Северо-Кавказский филиал
Ордена Трудового Красного Знамени федерального государственного
бюджетного образовательного учреждения высшего образования
"Московский технический университет связи и информатики"

Кафедра «Сети связи и системы коммутации»

Методические указания для проведения лабораторных работ по дисциплине «Информатика»

для студентов направления 38.03.02 Менеджмент.
Профиль производственный менеджмент (в инфокоммуникациях).

Ростов-на-Дону
2016

План УМД 2015/16 уч. год

Методические указания для проведения лабораторных работ по дисциплине «Информатика»

для студентов направления 38.03.02 Менеджмент. Профиль производственный менеджмент (в инфокоммуникациях).

Составители:

Сосновский И.А., доцент кафедры «Сети связи и системы коммутации», к.т.н.

Методические указания рассмотрены и утверждены на заседании кафедры «Сети связи и системы коммутации», протокол № 6 от 25.01.2016 г.

Рецензент: доцент кафедры «систем передачи и обработки информации» к.т.н. Смоляков В.Н.

Лабораторная работа 1. Вычисление количества и объема информации.

Цель работы: изучить понятие информации, три уровня рассмотрения информации, способы определения количества информации на этих уровнях; единицы измерения количества информации.

Теоретическая часть.

Понятие информации можно рассматривать при различных ограничениях, накладываемых на ее свойства, т. е. при различных уровнях рассмотрения. В основном выделяют три уровня – синтаксический, семантический и прагматический. Соответственно на каждом из них для определения количества информации применяют различные оценки.

На *синтаксическом уровне* для оценки количества информации используют вероятностные методы, которые принимают во внимание только вероятностные свойства информации и не учитывают другие (смысловое содержание, полезность, актуальность и т. д.).

Разработанные в середине XX в. математические и, в частности, вероятностные методы позволили сформировать подход к оценке количества информации как к мере уменьшения неопределенности знаний. Такой подход, называемый также вероятностным, постулирует принцип: если некоторое сообщение приводит к уменьшению неопределенности наших знаний, то можно утверждать, что такое сообщение содержит информацию. При этом сообщения содержат информацию о каких - либо событиях, которые могут реализоваться с различными вероятностями. Формулу для определения количества информации для событий с различными вероятностями и получаемых от дискретного источника информации предложил американский ученый К. Шеннон в 1948 г. Согласно этой формуле количество информации может быть определено следующим образом [2,6,7]:

$$I = -\sum_{i=1}^{N} p_i \log_2 p_i, \qquad (1.1)$$

где I –количество информации; N –количество возможных событий (сообщений); pi –вероятность отдельных событий (сообщений); Σ – математический знак суммы чисел.

Определяемое с помощью формулы (1.1) количество информации принимает только положительное значение. Поскольку вероятность отдельных событий меньше единицы, то соответственно выражение log pi является отрицательной величиной и для получения положительного значения количества информации в формуле (1.1) перед знаком суммы стоит знак минус.

Если вероятность появления отдельных событий одинаковая, и они образуют полную группу событий, т. е [2,6,7].

$$\sum_{i=1}^{N} p_i = 1,$$

то формула (1.1) преобразуется в формулу Р. Хартли:

$$I = \log_2 N, \qquad (1.2)$$

В формулах (1.1) и (1.2) отношение между количеством информации и соответственно вероятностью, или количеством, отдельных событий выражается с помощью логарифма. Применение логарифмов в формулах (1.1) и (1.2) можно объяснить следующим образом. Для простоты рассуждений воспользуемся соотношением (1.2). Будем последовательно присваивать аргументу N значения, выбираемые, например, из ряда чисел: 1, 2, 4, 8, 16, 32, 64 и т. д. Чтобы определить, какое событие из N равновероятных событий произошло, для каждого числа ряда необходимо последовательно производить операции выбора из двух возможных событий.

Так, при N =1 количество операций будет равно 0 (вероятность события равна 1), при N =2, количество операций будет равно 1, при N =4 количество операций будет равно 2, при N =8, количество операций будет равно 3 и т. д. Таким образом получим следующий ряд чисел: 0, 1, 2, 3, 4, 5, 6 и т. д., который можно считать соответствующим значениям функции I в соотношении (1.2). Последовательность значений чисел, которые принимает аргумент N, представляет собой ряд, известный в математике как ряд чисел, образующих геометрическую прогрессию, а последовательность значений чисел, которые принимает функция I, будет являться рядом, образующим арифметическую прогрессию. Таким образом, логарифм в формулах (1.1) и (1.2) устанавливает соотношение между рядами, представляющими геометрическую и арифметическую прогрессии, что достаточно хорошо известно в математике [2,6,7].

В информатике, изучающей процессы получения, обработки, передачи и хранения информации с помощью вычислительных (компьютерных) систем, в основном используется двоичное кодирование, при котором используется знаковая система, состоящая из двух символов 0 и 1. По этой причине в формулах (1.1) и (1.2) в качестве основания логарифма используется цифра 2.

Исходя из вероятностного подхода к определению количества информации эти два символа двоичной знаковой системы можно рассматривать как два различных возможных события, поэтому за единицу количества информации принято такое количество информации, которое содержит сообщение, уменьшающее неопределенность знания в два раза (до получения событий их вероятность равна 0,5, после получения – 1, неопределенность уменьшается соответственно: 1/0,5 = 2, т. е. в 2 раза). Такая единица измерения информации называется битом (от англ. слова binary digit –двоичная цифра). Таким образом, в качестве меры для оценки количества информации на синтаксическом уровне, при условии двоичного кодирования, принят один бит.

Следующей по величине единицей измерения количества информации является байт, представляющий собой последовательность, составленную из восьми бит, т. е.

1 байт = 23 бит = 8 бит.

В информатике также широко используются кратные байту единицы измерения количества информации, однако в отличие от метрической системы мер, где в качестве множителей кратных единиц применяют коэффициент 10n, где п =3, 6, 9 и т. д., в кратных единицах измерения количества информации

используется коэффициент 2n. Выбор этот объясняется тем, что компьютер в основном оперирует числами не в десятичной, а в двоичной системе счисления.

Кратные байту единицы измерения количества информации вводятся следующим образом:

1 Килобайт (Кбайт) = 2^{10}байт = 1024 байт,

1 Мегабайт (Мбайт) = 2^{10}Кбайт = 1024 Кбайт,

1 Гигабайт (Гбайт) = 2^{10}Мбайт = 1024 Мбайт,

1 Терабайт (Тбайт) = 2^{10}Гбайт = 1024 Гбайт,

1 Петабайт (Пбайт) = 2^{10}Тбайт = 1024 Тбайт,

1 Экзабайт (Эбайт) = 2^{10}Пбайт = 1024 Пбайт.

Единицы измерения количества информации, в названии которых есть приставки «кило», «мега» и т. д., с точки зрения теории измерений не являются корректными, поскольку эти приставки используются в метрической системе мер, в которой в качестве множителей кратных единиц используется коэффициент 10n, где п =3, 6, 9 и т. д. Для устранения этой некорректности международная организацией International Electrotechnical Commission, занимающаяся созданием стандартов для отрасли электронных технологий, утвердила ряд новых приставок для единиц измерения количества информации: киби (kibi), меби (mebi), гиби (gibi), теби (tebi), пети (peti), эксби (exbi). Однако пока используются старые обозначения единиц измерения количества информации, и требуется время, чтобы новые названия начали широко применяться.

Вероятностный подход используется и при определении количества информации, представленной с помощью знаковых систем. Если рассматривать символы алфавита как множество возможных сообщений N, то количество информации, которое несет один знак алфавита, можно определить по формуле (1.1). При равновероятном появлении каждого знака алфавита в тексте сообщения для определения количества информации можно воспользоваться формулой (1.2).

Количество информации, которое несет один знак алфавита, тем больше, чем больше знаков входит в этот алфавит. Количество знаков, входящих в алфавит, называется мощностью алфавита. Количество информации (информационный объем), содержащееся в сообщении, закодированном с помощью знаковой системы и содержащем определенное количество знаков (символов), определяется с помощью формулы [2,6,7]:

$$V = I \cdot K, \qquad (1.3)$$

где V – информационный объем сообщения; I = log2N, информационный объем одного символа (знака); К – количество символов (знаков) в сообщении; N –мощность алфавита (количество знаков в алфавите).

Рассмотрим примеры.

Пример 1.1

Определим, какое количество информации можно получить после реализации одного из шести событий. Вероятность первого события составляет 0,15; второго – 0,25; третьего – 0,2; четвертого – 0,12; пятого – 0,12; шестого – 0,1, т. е. Р1 =0,15; Р2 =0,25; Р3 = 0,2; Р4 =0,18; Р5 =0,12; Р6 =0,1.

Решение.

Для определения количества информации применим формулу (1.1)

$$I = -(P_1 \cdot \log_2 P_1 + P_2 \cdot \log_2 P_2 + P_3 \cdot \log_2 P_3 + P_4 \cdot \log_2 P_4 +$$
$$+ P_5 \cdot \log_2 P_5 + P_6 \cdot \log_2 P_6) = -(0{,}15 \cdot \log_2 0{,}15 + 0{,}25 \cdot \log_2 0{,}25 +$$
$$+ 0{,}2 \cdot \log_2 0{,}2 + 0{,}18 \cdot \log_2 0{,}18 + 0{,}12 \cdot \log_2 0{,}12 + 0{,}1 \cdot \log_2 0{,}1) \text{ бит.}$$

Для вычисления этого выражения, содержащего логарифмы, воспользуемся сначала компьютерным калькулятором, а затем табличным процессором Microsoft (MS) Excel, входящим в интегрированный пакет программ MS Office XP.

Для вычисления с помощью компьютерного калькулятора выполним следующие действия. С помощью команды: [Кнопка Пуск– Программы – Стандартные – Калькулятор] запустим программу Калькулятор. После запуска программы выполним команду: [Вид – Инженерный] (рис. 1.3).

Рис. 1.3. Инженерный калькулятор

Кнопка log калькулятора производит вычисление десятичного (по основанию 10) логарифма отображаемого числа. Поскольку в нашем случае необходимо производить вычисления логарифмов по основанию 2, а данный калькулятор не позволяет этого делать, то необходимо воспользоваться известной формулой:

logbN =M · logaN,

где

$$M = \frac{1}{\log_a b} \text{ (модуль перевода).}$$

В нашем случае соотношение примет вид: $log_2N = M\log_{10}N$,

где

$$M = \frac{1}{\log_{10} 2} = 3{,}322,$$

т. е $\log_2 N = 3{,}322 \cdot \log_{10} N$, и выражение для вычисления количества информации примет вид:

$$I = -3{,}322(0{,}15 \cdot \log_{10} 0{,}15 + 0{,}25 \cdot \log_{10} 0{,}25 + 0{,}2 \cdot \log_{10} 0{,}2 +$$
$$+ 0{,}18 \cdot \log_{10} 0{,}18 + 0{,}12 \cdot \log_{10} 0{,}12 + 0{,}1 \cdot \log_{10} 0{,}1) \text{ бит.}$$

При вычислении на калькуляторе используем кнопки: +/ (изменение знака отображаемого числа), ()(открывающие и закрывающие скобки), log (логарифм

числа по основанию 10) и т. д. Результат вычисления показан на рис. 1.3. Таким образом, количество информации I = 2,52 бит.

Воспользуемся теперь табличным процессором MS Excel. Для запуска программы Excel выполним команду: [Кнопка Пуск – Программы – MS Office XP – Microsoft Excel].В ячейки A1, B1, C1, D1, E1, F1 открывшегося окна Excel запишем буквенные обозначения вероятностей P1, P2, P3, P4, P5, P6 а в ячейку G1 – количество информации I, которое необходимо определить. Для написания нижних индексов у вероятностей P1 ÷ P6 в ячейках A1, B1, C1, D1, E1, F1 выполним следующую команду: [Формат – Ячейки – Шрифт – Видоизменение (поставим флажок напротив нижнего индекса)]. В ячейки A2, B2, C2, D2, E2, F2 запишем соответствующие значения вероятностей.

После записи значений в ячейки необходимо установить в них формат числа. Для этого необходимо выполнить следующую команду: [Формат – Ячейки – Число – Числовой (устанавливаем число десятичных знаков, равное двум)]. Устанавливаем в ячейке G2 тот же числовой формат. В ячейку G2 записываем выражение = – (A2*LOG(A2;2) + B2*LOG(B2;2) + C2*LOG(C2;2) + D2*LOG(D2;2) + E2*LOG(E2;2) + F2*LOG(F2;2)). После нажатия на клавиатуре компьютера клавиши <Enter>,в ячейке G2 получим искомый результат – I =2,52 бит (рис. 1.4).

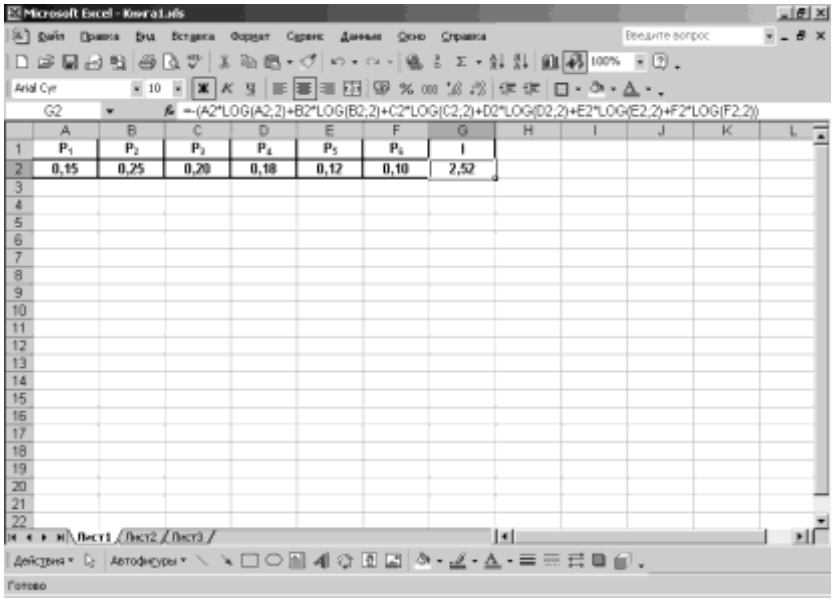


Рис. 1.4. Результат вычисления количества информации

Пример 1.2

Определим, какое количество байт и бит информации содержится в сообщении, если его объем составляет 0,25 Кбайта.

Решение.

С помощью калькулятора определим количество байт и бит информации, которое содержится в данном сообщении:

I =0,25 Кбайт · 1024 байт/1 Кбайт = 256 байт;

I =256 байт · 8 бит/1 байт = 2048 бит.

На *семантическом уровне* информация рассматривается по ее содержанию, отражающему состояние отдельного объекта или системы в целом. При этом не учитывается ее полезность для получателя информации.

Данный подход предполагает, что для понимания (осмысливания) и использования полученной информации приемник (получатель) должен обладать априорной информацией (тезаурусом), т. е. определенным запасом знаков, наполненных смыслом, слов, понятий, названий явлений и объектов, между которыми установлены связи на смысловом уровне. Таким образом, если принять знания о данном объекте или явлении за тезаурус, то количество информации, содержащееся в новом сообщении о данном предмете, можно оценить по изменению индивидуального тезауруса под воздействием данного сообщения. В зависимости от соотношений между смысловым содержанием сообщения и тезаурусом пользователя изменяется количество семантической информации, при этом характер такой зависимости не поддается строгому математическому описанию и сводится к рассмотрению трех основных условий, при которых тезаурус пользователя:

• стремится к нулю, т. е. пользователь не воспринимает поступившее сообщение;

• стремится к бесконечности, т. е. пользователь досконально знает все об объекте или явлении и поступившее сообщение его не интересует;

• согласован со смысловым содержанием сообщения, т. е. поступившее сообщение понятно пользователю и несет новые сведения.

Два первых предельных случая соответствуют состоянию, при котором количество семантической информации, получаемое пользователем, минимально. Третий случай связан с получением максимального количества семантической информации. Таким образом, количество семантической информации, получаемой пользователем, является величиной относительной, поскольку одно и то же сообщение может иметь смысловое содержание для компетентного и быть бессмысленным для некомпетентного пользователя.

На *прагматическом уровне*информация рассматривается с точки зрения ее полезности (ценности) для достижения потребителем информации (человеком) поставленной практической цели. Данный подход при определении полезности информации основан на расчете приращения вероятности достижения цели до и после получения информации.

Количество информации, определяющее ее ценность (полезность), находится по формуле:

$$I = \log_2 P_1 - \log_2 P_0 = \log_2 \frac{P_1}{P_0}, \qquad (1.4)$$

где P_0, P_1 –вероятность достижения цели соответственно до и после получения информации.

В качестве единицы измерения (меры) количества информации, определяющей ее ценность, может быть принят 1 бит (при основании логарифма, равном 2), т. е. это такое количество полученной информации, при котором отношение вероятностей достижения цели равно 2.

Рассмотрим три случая, когда количество информации, определяющее ее ценность, равно нулю и когда она принимает положительное и отрицательное значение.

Количество информации равно нулю при $P_0=P_1$, т.е. полученная информация не увеличивает и не уменьшает вероятность достижения цели.

Значение информации является положительной величиной при $P_1 > P_0$, т. е. полученная информация уменьшает исходную неопределенность и увеличивает вероятность достижения цели [2,6,7].

Значение информации является отрицательной величиной при $P_1 < P_0$, т. е. полученная информация увеличивает исходную неопределенность и уменьшает вероятность достижения цели. Такую информацию называют дезинформацией.

Общая постановка задачи

В данной лабораторной работе необходимо выполнить все разобранные в теоретической части примеры, параллельно знакомясь с теорией. Затем выполнить все задания, представленные ниже.

Список индивидуальных данных

1. По каналу связи передается пять сообщений, вероятность получения первого сообщения составляет 0,3; второго – 0,2; третьего – 0,14, а вероятности получения четвертого и пятого сообщений равны между собой. Какое количество информации мы получим после приема одного из сообщений?

$$(\sum_{i=1}^{N} p_i = 1).$$

2. Совершаются два события. При каких вероятностях этих событий мы получим минимальное и максимальное количество информации?

3. Какое количество информации несет в себе сообщение о том, что нужная вам компьютерная программа находится на одной из семи дискет?

4. С помощью компьютерного калькулятора заполнить пропуски числами:

а) 2 Кбайт = ___ байт = ___ бит;

б) ___ Гбайт = 2357 Мбайт = ___ Кбайт;

в) ___ Кбайт = ___ байт = 14567 бит;

г) 3 Гбайт = __ Мбайт = ___ Кбайт;

д) ___ Тбайт = 8 Гбайт = ___ Мбайт.

5. Определить информационную емкость буквы в русском и латинском алфавитах.

6. Сколько символов содержит сообщение, если его информационный объем составляет 1,25 Кбайта и мощность алфавита, с помощью которого записано сообщение, равна 32?

7. Опытный пользователь компьютера может вводить в минуту 110 знаков. Мощность алфавита, используемого в компьютере, равна 256. Какое количество информации в байтах может ввести пользователь в компьютер за 1 и 1,5 минуты?

8. Определить количество информации, определяющее ее ценность, если вероятность достижения цели до получения информации равна 0,5, а после получения информации – 0,3.

Пример выполнения работы

Определим мощность алфавита, с помощью которого передано сообщение, содержащее 4096 символов, если информационный объем сообщения составляет 2 Кбайта.

Решение.

С помощью калькулятора переведем информационный объем сообщения из килобайт в биты:

V = 2Кбайт 1024 байт/1 Кбайт = 2048 байт 8 бит/1 байт = 16384 бит.

Определим количество бит, приходящееся на один символ (информационный объем одного символа) в алфавите:

I= 16 384 бит/4096 = 4 бит.

Используя формулу (1.3), определим мощность алфавита (количество символов в алфавите):

$N = 2^I = 2^4 = 16$.

Решение задачи описывается и приводится копия экрана с решением каждого задания. Задание для каждого студента выдаётся преподавателем.

Контрольные вопросы

1. Определение информации.
2. Измерение информации.
3. Что такое количество информации?
4. Вычисление количества информации.
5. Что такое объем информации?
6. Вычисление объема информации.

Лабораторная работа 2. *Перевод чисел из одной ПСС в другую.*

Цель: научиться переводить числа из одной системы счисления в другую.

Методические указания.

Под системой счисления понимается способ представления любого числа с помощью некоторого алфавита символов, называемых цифрами.

Все системы счисления делятся на позиционные и непозиционные.

Непозиционными системами являются такие системы счисления, в которых каждый символ сохраняет свое значение независимо от места его положения в числе. Примером непозиционной системы счисления является римская система. К недостаткам таких систем относятся наличие большого количества знаков и сложность выполнения арифметических операций.

Система счисления называется позиционной, если одна и та же цифра имеет различное значение, определяющееся позицией цифры в последовательности цифр, изображающей число. Это значение меняется в однозначной зависимости от позиции, занимаемой цифрой, по некоторому закону. Примером позиционной системы счисления является десятичная система, используемая в повседневной жизни.

Количество *p* различных цифр, употребляемых в позиционной системе определяет название системы счисления и называется основанием системы счисления *"p"*.

В десятичной системе используются десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; эта система имеет основанием число десять.

Задание 1. Запишите развернутую и краткую формы записи любого числа.

В ЭВМ применяют позиционные системы счисления с недесятичным основанием: двоичную, восьмеричную, шестнадцатеричную. В аппаратной основе ЭВМ лежат двухпозиционные элементы, которые могут находиться только в двух состояниях; одно из них обозначается 0, а другое 1. Поэтому основной системой счисления применяемой в ЭВМ является двоичная система.

Двоичная система счисления. Используется две цифры: 0 и 1.

Восьмеричная система счисления. Используется восемь цифр: 0, 1, 2, 3, 4, 5, 6, 7.

Употребляется в ЭВМ как вспомогательная для записи информации в сокращенном виде. Для представления одной цифры восьмеричной системы используется три двоичных разряда (триада) (Таблица 1).

Шестнадцатеричная система счисления. Для изображения чисел употребляются 16 цифр. Первые десять цифр этой системы обозначаются цифрами от 0 до 9, а старшие шесть цифр латинскими

буквами: 10=A,

11=B,

12=C,

13=D,
14=E,
15=F.

Шестнадцатеричная система используется для записи информации в сокращенном виде. Для представления одной цифры шестнадцатеричной системы счисления используется четыре двоичных разряда (тетрада) (Таблица 1).

Таблица 1 - Наиболее важные системы счисления.

Двоичная (Основание 2)	Восьмеричная (Основание 8)		Десятичная (Основание 10)	Шестнадцатиричная (Основание 16)	
		триады			тетрады
0	0	000	0	0	0000
1	1	001	1	1	0001
	2	010	2	2	0010
	3	011	3	3	0011
	4	100	4	4	0100
	5	101	5	5	0101
	6	110	6	6	0110
	7	111	7	7	0111
			8	8	1000
			9	9	1001
				A	1010
				B	1011
				C	1100
				D	1101
				E	1110
				F	1111

Перевод чисел из одной системы счисления в другую.

Перевод чисел в десятичную систему осуществляется путем составления степенного ряда с основанием той системы, из которой число переводится. Затем подсчитывается значение суммы.

Задание 2.

Перевести 10101101.101 из «2» в «16», «8» и «10» с.с.

При одновременном использовании нескольких различных систем счисления основание системы, к которой относится число, указывается в виде нижнего индекса.

Задание 3. Переведите самостоятельно.

а) Перевести 703.048 из «10» в «2», затем в «8» и наконец, в «16»

б) Перевести B2E.416 из «16» в «10», затем в «8».

Перевод целых десятичных чисел в недесятичную систему счисления осуществляется последовательным делением десятичного числа на основание той системы, в которую оно переводится, до тех пор, пока не получится частное меньшее этого основания. Число в новой системе записывается в виде остатков деления, начиная с последнего.

Задание 4.

а) Перевести 18110 из «10» в «2».

б) Перевести 62210 из «8» в «2», затем в «10».

Перевод правильных дробей из десятичной системы счисления в недесятичную.

Для перевода правильной десятичной дроби в другую систему эту дробь надо последовательно умножать на основание той системы, в которую она переводится. При этом умножаются только дробные части. Дробь в новой системе записывается в виде целых частей произведений, начиная с первого.

Задание 5. Перевести 0.312510

Замечание. Конечной десятичной дроби в другой системе счисления может соответствовать бесконечная (иногда периодическая) дробь. В этом случае количество знаков в представлении дроби в новой системе берется в зависимости от требуемой точности.

Задание 6. Перевести 0.6510 из «10» в «2» с.с. Точность 6 знаков.

Для перевода неправильной десятичной дроби в систему счисления с недесятичным основанием необходимо отдельно перевести целую часть и отдельно дробную.

Задание 7.

Перевести 23.12510 из «10» в «2» с.с.

Необходимо отметить, что целые числа остаются целыми, а правильные дроби дробями в любой системе счисления. Для перевода восьмеричного или шестнадцатеричного числа в двоичную форму достаточно заменить каждую цифру этого числа соответствующим трехразрядным двоичным числом (триадой) (Таб. 1) или четырехразрядным двоичным числом (тетрадой) (Таб. 1), при этом отбрасывают ненужные нули в старших и младших разрядах.

Задание 8.

а)Перевести 305.47 из «8» в «10» с.с.

б)Перевести 7B2.E16 из «16» в «10».

Для перехода от двоичной к восьмеричной (шестнадцатеричной) системе поступают следующим образом: двигаясь от точки влево и вправо, разбивают двоичное число на группы по три (четыре) разряда, дополняя при необходимости нулями крайние левую и правую группы. Затем триаду (тетраду) заменяют соответствующей восьмеричной (шестнадцатеричной) цифрой.

Задание 9.

а) Перевести 1101111001.1101 из «2» в «8» с.с.

б) Перевести 11111111011.100111 из «2» в «16» с.с.

Перевод из восьмеричной в шестнадцатеричную систему и обратно осуществляется через двоичную систему с помощью триад и тетрад.

Задание 10.

Перевести 175.248 "16" с.с.

Двоичная арифметика.

При сложении двоичных чисел в каждом разряде производится сложение цифр слагаемых и переноса из соседнего младшего разряда, если он имеется. При этом необходимо учитывать, что 1+1 дают нуль в данном разряде и единицу переноса в следующий.

Задание 11. Выполнить сложение двоичных чисел:

а) X=1101, Y=101;

б) X=1101, Y=101, Z=111;

При вычитании двоичных чисел в данном разряде при необходимости занимается 1 из старшего разряда. Эта занимаемая 1 равна двум 1 данного разряда.

Задание 12. Заданы двоичные числа X=10010 и Y=101. Вычислить X-Y.

Умножение двоичных чисел производится по тем же правилам, что и для десятичных с помощью таблиц двоичного умножения и сложения.

Пример. 1001* 101 = ?

Деление двоичных чисел производится по тем же правилам, что и для десятичных. При этом используются таблицы двоичного умножения и вычитания.

Пример. 1100.011 : 10.01= ?

Самостоятельная работа.

Выполнить перевод числа в соответствии с вариантом.

1. Перевести десятичное число A=121 в двоичную систему счисления.
2. Перевести двоичное число A=10001010111,01 в десятичную систему счисления.
3. Перевести десятичное число A=135,656 в двоичную систему счисления с точностью до пяти знаков запятой.
4. Перевести двоичное число A=10111011 в десятичную систему счисления методом деления на основание.
5. Перевести восьмеричное число A=345,766 в двоичную систему счисления.
6. Записать десятичное число A=79,346 в двоичнодесятичной форме.
7. Перевести десятичную дробь 64 *A* = 63 9 в двоичную систему счисления.
8. Перевести десятичное число A=326 в троичную систему счисления.
9. Перевести десятичную дробь 40 *A* = 63 5 в двоичную систему счисления.
10. Перевести десятичное число A=15,647 в двоичную систему счисления.
11. Перевести десятичное число A=1211 в пятеричную систему счисления.
12. Перевести десятичную дробь A=0,625 в двоичную систему счисления.

Контрольные вопросы

1. Что такое система счисления?
2. Опишите виды систем счисления.
3. Преимущества позиционной системы счисления?
4. Запишите степенную формулу представления числа в ПСС.

5. Перевод числа из любой ПСС в десятичную.

6. Как перевести целое восьмеричное число в двоичное и обратно?

7. Как перевести целое восьмеричное число в шестнадцатеричное и обратно?

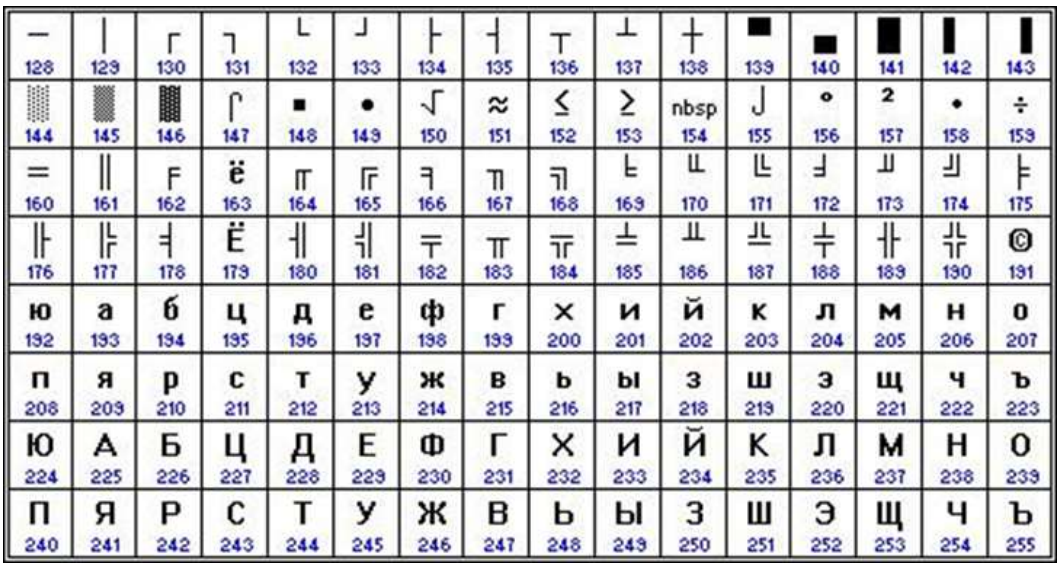
8. Перевод дробных чисел из системы счисления с основанием a в дробь с основанием b.

9. Выполнение арифметических операции с числами в разных ПСС.

Лабораторная работа 3. *Использование различных таблиц кодировки данных.*

Цель: научиться переводить числа из одной системы счисления в другую.

КОИ-7, КОИ-8 – кодирование русских букв и символов (семи-, восьми - битное кодирование).



─ 128	│ 129	┌ 130	┐ 131	└ 132	┘ 133	├ 134	┤ 135	┬ 136	┴ 137	┼ 138	▀ 139	▄ 140	▌ 141	▐ 142	▀ 143
▒ 144	▒ 145	▒ 146	⌠ 147	■ 148	• 149	√ 150	≈ 151	≤ 152	≥ 153	nbsp 154	⌡ 155	° 156	² 157	• 158	÷ 159
═ 160	║ 161	╒ 162	ё 163	╓ 164	╔ 165	╕ 166	╖ 167	╗ 168	╘ 169	╙ 170	╚ 171	╛ 172	╜ 173	╝ 174	╞ 175
╟ 176	╠ 177	╡ 178	Ё 179	╢ 180	╣ 181	╤ 182	╥ 183	╦ 184	╧ 185	╨ 186	╩ 187	╪ 188	╫ 189	╬ 190	© 191
ю 192	а 193	б 194	ц 195	д 196	е 197	ф 198	г 199	х 200	и 201	й 202	к 203	л 204	м 205	н 206	о 207
п 208	я 209	р 210	с 211	т 212	у 213	ж 214	в 215	ь 216	ы 217	з 218	ш 219	э 220	щ 221	ч 222	ъ 223
Ю 224	А 225	Б 226	Ц 227	Д 228	Е 229	Ф 230	Г 231	Х 232	И 233	Й 234	К 235	Л 236	М 237	Н 238	О 239
П 240	Я 241	Р 242	С 243	Т 244	У 245	Ж 246	В 247	Ь 248	Ы 249	З 250	Ш 251	Э 252	Щ 253	Ч 254	Ъ 255

1) #154 неразрывный пробел.

Задание. С учётом указанной таблицы кодировки записать следующую информацию о себе. «№ учебной группы, № студенческого билета, фамилия имя отчество».

ASCII –American Standard Code for Information Interchange (американский стандарт кодов для обмена информацией) – это восьмиразрядная кодовая таблица, в ней закодировано 256 символов (127-стандартные коды символов английского языка, спецсимволы, цифры, а коды от 128 до 255 – национальный стандарт, алфавит языка, символы псевдографики, научные символы, коды от 0 до 32 отведены не символам, а функциональным клавишам) [2,6,7].

