* – файловая переменная. Перед использованием RESET переменная *f* должна быть связана с файлом на диске с помощью процедуры ASSIGN.

- Процедура READ(*f*, *v*) – считывает в переменную *v* очередной элемент файла. Тип переменной *v* должен соответствовать файловой переменной *f*.

- Процедура WRITE(*f*, *v*) – записывает переменную *v* в файл. *f* – файловая переменная, *v* – переменная того же типа, что и элемент файла *f*.

- Процедура CLOSE(*f*) – закрывает открытый файл.

- Функция EOF в случае, если достигнут конец файла возвращает значение TRUE и FALSE, если конец файла не достигнут.

- rename(*f*, *newname*); - переименовать файл; *newname* - переменная типа *string*, содержащая новое имя файла;

- erase(F); - удалить файл.

Рассмотрим снова предыдущую программу. С этой программой, было не удобно работать, т.к. постоянная информация (фамилии, количество проданных компьютеров) не сохранялись и при каждом запуске программы приходилось заново их вводить.

Проблема решается, если постоянную информацию сохранить на диске. Напишем программу создания файлов на диске.

```
PROGRAM CREATE;  
USES CRT;  
VAR NAME: FILE OF STRING[30];  
KOL: FILE OF INTEGER;  
BEGIN  
  ASSIGN(NAME, 'NAME.DAT');  
  ASSIGN(KOL, 'KOL.DAT');  
  REWRITE(NAME); REWRITE(KOL);  
  WRITELN('ФАЙЛЫ НА ДИСКЕ СОЗДАНЫ');  
  READLN; CLOSE(NAME); CLOSE(KOL);  
END;
```

Напишем программу записи информации в созданные файлы.

```
PROGRAM INPUT;  
USES CRT;  
VAR NAME: FILE OF STRING[30];  
KOL: FILE OF INTEGER;  
COMP, K: INTEGER; FAM: STRING[30];  
BEGIN  
  ASSIGN(NAME, 'NAME.DAT');  
  ASSIGN(KOL, 'KOL.DAT');  
  RESET(NAME); RESET(KOL);  
  WRITELN('Введите фамилию и количество проданных компьютеров');  
  FOR K:=1 TO 20 DO  
    BEGIN  
      READLN(FAM, COMP);  
      WRITE(NAME, FAM);  
      WRITE(KOL, COMP);  
    END;  
  WRITELN('Информация на диск записана');  
  CLOSE(NAME); CLOSE(KOL);  
  READLN;  
END;
```

Напишем главную программу

```

PROGRAM MAIN;
USES CRT;
VAR NAME: FILE OF STRING[30];
KOL: FILE OF INTEGER;
FAM: STRING[30];
COMP, COST, PREM, SUM, SUM1, SUMC, SUMV, SUMP: INTEGER;
FAM: STRING[30];
BEGIN
    ASSIGN(NAME, 'NAME.DAT');
    ASSIGN(KOL, 'KOL.DAT');
    RESET(NAME); RESET(KOL);
    COST:=1000; PREM:=25;
    SUMC:=0; SUMV:=0; SUMP:=0;
    WRITELN('Фамилия    количество    сумма    выручки    сумма
            премии');
    WHILE NOT EOF(NAME) DO
    BEGIN READ(NAME, FAM);
        READ(KOL, COMP);
        SUM:=COMP*COST;
        SUM1:=COMP*PREM;
        SUMC:=SUMC+COMP;
        SUMV:=SUMV+SUM;
        SUMP:=SUMP+SUM1;
        WRITELN(FAM, COMP, SUM, SUM1);
    END;
    WRITELN('ИТОГО: ', SUMC, SUMV, SUMP);
    READLN;
END.

```

## Литература

1. Фигурнов В.Э.  
“ IBM PC для пользователя”, М., 1995 г.
2. Левин А.  
“ Самоучитель работы на компьютере”, М., 1996 г.
3. Брябрин В.М.  
“ Программное обеспечение ПЭВМ”, М., 1989 г.
4. Фролов Г.Д., Кузнецов Э.И.  
“ Элементы информатики”, М., 1989
5. Дж. Мартин  
“ Организация баз данных в вычислительных системах”, М., 1976 г.
6. Громов Г.Р.  
“ Национальные информационные ресурсы”, М., 1985 г.
7. Глушков В.М.  
“ Основы безбумажной технологии”, М., 1982 г.
8. Г. Симпсон  
"Профессиональная работа на персональном компьютере", М.:  
"Финансы и статистика", 1988 г.
9. А. Файсман  
"Профессиональное программирование на Турбо Паскале", М.:  
"Инфомэкс", 1992 г.

## Содержание

Предисловие .....	4
Введение .....	7
Информация и информатика. ....	7
Формы и виды представления информации .....	8
Глава I.    Теоретические основы информатики.....	10
§1.1. Системы счисления .....	10
Представление чисел в памяти компьютера.....	14
§1.2. Кодирование чисел.....	18
Обратный код чисел.....	20
§1.3. Логические основы вычислительной техники .....	29
Законы алгебры логики.....	33
Переместительный закон .....	34
Глава II.  Технические средства вычислительной техники .....	36
§2.1. Краткая история развития средств вычислительной техники.....	36
§2.2. Логические элементы вычислительной техники, реализующие основные логические функции.....	37
§2.3. Принципы работы компьютера.....	44
Что такое "компьютер"?.....	44
Простейшая структура компьютера .....	44
Принципы фон - Неймана.....	46
§2.4. Классификация компьютеров .....	49
Классификация по назначению .....	49
Другие системы классификации компьютеров .....	51
§2.5. Аппаратное обеспечение.....	53
Глава III. Программное обеспечение компьютеров .....	63
§3.1. Структура программного обеспечения .....	63
§3.2. Основы Windows-95 .....	65
Графический интерфейс Windows – 95.....	65

Файловая система Windows – 95 .....	71
Глава IV. Текстовый редактор Word – 97 .....	78
§4.1. Введение в Word – 97 .....	78
Элементы окна Word – 97 .....	78
Панели инструментов Word.....	79
Кнопки просмотра .....	80
Справка Word – 97. Работа с "Помощником" .....	81
§4.2 Практикум по Word – 97 .....	84
Урок 1.....	84
Урок 2.....	90
Урок 3.....	97
Урок 4.....	102
Урок 5.....	107
Урок 6.....	117
Урок 7.....	128
Глава V. Основы программирования .....	136
§5.1. Понятие алгоритма. ....	136
§5.2. Введение в язык Турбо – Паскаль .....	141
Переменные. Стандартные типы. ....	142
Операции отношения .....	143
Раздел описаний переменных.....	143
Выражения. Порядок выполнения операций.....	143
Константы .....	144
Основные операторы Паскаля.....	144
Оператор присваивания .....	144
Операторы ввода/вывода .....	145
Операции с целыми числами .....	146
Стандартные функции с целыми переменными (аргументами).....	147
Операции с вещественными числами (тип Real).....	147
Одновременное использование вещественных и целых чисел.....	148

Другие стандартные функции с вещественными аргументами.....	149
Булевы переменные и условные операторы .....	149
Условные операторы.....	150
Оператор IF ....THEN.....	150
Оператор IF ...THEN.....ELSE .....	151
Операторы цикла.....	154
Оператор цикла с постусловием.....	155
Оператор цикла с параметром.....	158
Второй вариант оператора цикла с параметром.....	159
Символьные и строковые типы данных.....	161
Строковые процедуры и функции.....	162
Массивы.....	166
Модули в Турбо Паскале.....	169
Создание и обработка последовательных файлов.....	172
Файловые типы.....	173
Литература.....	177
Содержание .....	178

**Мансуров К.Т.**

# Информатика для программиста

введение в специальность

Редактор Маматалиев А.

Корректор Курбанова Г.

Подписано к печати 18.06.02. Формат 60x84/16

Усл. п.л. 11,5. Тираж 200

Отпечатано офсетным способом

---

АО "Ошоблтипография", г. Ош, ул. Курманжан датка, 209.