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

1) #32 - пробел.

Задание. С учётом указанной таблицы кодировки записать следующую информацию о себе. «№ учебной группы, № студенческого билета, фамилия имя отчество».

Unicode – стандарт, согласно которому для представления каждого символа используется 2 байта. (можно кодировать математические символы, русские, английские, греческие и даже китайские). С его помощью можно закодировать не 256, а 65536 различных символов. Полная спецификация стандарта Unicode включает в себя все существующие, вымершие и искусственно созданные алфавиты мира, а также множество математических, музыкальных, химических и прочих символов [2,6,7].

CP1251 - наиболее распространенной в настоящее время является кодировка Microsoft Windows, ("CP" означает "Code Page", "кодовая страница").

А 128	Б 129	В 130	Г 131	Д 132	Е 133	Ж 134	З 135	И 136	Й 137	К 138	Л 139	М 140	Н 141	О 142	П 143
Р 144	С 145	Т 146	У 147	Ф 148	Х 149	Ц 150	Ч 151	Ш 152	Щ 153	Ъ 154	Ы 155	Ь 156	Э 157	Ю 158	Я 159
† 160	° 161	ы̀ 162	£ 163	§ 164	• 165	¶ 166	ы́ 167	® 168	© 169	™ 170	А́ 171	а́ 172	ё̀ 173	а̀ 174	è 175
Ё̀ 176	± 177	≤ 178	≥ 179	э́ 180	µ 181	ґ 182	Э́ 183	Ю́ 184	ю́ 185	Я́ 186	я́ 187	Е́ 188	е́ 189	ѝ 190	ò 191
э̀ 192	ю̀ 193	¬ 194	√ 195	ƒ 196	≈ 197	Δ 198	« 199	» 200	... 201	nbsp 202	О́ 203	о́ 204	И́ 205	и́ 206	я̀ 207
– 208	— 209	“ 210	” 211	‘ 212	’ 213	÷ 214	„ 215	у̀ 216	Ы́ 217	У́ 218	у́ 219	№ 220	Ё 221	ё 222	я 223
а 224	б 225	в 226	г 227	д 228	е 229	ж 230	з 231	и 232	й 233	к 234	л 235	м 236	н 237	о 238	п 239
р 240	с 241	т 242	у 243	ф 244	х 245	ц 246	ч 247	ш 248	щ 249	ъ 250	ы 251	ь 252	э 253	ю 254	¤ 255

1) #160 неразрывный пробел,
2) #173 мягкий перенос.

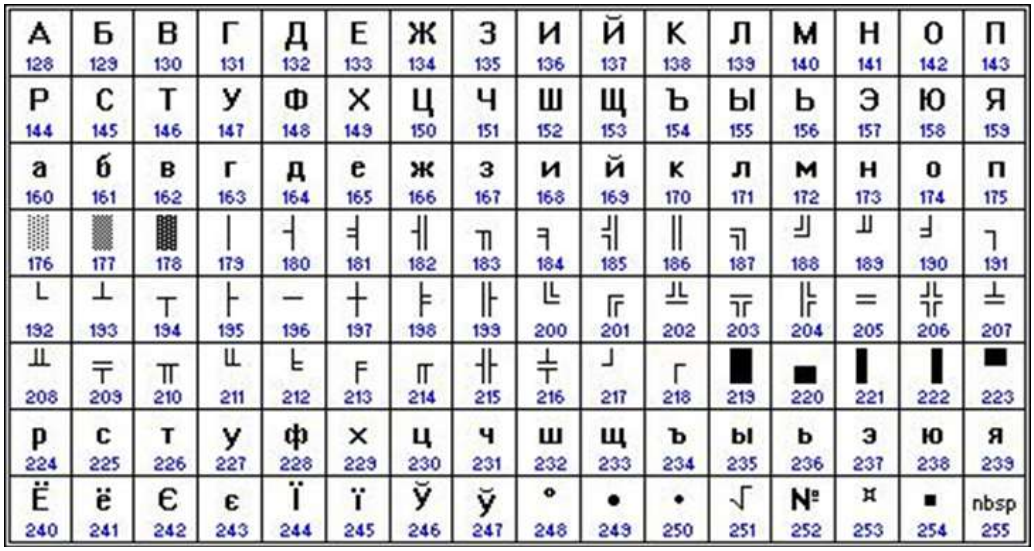
Задание. С учётом указанной таблицы кодировки записать следующую информацию о себе. «№ учебной группы, № студенческого билета, фамилия имя отчество».

CP866 - кодировка под MS DOS

А́ 128	а̀ 129	, 130	è 131	„ 132	... 133	† 134	‡ 135	€ 136	‰ 137	Е́ 138	< 139	ѝ 140	И́ 141	О́ 142	У́ 143
а́ 144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	— 151	ё̀ 152	™ 153	е́ 154	> 155	о̀ 156	и́ 157	о́ 158	у́ 159
nbsp 160	у̀ 161	Ы́ 162	Э́ 163	¤ 164	ы̀ 165	¦ 166	§ 167	Ё 168	© 169	Ю́ 170	« 171	¬ 172	shy 173	® 174	Я́ 175
° 176	± 177	ы́ 178	э́ 179	´ 180	µ 181	¶ 182	• 183	ё 184	№ 185	ю́ 186	» 187	э̀ 188	ю̀ 189	я̀ 190	я́ 191
А 192	Б 193	В 194	Г 195	Д 196	Е 197	Ж 198	З 199	И 200	Й 201	К 202	Л 203	М 204	Н 205	О 206	П 207
Р 208	С 209	Т 210	У 211	Ф 212	Х 213	Ц 214	Ч 215	Ш 216	Щ 217	Ъ 218	Ы 219	Ь 220	Э 221	Ю 222	Я 223
а 224	б 225	в 226	г 227	д 228	е 229	ж 230	з 231	и 232	й 233	к 234	л 235	м 236	н 237	о 238	п 239
р 240	с 241	т 242	у 243	ф 244	х 245	ц 246	ч 247	ш 248	щ 249	ъ 250	ы 251	ь 252	э 253	ю 254	я 255

1) #255 неразрывный пробел.

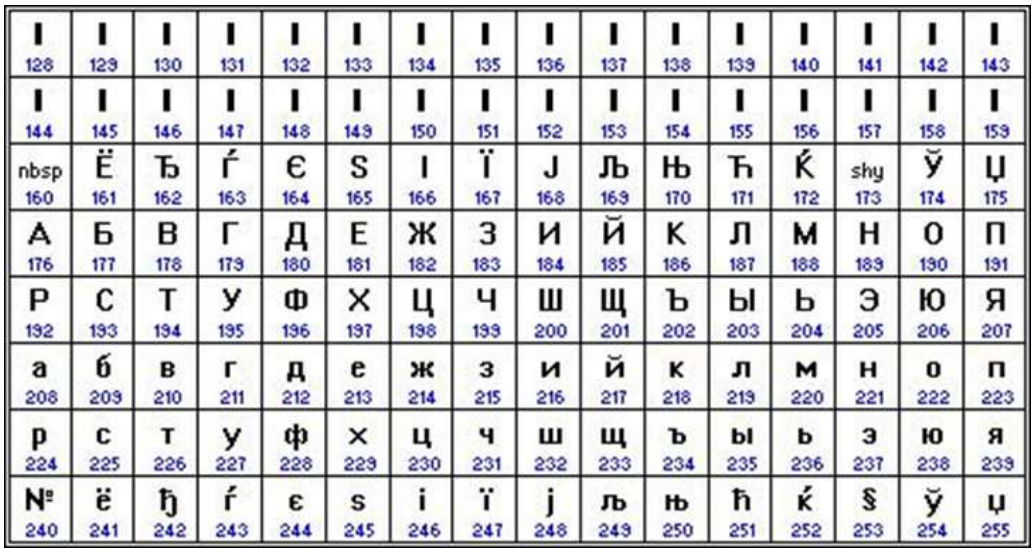
Mac – кодировка в ПК фирмы Apple, работающих под управлением операционной системы Mac OS [2,6,7].



А 128	Б 129	В 130	Г 131	Д 132	Е 133	Ж 134	З 135	И 136	Й 137	К 138	Л 139	М 140	Н 141	О 142	П 143
Р 144	С 145	Т 146	У 147	Ф 148	Х 149	Ц 150	Ч 151	Ш 152	Щ 153	Ъ 154	Ы 155	Ь 156	Э 157	Ю 158	Я 159
а 160	б 161	в 162	г 163	д 164	е 165	ж 166	з 167	и 168	й 169	к 170	л 171	м 172	н 173	о 174	п 175
▒ 176	▒ 177	▒ 178	─ 179	┤ 180	┤ 181	┤ 182	┐ 183	┐ 184	┐ 185	┐ 186	┐ 187	┘ 188	┘ 189	┘ 190	┐ 191
└ 192	┴ 193	┬ 194	├ 195	─ 196	┼ 197	├ 198	├ 199	└ 200	┌ 201	┴ 202	┬ 203	├ 204	═ 205	┼ 206	┴ 207
┴ 208	┬ 209	┬ 210	└ 211	└ 212	┌ 213	┌ 214	┼ 215	┼ 216	┘ 217	┌ 218	▉ 219	▄ 220	▌ 221	▐ 222	▀ 223
р 224	с 225	т 226	у 227	ф 228	х 229	ц 230	ч 231	ш 232	щ 233	ъ 234	ы 235	ь 236	э 237	ю 238	я 239
Ё 240	ё 241	Є 242	є 243	Ї 244	ї 245	Ў 246	ў 247	° 248	∙ 249	· 250	√ 251	№ 252	¤ 253	■ 254	nbsp 255

#202 неразрывный пробел.

ISO 8859-5 – Международная организация по стандартизации (International Standards Organization, ISO) утвердила в качестве стандарта для русского языка еще одну кодировку [2,6,7].



128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
nbsp 160	Ё 161	Ђ 162	Ѓ 163	Є 164	S 165	I 166	Ї 167	J 168	Љ 169	Њ 170	Ћ 171	Ќ 172	shy 173	Ў 174	Џ 175
А 176	Б 177	В 178	Г 179	Д 180	Е 181	Ж 182	З 183	И 184	Й 185	К 186	Л 187	М 188	Н 189	О 190	П 191
Р 192	С 193	Т 194	У 195	Ф 196	Х 197	Ц 198	Ч 199	Ш 200	Щ 201	Ъ 202	Ы 203	Ь 204	Э 205	Ю 206	Я 207
а 208	б 209	в 210	г 211	д 212	е 213	ж 214	з 215	и 216	й 217	к 218	л 219	м 220	н 221	о 222	п 223
р 224	с 225	т 226	у 227	ф 228	х 229	ц 230	ч 231	ш 232	щ 233	ъ 234	ы 235	ь 236	э 237	ю 238	я 239
№ 240	ё 241	ђ 242	ѓ 243	є 244	s 245	i 246	ї 247	j 248	љ 249	њ 250	ћ 251	ќ 252	§ 253	ў 254	џ 255

1) Коды 128-159 не используются;
2) #160 неразрывный пробел,
3) #173 мягкий перенос.

Задание. С учётом указанной таблицы кодировки записать следующую информацию о себе. «№ учебной группы, № студенческого билета, фамилия имя отчество».

Контрольные вопросы

1. Что такое код?
2. Что такое кодирование?
3. Что такое декодирование?

4. Прямой, обратный и дополнительный коды.
4. Что такое набор символов таблицы кодирования ?
5. Что такое кодовая страница?
6. Характеристика таблицы кодирования ASCII.
7. Характеристика таблицы кодирования ANSI.

Лабораторная работа 4. *Вычисление и преобразование логических функций. Структурные схемы логических функций.*

Цель: Получить теоретические знания и первичные навыки преобразования логических функций.

Основа работы логической схемы и устройств П.К - логика. В логике суждения – высказывание - повествовательное предложение - истинное или ложное.

2+8<5,

*5*5=25,*

*2*2=5.*

Квадрат есть параллелограмм.

Параллелограмм есть квадрат. -простые.

Сложные (с использованием связок и, или и частицы не.)

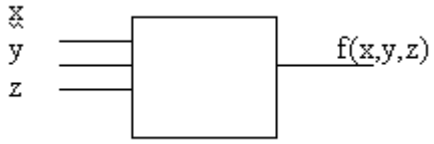
В М. Л. не рассматривается конкретное содержание высказывания, важно только истинно оно или ложно, поэтому высказывание можно представить некоторой ~ величиной, значение которой может быть 0 или 1.

0 - ложно, 1- истинно.

Для простоты записи высказывание обозначается латинскими буквами. У кошки 4 ноги A=1.

Москва расположена на 2 холмах B=0

Устройство П.К, выполняющее действие над двоичными числами, можно рассмотреть как некоторый функциональный преобразователь, причем входные числа это значения входных логических переменных, а выходное число значение логической функции, которое получено в результате выполнения определенных операций. Таким образом, этот преобразователь реализует некоторую логическую функцию.



Значения логических функций при разных сочетаниях значений входных переменных (наборах входных ~) - обычно задаются специальной таблицей - таблицей истинности.

Количество наборов входных ~ (Q) определяется выражением [2,6,7]: (Q)=2n – где n количество входных ~. таблица истинности может иметь вид

X	Y	Z	F (x, y, z)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

В алгебре высказываний любую логическую функцию можно выразить через основные логические операции, записать ее в виде логического выражения и упростить ее, применяя законы логики и свойства логических операций. По формуле логической функции легко рассчитать ее таблицу истинности. Необходимо только учитывать порядок выполнения логических операций (приоритет) и скобки. Операции в логическом выражении выполняются слева направо с учетом скобок.

Приоритет логических операций:

- ИНВЕРСИЯ,
- КОНЪЮНКЦИЯ,
- ДИЗЪЮНКЦИЯ

КОНЪЮНКЦИЯ

Конъюнкция: соответствует союзу: «и», обозначается знаком^, обозначает логическое умножение.

Конъюнкция двух логических ~ истинна тогда и только тогда , когда оба высказываний истинны. Можно обобщить для любого количества переменных A^B^C = 1 если A=1, B=1, C=1.

A	B	A^B
1	1	1
1	0	0
0	1	0
0	0	0

ДИЗЪЮНКЦИЯ

Логическая операция соответствует союзу ИЛИ, обозначается знаком v, иначе называется ЛОГИЧЕСКОЕ СЛОЖЕНИЕ.

Дизъюнкция двух логических переменных ложна тогда и галька тогда, когда оба высказывавия ложны.

Это определение можно обобщить для любого количества логических переменных, объединенных дизъюнкцией.

A v B v C = 0, только если A = O, B = O, C - 0.

Таблица истинности дизъюнкции имеет следующий вид:

A	B	A v B
1	1	1
1	0	1
0	1	1
0	0	0

ИНВЕРСИЯ

Логическая операция соответствует частице не, обозначается ¬ или ¯ и является логическим отрицанием.

Инверсия логической переменной истинна, если переменная ложна и наоборот: инверсия ложна, если переменная истинна.

A	¬A
1	0
0	1

высказывания у которых таблицы истинности совпадают называются равносильными.

ИМПЛИКАЦИЯ и ЭКВИВАЛЕНТНОСТЬ

Импликация «если А, то В», обозначается А → В

A	B	A → B
0	0	1
0	1	1
1	0	0
1	1	1

Эквиваленткость «А тогда В и только тогда», обозначается А ~ В

A	B	A~ B
0	0	1
0	1	0
1	0	0
1	1	1

При вычислении значения логического выражения (формулы) логические операции вычисляются в определенном порядке, согласно их приоритету:

1. инверсия,
2. конъюнкция,
3. дизъюнкция,
4. импликация и эквивалентность.

Операции одного приоритета выполняются слева направо. Для изменения порядка действий используются скобки.

Например: дана формула $A \vee B \to C \wedge D \Leftrightarrow \neg A$

Порядок вычисления:

$\neg A$ - инверсия.

$C \wedge D$ - конъюнкция.

$A \vee B$ - дизъюнкция.

$A \vee B \to C \wedge D$ - импликация.

$A \vee B \to C \wedge D \Leftrightarrow \neg A$ - эквивалентность.

Задание:

1. Определить таблицу истинности логической функции: F (A, B, C) = A v (C ^ B) , Определяем количество строк в таблице: Q = 2^3 = 8
2. Определяем количество логических операций (3) и последовательность их выполнения
3. Определяем количество столбцов: три переменные + три логические операции = 6.

A	**B**	**C**	**C**	**C ^ B**	A v (C ^ B)

Пример. Постройте таблицу истинности высказываний «Саша не выполнил задание» и «Саша получил выговор» .Решение:

Саша не выполнил задание	Саша получил выговор	Результат
Истина	Истина	Истина
Истина	Ложь	Ложь
Ложь	Истина	Ложь
ложь	Ложь	Ложь

Пример. В приведенном ниже высказывании выделите простые. Запишите сложные высказывания в виде формулы, приведите таблицы истинности.

Пришла весна, и грачи прилетели.

F=A^B.

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

Пример. Определите истинность составного высказывания: ($\overline{A}$ & $\overline{B}$) & (*C* Ú*D*), состоящего из простых высказываний:

A = {Принтер – устройство вывода информации},

B = {Процессор – устройство хранения информации},

C = {Монитор – устройство вывода информации},

D = {Клавиатура – устройство обработки информации}.

Сначала на основании знания устройства компьютера устанавливаем истинность простых высказываний: *A* = 1, *B* = 0, C = 1, *D* = 0.

Определим теперь истинность составного высказывания, используя таблицы истинности логических операций:

(&) &(1Ú 0) = (0&1) & (1Ú 0) = 0

Составное высказывание ложно.

Задание. *Даны два простых высказывания:*

A= “Щука – рыба”;

B=“Ворона – певчая птица”.

Составьте из них все возможные составные (сложные) высказывания и определите их истинность.

А. Среди следующих высказываний укажите составные, выделите в них простые, обозначьте их каждое из них буквой. Запишите с помощью логических операций каждое составное высказывание.

1. Число 456 трехзначное и четное.
2. Неверно, что Солнце движется вокруг Земли.
3. Число делится на 9 тогда и только тогда, когда сумма его цифр делится на 9.
4. Луна – спутник Земли.
5. На уроке химии ученики выполняли лабораторную работу, и результаты исследований записывали в тетрадь.

6. Если число оканчивается на 0, то оно делится на 10.
7. Чтобы погода была солнечной, достаточно, чтобы не было ни ветра, ни дождя.
8. Если у меня будет свободное время и не будет дождя, тоя не буду писать сочинения, а пойду на дискотеку.
9. Без Вас хочу сказать Вам много
При Вас я слушать Вас хочу.
10. Если человек с детства и юности своей не давал нервам властвовать над собой, то они не привыкнут раздражаться и будут ему послушны.

Б. Постройте отрицания следующих высказываний.
1. На улице сухо.
2. Сегодня выходной день.
3. Ваня не был готов сегодня к урокам.
4. Неверно, что число 3 не является делителем числа 198.
5. Некоторые млекопитающие не живут на суше.
6. Неверно, что число 17 – простое.

В. Из каждых трех выберите пару высказываний, являющихся отрицаниями друг друга.
1. “Луна – спутник Земли”, “Неверно, что Луна спутник Земли”, “Неверно, что Луна не является спутником Земли”;
2. “2007 < 2008”, “2007 > 2008”, “2007 ? 2008”;
3. “Прямая *a* перпендикулярна прямой *c*”; “Прямая *a* не параллельна прямой *c*”; “Прямая *a* не пересекается с прямой *c*”.

Г. По данным формам сложных высказываний запишите высказывания на русском языке.
1. $E = A \to B$
2. $E = A \wedge B$
3. $E = \neg A \wedge \neg B$
4. $E = A \leftrightarrow (B \vee C \vee D)$
5. $E = B \wedge \neg A \to C$

Д. Найдите значения логических выражений:
1. $(0 \vee 1) \vee (1 \vee 1)$
2. $((1 \vee 1) \vee 0) \vee 1$
3. $(1 \to 1) \vee (0 \leftrightarrow 0)$
4. $(0 \wedge 1) \wedge 1$
5. $1 \wedge (1 \wedge 1) \wedge 1$
6. $((0 \vee 1) \wedge (1 \wedge 1)) \wedge (0 \vee 1)$
7. $((1 \wedge 0) \to (0 \vee 1)) \vee 1)$
8. $((1 \wedge \neg 1) \vee 0) \wedge (0 \vee \neg 1)$
9. $((0 \wedge 0) \vee 0) \leftrightarrow (1 \vee 1)$
10. $\neg 1 \wedge (1 \vee 1) \vee (\neg 0 \wedge 1)$
11. $(\neg 1 \to 1) \wedge (1 \vee \neg 1) \leftrightarrow (\neg 1 \vee 0)$

Ж. Даны два высказывания: А = “2 х 2 = 4”, В = “2 х 2 = 5”. Очевидно, что А=1, В=0. Какие из высказываний истинны?

а) $\neg A$

б) $\neg B$

в) А

г) $A \vee B$

д)

е) $A \Leftrightarrow B$

З. Даны простые высказывания: А= {15>13}, В={4=5}, С= {7<4}. Определите истинность составных высказываний:

$(A \vee B) \wedge C \rightarrow (A \wedge C) \vee (B \wedge C)$

$(A \wedge B) \vee C \Leftrightarrow (A \vee C) \wedge (A \wedge B)$

И. При каких значениях числа Х логическое выражение *не ((X>15) или (X<-5))* примет значение:

1. ложь,
2. истинна.

К. Какие из высказываний А, В должны быть истинны и какие ложны, чтобы было ложное высказывание $(A \wedge B) \Leftrightarrow 1$?

	Высказывания, записанные на естественном языке	Высказывания, записанные на языке алгебры логики
1.	Не А: Неверно, что А; А не имеет места	$\overline{A}$
2.	А и В Как А, так и В; Не только А, но и В; А вместе с В; А, несмотря на В; А в то время, как В	$A \wedge B$
3.	А, но не В; Не В, а А	$A \wedge \overline{B}$
4.	А или В; А или В, или оба	$A \vee B$
5.	Либо А, либо В	$A \wedge \overline{B} \vee \overline{A} \wedge B$
6.	Либо А, либо В и С	$A \wedge \overline{B} \wedge \overline{C} \vee \overline{A} \wedge B \wedge C$
7.	Либо А и В, либо С и Д	$A \wedge B \wedge \overline{C} \wedge \overline{D} \vee \overline{A} \wedge \overline{B} \wedge C \wedge D$
8.	Если А, то В; А только, если В; А только, когда В; А достаточно для В;	$A \rightarrow B$

	A только при условии, что B; B необходимо для A; A значит B; Для A необходимо B; Из A следует B; B тогда, когда A	
9.	A эквивалентно B; A тогда и только тогда, когда B; A если и только если B; A необходимо и достаточно для B	$A \Leftrightarrow B$ $A \equiv B$

Контрольные вопросы:

1. Что такое функция алгебры логики?
2. Что такое область определения функции алгебры логики?
3. Что такое область значений функции?
4. Что такое таблица истинности функции?
5. Одноместные функции. Определение. Таблицы истинности.
6. Двуместные функции. Определение. Таблицы истинности.

Лабораторная работа 5. *Работа в среде MS Windows (XP, 7), Linux Ubuntu. Исследование работы встроенных и внешних устройств ПК.*

Цель работы. Приобретение навыков работы в операционной системе.

Учебные вопросы.

1. Регистрация пользователя в системе.
2. Изучение папок профиля пользователя и библиотек.
3. Настройка компонентов рабочего стола (меню кнопки Пуск, значки панели задач, ярлыки Рабочего стола, гаджеты).
4. Настройка и использование Проводника. Операции с папками и файлами.
5. Способы запуска программ и открытия файлов данных.

Краткое изложение учебного материала *1 Регистрация пользователя в системе.*

В компьютерном классе преподаватель сообщает студентам данные **учётной записи**, которую они должны использовать при работе на компьютерах (**имя пользователя и пароль**). Обычно используется учетная запись, зарегистрированная на контроллере домена университета.

В начале работы необходимо правильно задать эти данные на экране регистрации, показанном на рисунке 1.1 [1,2].

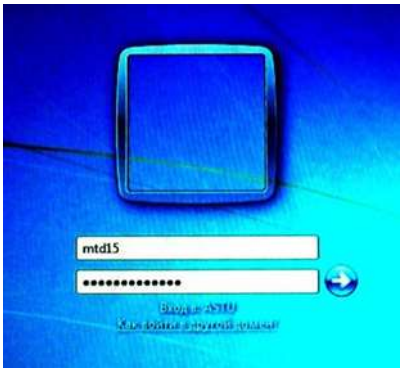


Рисунок 1.1 – Экран регистрации пользователя

При этом следует учитывать, что в имени пользователя нельзя ставить лишние пробелы в конце слова, а в пароле – необходимо правильно указывать прописные и строчные буквы (пароль чувствителен к регистру букв!) [1,2].

2 Изучение папок профиля пользователя и библиотек

При первом входе пользователя в систему создается **локальный профиль** учётной записи (Ваши папки на том компьютере, где Вы регистрируетесь). На правой панели кнопки **Пуск** верхняя позиция – *Открытие персональной папки* (рисунок 1.2). Именно в этой папке и находятся все папки профиля пользователя.

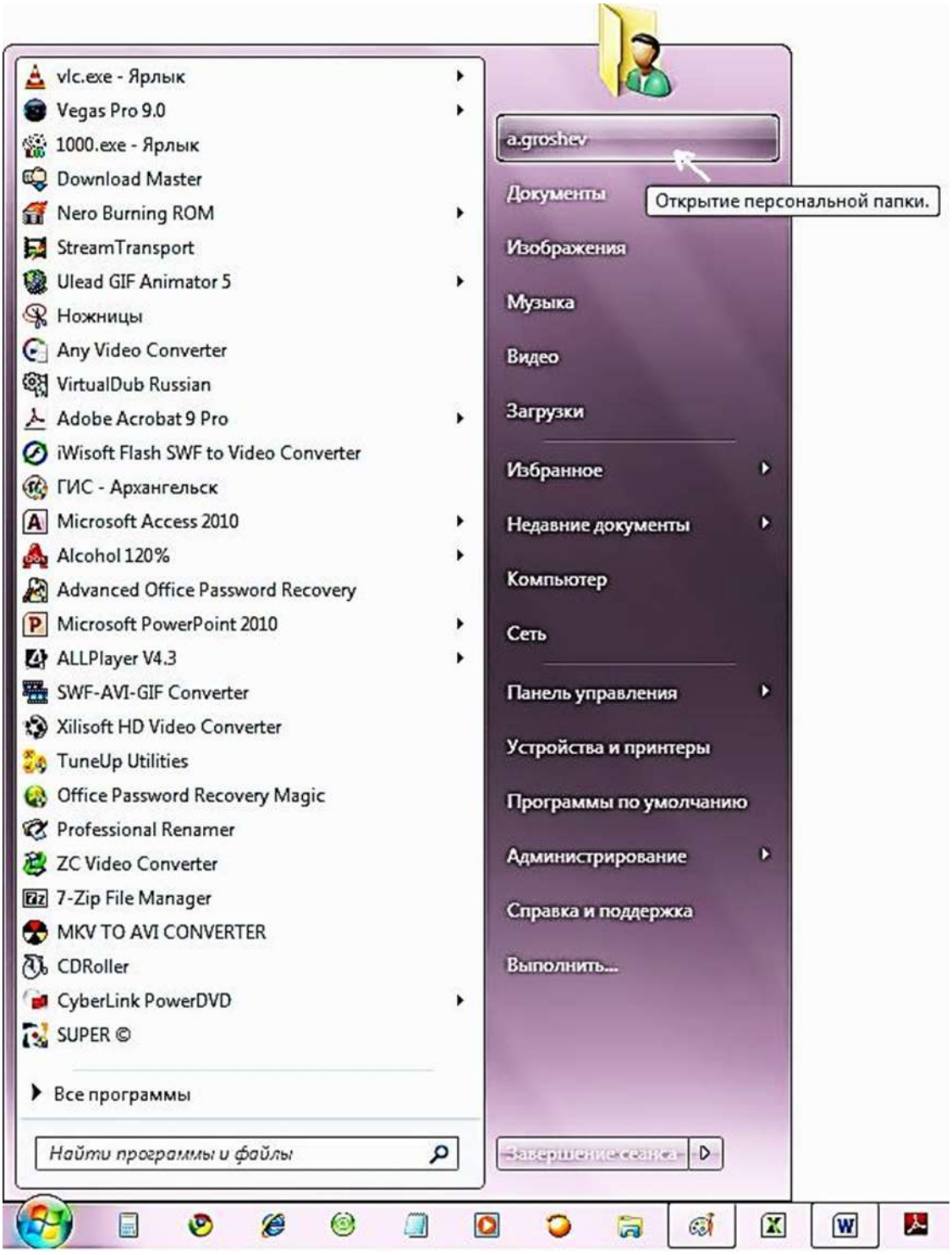


Рисунок 1.2 – Меню кнопки Пуск

Папки профилей находятся на диске, где установлена операционная система, в папке *Пользователи* (рисунок 1.3) [1,2].

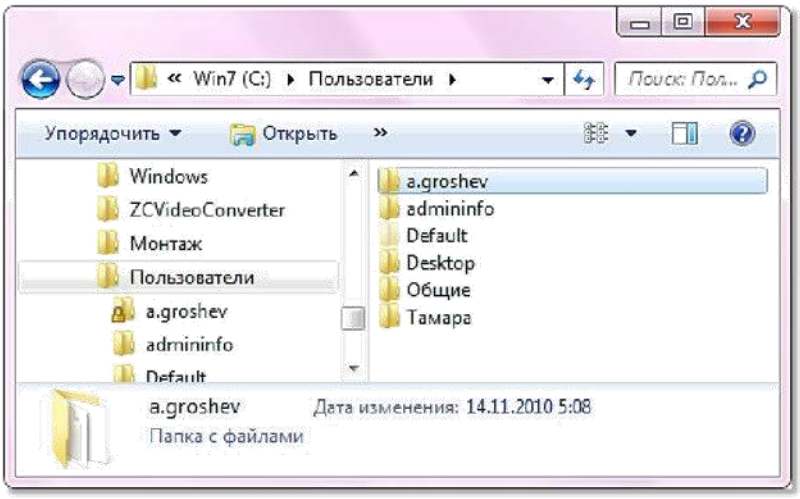


Рисунок 1.3 – Папка профилей пользователей в Проводнике

В папке каждого пользователя находятся вложенные папки *Загрузки, Изображения, Контакты, Мои видеозаписи, Мои документы, Моя музыка, Поиски, Рабочий стол, Ссылки* (рисунок 1.4) и несколько скрытых системных папок.

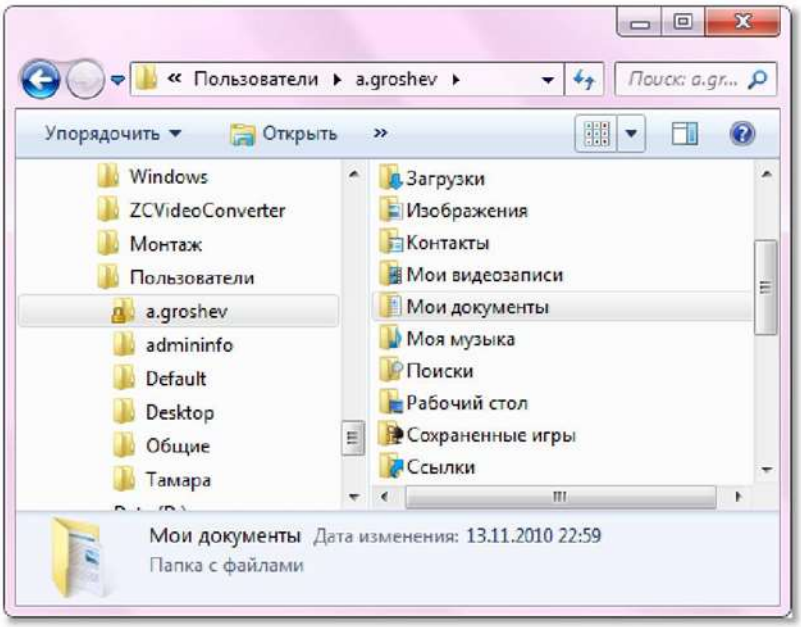


Рисунок 1.4 – Папки в профиле пользователя

Следует знать, что в меню кнопки ***Пуск*** присутствуют пункты *Доку-менты, Изображения, Музыка, Видео, Загрузки* (см. ранее рисунок 1.2) [1,2], которые открывают библиотеки пользователя с соответствующими названиями. В состав библиотек могут входить различные папки, список которых пользователь может изменять. Например, в состав библиотеки **Документы** входят папки **Мои документы** и **Общие документы**.

Для сохранения файлов пользователя, которые будут созданы во время выполнения лабораторных работ, следует внутри папки **Мои документы** создать папку с именем <*фамилия и инициалы пользователя*>.

3 Настройка компонентов рабочего стола

Для настройки Рабочего стола [1, стр. 153 – 156] следует использо-вать контекстное меню, которое появляется при щелчке правой кнопкой мыши на экране компьютера (рисунок 1.5) [1,2].

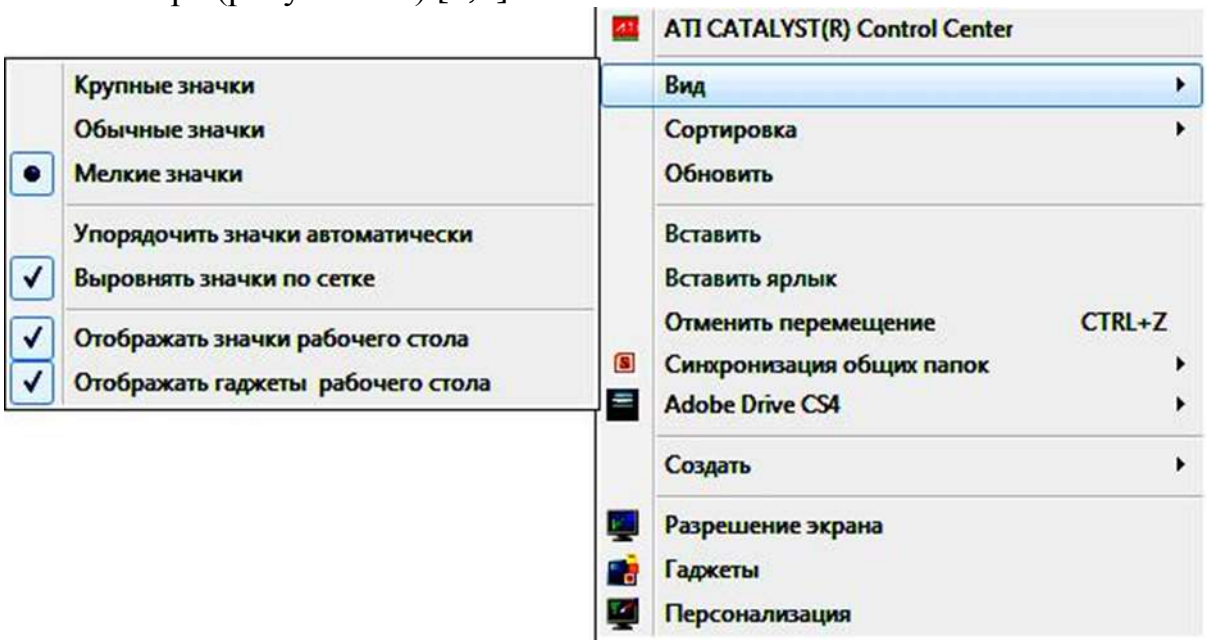


Рисунок 1.5 – Контекстное меню настройки Рабочего стола

Как видно из рисунка 1.5, меню позволяет настроить вид значков (ярлыков) на **Рабочем столе**, разрешение экрана (для жидкокристаллических мониторов следует задавать их штатное разрешение), используемые гаджеты (миниприложения) и обратиться к диалоговому окну **Персонализация**, содержащему настройки тем **Рабочего стола** и других параметров (рисунок 1.6).

Для настройки **Панели задач**, меню кнопки **Пуск** и **Панелей инструментов** (которые могут располагаться на **Панели задач**) следует использовать пункт ***Свойства*** контекстного меню этих элементов.

В настройках меню **Пуск** можно задать наличие или отсутствие пунктов меню и их вид (подменю или ссылка). По умолчанию некоторые пункты меню отключены (*Администрирование, Выполнить* и пр.), при необходимости можно задать их использование (рисунок 1.7). В этом окне можно настроить также количество строк в списке недавних программ (левая панель при нажатии на кнопку **Пуск**) и в списке **Недавние документы** (см. ранее на рисунке 1.2) [1,2].

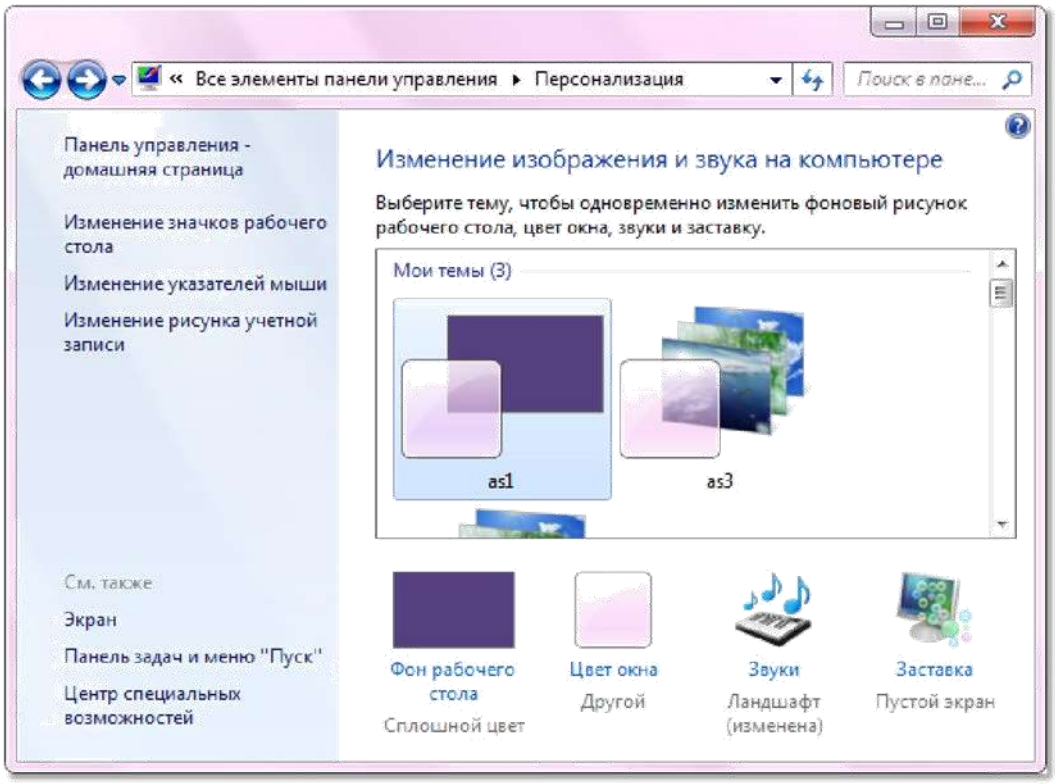


Рисунок 1.6 – Окно **Персонализация** для настройки **Рабочего стола**

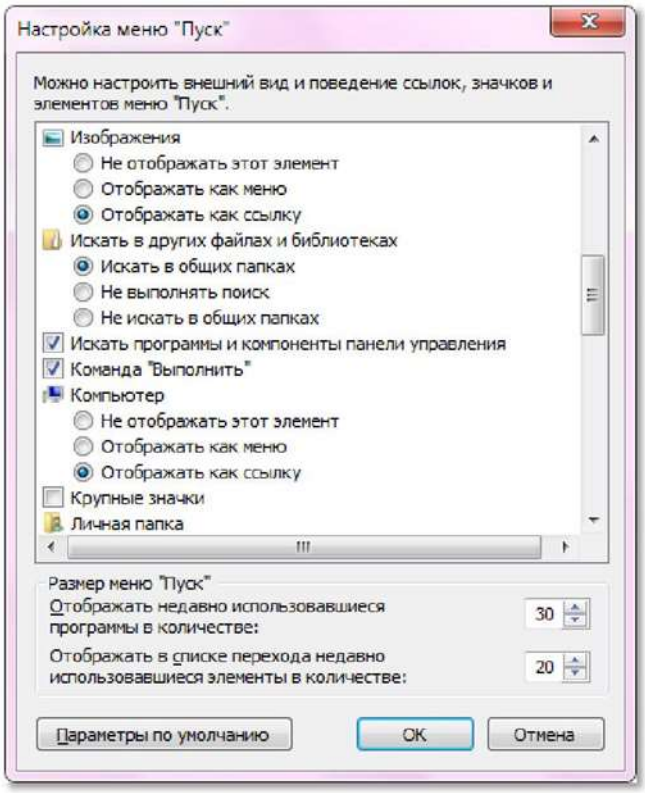
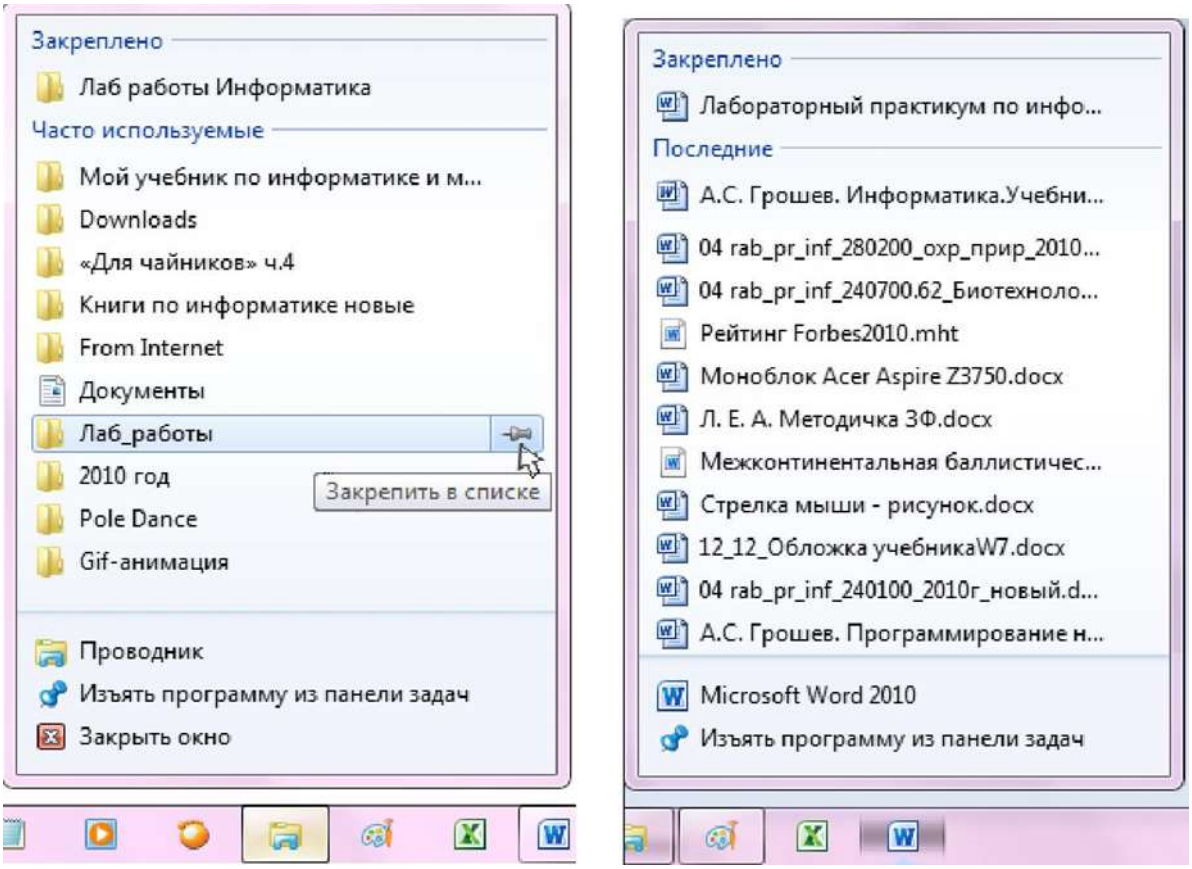


Рисунок 1.7 – Окно настройки меню **Пуск**

Значки на **Панели задач** появляются для запущенных программ. Для тех программ, которыми Вы часто пользуетесь (в том числе и для **Провод**

ника), полезно закрепить значок на **Панели задач** (в контекстном меню для значка программы команда *Закрепить программу в панели задач*). Это позволит не только быстро запускать нужные программы, но и сразу выбирать нужные файлы и папки для **Проводника** в **Списке переходов** (Jump List – список последних документов, например, для Word, или часто используемых папок, например , для **Проводника**), который появляется при щелчке правой кнопкой мыши на значке программы (рисунок 1.8). Необ-ходимые строки списка можно закрепить как постоянные в **Списке пере-ходов**. Видеодемонстрацию процесса использования **Списков переходов** можно посмотреть на сайте Microsoft (*http://windows.microsoft.com/ ru-RU/windows7/Using-Jump-Lists-to-open-programs-and-items*) [1,2].



1.8 – Списки переходов для Проводника и для Word

Ярлыки программ (значки, содержащие ссылку на объект) появляются на Рабочем столе при инсталляции многих программ. Первоначально присутствует один ярлык – Корзина, в окне Персонализация (см. ранее рисунок 1.6) есть ссылка *Изменение значков рабочего стола*, которая позволяет разместить на Рабочем столе дополнительные ярлыки (с особыми свойствами!): **Компьютер**, **Файлы пользователя**, **Сеть**, **Панель управления** [1,2].

Гаджеты **Рабочего стола** используются обычно для показа некоторой информации: часы и календарь, загрузка процессоров и оперативной памя

ти, мониторинг работы сети, показ погоды, теле- и радиоканалов Интернета и пр. Среди них есть такие, которые работают только при наличии подключения к Интернету (погода, курсы валют, Интернет-телевидение и пр.). Новые гаджеты, можно скачать и установить с сайта Microsoft [1,2].

4 Настройка и использование ***Проводника****. Операции с папками и файлами.*

Проводник или **Windows Explorer** – стандартная программа, которая позволяет работать с папками и файлами Вашего компьютера и компьюте-ров локальной сети. Вид окна **Проводника** показан ранее на рисунках 1.3 и 1.4, однако возможности по его настройке очень большие.

Наиболее часто **Проводник** используется для нахождения нужных файлов и папок и выполнения с ними различных операций: открытия, со-здания новых файлов и папок, их копирования, перемещения, удаления и других операция. Список возможных операций для файлов и папок можно увидеть в контекстном меню (рисунок 1.9). Состав команд контекстного меню изменяется в зависимости от типа выбранных файлов и наличия прикладных программ, которые вписывают свои команды в контекстное меню, например, команды для программ ESET NOD32 и WinRar на рисунке 1.10. На том же рисунке показано подменю для пункта ***Отправить***, кото-рое содержит, в частности, команду архивации средствами Windows – ***Сжатая ZIP-папка*** [1,2].

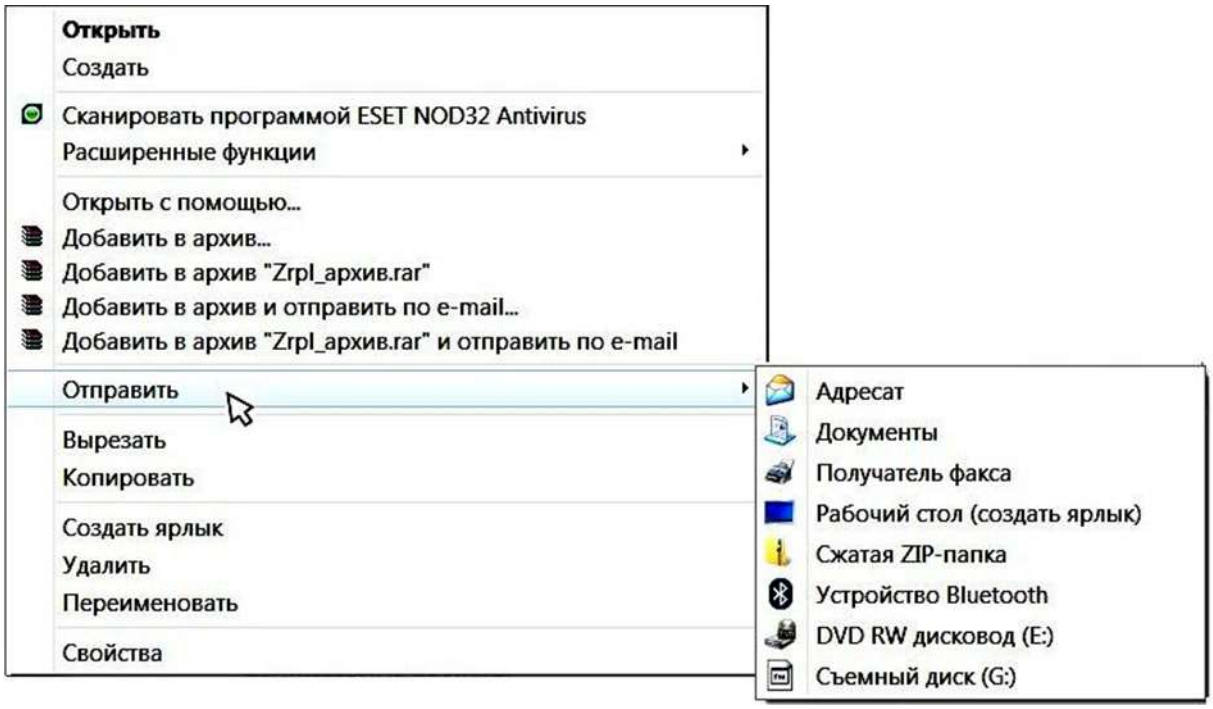


Рисунок 1.9 – Контекстное меню для файлов Проводника

Команду **Создать** контекстного меню (рисунок 1.10) удобно использовать для создания новых файлов определенного типа в выбранной папке. Для создания папок можно использовать кнопку **Новая папка** на панели команд **Проводника** [1,2].

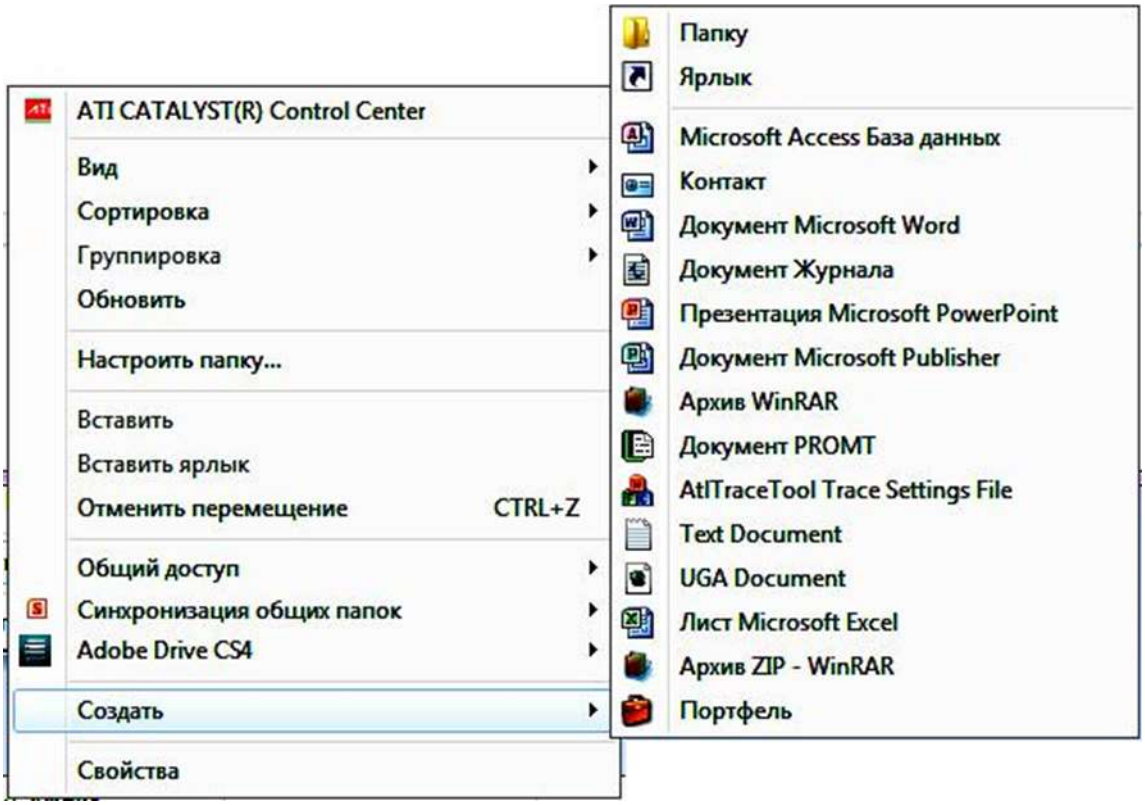


Рисунок 1.10 – Контекстное меню **Создать Проводника**

Для копирования или перемещения файлов из одной папки в другую, можно использовать как контекстное меню **Проводника** (команды Копировать и Вставить или Вырезать и Вставить), горячие клавиши CTRL+C и CTRL+V или CTRL+X и CTRL+V (другие горячие клавиши Windows см. в Приложении 1).

Другой способ, который часто используется – перетаскивание мышкой из одной папки в другую. При перетаскивании с нажатой левой кноп-кой мыши у курсора появляется сообщение о виде операции (Копировать в …, Переместить в … или Создать ссылку в …), как показано на рисунке 1.11. Вид операции меняется при нажатии кнопки CTRL или CTRL+SHIFT для последней операции. При перетаскивании с нажатой правой кнопкой мыши после выбора места, куда выполняется эта операция и отпускания кнопки мыши, появляется контекстное меню для выбора одной из трех названных выше операций и команда Отмена.

Выбор группы файлов для выполнения операций копирования, перемещения, архивации, удаления и пр. можно выполнить несколькими способами [1,2]:

1) выделение с нажатой левой кнопкой мыши; 11.

2) выделение первого и последнего файлов щелчком мыши с нажа-той кнопкой SHIFT;

3) выделение любых файлов в **Проводнике** при нажатой кнопке CTRL.

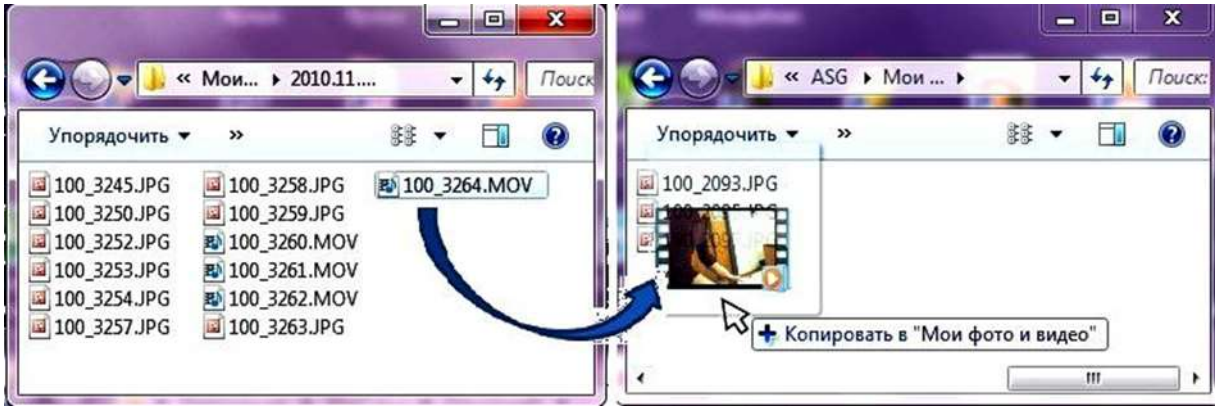


Рисунок 1.11 – Копирование файлов перетаскиванием мышкой

5 Способы запуска программ и открытия файлов данных

Запуск программ (приложений Windows) – операция, которую пользователь выполняет множество раз при работе на компьютере. Установленных в системе программ для работы с различными видами информации (с текстовой и числовой, с базами данных и информационными системами, с графической и конструкторской, с видео и Интернетом…) может быть несколько сотен. Поэтому нужно уметь пользоваться способами, которые позволяют наиболее быстро найти и запустить необходимую программу или открыть файл данных в программе по умолчанию для данного типа информации.

Способов запуска программ в операционной системе Windows достаточно много. Для этого можно использовать различные элементы **Рабочего стола**, о которых говорилось в предыдущем разделе [1,2]:

о Меню кнопки **Пуск** (см. ранее рисунок 1.2) содержит:

• на левой панели список наиболее часто используемых программ, для многих из них есть список переходов со ссылками на последние использованные файлы данных, и внизу кнопку **Все программы**, которая покажет на этой панели полный список установленных программ;

• на правой панели список библиотек, Избранное, Недавние документы, команда **Выполнить**, которые также можно использовать для поиска данных и запуска программ;

о Ярлыки программ на **Рабочем столе**;

о Ярлыки на **Панели задач**, для многих из них могут существовать **Списки переходов,**

а также файлы исполняемых программ или данных в Проводнике.

Обобщая, можно говорить о следующих основных способах [1,2]:

1) Запуск программы с использованием ссылок на нее (в меню пуск, в ярлыках), в том числе с выбором файлов данных в **Списках переходов** ярлыков программ (см. ранее рисунок 1.8).

2) Открытие файла данных с использованием программы, задан-ной по умолчанию для этого типа данных, или с выбором про-граммы с использованием команды Открыть с помощью (рисунок 1.12).

3) Запуск программы с использованием найденного исполняемого файла или с написанием его имени в окне команды Выполнить.

Для 3-го варианта можно использовать, например, имена программ Notepad, WordPad, Calc, Mspaint, Explorer, Iexplore, Winword, Excel и пр.

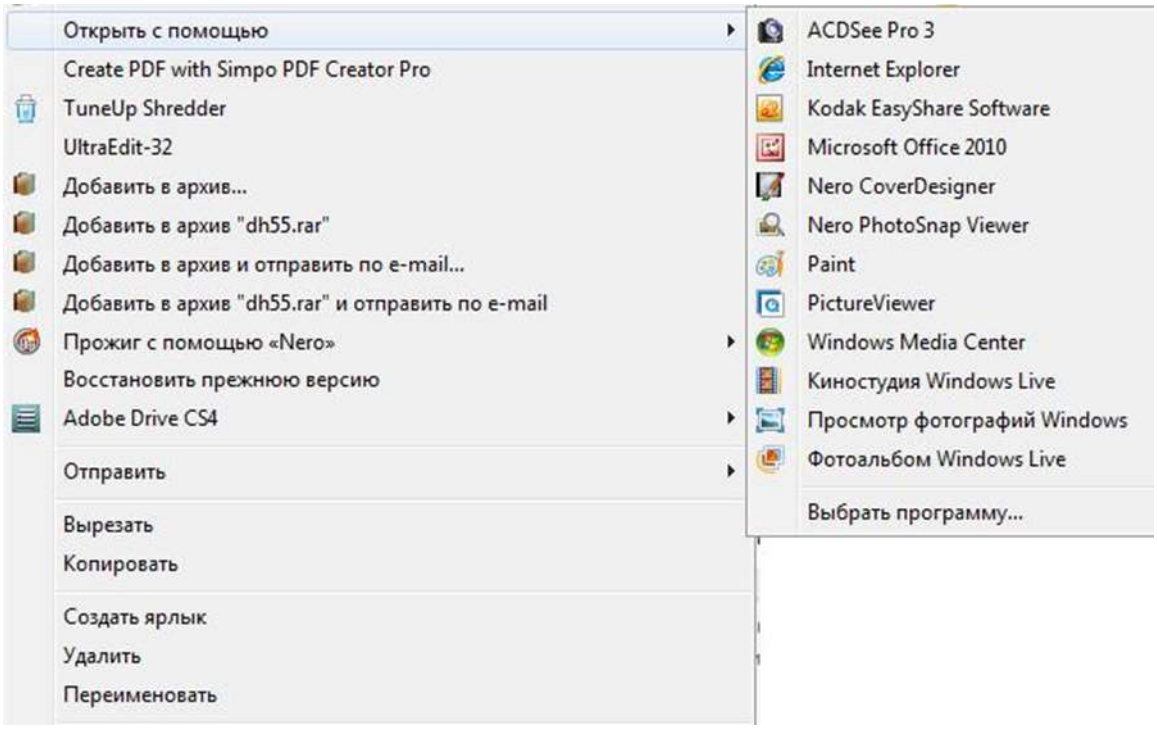


Рисунок 1.12 – Использование команды Открыть с помощью.

Войти в систему с использованием имени пользователя и пароля, выданных преподавателем [1,2].

1) Изучить состав папок профиля пользователя, скопировать с экрана окно этой папки и вставить в отчет о работе.

2) Закрепить на Панели задач значок Проводника.

3) Показать на Рабочем столе стандартные ярлыки Компьютер, Файлы пользователя и Сеть.

4) Установить гаджеты Часы, Календарь, Индикатор ЦП, прочие уда-лить с Рабочего стола.

5) Создать в папке Мои документы профиля пользователя папку с именем – *Ваша фамилия и инициалы* (например, Иванов П.С.).

6) В созданной папке создать новую папку с именем Архив.

7) Создать в своей папке с использованием контекстного меню пустые файлы следующих типов: текстовый документ, точечный рисунок, документ Word, лист Excel, база данных Access. Имя всех файлов – *Ваша фамилия*.

8) Выполнить копирование всех файлов разными способами в папку *Архив*.

9) Выполнить архивацию файлов в папке Архив командой контекстного меню Отправить → Сжатая Zip-папка.

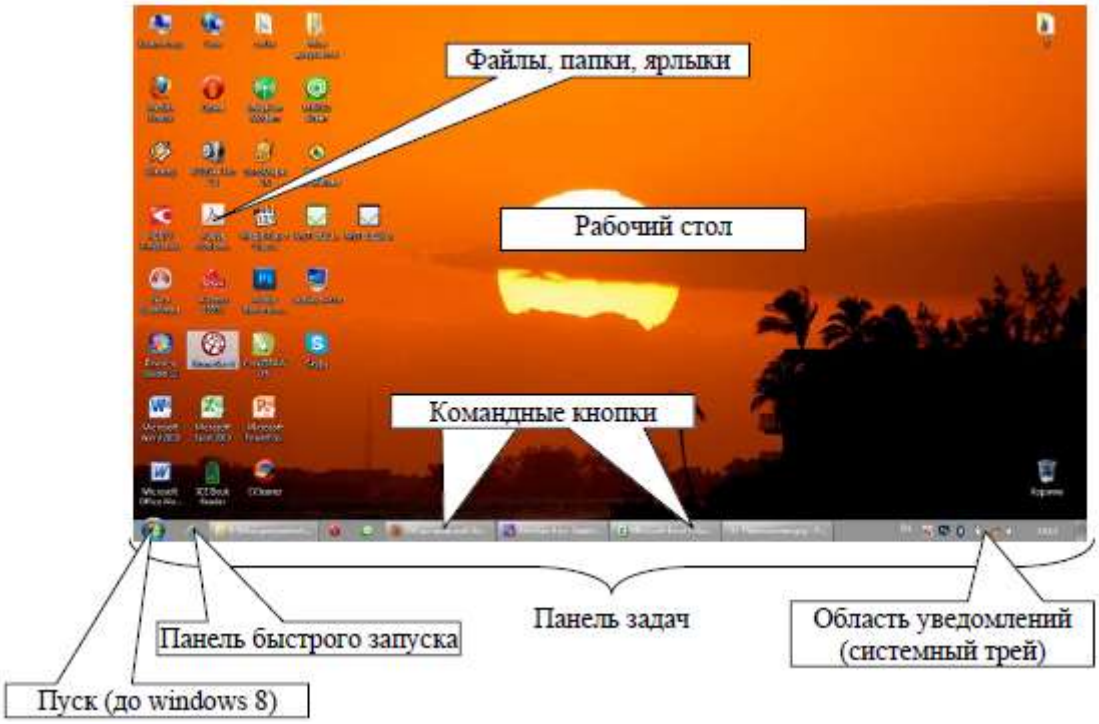
10) Переименовать файл документа Word в *РГР-<Ваша Фамилия>*, базы данных в *БД-<Ваша Фамилия>*.

Контрольные вопросы:

1. Что такое архитектура фон Неймана?
2. Что такое Гарвардская архитектура?
3. Что такое центральный процессор?
4. Что такое оперативная память?
5. Что такое запоминающее устройство с произвольным доступом?
6. Что такое USB-флеш-накопитель?
7. Общие характеристики ОС Windows XP.
8. Общие характеристики ОС Windows 7 .

Лабораторная работа 6. *Основные приемы работы в различных ОС. Использование системного и прикладного программного обеспечения.*

Элементы графического интерфейса ОС Windows.



Интерфейс – это способ общения пользователя с компьютером. То есть это совокупность правил, элементов и команд посредством которых человек даёт задание компьютеру и получает ответную реакцию. До 1991 года не было привычной нам графической оболочки. Все указания компьютеру задавались с помощью консольных команд. Сейчас с появлением графического интерфейса управление компьютером стало значительно проще.

Рассмотрим основные элементы этого интерфейса.

Самое большое пространство экрана после загрузки операционной системы (ОС) занимает

«**Рабочий стол**».

На нём могут находиться **файлы, папки и ярлыки**. Различают папки двух видов – **системные** (они предусмотрены изначально разработчиками и создаются автоматически) и **пользовательские** (те, которые создаём мы с вами).

Ярлык – это ссылка на тот или иной объект, то есть на файл или папку. Его можно отличить от фала или папки по наличию **стрелки** в левом нижнем углу. Однако от этой стрелки можно при желании избавиться. Тогда для идентификации объекта можно обратиться к его свойствам, вызвать которые можно нажав на него правой кнопкой мыши и выбрав пункт контекстного меню «свойства».

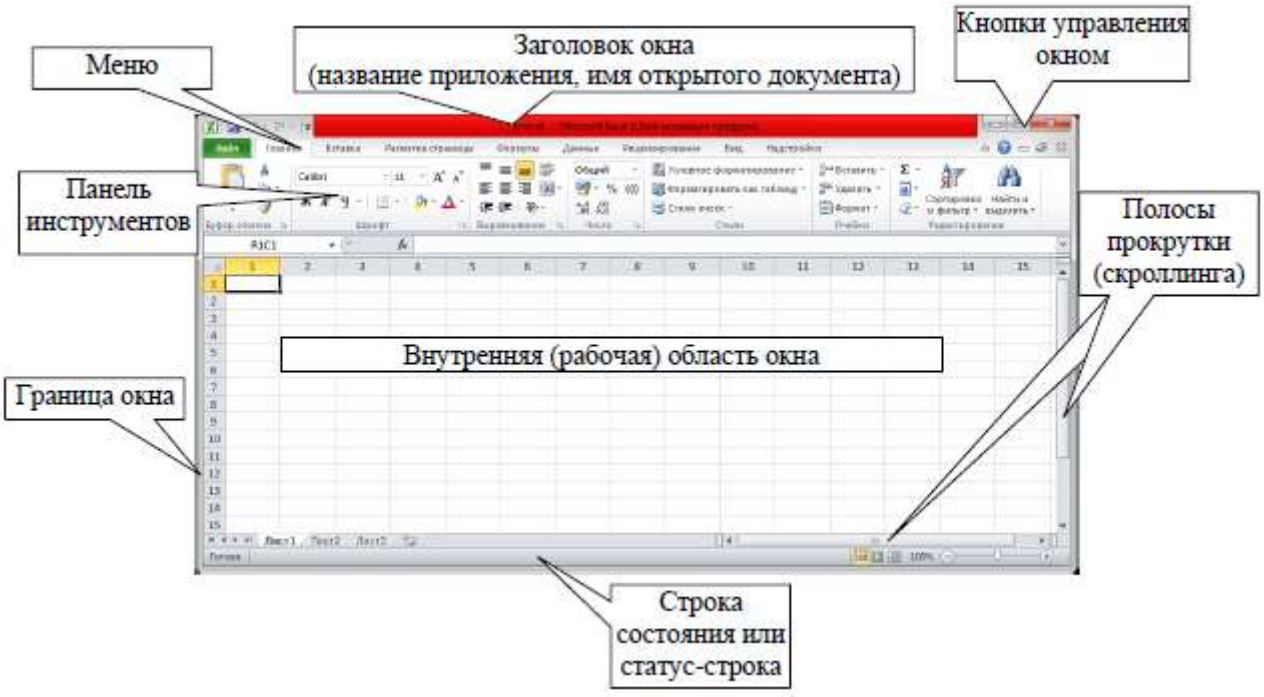
В нижней части экрана находится «**Панель задач**». На ней располагаются некоторые важные элементы. В частности слева до 8 версии windows есть кнопка «**Пуск**», при нажатии на которую вызывается соответствующее меню. Правее

находится «**Панель быстрого запуска**». Её при желании можно отключить (то есть это не обязательный элемент интерфейса) . Интерфейс, придуманный компанией Microsoft, называется **оконным** (отсюда и название семейства операционных систем) . Все программы, которые мы запускаем, открываются в **окнах**. На панели задач они отображаются в виде **командных кнопок.** И, как вы прекрасно знаете, для переключения между открытыми окнами достаточно щёлкнуть по соответствующей командной кнопке [1,2].

Наконец, в правом углу панели задач находится так называемый **системный трей** или **область уведомлений,** в которой помимо индикатора раскладки клавиатуры, времени и прочих уведомлений находятся значки **резидентных программ**. Это те программы, которые стартуют после загрузки операционной системы и постоянно находятся в оперативной памяти компьютера, выполняя те или иные задачи. Самой распространённой резидентной программой, безусловно, является **антивирус**, который осуществляет контроль запуска программ и операций с файловой системой.

Теперь разберём элементы окна.

Элементы окна ОС Windows



Некоторые из элементов могут отсутствовать. В качестве примера можно привести окно браузера, которое развёрнуто на весь экран. То есть из всех элементов присутствует только внутренняя область окна и, возможно, полоса прокрутки.

Файловая система Windows

Файл – однотипный набор данных, которому присвоено имя и хранящийся во внешней памяти. К внешней памяти относятся устройства: жёсткие диски, флеш-накопители, компакт-диски.

Имя файла состоит из названия и расширения. Название файла может содержать до 255 символов и может состоять из любых символов, кроме служебных (таких как: / \ * : ? « » < > |).

Расширение файла следует после названия через последнюю точку и указывает на его тип. Обычно состоит из 3-4 символов. Если конкретный тип файла зарегистрирован в системе, то двойной клик мыши по такому файлу приведёт к запуску приложения, с которым он связан, а затем в этом приложении откроется сам файл. По умолчанию расширения зарегистрированных типов файлов в ОС Windows 7(8) скрыто, а тип таких файлов можно определить по соответствующему значку связанного с ним приложения [1,2].

Все файлы можно разделить на большие группы по типу хранящихся в них данных.

текстовые файлы: *.txt, *.doc, *.docx, *.rtf

графические файлы: *.jpg, *.gif, *.png, *.pdf

файлы видео: *.avi, *.mp4, *.mkv

звуковые файлы: *.mp3, *.wav

Примеры расширений файлов и приложений, в которых они создаются и открываются:

pdf – Adobe Acrobat Reader (Adobe Acrobat Pro); doc (docx) – MS Word;

xls (xlsx) – MS Excel;

ppt (pptx) – MS Power Point;

txt – Блокнот.

В системе могут быть несколько приложений, открывающих один тип файла. При желании пользователь может поменять программу, в которой конкретный тип файла открывается по умолчанию.

Существуют сотни различных типов файлов. Время от времени встречаются файлы, которые трудно идентифицировать, то есть не понятно в каком приложении их можно открыть. В этом случае подсказку следует искать в сети Интернет.

Папка – специально выделенная область внешней памяти, которой присвоено имя. В ней могут находиться другие папки и/или файлы. Папка может быть пустой.

Корневая папка – это любой **логический диск.** Например, флешка или раздел жёсткого диска.

Файловая система имеет вид **дерева.** Говорят «древовидная структура».

К одной папке существует только один единственный путь! Например: C:\Папка 1.1\Папка 2.1\Папка 3.2

Каждый файл находится в определённом месте. **Полное имя файла** – это путь к нему + собственно имя. Сначала указывается имя логического диска, потом «:», а затем через слеш (\) указывается последовательность вложенных папок и, наконец, имя файла с его расширением [1,2].

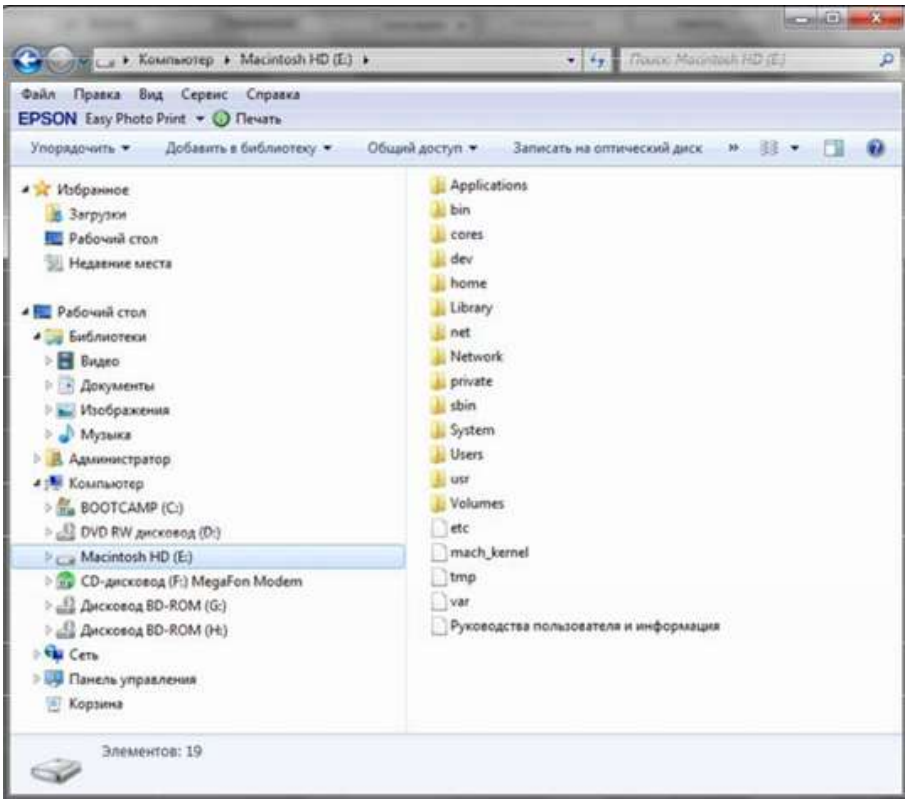
Например, **C:\Windows\System32\cmd.exe**.

Следует отметить, что любой файл или папка могут иметь 2 различных атрибута «Только для чтения» и «Скрытый». Названия говорят сами за себя. Если вы попытаетесь внести изменения в файл, имеющий атрибут «Только для чтения», а затем сохранить его, то система сообщит вам о том, что невозможно сохранить этот файл под тем же самым именем и предложит сохранить изменённую копию файла под другим именем. Кстати, подобная ситуация может возникнуть и в том случае, если вы по сети откроете файл, который уже открыт на каком- то другом компьютере. Дело в том, что изменения может вносить только тот, кто первым открыл файл.

Скрытые файлы и папки по умолчанию не отображаются в системе. О том, как сделать их видимыми речь пойдёт немного ниже.

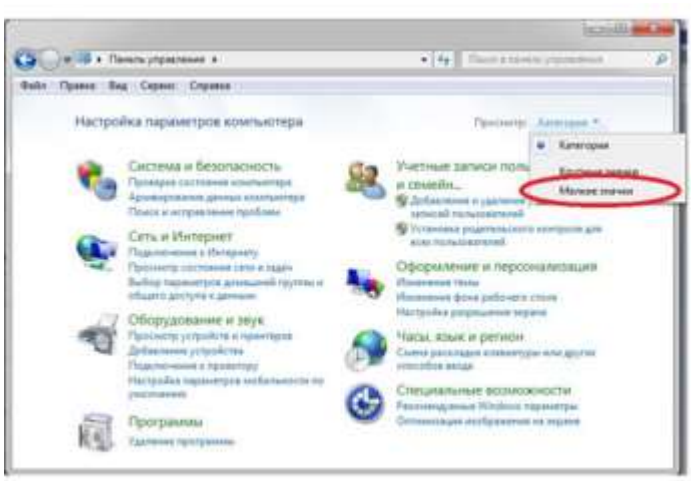
Проводник Windows

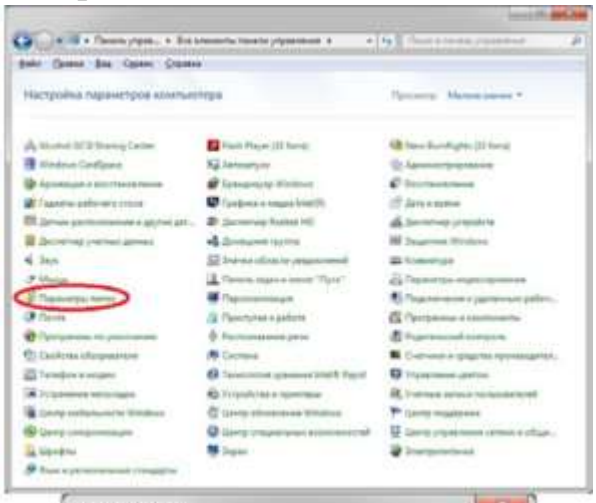
Для управления файловой системой в ОС Windows есть программа **проводник**. Она запускается при открытии любой папки. Кроме того, проводник можно запустить из меню «Пуск», с панели быстрого запуска или сочетанием клавиш **Win+E** [1,2].

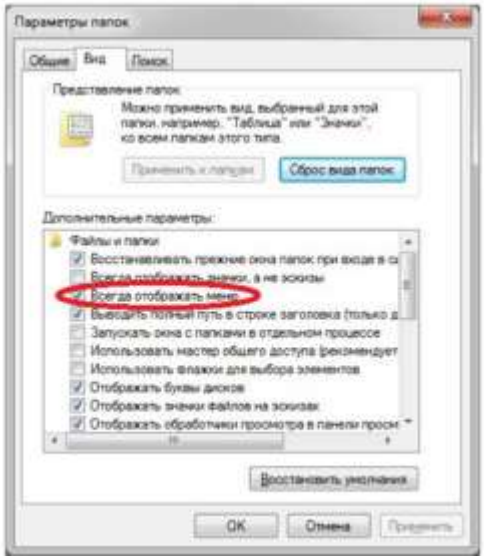


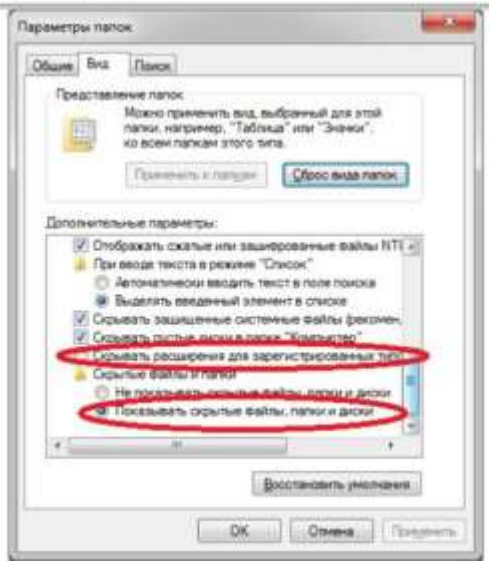
По умолчанию в проводнике меню спрятано. Чтобы оно стало видимым, нужно проделать следующее: Пуск → Панель управления. Далее вместо «Категорий» удобнее использовать «Маленькие значки». Выбираем «Параметры папок». В дополнительных параметрах вкладки «Вид» ставим галочку напротив пункта «Всегда отображать меню» (для Windows 7). В Windows XP меню проводника по умолчанию есть. Поэтому чтобы попасть в соответствующие настройки, нужно выбрать пункт меню «Сервис», а затем «Параметры папок».

Здесь же можно сделать так, чтобы отображались скрытые файлы и папки, а также расширения для зарегистрированных типов файлов.









Способы создания, переименования, копирования, перемещения и удаления файлов и папок

1. Создание папок

а) ПКМ* → Создать → Папку; б) Из меню проводника: Файл → Создать → Папку;

в) Нажав на кнопку на панели инструментов проводника «Новая папка» (для Windows 7(8)).

2. Переименование папок (файлов)

а) ПКМ → Переименовать; б) ЛКМ → Задержка → ЛКМ;

в) Функциональная клавиша F2.

3. Выделение файлов (папок):

а) Растянуть область выделения ЛКМ; б) CTRL+A – выделить всё;

в) ЛКМ с удержанием кнопки SHIFT – последовательное выделение;

г) ЛКМ с удержанием кнопки CTRL – выделение в произвольном порядке.

4. Копирование файлов (папок)

Все действия производятся после выделения.

а) ПКМ по выделению → Копировать, в нужном месте ПКМ → Вставить;

б) CTRL+C, в нужном месте CTRL+V;

в) Перетаскиванием ПКМ → Копировать;

г) Перетаскивание ЛКМ с удержанием CTRL.

В случае перетаскивания ЛКМ с одного логического диска на другой CTRL удерживать не нужно.

5. Перемещение файлов (папок)

Все действия производятся после выделения.

а) ПКМ по выделению → Вырезать, в нужном месте ПКМ → Вставить;

б) CTRL+X, в нужном месте CTRL+V;

в) Перетаскиванием ПКМ → Переместить; г) Перетаскивание ЛКМ в пределах одного логического диска.

6. Удаление файлов (папок)

Все действия производятся после выделения. а) Перетащить ЛКМ в корзину;

б) Delete (Del);

в) ПКМ → Удалить; г) Из меню проводника Файл → Удалить.

Чтобы удалить папки (файлы) совсем (то есть, **минуя корзину**) при использовании любого способа надо удерживать кнопку **SHIFT**.

Задание

Используя различные приёмы создания папок, сделайте на своей флешке дерево, показанное на рисунке.

Папки выделены зелёным, файлы – розовым!

После того как вы убедитесь в том, что дерево папок создано верно, попробуйте различные способы переименования, копирования, перемещения и

удаления файлов и папок. После этого посмотрите описанные выше настройки проводника Windows.

Файловая система FAT. Файловые системы VFAT и FAT32.

В файловой системе FAT дисковое пространство любого логического диска делится на две области:

- системную область и
- область данных.

Системная область создается и инициализируется при форматировании, а впоследствии обновляется при манипулировании файловой структурой.

Системная область состоит из следующих компонентов:

- загрузочного сектора, содержащего загрузочную запись (boot record);
- зарезервированных секторов (их может и не быть);
- таблицы размещения файлов (FAT, File Allocation Table);
- корневого каталога (Root directory, ROOT).

Эти компоненты расположены на диске друг за другом.

Область данных содержит файлы каталоги, подчиненные корневому.

В отличие от системной области, область данных доступна через пользовательский интерфейс DOS.

Загрузочный сектор

Формирование загрузочной записи происходит при форматировании (например, FORMAT). Формат загрузочного сектора зависит от ОС и даже от версии [1,2].

Загрузочный сектор является самым первым на логическом диске. Он содержит загрузочную запись (boot record).

Загрузочная запись состоит из двух частей:

- блока параметров диска (disk parameter block) - часто наз. блоком параметров BIOS (BPB) или Extended BPB (для более старших версий ОС)
- программы начальной загрузки ОС (system bootstrap).

Первые два байта загрузочной записи - команда безусловного перехода на системный загрузчик - JMP 3Eh. В третьем байте - NOP (90h).

Далее - 8 байтовый системный идентификатор (информация о фирме-разработчике и версии ОС). Обратим внимание - ОС не используется.

Затем следует блок параметров диска, а после него - загрузчик ОС.

Блок параметров диска содержит следующую информацию

- размер сектора,
- число секторов в кластере,
- число зарезервированных секторов,
- количество копий FAT,
- максимальное количество элементов ROOT,

- количество секторов в таблице FAT,
- число секторов на дорожке,
- метку тома,
- имя файловой системы
- и другие параметры (байт-описатель среды по смещ 0Ah = F8H- ж.д. любой емкости; F0-дискета 1.44, 3.5’’).

Загрузочные записи различных операционных систем отличаются обычно структурой блока параметров. В некоторых есть и дополнительные поля.

Подробно мы познакомимся с загрузочной записью на лабораторных работах.

Между загрузочным сектором и FAT могут находиться зарезервированные секторы, которые являются служебными для файловой системы или не используются. Количество зарезервированных секторов определено в BPB. (В форматном просмотре - Reserved sectors at beginning - если =1, то это MBR)

Для просмотра и редактирования, а также сохранения и восстановления загрузочной записи можно использовать **утилиту Disk Editor.**

Таблица размещения файлов

Таблица размещения файлов (File Allocation Table - FAT) по сути является картой области данных.

Область данных разбивают на так называемые кластеры. Кластер - это один или несколько смежных секторов области данных. С другой стороны, кластер - это минимальная адресуемая единица дисковой памяти, выделяемая файлу. Т.е. файл или каталог занимает целое число кластеров. Для создания и записи на диск нового файла операционная система отводит для него несколько свободных кластеров диска. Эти кластеры не обязательно должны следовать друг за другом. Для каждого файла хранится список всех номеров кластеров, которые предоставлены данному файлу [1,2].

На дискетах кластер занимает один или два сектора, а на жестких дисках - в зависимости от объема раздела:

для разделов емкостью 16-127 Мбайт - 4 сектора в кластере (размер кластера - 2 Кбайта);

для разделов емкостью 128-255 Мбайт - 8 секторов в кластере (4 Кб);

для разделов емкостью 256-511 Мбайт - 16 секторов в кластере (8 Кб);

для разделов емкостью 512-1023 Мбайт - 32 сектора в кластере (16 Кб);

для разделов емкостью 1024-2047 Мбайт - 64 сектора в кластере (32 Кб).

Разбиение области данных на кластеры вместо использования секторов позволяет:

- уменьшить размер таблицы FAT;
- уменьшить фрагментацию файлов;
- сокращается длина цепочек файла Þ ускоряется доступ к файлу.

Однако слишком большой размер кластера ведет к неэффективному использованию области данных, особенно в случае большого количества маленьких файлов (ведь на каждый файл теряется в среднем полкластера).

В современных файловых системах (FAT32, HPFS, NTFS) эта проблема решается за счет ограничения размера кластера (максимум 4 Кбайта)

Каждый элемент таблицы FAT (12, 16 или 32 бит) соответствует одному кластеру диска и характеризует его состояние: свободен, занят или является сбойным кластером (bad cluster).

- Если кластер распределен какому-либо файлу (т.е., занят), то соответствующий элемент FAT содержит номер следующего кластера файла;
- последний кластер файла отмечается числом в диапазоне FF8h - FFFh (FFF8h - FFFFh);
- если кластер является свободным, он содержит нулевое значение 000h (0000h);
- кластер, непригодный для использования (сбойный), отмечается числом FF7h (FFF7h).

Таким образом, в таблице FAT кластеры, принадлежащие одному файлу связываются в цепочки.

Первый элемент FAT описывает среду загрузочного сектора. Его первый байт совпадает с байтом-описателем среды носителя данных (смещение 0Ah - см. табл.4) и равен 0F0h для гибкого магнитного 3,5 дм. диска или 0F8h для жесткого диска. Следующие 5 байт (7 байт) для 12-разрядного (16-разрядного) формата содержат значение 0FFh.

Таблица размещения файлов хранится сразу после загрузочной записи логического диска, ее точное расположение описано в специальном поле в загрузочном секторе.

Она хранится в двух идентичных экземплярах, которые следуют друг за другом. При разрушении первой копии таблицы используется вторая.

В связи с тем, что FAT используется очень интенсивно при доступе к диску, она обычно загружается в ОП (в буфера ввода/вывода или кэш) и остается там настолько долго, насколько это возможно.

Основной недостаток FAT - медленная работа с файлами. При создании файла работает правило - выделяется первый свободный кластер. Это ведет к фрагментации диска и сложным цепочкам файлов. Отсюда следует замедление работы с файлами.

Для просмотра и редактирования таблицы FAT можно использовать **утилиту Disk Editor**.

Корневой каталог ROOT

Подробная информация о самом файле хранится в другой структуре, которая называется корневым каталогом. Каждый логический диск имеет свой корневой каталог (ROOT, англ. - корень).

Корневой каталог описывает файлы и другие каталоги. Элементом каталога является дескриптор (описатель) файла [1,2].

Дескриптор каждого файла и каталога включает его:

- имя (8 байт).

- расширение (3).
- дату создания или последней модификации (2).
- время создания или последней модификации (2).
- атрибуты (1) (архивный, атрибут каталога, атрибут тома, системный, скрытый, только для чтения).
- длину файла (для каталога - 0) (4).
- зарезервированное поле, которое не используется (10).
- номер первого кластера в цепочке кластеров, отведенных файлу или каталогу; получив этот номер, операционная система, обращаясь к таблице FAT, узнает и все остальные номера кластеров файла (2 байта).

Итак, пользователь запускает файл на выполнение. Операционная система ищет файл с нужным именем, просматривая описания файлов в текущем каталоге. Когда найден требуемый элемент в текущем каталоге, операционная система считывает номер первого кластера данного файла, а затем по таблице FAT определяет остальные номера кластеров. Данные из этих кластеров считываются в оперативную память, объединяясь в один непрерывный участок. Операционная система передает управление файлу, и программа начинает работать.

Для просмотра и редактирования корневого каталога ROOT можно также использовать **утилиту Disk Editor**.

Файловая система VFAT

Файловая система VFAT (виртуальная FAT) впервые появилась в Windows for Workgroups 3.11 и была предназначена для файлового ввода/вывода в защищенном режиме.

Используется эта файловая система в Windows 95. Поддерживается она также и в Windows NT 4.

VFAT - это «родная» 32-разрядная файловая система Windows95. Ее контролирует драйвер VFAT.VXD.

Управляет различными файловыми системами и приложениями диспетчер устанавливаемых файловых систем - Installable File Systems Manager.

VFAT использует 32-разрядный код для всех файловых операций, может использовать 32-разрядные драйверы защищенного режима.

НО, элементы таблицы размещения файлов остаются 12- или 16-разрядными, поэтому на диске используется та же структура данных (FAT). Т.е. ф*ормат таблицы VFAT такой же*, как и формат FAT.

VFAT наряду с именами «8.3» поддерживает длинные имена файлов. (Часто говорят, что VFAT - это FAT с поддержкой длинных имен).

Существует специальный механизм преобразования длинных имен в короткие и наоборот.

Вы помните, что длина имени для DOS подчиняется правилу “8.3”, то есть, длина имени не должна превышать 8 символов, а расширения - 3. Главной особенностью файловой системы Windows 95 для пользователя является то, что *максимальная длина имени файла в Windows 95 может достигать 256*

символов, включая пробелы. Ограничением длины имени файла служит путь к файлу: суммарная длина пути и имени файла не может превышать 260 символов.

При создании файла в Windows95 ему присваивается сразу два имени - длинное и короткое (8.3). Короткое имя образуется из длинного путем удаления из него пробелов и символов / : * ? “ “< > I . Для восьмибуквенного имени файла используются первые шесть оставшихся символов длинного имени, символ “~“ (тильда) и порядковый номер. Для трехбуквенного расширения используются первые три символа после последней точки в длинном имени файла.

Например, короткие имена для файлов (в следующем порядке)

Article about Windows 95.DOS.

Next article about Windows 95.DOS.

Article about Windows NT.DOS.

Microsoft office.HTML.

Microsoft Windows. HTML.

будут выглядеть так

ARTICL~1.DOC.

NEXTAR~1.DOC.

ARTICL~2.DOC.

MICROS~1.HTM.

MICROS~2.HTM.

При этом в структуре ROOT, наряду с обычным дескриптором (называемым алиасным) для файла или каталога создаются дескрипторы специального вида, в которых и хранится длинное имя. Для специальных дескрипторов установлены атрибуты Read Only (только для чтения), System (системный), Hidden (скрытый), Volume Label (Метка Тома). Количество специальных дескрипторов зависит от длины имени.

Специальный дескриптор ссылается на кластер с номером О. Настоящий номер первого кластера, распределенного файлу или каталогу, находится в стандартном (алиасном) дескрипторе, расположенном непосредственно за специальными.

Для томов VFAT нельзя пользоваться никакими утилитами, кроме утилит «понимающих» VFAT

Основной недостаток VFAT - большие потери на кластеризацию при больших размерах логического диска и ограничения на сам размер логического диска.

Файловая система FAT32

Это новая реализация идеи использования таблицы FAT.

FAT32 - это полностью самостоятельная 32-разрядная файловая система.

Впервые использовалась в Windows OSR2 (OEM Service Release 2).

В настоящее время FAT32 используется в Windows 98 и Windows ME.

Она содержит многочисленные усовершенствования и дополнения по сравнению с предыдущими реализациями FAT.

1. Намного эффективнее расходует дисковое пространство за счет того, что использует кластеры меньшего размера (4 Кб) - подсчитано, что экономится до 15%.

2. Имеет расширенную загрузочную запись, которая позволяет создавать копии критических структур данных Þ повышает устойчивость диска к нарушениям структур диска

3. Может использовать резервную копию FAT вместо стандартной.

4. Может перемещать корневой каталог, другими словами, корневой каталог может находиться в произвольном месте Þ снимает ограничение на размер корневого каталога (512 элементов, т.к. ROOT должен был занимать один кластер).

5. Усовершенствована структура корневого каталога

Появились дополнительные поля:

- время создания (2)
- дата создания (2)
- дата последнего доступа (2)
- старшее слово номера начального кластера
- младшее слово номера начального кластера
- контрольная сумма

По-прежнему для длинного имени файла используется несколько дескрипторов.

Для томов FAT32 нельзя пользоваться никакими утилитами, кроме утилит «понимающих» FAT32

NTFS

В *Windows Server* 2003 используется система *NTFS*, которая впервые появилась в *Windows* 2000 (иногда ее называют *NTFS* 5). Эта *файловая система* содержит встроенную поддержку *множества* средств, которые помогают управлять доменами, пользовательскими учетными записями и средствами безопасности. Данная система *NTFS* отличается от *NTFS*, используемой в *Windows* NT 4. Средства, которые поддерживает *NTFS* 5, являются частью структуры *Windows Server* 2003 (и *Windows* 2000), включая (но не ограничиваясь) следующим.

— Active Directory.Содержит все объекты операционной системы, разрешая или запрещая доступ с помощью полномочий. Описание Active Directory см. в "Описание Active Directory" .

— Дисковые квоты.Ограничивают использование дискового пространства на уровне пользователя или группы. Подробнее о дисковых квотах см. в "Управление файлами и дисками" .

— Шифрование. Обеспечивает автоматическое шифрование и дешифрование данных файла при записи или чтении файла. Информацию по использованию шифрующей файловой системы *EFS* (Encrypting File System) см. в "Безопасность Windows Server 2003" .

— DFS (Distributed File System – Распределенная файловая система).Позволяет вам использовать одно дерево папок, охватывающее несколько серверов и разделяемых ресурсов. Подробнее о настройке и поддержке *DFS* см. в "Управление файлами и дисками" .

Кроме мощных средств и возможностей, упрощающих для администраторов управление предприятием, *NTFS* имеет важную и удобную для пользователей

возможность – внесение изменений в значок документа при изменении его местоположения. Эта *функция*, о которой даже не знают некоторые администраторы, очень удобна для пользователей, которые поддерживают значки для часто используемых документов. Например, я держу в моей панели Quick *Launch* (Быстрый *запуск*) значки документов, над которыми работаю длительное время (например, главы книг). Эти значки создаются и удаляются *по* мере необходимости. Кроме того, я держу значок для любого документа, которому требуется постоянное обновление, например, документ, где я держу пароли для доступа к веб-сайтам.

В диалоговом окне Properties (Свойства) для значка документа представлен *путь* к этому документу. Если переместить документ в другую папку и затем щелкнуть на его значке, то откроется именно этот документ. При следующем входе в *диалоговое окно* Properties вы увидите, что *путь* к этому документу изменен соответствующим образом.

Главная таблица файлов (Master File Table) NTFS

Вместо таблицы FAT (File Allocation Table) NTFS использует специальный файл, который называется Master File Table (*MFT*), позволяющий отслеживать все файлы и папки на томе. *MFT* имеет переменный размер, и она автоматически расширяется по мере необходимости. Фактически *MFT* является массивом записей, который вы можете рассматривать как базу данных по всем файлам системы.

Каждая запись *MFT* обычно имеет фиксированный размер 1 Кб, и первые 16 записей содержат информацию о данном томе. Эти относящиеся к тому записи называются файлами метаданных (эта терминология используется для структур служебной информации в файловой системе).

Обычно одна запись в *MFT* соответствует одному файлу или папке в этой файловой системе. Запись содержит атрибуты файла, включая такие элементы, как readonly (доступ только по чтению) и флаги архивации, даты создания и последнего доступа, имя файла и дескриптор безопасности.

Примечание. Обычно в запись включаются два имени файла: длинное имя и совместимое с DOS имя в формате 8.3.

Более существенным отличием от FAT является то, что сами данные файла являются просто еще одним атрибутом NTFS. Имеется ограничение на количество данных, которые можно поместить в запись *MFT*, и все, что превышает этот предел, заменяется в записи указателями на местоположение данных этого файла на диске. Запись *MFT* может содержать примерно 750 байтов данных файла (точное количество зависит от количества атрибутов, хранящихся в записи *MFT*). Небольшие файлы данных (меньше 750 байтов) могут умещаться целиком в своих записях *MFT*, чтобы дает невероятно высокую производительность без какого-либо риска фрагментации этих файлов.

Хотя бы одна запись таблицы *MFT* имеется для каждого файла на томе NTFS, включая файл самой *MFT* и другие файлы метаданных, такие как файл журнала, карта сбойных кластеров и корневая папка.

Конечно, большинство файлов не могут уместиться непосредственно в своих записях *MFT*, поэтому *MFT* сохраняет их данные на диске. Единицей размещения файлов в NTFS являются кластеры, которые указываются двумя способами:

- *VCN* (Virtual Cluster Numbers – Виртуальные номера кластеров) от 0 до n-1, где n – количество кластеров в файле;
- *LCN* (Logical Cluster Numbers – Логические номера кластеров), соответствующие номерам кластеров на томе.

VCN аналогичен смещению в файле, которое запрашивается приложением. Приложение использует в файле определенный формат данных, и оно рассчитывает с помощью этого формата смещение в байтах относительно логического формата этого файла. Когда приложению требуется чтение или запись по определенному адресу в файле, NTFS делит это число на размер кластера, чтобы определить *VCN* для чтения или записи.

LCN – это индекс (указатель) кластеров на томе, и когда требуется чтение или запись, NTFS использует *LCN* для расчета адреса на диске. При этом расчете *LCN* умножается на количество секторов в кластере, после чего происходит чтение или запись секторов, начиная с этого адреса на диске. Связывая номера *VCN* с их *LCN*, система NTFS связывает логическую адресацию файла с физическим местоположением на диске.

Если какой-либо атрибут не умещается в записи *MFT*, то NTFS сохраняет его в новом, отдельном наборе кластеров на диске, который называется *непрерывной областью*, или *экстентом*. Обычно атрибут, представляющий данные файла, слишком велик, чтобы уместиться в записи *MFT*. Однако и другие атрибуты могут оказаться достаточно большими, чтобы образовать новые экстенты. Например, длинные имена файлов могут содержать до 255 символов, каждый из которых занимает 2 байта (поскольку имена файлов хранятся в кодировке Unicode) [1,2].

Атрибут, который хранится в самой записи *MFT*, называется *резидентным* атрибутом. Если атрибут вытесняется в экстент, то он называется *нерезидентным* атрибутом. Это означает, что если пользователи не создают все время очень короткие файлы с короткими именами, то большинство файлов на томе имеют данные, которые соответствуют нерезидентному атрибуту.

Если экстент требуется увеличить (обычно потому, что пользователь добавляет данные к файлу, который уже имеет большие размеры), то NTFS пытается выделить физически смежные кластеры для того же экстента. Если не удается выделить непрерывное пространство достаточного размера, то NTFS создает новый экстент где-либо на диске, разделяя файл на два фрагмента. Если новый экстент тоже не может обеспечить достаточный размер непрерывного пространства, то добавляется еще один экстент. Заголовок атрибутов данных (который хранится в записи *MFT*) содержит эту информацию в форме номеров *LCN* и длин экстентов, а NTFS использует эту информацию для поиска экстентов [1,2].

В некоторых случаях (обычно при крайне большом количестве атрибутов) система NTFS вынуждена выделить дополнительную запись *MFT* для данного файла. В этом случае NTFS создает атрибут, который называется *списком атрибутов* и действует как индекс (указатель) для всех атрибутов данного файла. Хотя это необычная ситуация, присутствие дополнительных записей *MFT* может существенно снизить производительность работы с соответствующими файлами.

Папки (директории) обрабатываются в NTFS почти так же, как файлы. Если папка достаточно мала, то индекс для файлов, на которые указывает эта папка,

может уместиться в записи *MFT*. Эта информация является атрибутом, который называется Index Root (Корень индекса).

Если в папке (индексе) имеется больше элементов, чем может уместиться в записи *MFT*, то NTFS создает новый экстент с нерезидентным атрибутом, который называется *буфером индекса*. Для таких папок буферы индексов содержат то, что называется *деревом b+*, которое является структурой данных, предназначенной для минимизации количества сравнений, необходимых для поиска определенного файла [1,2].

В дереве b+ хранится информация (или указатели на информацию) в отсортированном порядке. Запрос отсортированного списка элементов папки выполняется быстро, поскольку это порядок хранения в буфере индекса. Поиск определенного элемента тоже выполняется быстро, поскольку эти деревья больше разрастаются "вширь", чем "вглубь", что сводит к минимуму количество попыток доступа, необходимых для достижения определенной точки дерева.

Фрагментация NTFS

Все разговоры о том, что NTFS препятствует фрагментации, неверны. Сами системы Windows Server 2003, Windows 2000 и Windows NT намного лучше в выделении дискового пространства для файлов, чем более ранние версии Windows (и DOS), и, в результате, эти системы менее подвержены фрагментации файлов.

Однако NTFS не защищена от источников, вызывающих фрагментацию отдельных файлов, и со временем файлы на томе NTFS становятся фрагментированными. Кроме того, побочным эффектом того, что операционная система пытается воспрепятствовать фрагментации файлов, становится фрагментация свободного пространства на диске.

Форматируя том с помощью NTFS, вы можете выбрать *размер кластеров*. В Windows NT предлагается *размер кластеров* по умолчанию, основанный на размере данного тома, но если вы знаете, каким образом будет использоваться данный том, то можете выбрать наиболее подходящий для этого использования *размер кластеров*. Но вы должны быть очень осторожны, выбирая размер, отличный от размера по умолчанию. При выборе меньшего размера кластеров будут меньше потери пространства на диске, но это вызовет, скорее всего, фрагментацию файлов. При большем размере кластеров вероятность фрагментации будет меньше, но будут больше потери пространства на диске.

Внимание. Выбор 512-байтовых кластеров может вызвать проблемы, поскольку таблица *MFT* состоит из записей, имеющих длину 1024 байта. В случае кластеров размером 512 байтов отдельные записи *MFT* могут оказаться фрагментированными. Это невозможно при более высоких размерах кластеров, которые могут содержать более одной записи *MFT* без каких-либо потерь.

Если файл или папка занимает непрерывную область, то размер кластера не имеет значения (если вас не волнует потеря небольшой доли пространства на диске). Если вы знаете, что том будет содержать очень большое число небольших файлов, или предполагаете, что почти все файлы будут очень большими, то вы обладаете информацией, которую можете использовать для выбора наиболее подходящего размера кластеров. Кроме того, при очень большом общем числе файлов (порядка 100000) фрагментация *MFT* становится более вероятной. В этом случае

больший *размер кластеров* ограничит фрагментацию в таблице *MFT*, поскольку она увеличивается, чтобы вместить все файлы.

Примечание. Можно создать том NTFS с размером кластеров больше 4K, но деф-рагментировать том с помощью встроенной программы дефрагментации будет невозможно. Вы не сможете также использовать сжатие NTFS.

Важно отметить, что существуют отличия в фрагментации на различных уровнях хранения данных. Отдельные приложения, такие как Microsoft Office, и серверы баз данных, такие как Oracle, имеют проблемы фрагментации в рамках их собственного хранилища данных. Эти проблемы возникают независимо от файловой системы или операционной системы.

NTFS не имеет сведений о логической организации пользовательских данных. Независимо от местоположения файла на диске и его фрагментированности файловая система представляет его приложениям как одну непрерывную область хранения. Но приложение интерпретирует данные файла как логическую структуру. Для программы почтовой рассылки файл – это группа фамилий, имен, адресов и т.д. Для NTFS это просто группа кластеров данных.

Приложения, имея собственную организацию данных, могут создавать неиспользуемые промежутки в данных, что фактически фрагментирует эти данные. Как и в файловой системе, удаление данных в приложении не обязательно является их фактическим удалением; эти данные только маркируются как удаленные. Появление промежутков в логическом хранилище данных называют *внутренней фрагментацией*. Чтобы избежать внутренней фрагментации данных, в некоторые приложения, такие как Microsoft Access, включаются утилиты сжатия данных в файле, то есть фактически происходит дефрагментация внутреннего файла. К сожалению, эти утилиты сжатия часто увеличивают степень фрагментированности на уровне файловой системы, поскольку они обычно создают совершенно новую копию этого файла, используя большие объемы пространства на диске в этом процессе. Таким образом, частая дефрагментация ваших файлов данных может увеличить степень фрагментированности вашей файловой системы.

Приложения часто создают временные файлы, которые занимают определенное пространство, когда пользователь работает с этим ПО, открывая и сохраняя файлы. При закрытии приложения временные файлы удаляются, оставляя после себя пустое выделявшееся пространство. Файлы данных могут также иметь занятое, но неиспользуемое пространство, поскольку программы занимают такое пространство для своих организационных целей или для повышения производительности. Кроме того, отдельные файлы, связанные с каким-либо приложением, со временем могут оказаться физически разбросанными по всему диску. Этот тип фрагментации, известной также под названием *фрагментации использования*, представляет особенно сложную проблему для программы дефрагментации, поскольку обычные методы анализа фрагментации могут не выявить ее.

Фрагментация и дефрагментация папок (директорий) происходит аналогично файлам. Дело в том, что для NTFS папка – это еще один файл, хотя папки имеют особые типы атрибутов в своих записях *MFT*. В то время как приложения управляют содержимым файлов данных, NTFS управляет содержимым папок, то есть

деревьями b+, которые обеспечивают индексированный доступ к файлам, содержащимся в папках.

Папки, содержащие программные файлы, мало увеличиваются или уменьшаются в течение периода их использования. Но папки пользовательских документов и системная папка TEMP изменяют свои размеры весьма существенно. По мере роста числа файлов в папке NTFS может реагировать на этот рост путем увеличения пространства хранения папки. Если содержимое папки становится меньше, то NTFS может также освобождать неиспользуемое пространство в папке, но это происходит не всегда, поскольку является достаточно сложной операцией. Папки, которые, скорее всего, будут расти или сжиматься, создаются обычно на стадии конфигурирования системы (например, папки My Documents и TEMP). Эти папки разрастаются, занимая, скорее всего, область, которая не является непрерывной. Поскольку такие папки интенсивно используются, их фрагментация оказывает реальное влияние на производительность компьютера.

Что касается производительности, то вам следует знать, что большая вложенность папок может быть удобной с организационной точки зрения, но за это вы платите снижением производительности. Когда NTFS выполняет поиск данных в своих деревьях b+, то она проходит с самого начала через все уровни поддерева данной папки. Поэтому производительность для деревьев с меньшей вложенностью, имеющих большее число файлов, будет выше, чем для деревьев с большей вложенностью и меньшим числом файлов. Поддеревья с очень большими уровнями вложенности могут также представлять проблемы для приложений, которые имеют ограничение на общее число символов в полном пути к файлу (обычно это предел в 255 символов).

Контрольные вопросы:

1. Что такое файловая система?
2. Особенности файловой системы FAT.
3. Особенности файловой системы FAT32.
4. Особенности файловой системы NTFS.
5. Особенности файловой системы ext3.
6. Особенности файловой системы ext4.

Лабораторная работа 7. *Создание документа с графическими элементами. Презентация. Использование средств MS Office и Open (Libre) Office.*

Цель работы: Научиться работать в программе MS Power Point и создавать презентацию. Изучение информационной технологии разработки презентации в MSPowerPoint. Изучение информационной технологии задания эффектов и демонстрации презентации в MS Power Point.

Задания для выполнения № 1.

Тема: разработка презентации в ms power point.

Этапы создания презентации

1. Сформулировать тему будущей презентации — изученные программы MicrosoftOffice.
2. Определить количество слайдов — 7-10 слайдов.
3. Разработать структуру слайдов: 1-й слайд — титульный лист; 2, 3, 4, 5-й слайды посвящены программам MSWord,MSExcel,MSAccess,MSPowerPoint; 6-й, 7-й,— структурная схема информационного обмена при создании презентации; 8-й и 9-й слайды — резюме.

Задание 1. Создание титульного слайда презентации.

1. Запустите программу MicrosoftPowerPoint. Для этого при стандартной установкеMSOfficeвыполните*Пуск/Программы/ Microsoft Power Point.* Откроется окно с Главной лентойPowerPoint [1,4].
2. Следующим шагом будет Создание нового слайда: *Главная лента– Создать слайд* или *Макет,* на котором представлены различные варианты разметки слайдов (рис. 1).
3. Выберите самый первый тип — титульный слайд (первый образец слева в верхнем ряду). На экране появится первый слайд с разметкой для ввода текста (метками-заполнителями) (рис. 1).

Краткая справка. Метки-заполнители — это рамки с пунктирным контуром, появляющиеся при создании нового слайда. Эти рамки служат метками-заполнителями для ввода текста, таблиц, диаграмм и графиков. Для добавления текста в метку-заполнитель, необходимо щелкнуть мышью и ввести текст, а для ввода объекта надо выполнить двойной щелчок мышью [1,4].

4. Изучите интерфейс программы, подводя мышь к различным элементам экрана.
5. Выберите цветовое оформление слайдов, воспользовавшись шаблонами дизайна оформления *(лента Дизайн/Тема/Цвета)* (рис. 2, 3).

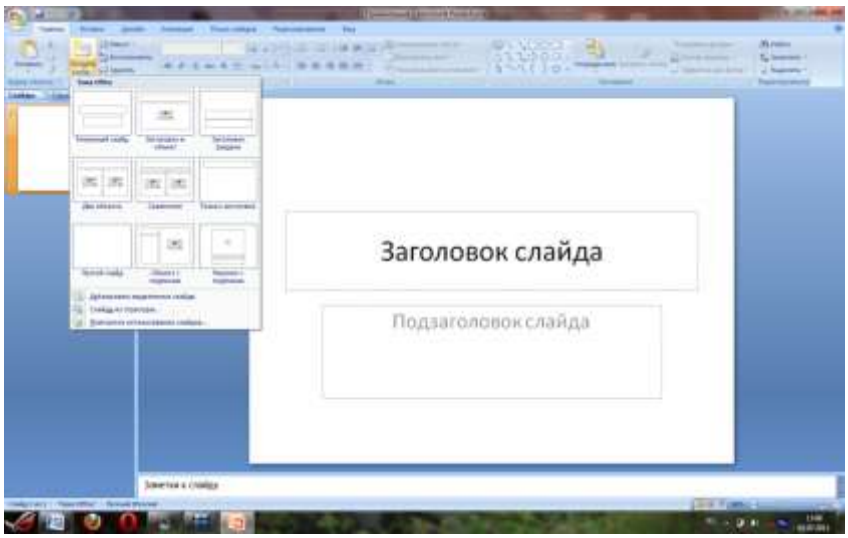


Рис. 1. Окно диалога *Создание слайда*

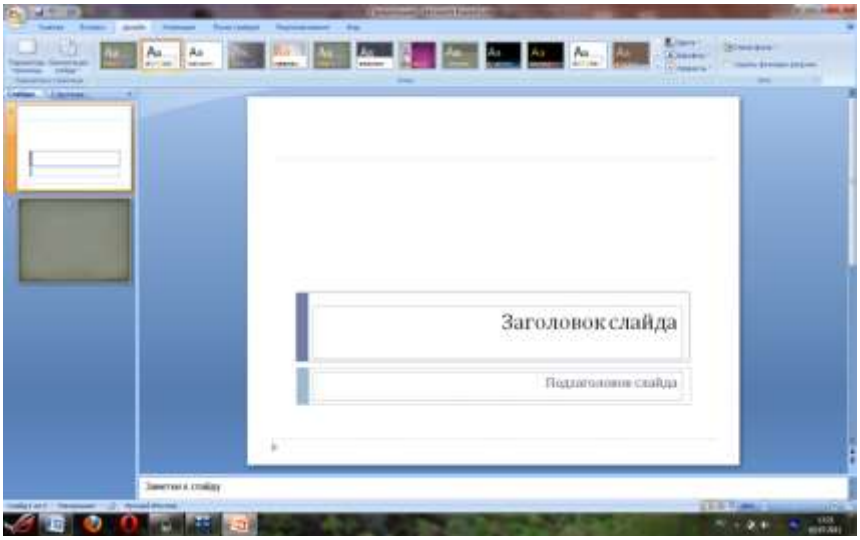


Рис. 2. Окно диалога оформления слайда

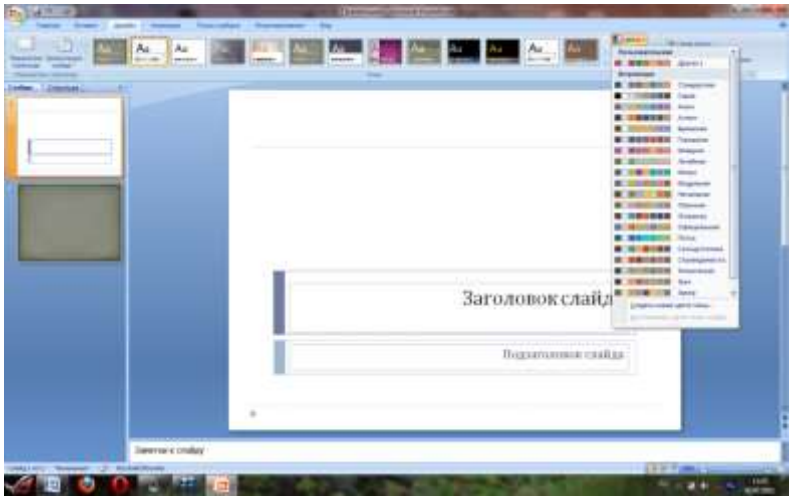


Рис. 3. Выбор цветового оформления слайдов

Введите с клавиатуры текст заголовка — *Microsoft Office*и подзаголовка — *Краткая характеристика изученных программ*. Для этого достаточно щелкнуть

мышью по метке-заполнителю и ввести текст, который автоматически будет оформлен в соответствии с установками выбранного шаблона (рис. 4).

Рис. 4. Титульный слайд презентации

4. Сохраните созданный файл с именем «Моя презентация» в своей папке командой *Кнопка MS Office/ Сохранить как* (рис.5).

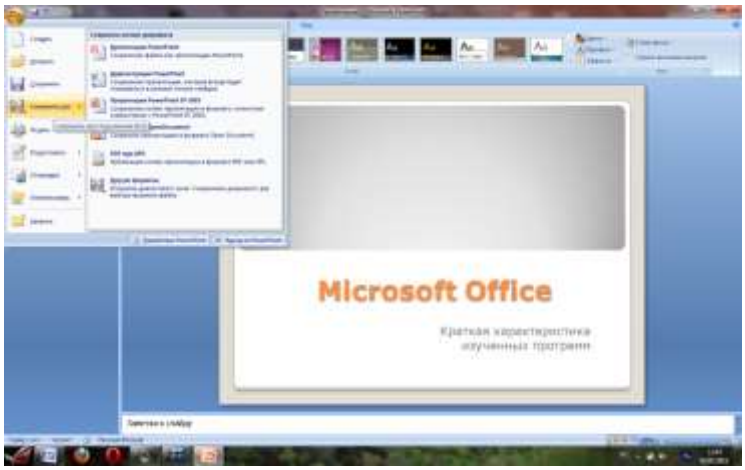


Рис.5. Сохранение презентации

Задание 2. Создание второго слайда презентации — текста со списком.

1. Выполните команду *Главная лента/Создать слайд.*
2. В верхнюю строку введите название программы *Текстовый редактор MSWord*.
3. В нижнюю рамку введите текст в виде списка. Щелчок мыши по метке-заполнителю позволяет ввести маркированный список. Переход к новому абзацу осуществляется нажатием клавиши [Enter] [1,4].

Образец текста

Текстовый редактор позволяет:

— создавать текстовые документы;

— форматировать текст и оформлять абзацы документов;

— вводить колонтитулы в документ;

— создавать и форматировать таблицы;

— оформлять списки в текстовых документах;
— представлять текст в виде нескольких колонок;
— вставлять в документ рисунки;
— готовить документ к печати.

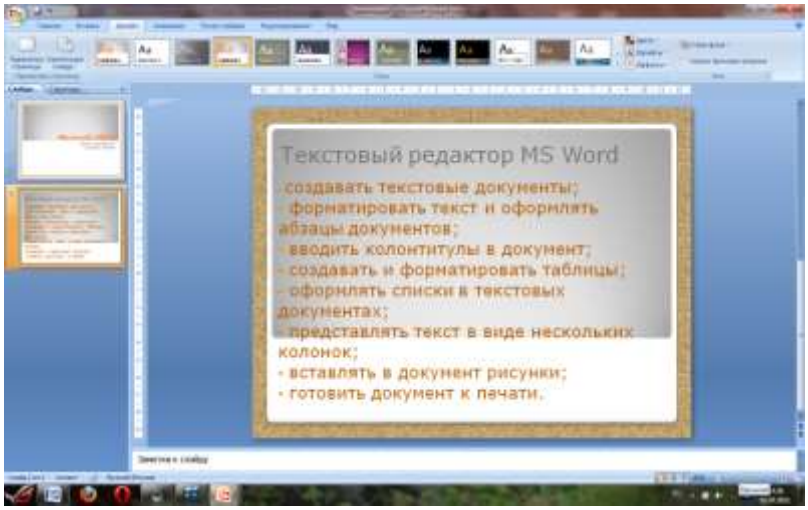


Рис. 5. Текстовый слайд со списком

4. Готовый слайд будет иметь вид, как на рис. 5.
5. Выполните текущее сохранение файла.

Задание 3. Создание третьего слайда презентации — текста в две колонки.

1. Создайте новый слайд.
2. В верхнюю строку введите название программы *Табличный процессор* ***MS Excel***. При необходимости уменьшите размер шрифта и измените тип шрифта (рис. 6).
3. Введите содержание в колонки. Щелчок мыши по метке-заполнителю колонки позволяет вводить в нее текст.

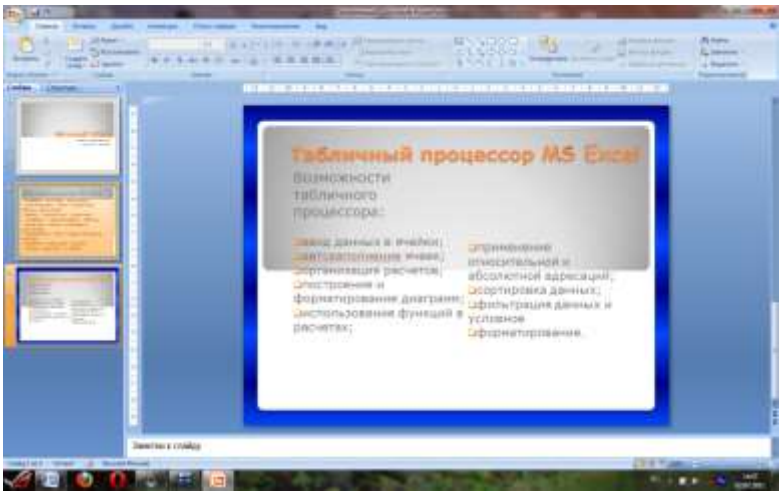


Рис. 6. Третий слайд презентации — текст в две колонки

Образец текста

Возможности табличного процессора:

ввод данных в ячейки; автозаполнение ячеек; организация расчетов; построение и форматирование диаграмм использование функций в расчетах; применение относительной и абсолютной адресаций; сортировка данных; фильтрация данных и условное форматирование.

4. Выполните текущее сохранение файла.

Задание 4. Создание четвертого слайда презентации — текста с таблицей.

1. Создайте новый слайд.
2. В верхнюю строку введите название программы *СУБД MS Access*. При необходимости измените размер шрифта.
3. В нижней рамке выполните двойной щелчок — появится окно задания параметров таблицы данных. Задайте количество столбцов — 2, строк — 5.
4. В появившейся таблице выполните объединение ячеек в первой строке таблицы и заливку, используя панель инструментов.
5. Введите исходные данные, представленные в табл. 1. Для удобства работы откройте ленту *Работа с таблицами/Конструктор и Макет.*

Таблица 1.

Проектирование базы данных	
Таблицы	для хранения данных
Формы	для ввода данных
Запросы	для работы с данными
Отчеты	для ввода информации из БД

6. Конечный вид четвертого слайда приведен на рис. 7.
7. Выполните текущее сохранение файла.

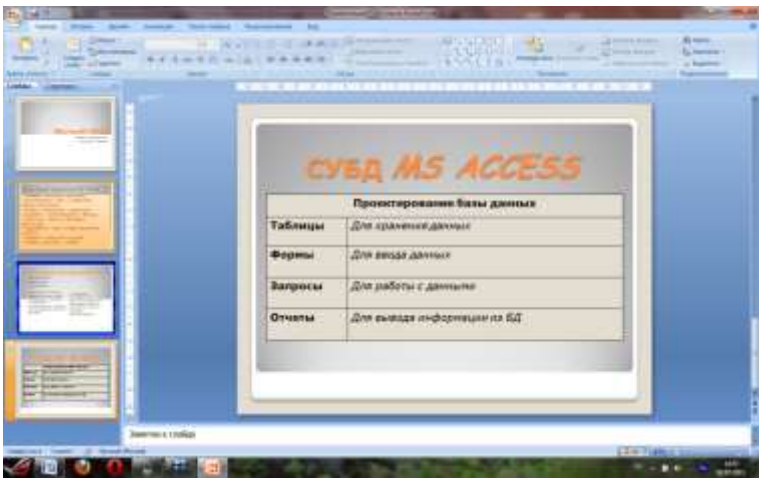


Рис. 7. Конечный вид четвертого слайда с таблицей

Задание 5. Создание пятого слайда презентации — текста с рисунком.

1. Создайте новый слайд.
2. В верхнюю строку введите название программы *MS Power Point*. При необходимости измените размер шрифта.

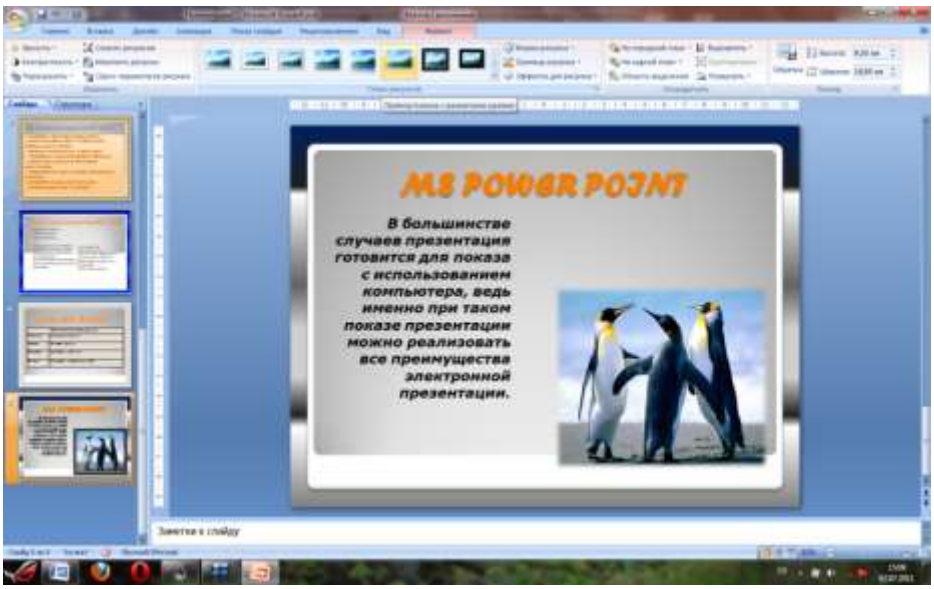


Рис. 8. Пятый слайд презентации — текст с рисунком

3. В левую рамку введите текст по образцу. Выполните правостороннее выравнивание текста (рис. 8).

Образец текста

В большинстве случаев презентация готовится для показа с использованием компьютера, ведь именно при таком показе презентации можно реализовать все преимущества электронной презентации [1,4].

4. В правую рамку введите рисунок, выполнив двойной щелчок мышью по правой рамке, предназначенной для вставки рисунка

6. Перекрасьте рисунок. Для этого щелчком по рисунку выделите его и на ленте *Работа с рисунками/Формат* измените Стиль рисунка, форму, границу и перекрасьте его.

7. Выполните текущее сохранение файла.

Задание 6. Создание шестого слайда презентации — структурной схемы.

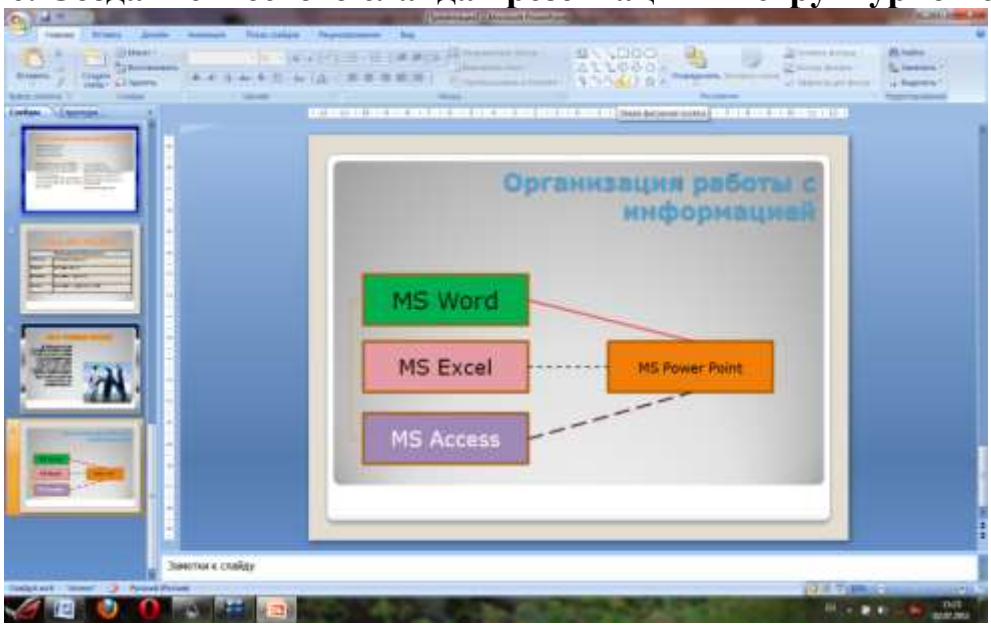


Рис. 9. Слайд презентации со структурной схемой

1. Создайте *Новый слайд.* Выберите *Только заголовок*.
2. Введите текст заголовка Организация работы с информацией. При необходимости измените размер шрифта.
1. На *Главной ленте*/*Рисование,* используя инструменты панели *Рисование* нарисуйте схему, как на рис.9.
2. Создайте *Новый слайд(№7),* используя вставку с элементом SmartArt.
1. Выполните текущее сохранение файла.

Задание 7. Создание восьмого слайда презентации — резюме.

Вставьте новый слайд и введите текст резюме по образцу, используя список (рис. 10). Девятый слайд создайте по образцу, используя вставку SmartArt(рис. 11).

Образец текста

К достоинствам слайдовой презентации можно отнести:

— последовательность изложения;
— возможность воспользоваться официальными шпаргалками;
— мультимедийные эффекты;
— копируемость;
— транспортабельность.

Выполните текущее сохранение файла.

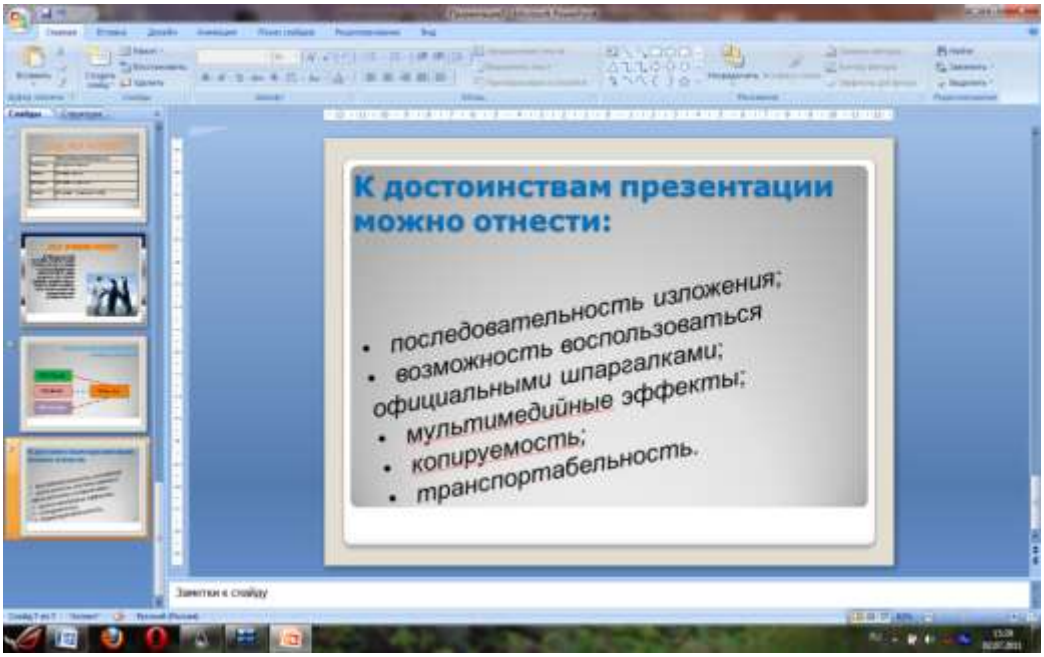


Рис. 10. Восьмой слайд

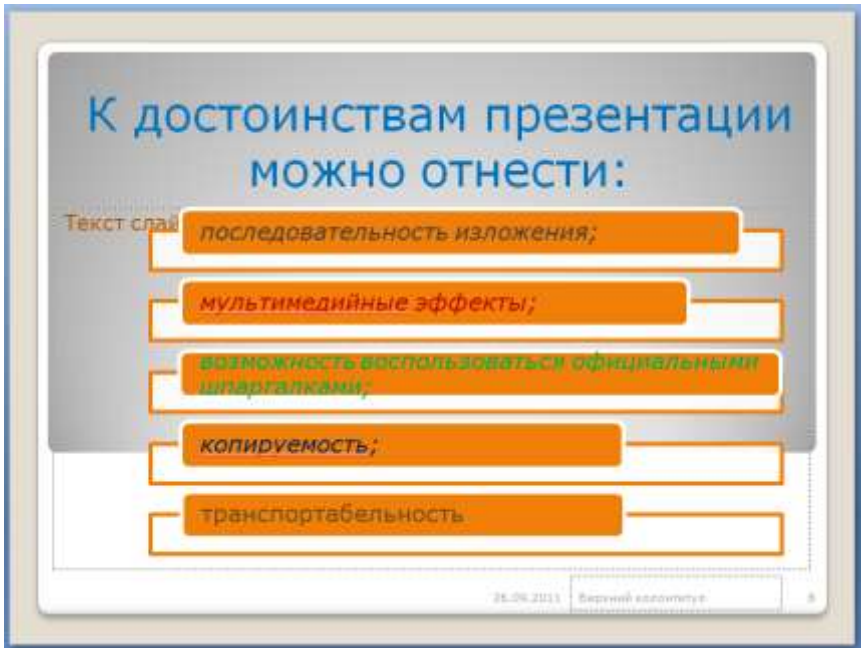


Рисунок 11. Девятый слайд

Аналогичные действия выполнить в LibreOffice Impress.

Контрольные вопросы:
1. Виды графических редакторов.
2. Представители основных видов графических редакторов.
3. Что такое презентация?
4. Что такое Microsoft PowerPoint?
5. Что такое LibreOffice Impress?

Лабораторная работа 8. *Обработка данных в MS Excel Open (Libre) Office Calc.*

Цель работы:

– познакомиться с основными понятиями электронной таблицы;
– освоить основные приёмы заполнения и редактирования таблицы;
– научиться строить диаграммы на основе данных электронной таблицы.

Для запуска табличного процессора Excel 5.0 необходимо дважды щелкнуть мышью на пиктограмме **Microsoft Excel**, которая обычно располагается в групповом окне **Microsoft Office** операционной оболочки Windows 3.1 (3.11). Для запуска табличного процессора Excel 97, работающего под управлением операционной системы Windows 95 или Windows 98, необходимо щёлкнуть на кнопку **Пуск** на панели задач и в открывшемся меню последовательно выбрать пункты **Программы - Microsoft Excel**.

Задание 1. **Изучите элементы окна табличного процессора Excel.**

Верхняя строка – заголовок окна с кнопками управления.

Вторая строка – меню табличного процессора Excel.

Третья и четвёртая строки – панели инструментов Стандартная и Форматирование [1,4].

Пятая строка – строка ввода и редактирования, или строка формул.

Строки между пятой и последней заняты рабочим листом электронной таблицы. Они образуют рабочее поле Excel.

Нижняя строка – строка состояния. В крайней левой позиции нижней строки отображается индикатор режима работы Excel. Он показывает состояние режима программы, например, ГОТОВ, когда Excel ожидает ввода данных.

Рабочее поле электронной таблицы состоит из строк и столбцов. Максимальное количество столбцов равно 256. Столбцы обозначаются буквами на верхней границе рабочего поля в следующем порядке: A, B, C, …, Z, затем AA, AB, …, AZ, BA, BB, BC, …, BZ, CA, CB, CC, …,CZ, DA и т.д. Максимальное количество строк равно 16384 для версии Excel 5.0 и 65536 для версии Excel 97. Строки обозначаются числами на левой границе рабочего поля [1,4].

Каждое пересечение строки и столбца образует ячейку, в которую можно вводить данные (текст, число, дату, время или формулу). Каждая ячейка имеет уникальный адрес, состоящий из буквенного обозначения столбца и номера строки. Например, адрес B3 определяет ячейку на пересечении столбца B и строки номер 3. Светящийся прямоугольник с чёрной рамкой определяет текущую ячейку и называется указателем ячейки. Его можно перемещать по таблице как при помощи

клавиатуры, так и мышью. Ввод данных и некоторые другие действия по умолчанию относятся к текущей ячейке.

Блок представляет собой прямоугольную область смежных ячеек. Блок может состоять из одной или нескольких ячеек, строк или столбцов. Адрес блока состоит из координат противоположных углов, разделенных двоеточием. Например, B13:C19; A12:D27 или D:F. Блок можно задать при выполнении различных команд или вводе формул посредством указания координат или выделения на экране.

Если Вы работаете с версией Excel 5.0, то запустите электронный учебник для изучения возможностей Excel. Для этого в меню **Справка** выберите команду **Быстрое знакомство с Excel**.

Задание 2.

Переход между ячейками, выделение столбцов, строк, блоков, листов.

Для выделения различных объектов с помощью мыши необходимо выполнить действия, указанные в приведённой ниже таблице.

Объект выделения	**Действия с мышью**
Столбец	Щёлкнуть мышью на буквенном имени столбца.
Несколько столбцов	Щёлкнуть мышью на буквенном имени первого столбца и, не отпуская после щелчка кнопку, протянуть курсор мыши до имени последнего выделяемого столбца.
Строка	Щёлкнуть мышью на числовом имени строки.
Несколько строк	Щёлкнуть мышью на числовом имени первой строки и, не отпуская после щелчка кнопку, протянуть курсор мыши до имени последней выделяемой строки.
Блок ячеек	Щёлкнуть мышью на начальной ячейке блока и, не отпуская после щелчка кнопку, протянуть курсор мыши до последней ячейки выделяемого блока.
Рабочий лист	Щёлкнуть мышью на пересечении имён столбцов и строк (левый верхний угол электронной таблицы).
Несколько несмежных блоков	Выделить первую ячейку или блок смежных ячеек, затем нажать клавишу **Ctrl** и, удерживая её, выделить все последующие ячейки и/или блоки.

Для снятия выделения достаточно щёлкнуть мышью по любому невыделенному участку рабочего листа. Новое выделение снимает предыдущее.

Выполните следующие операции.

2.1. Сделайте текущей ячейку **D4** при помощи мыши.

2.2. Сделайте текущей ячейку **A1** при помощи клавиш перемещения курсора на клавиатуре.

2.3. Выделите строку **3**.

2.4. Снимите выделение.

2.5. Выделите столбец **D**.

2.6. Выделите блок **A2:E13**.

2.7. Выделите столбцы **A, B, C, D**.

2.8. Снимите выделение.

2.9. Выделите блок **C4:F10**.

2.10. Выделите весь рабочий лист.

2.11. Снимите выделение.

2.12. Выделите одновременно следующие несмежные блоки: **F5:G10; H15:I15; C18:F20; H20**.

Задание 3.

Ввод данных в ячейки

В Excel существуют следующие типы данных, которые можно ввести в ячейки электронной таблицы: текст, число, дата, время и формула.

Для ввода данных необходимо переместиться в нужную ячейку и набрать данные, а затем нажать клавишу **Enter** или щёлкнуть мышью по зелёной галочке в строке формул. Excel определяет, являются ли вводимые данные текстом, числом или формулой по первому символу. Если первый символ буква или знак **‘**, то Excel считает, что вводится текст. Если первый символ цифра или знак **=, +** или **–**, то Excel считает, что вводится число или формула. Формула всегда должна начинаться со знака **=**.

Данные, которые вводятся, отображаются в ячейке и в строке формул и только при нажатии клавиши **Enter** или при щелчке мышью по зелёной галочке в строке формул они помещаются в ячейку.

Введите следующие данные.

3.1. В ячейку **A1** занесите текст:

Москва – древний город

3.2. В ячейку **B1** занесите число **1147** (год основания Москвы).

3.3. В ячейку **C1** занесите число – текущий год.

3.4. В ячейку **K1** занесите текущую дату в формате ДД.ММ.ГГ, где ДД – дата в диапазоне 01…31; ММ – месяц в диапазоне 01…12; ГГ – год в диапазоне 00…99.

3.5. В ячейку **K2** занесите дату 1 января текущего года в таком же формате.

3.6. В ячейку **D1** занесите формулу: **=C1–B1**. После завершения ввода в ячейке **D1** будет рассчитано по формуле число, указывающее возраст Москвы.

3.7. В ячейку **K3** занесите формулу: **=K1–K2**. После завершения ввода в ячейке **K3** будет по формуле рассчитано количество дней, которое прошло с начала текущего года до настоящего дня.

3.8. Самостоятельно в свободных ячейках электронной таблицы вычислите количество дней, которое Вы прожили со дня своего рождения по настоящий день.

3.9. Определите возраст Москвы в 2015 году. Для этого замените год в ячейке **C1**. Обратите внимание на важнейшее свойство электронной таблицы: автоматический пересчёт значений ячеек с формулами при вводе новых данных.

3.10. Отредактируйте текст в ячейке **A1**. После корректировки в ней должен быть новый текст:

Москва – столица России

Задание 4.

Операции со строками, столбцами, блоками

4.1. Выделите блок **A1:D1** и переместите его на строку ниже. Для выполнения перемещения с помощью мыши выделите указанный блок. Затем наведите указатель мыши на рамку блока (он должен принять форму белой стрелки) и переместите блок.

4.2. Верните блок на прежнее место.

4.3. Скопируйте блок **A1:D1** в строки **3**, **5**, **7**. Копирование выполняется аналогично перемещению, но при нажатой клавише **Ctrl**.

4.4. Выделите строку **7** и ***заполните*** выделенными данными строки с **8**-ой по **15**-ую включительно. ***Заполнение*** – это многократное копирование данных ячеек электронной таблицы. Заполнение с помощью мыши выполняется так же, как и перемещение, но при этом курсор должен наводиться на нижний правый угол ячейки или блока (указатель курсора принимает форму чёрного плюса и называется *маркером заполнения*).

4.5. Выделите блок **A10:G15** и очистите его. Для того, чтобы очистить только данные (числа, текст, формулы), необходимо выделить ячейку или блок и нажать клавишу **Del**.

4.6. Выделите блок **K1:K3** и очистите его.

Задание 5.

Функция автозаполнения

В Excel существует интересная функция, называемая автозаполнением, которая позволяет быстро вводить различные типовые последовательности (арифметическую и геометрическую прогрессии, даты, дни недели, месяцы, года и т.д.).

5.1. В ячейку **G10** занесите год – 1990.

5.2. В ячейку **H10** занесите год – 1991.

5.3. Выделите блок **G10:H10**.

5.4. Укажите на маленький чёрный квадратик в правом нижнем углу ячейки **H10** (экранный курсор при этом превращается в маркер заполнения – чёрный тонкий плюс).

5.5. Нажмите левую кнопку мыши и, не отпуская её, двигайте мышь вправо, пока рамка не охватит ячейки **G10:M10**. Блок должен заполниться годами с 1990 по 1996.

5.6. Введите в ячейки **G11:M11** дни недели с понедельника до воскресенья. При этом достаточно ввести один понедельник, а затем воспользоваться функцией автозаполнения.

5.7. Введите в ячейки **G12:M12** месяцы с января по июль. При этом достаточно ввести один январь, а затем воспользоваться функцией автозаполнения.

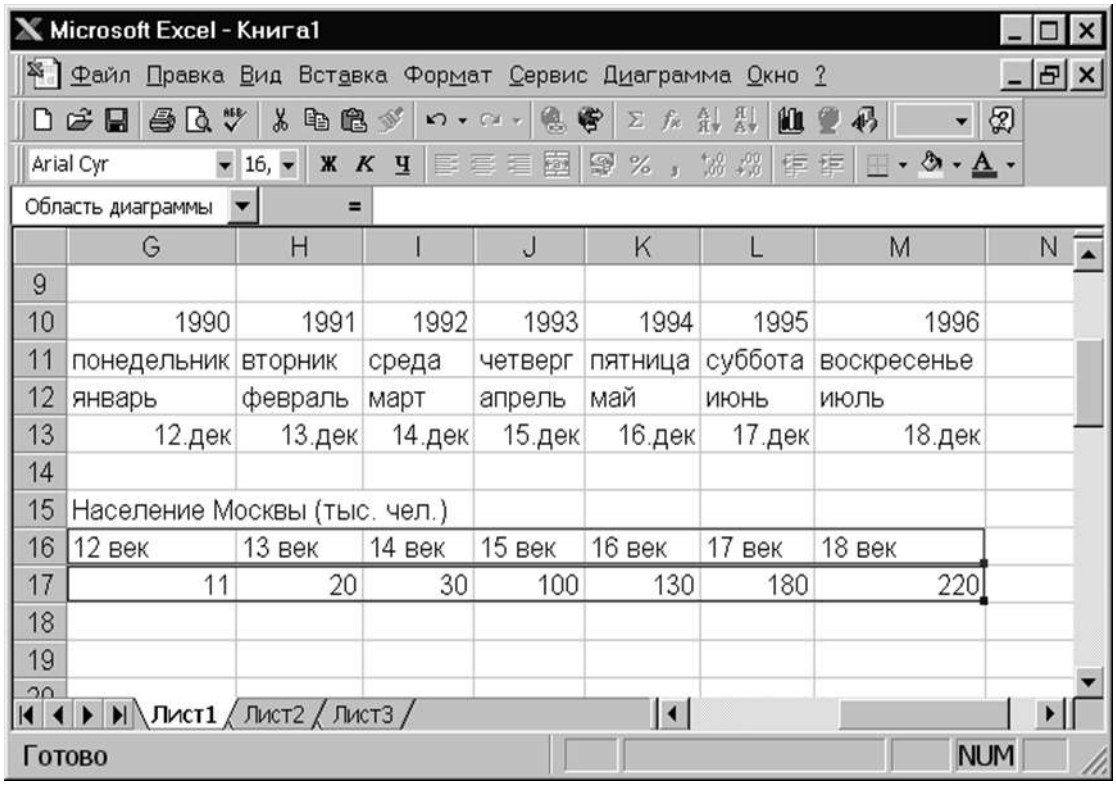
5.8. Введите в ячейки **G13:M13** даты, начиная с текущей. При этом достаточно ввести одну текущую дату, а затем воспользоваться функцией автозаполнения.

5.9. Введите с помощью функции автозаполнения в ячейки **G16:M16** века, начиная с 12 века, по 18 век включительно. В ячейку **G15** введите текст:

Население Москвы (тыс. чел.)

5.10. В ячейки **G17:M17** введите данные о численности населения Москвы по векам, заполнив их соответственно числами: 11; 20; 30; 100; 130; 180; 220.

В результате проделанной по данному заданию работы электронная таблица должна приблизительно иметь вид, представленный на рис.1.



	G	H	I	J	K	L	M	N
9								
10	1990	1991	1992	1993	1994	1995	1996	
11	понедельник	вторник	среда	четверг	пятница	суббота	воскресенье	
12	январь	февраль	март	апрель	май	июнь	июль	
13	12.дек	13.дек	14.дек	15.дек	16.дек	17.дек	18.дек	
14								
15	Население Москвы (тыс. чел.)							
16	12 век	13 век	14 век	15 век	16 век	17 век	18 век	
17	11	20	30	100	130	180	220	
18								
19								

Рис. 1 - Вид электронной таблицы после выполнения задания 5.

Задание 6.

Построение диаграммы на основе данных электронной таблицы

Создать диаграмму или график легче всего с помощью специального средства - Мастера диаграмм. В Excel 5.0 он производит построение диаграммы за 5 шагов, получая всю необходимую информацию для построения из диалоговых окон, открываемых на каждом шаге. (В Excel 97 Мастер диаграмм строит диаграмму за 4 шага.)

6.1. Выделите блок ячеек **G16:M17**.

6.2. Щёлкните по кнопке Мастера диаграмм на панели инструментов.

6.3. В первом диалоговом окне «Шаг 1 из 5» Мастер диаграмм спрашивает диапазон данных из таблицы, необходимых для построения. Но так как данные уже были указаны предварительным выделением ячеек (в п.6.1), переходите к следующему шагу, щёлкнув по кнопке **Шаг>.**

6.4. На втором шаге (в диалоговом окне «Шаг 2 из 5») следует выбрать тип диаграммы – Гистограмма, после чего снова щёлкнуть по кнопке **Шаг>.**

6.5. На третьем шаге выберите формат гистограммы – **7**, после чего снова щёлкните по кнопке **Шаг>.**

6.6. На четвёртом шаге, скорее всего, в диалоговом окне ничего изменять не надо. Excel обычно правильно определяет, где находятся ряды данных в таблице (в данном случае они должны быть в строках). Остаётся только проверить это. Убедитесь также, что установлены значения: «**Отвести 1 строку для меток оси X (категорий)**» и «**Отвести 0 столбцов для текста легенды»**. Затем снова щёлкните по кнопке **Шаг>.**

6.7. На пятом, последнем шаге необходимо указать, следует ли добавлять к диаграмме так называемую *легенду* с названиями и маркерами данных (укажите это по своему усмотрению – «Да» или «Нет»), а также можно ввести название диаграммы и осей X и. Введите название диаграммы – «Рост численности населения Москвы в 12-18 веках»; название по оси X – «Века» и название по оси Y – «Численность населения (тыс. чел.)». После этого щёлкните по кнопке **Закончить.** Диаграмма построена. Её вид должен примерно соответствовать рис.2.

6.8. Готовую диаграмму можно переместить в любое место рабочего листа. Для этого её надо сначала выделить, щёлкнув по ней мышью, а затем, установив на ней указатель мыши, переместить в новое место. Можно также изменить размеры диаграммы, для чего надо установить указатель мыши на размерные маркеры предварительно выделенной диаграммы, расположенные на углах или серединах сторон диаграммы, и потянуть за них. С учётом этих возможностей добейтесь хорошего вида и расположения диаграммы на рабочем листе Excel.

6.9. Результаты проделанной работы предъявите преподавателю для оценки.

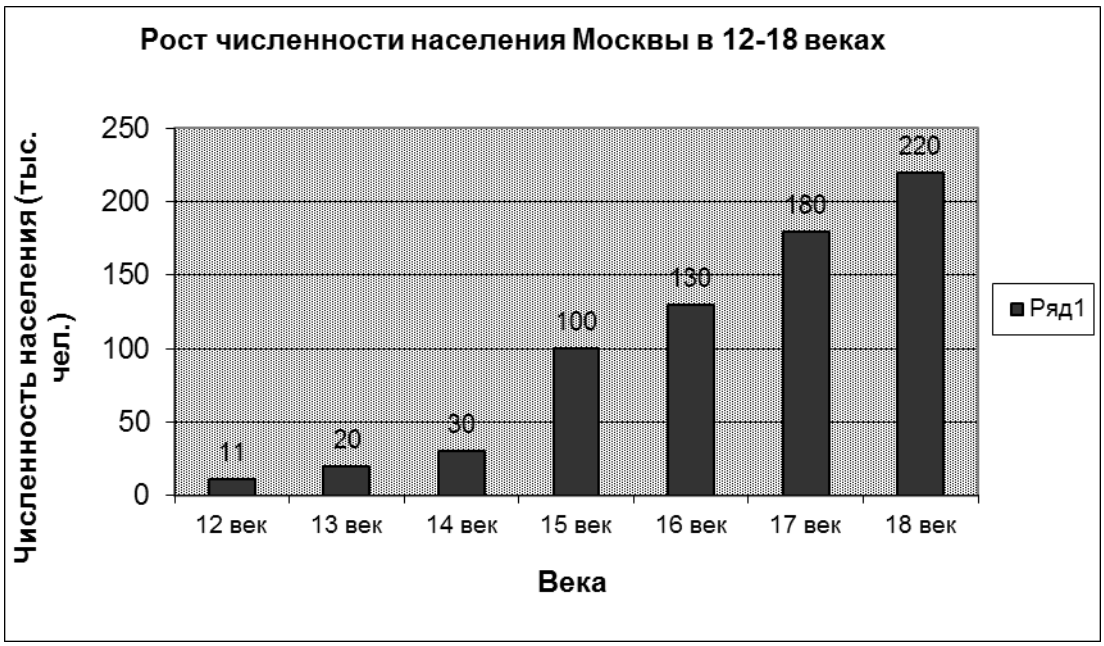


Рис. 2 - Вид диаграммы «Рост численности населения Москвы в 12-18 веках»

Контрольные вопросы:

1. Что такое электронная таблица?
2. Что такое книга?
3. Что такое лист?
4. Свойства ячейки.
5. Действия с ячейками.
6. Абсолютный и относительный адрес ячейки (ссылки).
7. Что такое формула?
8. Что такое функция?
9. Методы копирования в MS Excel.

Лабораторная работа 9. *Создание БД, запросы, формы, отчеты.*

Цель работы: научиться создавать таблицы базы данных в **СУБД MS Access** и связывать их между собой.

Задание 1.

Руководитель малого предприятия, выполняющего сборку персональных компьютеров из готовых компонентов, заказал разработку базы данных, основанной на двух таблицах комплектующих. Одна таблица содержит данные, которые могут отображаться для клиентов при согласовании спецификации изделия, — в ней указаны розничные цены на компоненты. Вторая таблица предназначена для анализа результатов деятельности предприятия — в ней содержатся оптовые цены на компоненты и краткая информация о поставщиках (клиенты предприятия не имеют доступа к данным этой таблицы) [1,4].

Создайте две таблицы «Комплектующие» и «Поставщики».

Алгоритм работы

1. Запустите программу ***Microsoft Access*** 2000 (**Пуск → Программы → Microsoft Access**).

2. В окне ***Microsoft Access*** включите переключатель **Новая база данных** и щелкните на кнопке **ОК**.

3. В окне **Файл** новой базы данных выберите свою личную папку (или папку ***Мои документы)*** и дайте файлу имя: ***Комплектующие.*** Убедитесь, что в качестве типа файла выбрано **Базы данных Microsoft Access**, и щелкните на кнопке **Создать**. Откроется окно новой базы —**- Комплектующие: база данных.**

4. Откройте панель **Таблицы.**

5. Дважды щелкните на значке **Создание таблицы** в режиме конструктора — откроется бланк создания структуры таблицы.

6. Для первой таблицы введите следующие поля:

Имя поля	**Тип поля**
№п/п	Счетчик
Компонент	Текстовый
Модель	Текстовый'
Основной	Числовой
Цена	Денежный

7. Щелкните на поле **Цена**. В нижней части бланка задайте свойство **Формат поля** Денежный, **Число десятичных знаков** равным 2.

8. Для связи с будущей таблицей поставщиков надо задать ключевое поле. Поскольку здесь ни одно поле явно не претендует на «уникальность», используем комбинацию полей **Компонент** и **Модель**. Выделите оба поля в верхней части

бланка (при нажатой клавише SHIFT). Щелчком правой кнопки мыши откройте контекстное меню и выберите в нем пункт **Ключевое поле**.

9. Закройте окно **Конструктора**. При закрытии окна дайте таблице имя ***Комплектующие.***

10. Повторив действия пунктов 5–9, создайте таблицу ***Поставщики***, в которую входят следующие поля:

Имя поля	Тип поля
№п./п	Счетчик
Компонент	Текстовый
Модель	Текстовый
Цена оптовая	Денежный
Поставщик	Текстовый
Телефон	Текстовый
Адрес	Текстовый
Примечание	Поле МЕМО

Таблица 1.1. Комплектующие

№ *а/а*	**Компонент**	Модель	**Основной параметр**	Цена
1	Монитор	LG SW 520Si	13,7	5415,00р.
2	Монитор	Samtron 55E	**13,8**	5318,00р.
3	Монитор	MAG57OFD	14	7993,00р.
4	Звуковая плата	ESS Solo-1 OEM	1938	1287,00р.
5	Звуковая плата	ESS Allegro	19895	1383,00р.
6	Системная плата	Socket A	100	2900,70р.
7	**Системная** плата	Slot A	133	3191,00р.
8	Накопитель	FDD	1,44	1431,00р.
9	Накопитель	HDD IDE	41	4061,00р.
10	Память	RIMM	64	3034,00р.
П	Винчестер	WD 100EB	8,00	493,00р.
12	Винчестер Винчестер	WD 200EB	20,00	607,00р.
13		WD 307AA	30,70	517,00р.

Обратите внимание на го, что ноле номера телефона является текстовым, несмотря па то, что обычно номера телефонов записывают цифрами. Это связано с тем, что они не имеют числового содержания. Номера телефонов не сравнивают по величине, не вычитают из одного номера другой и т. д. Это

типичное текстовое поле. Ключевое поле можно не задавать — для текущей задачи оно не требуется.

11. В окне **Комплектующие: база данных** откройте по очереди созданные таблицы и наполните их экспериментальным содержанием (табл. 1.1 и табл. 1.2). Закончив работу, закройте таблицы и завершите работу с программой [1,4].

Задание 2.

Определите ключевые поля и свяжите таблицы между собой с целью дальнейшего получения данных из обеих таблиц.

Алгоритм работы

1. Запустите программу ***Microsoft Access*** **2000 (Пуск → Программы → Microsoft Access).**

2. В окне ***Microsoft Access*** включите переключатель **Открыть базу данных**, выберите ранее созданную базу **Комплектующие** и щелкните на кнопке **ОК**.

3. В окне **Комплектующие: база данных** откройте панель **Таблицы.** Убедитесь, что на ней присутствуют значки ранее созданных таблиц ***Комплектующие*** и ***Поставщики*.**

4. Разыщите на панели инструментов кнопку **Схема данных**. Если есть сложности, найдите команду строки меню: **Сервис → Схема данных.** Воспользуйтесь любым из этих средств, чтобы открыть окно **Схема данных.** Одновременно с открытием этого окна, открывается диалоговое окно **Добавление таблицы**, на вкладке **Таблицы** которого можно выбрать таблицы, между которыми создаются связи.

5. Щелчком на кнопке **Добавить** выберите таблицы ***Комплектующие*** и ***Поставщики*** — в окне **Схема данных** откроются списки полей этих таблиц.

6. При нажатой клавише **SHIFT** выделите в таблице ***Комплектующие*** два поля **«Компонент»** и **«Модель».**

7. Перетащите эти поля на список полей таблицы ***Поставщики*.** При отпускании кнопки мыши автоматически откроется диалоговое окно **Изменение связей.**

8. На правой панели окна **Изменение связей** выберите поля **«Компонент»** и **«Модель»** таблицы ***Поставщики,*** включаемые в связь. Не устанавливайте флажок **Обеспечение целостности данных** — в данном упражнении это не требуется, но может препятствовать постановке учебных опытов с таблицами.

Таблица 1.2. Поставщики

п/п	Компонент	Модель	Цена оптовая	Поставщик	Телефон	Адрес поставщика	Примечание

1	Монитор	LO SW 520Si	4800,00	Заполните	произвольно	Заполните	произвольно,
2	Монитор	Samtron SSE	4550,00	Заполните	произвольно	Заполните	произвольно
3	Монитор	MAG570FD	6300,00	Заполните	произвольно	Заполните	произвольно
4	Звуковая ' плата	ESS Solo-1 OEM	1195,00	Заполните	произвольно	Заполните	произвольно
5	Звуковая плата	ESS Allegro	1278,00	Заполните	произвольно	Заполните	произвольно
6	Системная плата	Socket A	2340,70	Заполните	произвольно	Заполните	произвольно
7	Системная .плата	Slot A	2900,00	Заполните	произвольно	Заполните	произвольно
8	Накопитель	FDD	350,00	Заполните	произвольно	Заполните	произвольно
9	Накопитель	HDD IDE	3780,00	Заполните	произвольно	Заполните	произвольно
10	Память	RIMM	2700,00	Заполните	произвольно	Заполните.	произвольно |
И	Винчестер	WD 100EB	370,00	Заполните	произвольно	Заполните	произвольно |
12	Винчестер	WD 200EB	480,00	Заполните	произвольно	Заполните	произвольно
13	Винчестер	WD307AA	490,00	Заполните	произвольно	Заполните	произвольно

9. Закройте диалоговое окно **Изменение** связей и в окне **Схема данных;** рассмотрите образовавшуюся связь. Убедитесь в том, что линию связи можно выделить щелчком левой кнопки мыши, а щелчком правой кнопки мыши открывается контекстное меню, позволяющее разорвать связь или отредактировать ее.

10. Закройте окно **Схема данных**. Закройте программу ***Microsoft Access.***

Контрольные вопросы:

1. Что такое БД?
2. Виды БД.
3. Модели представления данных.
4. Что такое СУБД?
5. Что такое Запрос?
6. Что такое Форма?
7. Что такое Отчет?

Лабораторная работа 10. *Математическое моделирование. Компьютерное моделирование.*

В среде MS Excel создать информационную модель простейшей ЛВС.

Исходные данные:

1.1 Число серверов – 1.

1.2 Число отделов – 4.

1.3 Число ПЭВМ (рабочих станций, персональных компьютеров - PC) в каждой рабочей группе равно 4 + i.

1.4 Занимаемая полоса пропускания линии связи при придаче данных в процессе общения пары клиент – сервер равна:

- в направлении ПЭВМ – сервер – 1,35 МБит/с;
- в направлении сервер – ПЭВМ – 5 МБит/с.

1.5 Занимаемая полоса пропускания линии связи при придаче данных в процессе общения пары клиент – сервер равна:

- в направлении ПЭВМ -Internet - 1,9 МБит/с;
- в направлении Internet – ПЭВМ – 3,8 МБит/с.

1.6 Занимаемая полоса пропускания линии связи при придаче данных между компьютерами – 1,5 МБит/с.

Для представленных данных необходимо:

1. Построить схему сети;

2. Создать шаблон позволяющий рассчитывать значения трафика в каждой линии связи.

3. Вывести на график пропускные способности в различных линиях связи.

4. Увеличивая параметры занимаемой полосы пропускания на 30%, с шагом 0.1, представить изменение пропускной способности в точках подключения к серверу и выходу в интернет.

5. Сделать выводы о возможности процессов и целесообразности этого процесса.

Контрольные вопросы:

1. Классификация моделей.

2. Основные этапы разработки и исследования моделей на компьютере.

3. Сущность математического моделирования.

4. Сущность компьютерного моделирования.

Лабораторная работа 11. *Алгоритмы разветвляющейся структуры. Алгоритмы циклической структуры. Изображение блок-схемы алгоритма согласно ГОСТ РФ.*

Цель: Ознакомление студентов с организацией программ разветвляющихся и циклических типов, а также закрепление практических навыков работы в программировании.

КРАТКИИ СВЕДЕНИЯ ИЗ ТЕОРИИ

В большинстве встречающихся задач требуется использование определенного набора типовых приемов алгоритмизации, к которым можно отнести алгоритмы разветвляющейся и циклической структур.

В алгоритмах разветвляющейся структуры предусмотрено разветвление последовательности действия на несколько направлений в зависимости от итога проверки заданного условия.

В алгоритме циклической структуры предусмотрено неоднократное выполнение одной и той же последовательности действий.

1.1 Программирование алгоритмов разветвляющейся структуры

В языке ПАСКАЛЬ для программирования разветвляющейся структуры используются операторы безусловного перехода, условного перехода и оператор выбора.

Оператор безусловного перехода имеет вид GOTO *m*; где *m*- метка, представляющая собой целое число без знака. Метка должна быть описана в разделе описания. Этот оператор передает управления к оператору, описанному меткой *m*. Причем *m* может быть помечен пустой оператор, который не предписывает никаких действий и используется для выхода.

Условный оператор имеет следующие виды записи:

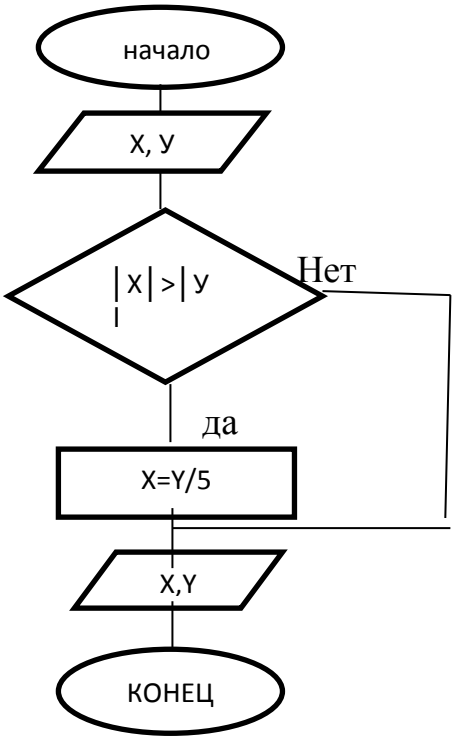
А) IF B THEN A;

Б) IF B THEN A ELSE C; где B - выражение логического типа; A, C- простые или сложные операторы.

При использовании условного оператора вида а) будет организовано вычисление оператора А, если логическое выражение В принимает значение TRUE(истина). Если же оно принимает значение FALSE (ложь), то будет выполняться оператор, который следует за этим условным оператором [2, 3, 5].

Пример 1. Пусть даны два число Х и У. Если первое больше второго по абсолютной величине, то необходимо уменьшить первое в 5 раз. Иначе оставить числа без изменения.

Блок-схема алгоритма и программа имеют вид [2, 3, 5]:




```
PROGRAM A;
VAR X, Y: REAL ;
BEGIN
    WRITELN (‘ ВВЕСТИ X,Y’);
    READ (X,Y);
    IF ABS(X)> ABS(Y) THEN X:= X/5;
    WRITELN (‘ X=’,X,’  ‘ , ‘Y=’Y);
END.
```

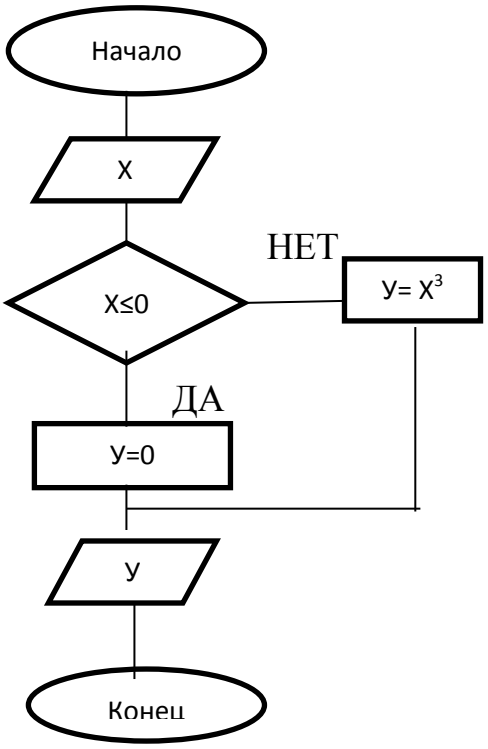
Рис.1

При использовании условного условного оператора вида б) будет организовано вычисление оператора А,если логическое выражение В принимает значение TRUE. Если логическое выражение принимает значение FALSE, то выполняется оператор С, стоящий после ELSE.

Пример 2. Вычислить У по значениям Х, если

$$У = \begin{cases} 0, \text{ если } X \le 0; \\ X^3, \text{ если } X>0; \end{cases}$$

Схема алгоритма и программа имеют вид:




```
PROGRAM B;

VAR X,Y :REAL;
BEGIN
WRITELN (‘ВВЕСТИХ’);

        READLN(X);

        IF X:=0 THEN Y:=0 ELSE

        Y:=X*X*X;

        WRITELN(‘Y=’,Y);

END.
```

Рис 2.

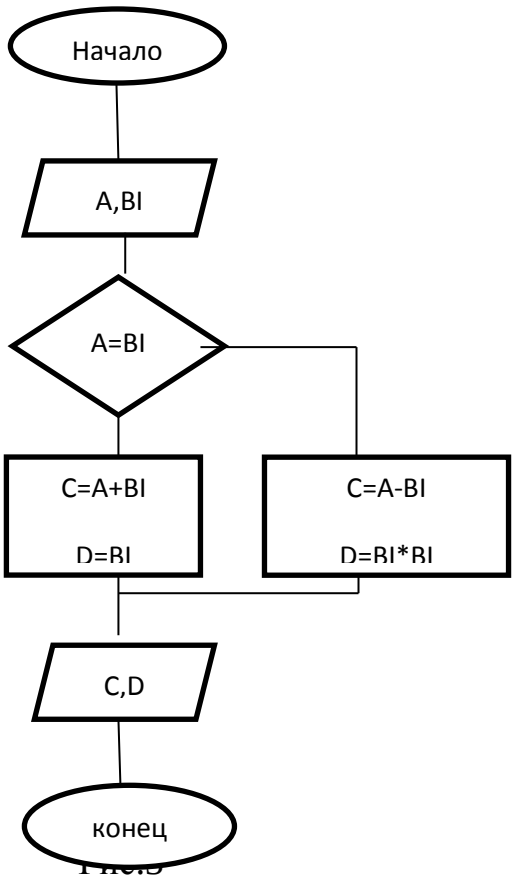
В случае, если необходимо выполнение операторов при тех или иных условиях, то это осуществляется объединением операторов в блок, формируемый операторами BEGIN и END

Например, IF B THEN BEGIN AI;A2;A3 END
ELSE BEGIN A4;A5; END;

В этом примере имена с A1 по A5символически обозначают операторы. Если логическое выражение B принимает значение TRUE, то выполняются операторы A1,A2,A3, в противном случае выполняются операторы A4 и A5. Операторы, заключенные между BEGINи END, образуют так называемый составной оператор, а сами операторы BEGINи ENDчасто именуют операторными скобками [2, 3, 5].

Пример 3.

Требуется установить значения переменных C и D в зависимости от состояния переменных A и B1, причем, если значение переменной A равно B1, то переменной C присваивается значение, равное сумме значений переменных A и B1,а переменной D- значение переменной B1. В случае, если значение переменной A не равно значению переменной B1, то переменной C присваивается значение, равное разности значений переменных A и B1,а переменной D - значение переменной B1,возведенное в квадрат. Схема алгоритма и программа имеют следующий вид [2, 3, 5]:




```
PROGRAMC(INPUT, OUTPUT);
      LABEL IØ;
      VAR A,BI,C,D:REAL;
BEGIN
      WRITELN( 'ВВЕСТИ A,BI');
      READ( A, BI);
      IF A=BI THEN BEGIN
      C=A+BI;
      D=BI;
      GOTO IØ;
ELSE BEGIN
            C=A-BI;
      D=BI*BI;
      END;
IØ: WRITE( 'C=',C,'   ','D=',D);
END.
```

Оператор выбора варианта имеет вид:

CASE C OF NI : AI;N2:A2;….NN:ANEND;

Где С-выражение скалярного типа (но не вещественного); NI,N2,…,NN –метка операторов; AI,A2,…,AN – операторы.

Выполнение оператора начинается с вычисления значения выражения С. Затем выполняется тот оператор, метка которого совпала с полученным значением. Если же вычисленное значение выражение С не совпало ни с одной из меток, то предусмотрено появление сообщения об ошибке.

Пример 4. Пусть требуется напечатать значение введенной с клавиатуры цифры, принимающей значение от единицы до пяти, с контролем введенного значения [2, 3, 5].

Программа имеет вид:

```
PROGRAM VES;
VAR I: INTEGER;
BEGIN
```

```
WRITELN( ‘ВВЕСТИ I’);
READ(I);
WRITELN( ‘I=’,I);
CASE I OF
1: WRITELN(‘ ВЫ ВВЕЛИ => 1’);
2: WRITELN(‘ ВЫ ВВЕЛИ =>2’);
3: WRITELN(‘ ВЫВВЕЛИ =>3’);
4,5: WRITELN( ‘ВЫ ВВЕЛИ=>4 ИЛИ 5’);
END;  END.
```

1.2 Программирование алгоритмов циклической структуры

Программы циклической структуры могут быть организованы с помощью следующих операторов IF, FOR, WHILE, REPEAT. Условный оператор IF требует задания начального параметра цикла, проведения в цикле вычисления с этим параметром, изменения его на величину шага и проверки условия окончания цикла. Если цикл не закончен, то вычисления продолжаются.

Пример 5.

Составить схему алгоритма и программу для вычисления выражения. Схема алгоритма и программа имеют вид:

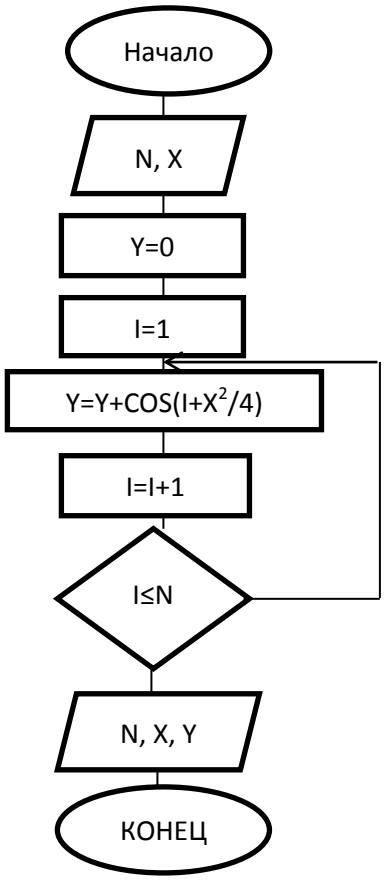


Рис. 4

```
PROGRAM SUMIP;
LABEL 3;

VAR  X,Y: REAL;

        I,N:INTEGER;

BEGIN

        WRITELN( ‘ВВЕСТИN,X’);

        READ(N, X);

        Y:=Ø;

        I:=1;

3: Y:=Y+COS(I+SQR(X)/4);

I:=I+I;

        IF X<=N THEN GOTO 3;

WRITELN( ‘N=’,N);
WRITELN(‘X=’,X, ‘   ‘,’Y=’,Y);
END.
```

Оператор цикла FOR используется для организации цикла с известным числом повторений. При этом различают две формы записи:

А) с организацией счета при изменении параметра цикла от начального (меньшего) значения до конечного (большего) значения этого параметра цикла. Такой оператор цикла имеет вид:

FOR K:=NZ TO KZ DO

Б) с организацией счета при изменении параметра цикла от начального(большего) значения до конечного(меньшего) значения этого параметра. Этот оператор имеет вид:

FOR K: = NZ DOWNTO KZ DO

Где К- параметр цикла (целочисленная переменная), NZ, KZ-выражения, задающие соответственно начальное и конечное значения параметра цикла; S–простой или составной оператор (тело цикла).

Оператор цикла выполняется следующим образом. Параметру цикла К присваивается начальное значение NZ. Затем уравнение передается в тело цикла и выполняется оператор S, после выполнения которого параметр цикла меняет свое значение на единицу(шаг изменения параметра цикла). При этом шаг равен +I, если используется оператор цикла с ключевым словом TO(случай а), а если используется оператор цикла с ключевым словом DOWNTO (случай б), то шаг равен –I. Далее измененное значение параметра цикла сравнивается с конечным значением KZ и, если параметр цикла не превышает К(случай а) или превышает KZ (случай б), то управление передается в тело цикла и выполняется оператор S; в противном случае осуществляется выход из цикла.

Рассмотрим фрагменты записи операторов цикла:

VARY,I: INTEGER;

1) Y:=0; FOR I=3 TO 5 DO Y=Y+1;

2) Y:=0; FOR I=12 DOWNTO 6 DO Y:=Y+1;

В результате выполнения первого оператора цикла параметр цикла I будет изменяться от начального значения I=3 до конечного значения I=5 с шагом, равным +I, причем после окончания цикла значения переменной Y=3. При выполнении второго оператора цикла параметр цикла I будет изменяться от начального значения I=I2 до конечного значения I=6 с шагом, равным –I. После окончания цикла Y=7.

Пример 6.

Используя условия примера 5 программу с изменением оператора цикла FOR.

```
PROGRAM SUM;
VAR    X,Y: REAL;
I,N: INTEGER;
BEGIN
   WRITELN( ‘ВВЕСТИN,X ‘);
   READ(N, X);
 Y:=Ø;
FOR   I:=1 TO N   DO
   Y:= Y+ COS (I+SQR(X)/4);
    WRITELN (‘N=’,  N  ,’    ‘, ‘X=’,X);
    WRITELN(‘Y=’,Y);
END.
```

Оператор цикла WHILE позволяет организовать цикл, количество повторений которого зависит от включенного в него условия, т.е цикл с неизвестным числом повторений. Этот оператор имеет вид

WHILE LV DO

Где LV- логическое выражение;

S - простой или составной оператор.

Выполнение оператора начинается с проверки выражения LV. Если логическое выражение имеет значение TRUE, то выполняется оператор S. Выполнение оператора S продолжается до тех пор, пока значение выражения LV не станет равным FALSE. В этом случае уравнение передается оператору, следующему за оператором S. Если же выражение LV имеет значение FALSE при первоначальной проверке, то оператор S не выполняется ни разу. При этом, чтобы выйти из цикла, внутри его должно быть организовано изменение значений переменных, входящих в логическое выражение. Иначе оператор цикла будет выполняться бесконечное число раз.

Пример 7.

Составим программу, используя условия примера 5. Программа запишется в виде:

```
PROGRAM  SUMWHL;
VARX,Y : REAL;
        I,N :INTEGER;
BEGIN
        WRITELN('ВВЕСТИN,X');
        READ(N,X);
        Y:=Ø;
        I:=1;
        WHILE I<=N   DO
        BEGIN
        Y=Y+ COS( I+SQR(X)/4);
        I:=I+1;
END;
        WRITELN( 'N=',N,'    ','X=',X);
        WRITELN( 'Y=',Y);
        END.
```

Оператор цикла REPEAT также позволяет организовать цикл с неизвестным числом повторений. Такой оператор имеет вид:

REPEATS
UNTIL LV

где S – простой или составной оператор;

LV- логическое выражение.

Выполнение оператора REPEAT начинается с вычисления оператора S и продолжается до тех пор, пока не выполняется LV т.е. когда значение этого логического выражения равно FALSE. Это означает, что проверка LV проводится после каждой итерации и в случае принятия LV значения TRUE осуществляется выход из цикла. При использовании составного оператора S операторные скобки (BEGIN и END) не требуются. Кроме того оператор, стоящий перед словом UNTIL, не имеет после себя точку с запятой.

Пример 8.

Задан массив вещественных чисел, состоящий из десяти элементов. Требуется вывести четные элементы этого массива. Программа имеет вид:

```
PROGRAM MAS;
VAR  I: INTEGER;
     A: ARRAY[1…10] OF REAL;
BEGIN
     WRITELN(‘ ВВЕСТИМАССИВ(1..10)’);
     FOR I:=1 TO 10  DO;
     READ (A[I]);
     I:=0;
     REPEAT
     I=I+2;
     WRITE (A[I]:5:2)
     UNTIL I>10;
END.
```

3.3 Программирование алгоритмов со структурой вложенных циклов.

Для реализации сложных алгоритмов возникает необходимость организовать в составе одного цикла еще несколько других циклов. При этом цикл, внутри которого находятся другие циклы, называется внешним, а вложенные в него-внутренними. Алгоритмы циклических вычислительных процессов, содержащие внутренние циклы, будем называть алгоритмами со структурой вложенных циклов [2, 3, 5].

Организовывать вложенные циклы необходимо таким образом, чтобы циклы не пересекались, т.е., чтобы тело внутреннего цикла полностью размещалось в теле внешнего цикла.

Изменение параметров внутреннего и внешнего циклов происходит не одновременно, следующим образом:

- при начальном значении параметра внешнего цикла параметра внутреннего цикла последовательно принимает все возможные значения(от своего начального значения до конечного), причем передача управления из внутреннего во внешний цикл происходит после принятия параметром внутреннего цикла своего конечного значения и выполнения тела этого цикла;

- происходит изменение параметра внешнего цикла на величину шага, а параметр внутреннего цикла опять последовательно принимает все свои значения;

- выход из внешнего цикла производят тогда, когда параметр этого цикла примет свое конечное значение, а параметр внутреннего цикла последовательно примет все свои значения [2, 3, 5].

Пример 9.

Составить программу для вычисления

$$P = \sum_{n=1}^{m} \sqrt{2 * S / n} * \prod_{i=1}^{n} N / \sqrt{i}$$

```
PROGRAM VLZ;
VAR    N,M, I: INTEGER;
       S,R,P,Y,Z: REAL;
BEGIN
       WRITELN (‘ВВЕСТИ M,S’);
       READ (M,S);
       R:=0;
       FOR N:=1 TO M DO
BEGIN
       P:=1;
       FOR I:=1 TO N DO
       BEGIN
       Y:=N/SQRT(I);
       P:=P*Y;
```

```
END;
        Z:=SQRT(2*S/N)*P;
        R:=R+Z;
        END;
        WRITELN( 'R=',R);
        END.
```

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

1. Вычислить значение

$$Z=\begin{cases} X^2+3, \text{ если } x<0, \\ X-0{,}8, \text{ если } x>\pi/2, \\ \cos X, \text{ если } 0\le X\le \Pi/2. \end{cases}$$

2. Вычислить значение

$$Z=\begin{cases} a-b, \text{ если } a>0, \\ a+b, \text{ если } a=0, \\ ab, \text{ если } a<0 \end{cases}$$

3. Вычислить значение

$$Z=\begin{cases} a/b, \text{ если } a>0, \\ a*b, \text{ если } a=0, \\ a-b, \text{ если } a<0 \end{cases}$$

4. $Z=a+b \sin x/ x^2-bx+0{,}5$, еслих≥ 0

5. Вычислить значение

$$Z=\begin{cases} X^2+Y^2, \text{ если } X^2+Y^2< 1, \\ X+Y, \text{ если } X^2+Y^2>1, \\ 0{,}8, \text{ если } X^2+Y^2=1. \end{cases}$$

6. Вычислить значение

$$Y=\begin{cases} a+ bx+cx^2, \text{ если К=1,} \\ d+cx+fx^2, \text{ если К=2,} \\ g+ hx+tx^2, \text{ в остальных случаях} \end{cases}$$

7. Вычислить значение

$Y= 18,9X^2-1,6X+1,97$, если $X<0$ и $Z=9,75X^2-0,35\text{arctg } x$, если $X\geq 0$.

8. Вычислить значение $Z=a+b \sin x/ x^2-bx+0,5$, если$x\geq 0$.

9. Вычислить значение $Z= \sin x/ x^2 -\sqrt{1+3x}$, если $x\leq 0,5$

10. Вычислить значение

$$Z=\begin{cases} \text{Arctg}\,x+y/1-xy \text{ , если } xy<1, \\ 3,14, \text{ если } xy=1, \\ 3,14+ (x+y/1-xy), \text{ если } xy>1. \end{cases}$$

11. Вычислить значение $Y^2-0,3$, если $y<0$,

$$Z=\begin{cases} 1, \text{ если } 0\leq y\leq 1, \\ \\ Y^2+0,3, \text{ если } y>1. \end{cases}$$

12. Вычислить значение $A_i=X^I/(2i)$, $i=1,2,\ldots,30$.

13. Вычислить значение $A_i=(X^{2i}\ \text{SIN}(X^i))/i^2$, $i=1,2,\ldots,30$.

14. Вычислить значение $A_i=((-1)^i*X^i)/i(i+1)(i+2)$, $i=1,2,\ldots,30$.

15. Вычислить значение $A_I=(X^{2i}*(i+3))/2i$, $i=1,2,\ldots,30$.

16. Вычислить значение $A_I=(X^I*(i+2))/i+1$, $i=1,2,\ldots,30$.

17. Даны вещественные числа У, С и целое n($1\leq n\leq 5$). Вычислить $Z=3$ COS(X)/X+C, где Хвычисляется по формулам:

1) $X=1,2\ Y^2$ 2)$X=3,5Y$ 3)$X=1,5$ 4)X=ARCTG(Y) 5) $X=6+0,5Y^2$, а значение nтрактуется как номер для вычисления X.

18. Дано целочисленное значение nи вещественные значения X,У и Z. При n<0,n=0 или n>0 изменить соответственно значения X на X-0,5, У на У/3,5, а Z на Z^2.

19. Вычислить значение $У_i=X_i+\sin x_i$ для всех $0\le X_i\le 1$, если $X_i=X_{i-1}+h$, $X_0=0$ и h=0,1

20. Все отрицательные компоненты вещественного вектора $a(a_1,a_2,\ldots,a_{10})$ заменить их квадратами.

21. Дан вещественный вектор X $(X_1,X_2,\ldots,X_{10})$. Умножить на 2 компонента с нечетными номерами.

22. Дан вещественный вектор b $(b_1,b_2,\ldots,b_{10})$. Все компоненты с четными номерами уменьшить на 0,5.

23. Все компоненты вещественного вектора C $(C_1,C_2,\ldots, C_{10})$, начиная с первой по порядку положительной компоненты увеличить в два раза.

24. Вычислить n! для n=7.

25. Присвоить переменной Тзначение " 3", если числа X,У,Z равны "2", если числа равны; "1", если все числа различны.

26. Определить количество тех членов последовательности

$A_k=SIN(K^2+C)$, K=1,2,….,10, которые удовлетворяют условию $0\le A_K\le 1/2$.

27. Определить значения последовательности $\{x_i\}$из условия $X_{i+1}=0{,}5X_{i-1}-X_i^2$, $X_1=X_i=0{,}65$ для i=3,4….10.

28. Вычислить значение $Y=X^n/3!$ Для n=1,2,…10

29. Вычислить значение $Y=(1)^n/2n(2n-1)$ для n=1,2,….10.

30. Вычислить значение $Y=2n/n^2\sqrt{x}+n$ для n=1,2,….,10.

Контрольные вопросы:

1. Какими свойствами обладает алгоритм?
2. Что понимается под результативностью алгоритма?
3. В чем заключается свойство массовости?
4. Что означает свойство определенности алгоритма?
5. Какими элементами задается алгоритм?

6. Что представляет собой «исполнитель алгоритма»?
7. Что является основной характеристикой исполнителя?
8. Что относится к среде исполнителя?
9. Поясните общую методику записи содержания алгоритма.
10. Какие существуют формы представления алгоритмов?
11. Что представляет собой словесный способ записи алгоритма?
12. Что называется блок-схемой алгоритма?
13. Поясните особенности составления блок-схемы алгоритма.
14. Когда используется межстраничный соединитель?
15. Что представляет собой псевдокод?

Лабораторная работа 12. *Использование инструментов сред ЯВУ (Lazarus-Delphi).*

Lazarus состоит из нескольких окон (которые стоит подравнять, чтобы они занимали весь *рабочий стол* и не мешали друг другу) [3]:

1. **Главное окно**
2. **Инспектор объектов**
3. **Редактор форм**
4. **Редактор кода**
5. **Окно сообщений**

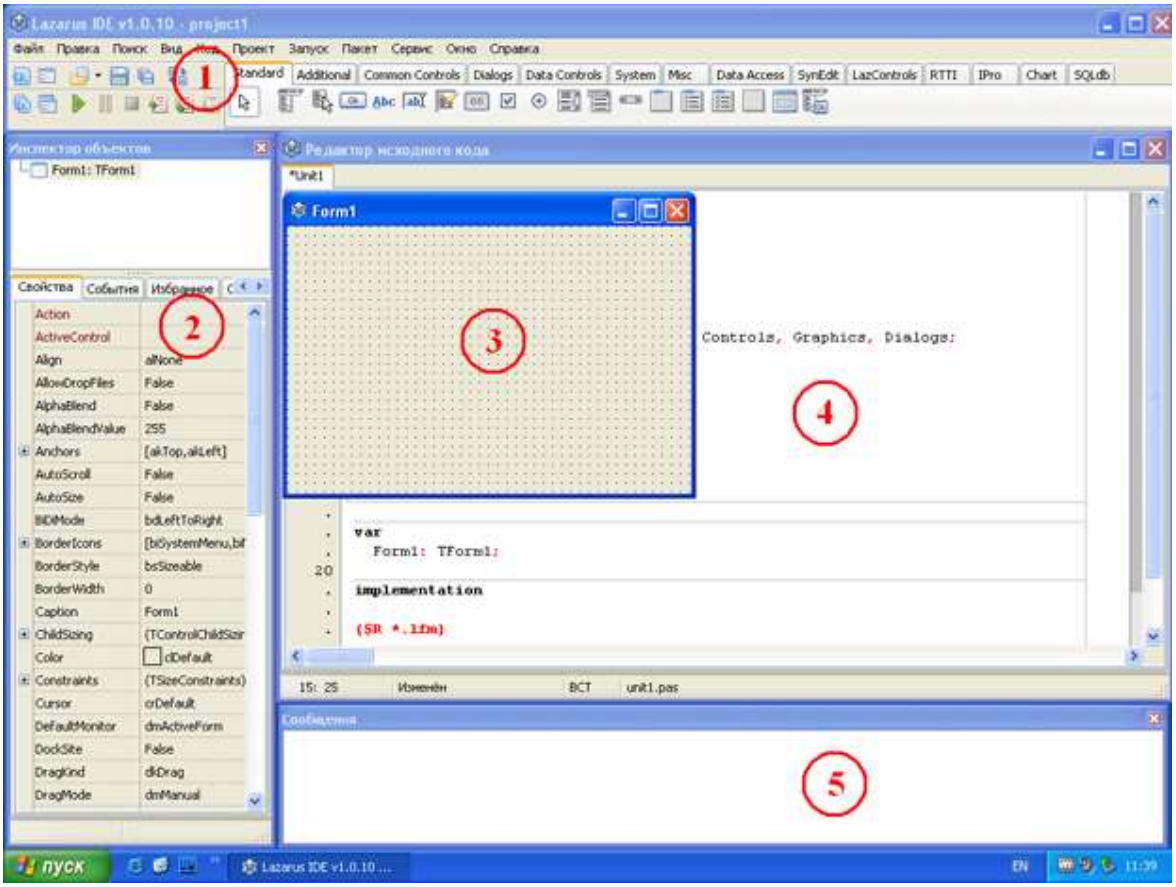


Рис. 1.3. Окна Lazarus

Главное окно

Главное окно состоит из следующих элементов:

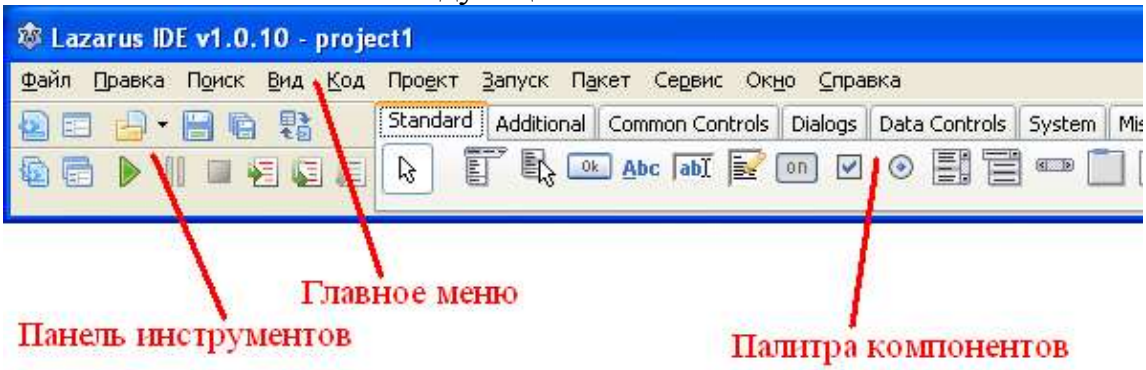


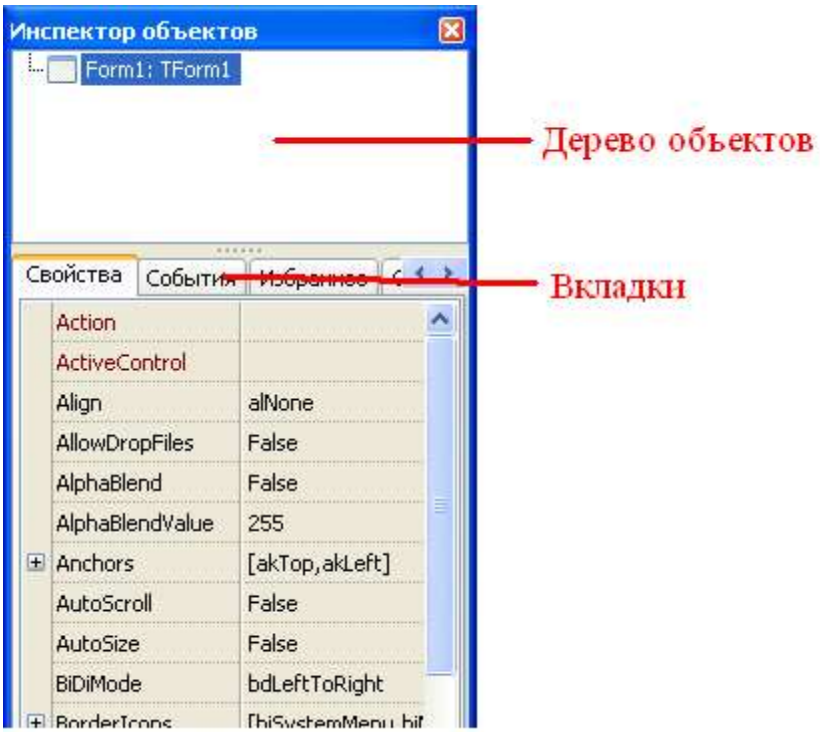
Рис. 1.4. Главное окно Lazarus

1. **лавное меню** содержит все команды, необходимые для правки, компиляции, отладки программы, для запуска различных вспомогательных утилит.
2. **Панель инструментов** содержит кнопки чаще всего применяемых команд (эти же команды можно выполнить и с помощью **Главного меню**).
3. **Палитра компонентов** содержит множество вкладок, на которых содержится богатый выбор компонентов из собственной библиотеки компонентов **Lazarus - LCL** (Lazarus Component Library).

Инспектор объектов

Окно **Инспектора объектов** состоит из двух частей:

- **Дерево объектов**, в котором в древовидной форме располагаются все объекты, используемые в текущей форме.
- **Окно с вкладками**, в котором можно настраивать различные свойства текущего объекта. Несмотря на то, что имеется 4 вкладки (**Свойства**, **События**, **Избранное**, **Ограничения**), чаще всего используются только первые две. О свойствах и событиях мы поговорим подробней в следующих лекциях [3].



Редактор форм, Редактор кода и Окно сообщений

Последние три окна проще. **Редактор форм** предназначен, соответственно, для редактирования формы - положения и размеров компонентов, размещенных на этой форме.

Несмотря на явную схожесть, форма и окно приложения - не одно и то же. **Форма** - это то, что видит программист в процессе разработки проекта, а окно - это то, что увидит *пользователь*, когда загрузит нашу программу.

Редактор кода содержит исходный код, который нам придется вводить и модифицировать. Редактор обладает рядом полезных умений:

подсвечивает *синтаксис* команд, делает авто-отступ и авто-завершение команд, выводит необходимые подсказки, в общем, сильно облегчает жизнь программисту.

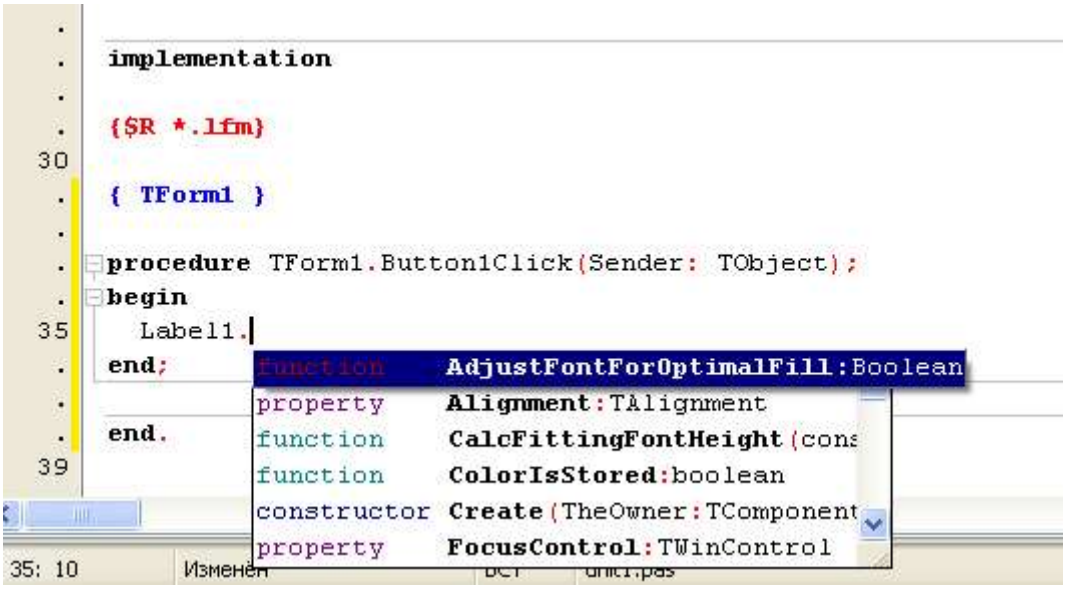


Рис. 1.6. Редактор кода

Нам придется часто переключаться между **Редактором форм** и **Редактором кода**. Проще всего это делать кнопкой **<F12>.**

И, наконец, **Окно сообщений** выводит различные сообщения: о найденных ошибках, о завершении компиляции, о наличии объявленных, но неиспользуемых переменных и т.п.

Схожесть и отличия с Delphi

Я не мог не упомянуть о **Delphi**, который является платным аналогом **Lazarus**, и его предшественником. И Lazarus, и Delphi поддерживают код Объектного Паскаля, используют визуальные и невизуальные компоненты, имеют схожий *интерфейс*. Взгляните на рисунок [3]:

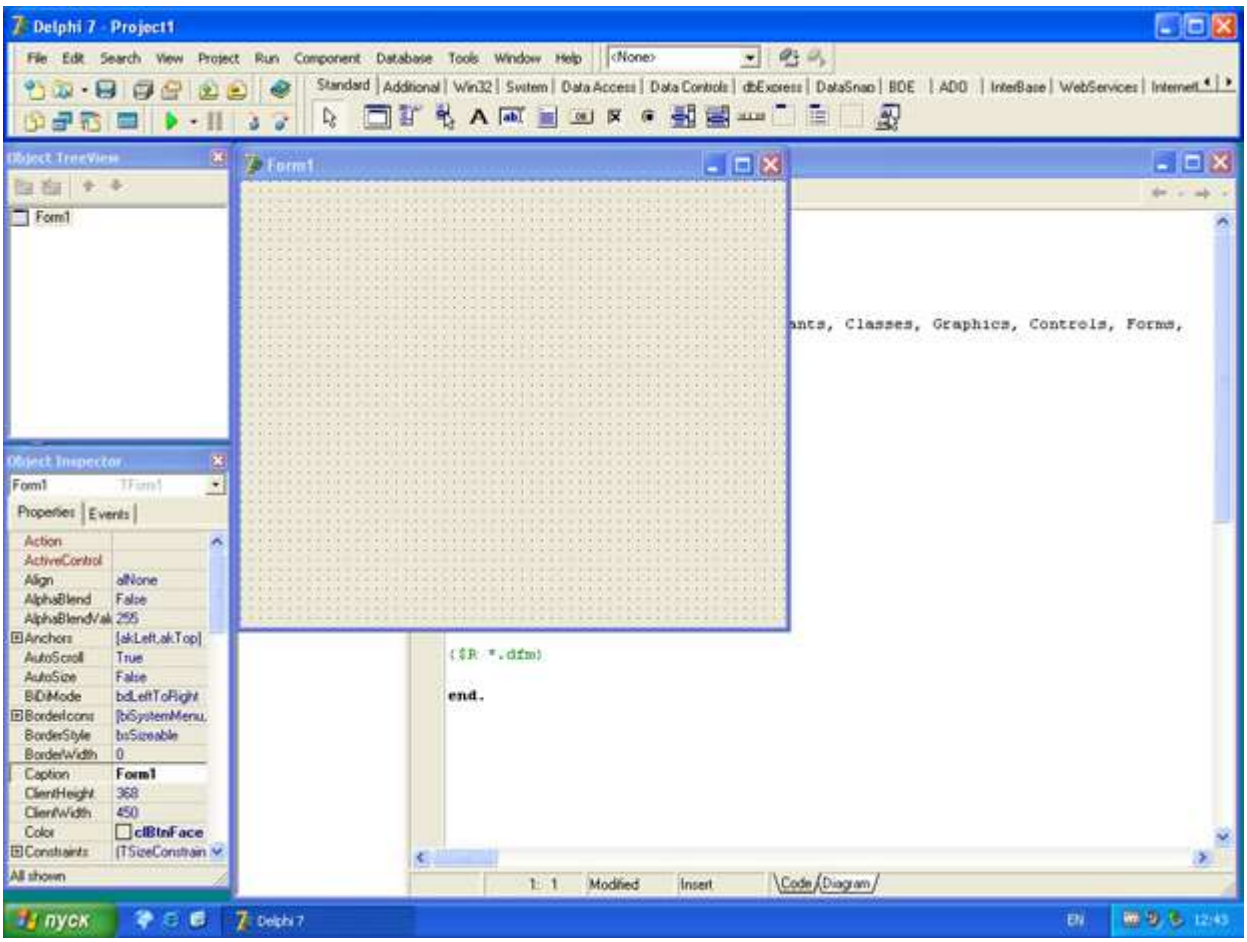


Рис. 1.7. Окна Delphi 7

Как видите, **Lazarus** очень похож на Delphi, однако есть и отличия.

- Lazarus, в отличие от Delphi, бесплатен, и может свободно и легально применяться в любом учебном, государственном или производственном учреждении, или дома.
- Lazarus имеет собственную библиотеку компонентов LCL (Lazarus Component Library), а Delphi - VCL (Visual Component Library). Однако VCL и LCL во многом так похожи, что программист при работе с компонентами почти не ощущает разницы. Часто (но не всегда) проекты, написанные на Delphi можно без потерь компилировать на **Lazarus.**
- Lazarus кросс-платформенная IDE, то есть, поддерживает различные операционные системы. Существует, правда, Kilyx - реализация Delphi для Linux, однако Lazarus имеет реализации для гораздо большего списка операционных систем, причем как 32-х, так и 64-х разрядных версий.
- Lazarus, в силу того, что моложе Delphi, пока имеет меньшую поддержку: дополнительные компоненты сторонних разработчиков, книги на русском языке, сайты, посвященные языку и т.п.
- Lazarus имеет менее развитые средства для работы с Базами Данных. Будем надеяться, это временный недостаток.
- Некоторые компоненты LCL в Lazarus еще "сырые" - иногда попадаются свойства, которые не работают. Чаще всего, это второстепенные свойства, так что можно смело использовать компоненты и без них.

Несмотря на все отличия, эти *IDE* так похожи, что можно смело утверждать - Delphi-программист почти без усилий сможет пользоваться **Lazarus**, и наоборот. А бурное развитие молодого Lazarus гарантирует, что в будущем его немногочисленные недостатки будут исправлены.

Первая программа

Прежде всего, давайте создадим где-нибудь общую папку, в которую затем будем складывать все наши проекты. Пусть это будет

C:\Education

(*education* -образование)

Каждый проект *по* правилам, должен сохраняться в отдельную папку. Чтобы не запутаться, будем давать этим папкам имена, соответствующие номеру лекции, и номеру проекта в ней (лекция-проект). То есть, в первой лекции первый проект будет сохранен в папку

C:\Education\01-01

Вы можете выработать и собственные правила наименования папок - суть от этого не изменится, лишь бы вы сами потом в них не запутались. Мы несколько раз упомянули *слово* **проект.**

Проект - *это то, что разрабатывает программист. Когда проект готов и* **скомпилирован**, *получается* **программа**, *с которой может работать пользователь. Проект - это набор связанных файлов различного типа, а программа - это полученный в результате компиляции* **исполняемый файл**. *Подробней о проектах мы поговорим в следующей лекции.*

По традиции, первая созданная *программа* должна просто выводить сообщение "Hello, world!" ("Привет, мир!"). Не будем отступать от традиций, и сделаем это тремя разными способами. Итак, создадим на диске указанные выше папки, куда сохраним наш первый проект. Загружаем **Lazarus**, если он еще не загружен, и выделяем редактор форм. В левой части, если вы не забыли, находится **Инспектор объектов**, и в нем выделена вкладка "**Свойства**" - это нам и нужно. Среди свойств найдите Caption, и вместо текста [3]

Form1

который там находится *по* умолчанию, впишите

Hello, world!

По окончании этой процедуры нажмите **<Enter>**. Как только вы это сделаете, текст в заголовке формы изменится:

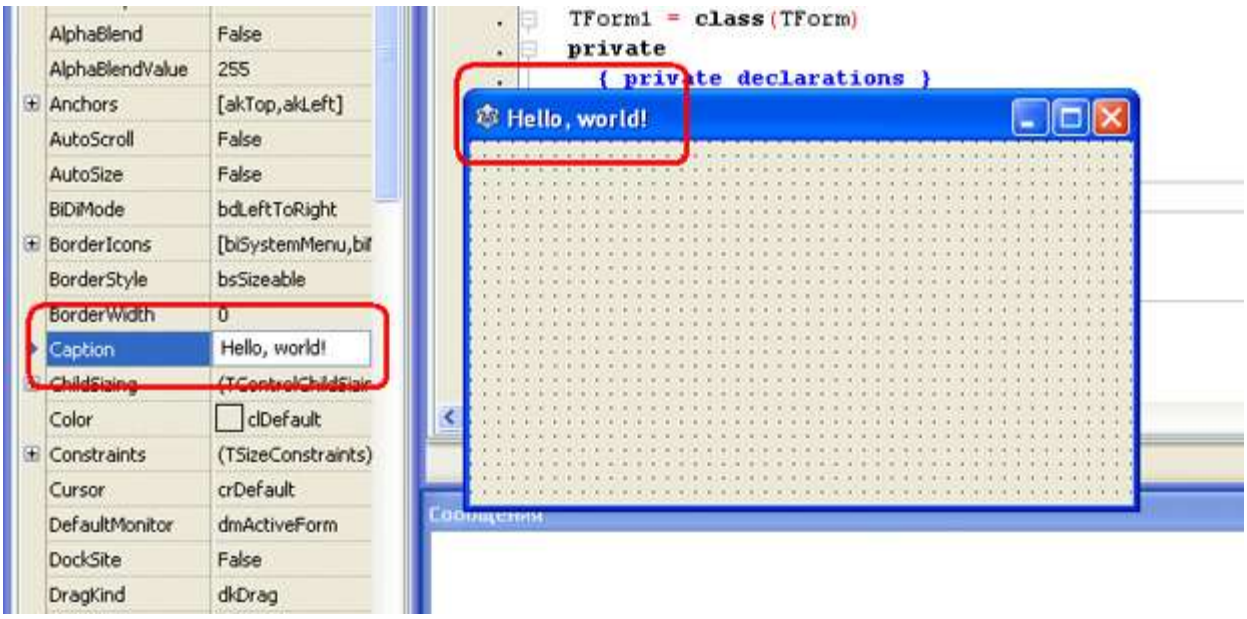


Рис. 1.8. Первые шаги

Теперь обратите внимание на **Палитру компонентов**. На вкладке **Standard**, четвертый значок изображает кнопку с надписью "**Ok**" на ней. Эта кнопка нам и нужна [3].

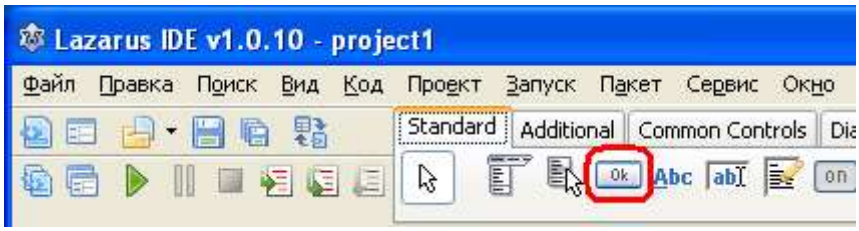
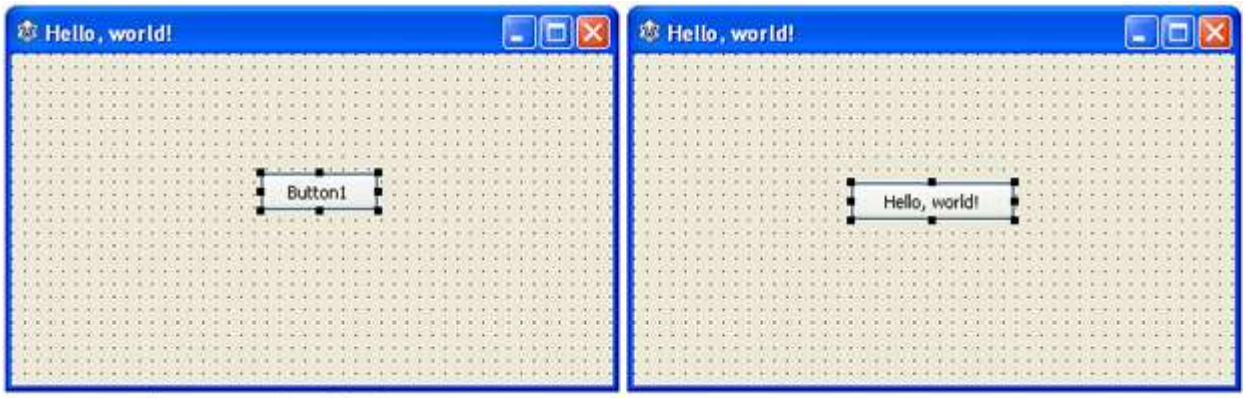


Рис. 1.9. Кнопка TButton в Палитре компонентов

Щелкните *по* ней мышью, а затем щелкните уже *по* форме, примерно *по* центру. На форме тут же появится кнопка, обрамленная рамочкой, с надписью *Button1* - такое имя **Lazarus** дал кнопке *по* умолчанию. Рамочка вокруг кнопки говорит о том, что ухватившись мышью за одну из ее сторон или углов, мы сможем менять размеры кнопки. Ухватившись за саму кнопку, мы сможем перемещать ее *по* форме [3].

Слева, в **Инспекторе объектов**, *список* свойств также изменился - некоторые остались прежними, другие добавились. Поступим, как и в прошлый раз - в свойстве Caption вместо *Button1* впишем *Hello, world!*. Затем мышью изменим размеры и расположение кнопки, чтобы у нас получилось примерно следующее:



а) кнопка до обработки *б) кнопка после обработки*

Рис. 1.10. Кнопка до и после обработки

Обратите внимание: перемещать кнопку *по* форме можно не только мышью, но и клавишами **<Ctrl> + <Кнопки управления курсором>** (стрелки вверх, вниз, влево, вправо). А изменять ее размеры - клавишами **<Shift> + <Кнопки управления курсором>**. Этот способ удобней для более точной настройки положения и размеров любых компонентов, не только кнопки.

Мы попробовали два способа вывести нужный текст. Теперь поработаем с исходным кодом. Щелкните дважды *по* кнопке, и **Lazarus** откроет **Редактор кода**, создав обработчик для этой кнопки, и установив *курсор* внутрь него. Здесь нам пока ничего понимать не нужно, просто впишите текст, прямо туда, где находится *курсор* [3]:

```
ShowMessage('Hello, world!');
```

Чтобы получилось так:

Рис. 1.11. Код обработчика кнопки

На этом наш проект закончен. Осталось сохранить его и скомпилировать в программу. Выберите команду *меню* **Файл -> Сохранить все**, или (что проще) нажмите соответствующую кнопку на **Панели инструментов**:

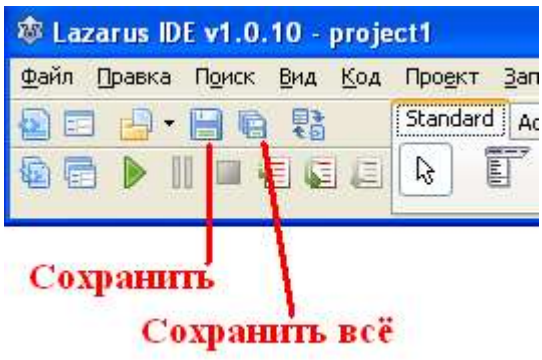


Рис. 1.12. Кнопки сохранения

Кнопка "**Сохранить**" сохраняет изменения текущей формы, кнопка "**Сохранить всё**" - изменения всех форм и модулей проекта. Поскольку форма у нас одна, то можно нажать любую из них. Как только вы это сделаете, выйдет окно сохранения проекта. Вы помните, в какую папку мы будем сохранять первый проект первой лекции? Туда и сохраняйте. Название проекта оставьте без изменений, подробней об этом мы поговорим в следующей лекции.

Как только вы сохраните проект, выйдет еще один *запрос* - на сохранение файла модуля **Unit1**. Здесь мы тоже оставляем название *по* умолчанию, нажмем лишь кнопку "**Сохранить**". Обратите внимание - кнопки "**Сохранить**" и "**Сохранить всё**" стали неактивными - это означает, что ни в форме, ни в проекте в целом у нас изменений нет [3].

Хорошо, проект мы сохранили, однако работающей программы у нас пока нет. Чтобы ее получить, нужно **скомпилировать** проект. Причем сделать это можно тремя командами **Главного меню**:

- **Запуск -> Компилировать**
- **Запуск -> Собрать**
- **Запуск -> Запустить**

Последняя *команда* не только компилирует проект и создает загрузочный *файл* программы, но и сразу запускает его на выполнение, так что в большинстве случаев предпочтительней именно эта *команда*. Удобнее всего воспользоваться соответствующей кнопкой на **Панели инструментов**:

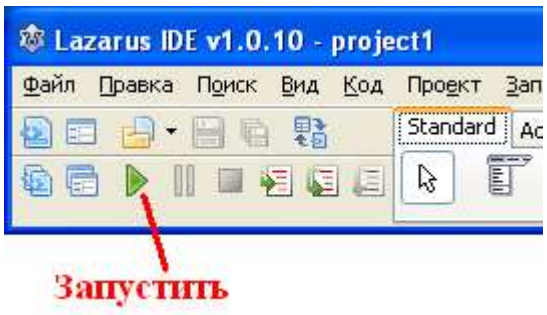


Рис. 1.13. Кнопка запуска

Нажмите эту кнопку, проект скомпилируется и сразу же запуститься. Как только мы нажмем кнопку "**Hello, world!**" в окне программы, выскочит записанное в **Редакторе кода** сообщение:

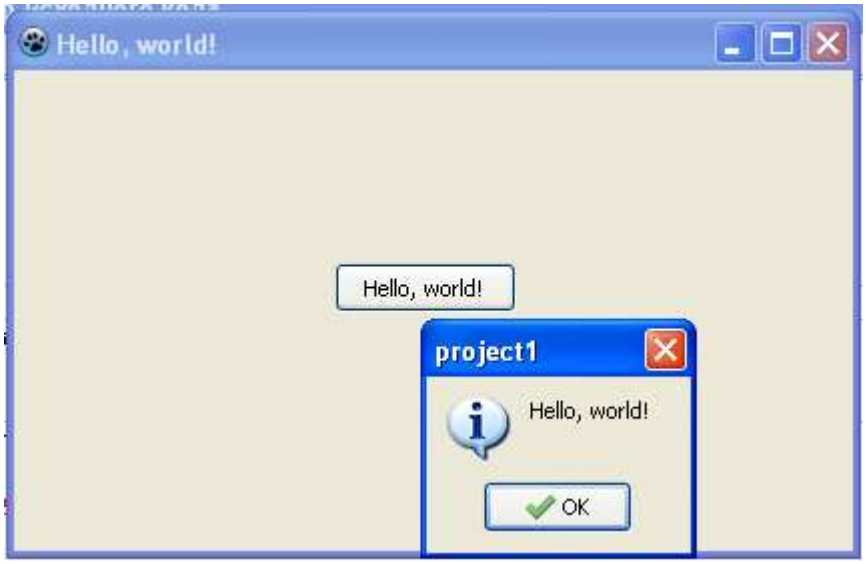


Рис. 1.14. Работа нашей программы

Нажмите кнопку "**Ок**", закрыв сообщение, после чего закройте саму программу (не проект!). Если выйдет подобное сообщение:

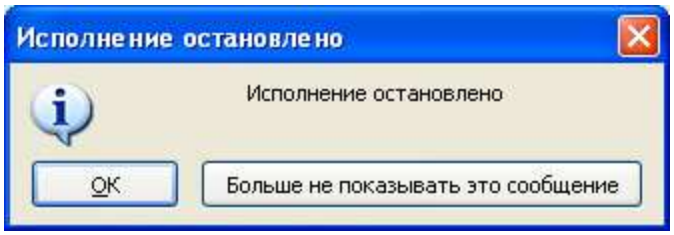


Рис. 1.15. Сообщение об остановке выполнения

то советую нажать кнопку "**Больше не показывать это сообщение**", чтобы в дальнейшем оно вам не надоедало. После этого можете закрыть и **Lazarus**. Исполняемый *файл* **project1.exe** с нашей программой вы найдете в папке, куда сохраняли проект. Это уже вполне работоспособная *программа*, её можно запустить прямо из этой папки или переслать другу, чтобы он позавидовал вашим новым знаниям (для выполнения программы требуется только один *файл* **project1.exe**, остальные файлы не нужны - это файлы проекта, которые нужны только программисту) [3].

Форма

Мы много раз употребляли *слово* **компонент**, но что это такое?

Компонент - это некий виртуальный объект, который обладает своими свойствами и событиями, находится на Палитре компонентов, и который можно установить на форму.

Форма не находится в **Палитре компонентов**, однако, это тоже *компонент*! Ведь и у формы есть свойства и события, а мы имеем возможность менять первые и программировать вторые. Познакомимся с основными свойствами формы. Откройте **Lazarus** и новое *приложение* в нем. Поскольку ничего, кроме формы, в

приложении пока нет, в Инспекторе объектов отображаются свойства формы. Причем все свойства нам не видны, придется прокручивать **Инспектор объектов**, чтобы рассмотреть остальные свойства или найти нужное.

Свойство Name нам уже знакомо - это имя компонента, в нашем случае, формы (англ. *name* - имя). *По* этому имени мы можем программно обращаться к его различным свойствам. Мы договорились, что главную форму мы будем называть *fMain*. Другими словами, свойству Name главной формы мы присвоим *значение* fMain. Так и сделаем. Как только мы переименовали форму, автоматически изменился и ее заголовок.

Свойство Caption - заголовок формы (англ. *caption* - заголовок). В это свойство впишем новый текст:

Моя форма

Как только мы впишем новое *значение* и нажмем **<Enter>**, текст заголовка изменится.

Поскольку нам придется сегодня много экспериментировать, будет нелишним сохранить новый проект. Сохраните его под именем **Proba** в папку

C:\Education\03-01

Вы еще не забыли, что *модуль* мы называем так же, как форму, но без начального "**f**"? В нашем случае, это будет **Main.**

Теперь давайте прогуляемся *по* основным свойствам формы, от начала к концу.

AlphaBlend - прозрачность. Это свойство не назовешь основным, однако в некоторых случаях оно может придать нашему приложению дополнительное изящество. Это логическое свойство, в нем мы можем выбрать либо False (англ. *false* - *ложь*) - *значение по* умолчанию, либо True (англ. *true* - *истина*). Другие значения в это свойство мы вписать не сможем. Если в этом свойстве установлено False, то форме запрещено быть прозрачной, если же True,- то разрешено. Давайте для пробы установим True.

Совет. Поменять False *на* True, *или наоборот, можно либо выбрав нужное значение, либо дважды щелкнув по изменяемому значению.*

AlphaBlendValue - *значение* прозрачности. Просто установить AlphaBlend в True недостаточно, чтобы сделать форму прозрачной. Нужно еще указать *значение* прозрачности в AlphaBlendValue. Это *значение* может изменяться от нуля (полная прозрачность) до 255 (полная непрозрачность). Другие значения вы просто не сможете установить. Имейте в виду, что если вы сделаете форму полностью прозрачной, то увидеть ее в рабочей программе не сможете. Закрывать придется либо с помощью кнопки программы в строке состояния (справа от кнопки "**Пуск**"), либо с помощью **Диспетчера задач** *Windows*, либо (если вы скомпилировали и загрузили программу командой "**Запустить**" или аналогичной кнопкой на **Панели инструментов**) выполнив команду **Lazarus** "**Запуск -> Сбросить отладчик**". Но лучше до этого не доводить. Установите *значение* в 125 (полупрозрачность), нажмите кнопку "**Запустить**" и посмотрите, что получилось:

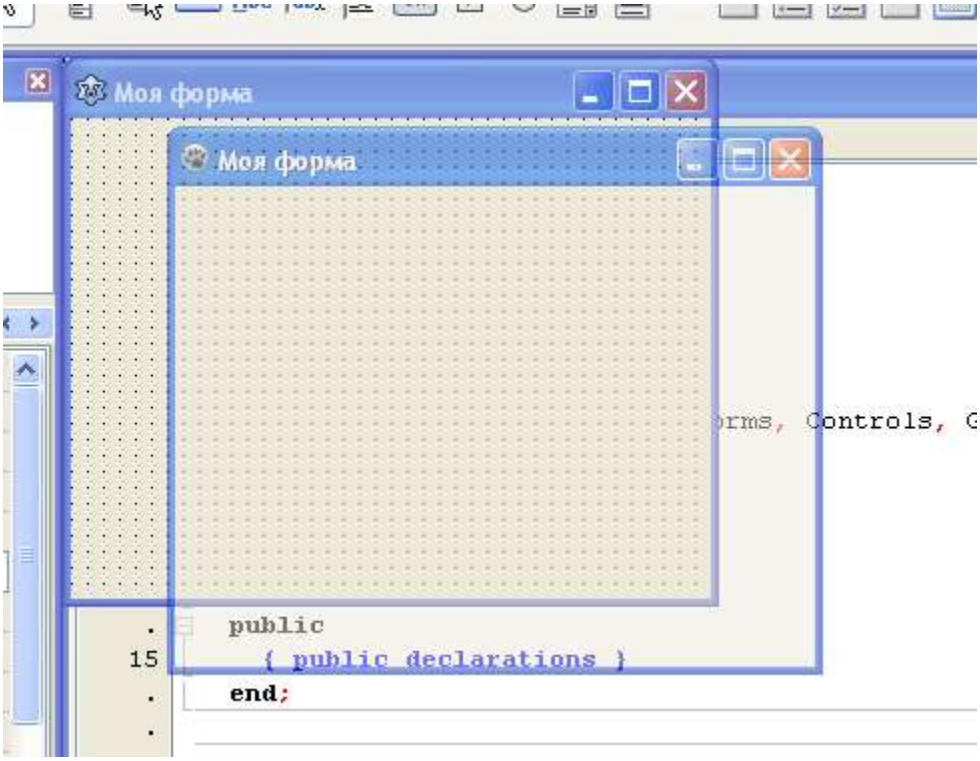


Рис. 3.1. Полупрозрачное окно

Верните в свойство AlphaBlend *значение* False, так как для дальнейших экспериментов прозрачность окна будет нам мешать [3].

AutoScroll - автоматическая прокрутка формы. Также логическое свойство. Если установлено в True, то прокрутка окна формы разрешена. При этом, если какие то компоненты на форме стали невидны, например, при уменьшении размера формы, то в правой части формы появляется *полоса прокрутки*, позволяя прокручивать форму. Оставим здесь *значение по* умолчанию False - прокрутка запрещена.

AutoSize - автоматический размер. Если установлено в True, то форма автоматически будет принимать размер, достаточный только для установленных компонентов. Имейте в виду, что увидеть этот эффект вы сможете только в работающем приложении, не в форме. Если, например, установить на форму кнопку, и запустить *приложение*, то окно сожмется *по* размеру кнопки. Не очень предсказуемый эффект, поэтому данное свойство используется редко. Можете поэкспериментировать с этим свойством, но в конце установите в нем *значение по* умолчанию False.

BorderStyle - стиль обрамления формы. Очень интересное свойство, сильно влияющее на вид окна. Может принимать следующие значения:

- bsDialog - Диалоговое окно с невозможностью изменять размеры.
- bsNone - Окно без строки заголовка. Невозможно изменить размеры, невозможно перемещать по экрану.
- bsSingle - Окно, размеры которого нельзя менять, но которое можно свернуть/развернуть с помощью кнопок строки заголовка.
- bsSizeable - Обычное окно Windows.

- bsSizeToolWin - Строка заголовка более узкая, кнопки свернуть/развернуть отсутствуют, но можно менять размеры окна мышью, ухватившись за край или угол окна.
- bsToolWindow - Как **bsSizeToolWin**, но нельзя мышью менять размеры окна.

Установите *значение* bsDialog и запустите программу, посмотрите на внешний вид полученного окна.

Font - *шрифт* формы и всех компонентов на ней. Когда вы щелкаете *по* свойству, в его правой части появляется кнопка с тремя точками [3]:

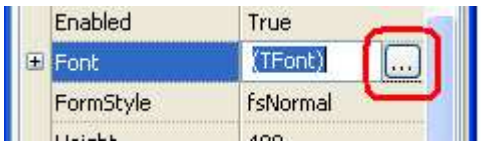


Рис. 3.2. Раскрывающее свойство Font

Если щелкнуть *по* этой кнопке, откроется диалог выбора шрифта. Если вы выберите, например, *шрифт* Times *New* Roman, обычный, размером 11 пикселей, то это скажется на всех компонентах, расположенных на форме. Они все получат этот *шрифт* "в наследство", хотя для отдельных компонентов затем можно будет выбрать и другой *шрифт*.

Height - *Высота* формы. Размер в пикселях от строки заголовка до нижней границы формы. Когда вы мышью меняете размеры формы, автоматически меняется и это *значение*. Изменить высоту окна можно и вручную, указав в этом свойстве нужный размер. Не работает, если окно показывается развернутым.

Left - Левая граница формы. *Расстояние* в пикселях от левой границы рабочего стола до левой границы формы. Не работает, если установлено положение формы *по* центру рабочего стола (экрана), или окно развернуто.

Position - Положение формы. Может принимать следующие значения:

- poDefault - положение окна определяет Windows.
- poDefaultPosOnly - размер окна соответствует проекту, а его положение определяет Windows.
- poDefaultSizeOnly - положение окна соответствует проекту, а его размер определяет Windows.
- poDesigned - значение по умолчанию, и размер, и положение определяет разработчик.
- poDesktopCenter - окно будет располагаться по центру рабочего стола.
- poMainFormCenter - данное значение применяют для дополнительных (неглавных) форм проекта. Окно будет располагаться по центру главного окна.
- poOwnerFormCenter - это значение также используется для дополнительных окон, располагая его по центру родительского окна, то есть, окна, из которого произошел вызов данного окна.
- poScreenCenter - окно располагается по центру экрана.

Совет: главное окно располагайте по центру рабочего стола, а остальные окна проекта - по центру главной формы. Такие настройки будут предпочтительней для подавляющего большинства проектов.

ShowHint - показ подсказок. Если вы подведете *указатель* мыши к какому-нибудь компоненту в **Палитре компонентов** (или к кнопке на **Панели инструментов**), но не будете нажимать кнопку, то через некоторое время появится *всплывающая подсказка* (хинт, от англ. *hint* - намёк, намекать) с именем компонента (кнопки). Свойство логическое, если установлено в True, то показ таких подсказок разрешен для всех установленных на форме компонентов, либо для самой формы.

Hint - текст самой подсказки. Попробуйте разрешить показ подсказок, а в этом свойстве напишите какой-нибудь текст, например, "*Наша всплывающая подсказка*". Запустите программу, наведите на форму *указатель* мыши, но не нажимайте на нее:

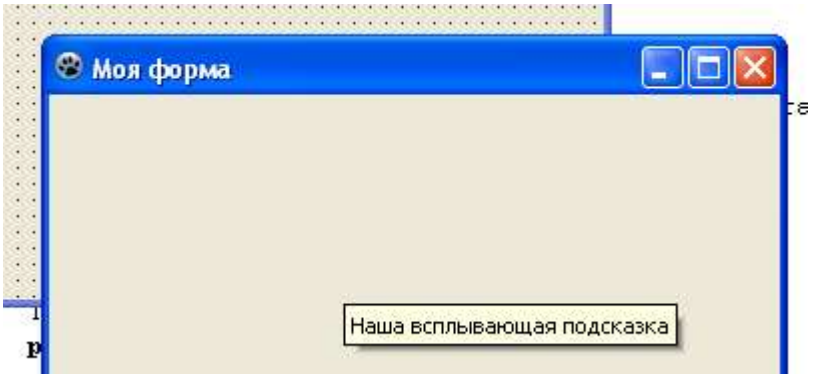


Рис. 3.3. Всплывающая подсказка

Top - *Верхняя граница* формы. *Расстояние* в пикселях от верхней границы рабочего стола до строки заголовка формы. Не работает, если установлено положение формы *по* центру рабочего стола (экрана), или окно развернуто.

Width - Ширина формы. Размер в пикселях от левой до правой границы формы. Когда вы мышью меняете размеры формы, автоматически меняется и это *значение*. Не работает, если окно показывается развернутым.

WindowState - Состояние окна. Может принимать следующие значения:

- wsFullScreen - во весь экран.
- wsMaximized - развернуто.
- wsMinimized - свернуто.
- wsNormal - обычное состояние. Это значение по умолчанию, чаще всего используется именно оно.

Для дальнейших экспериментов давайте установим следующие параметры формы:

BorderStyile = bsSizeable
Height = 400
Position = poDesktopCenter
Width = 600
WindowState = wsNormal

Панель и TSplitter

Панель - это своего рода *контейнер*, для размещения на ней связанных *по* какому-то признаку компонентов. Размещенные на панели

компоненты "привязываются" к ней. При перемещении панели *по* форме, будут перемещаться и эти компоненты. Панель нередко используют и для украшения формы. На форме может быть несколько панелей [3].

После изучения предыдущего раздела, у вас должен быть открыт проект с пустой формой, настроенной определенным образом. Если это так, установим на форму панель. Для этого щелкните *по* компоненту **TPanel** в палитре компонентов, а затем щелкните *по* форме, установив на нее панель [3]:

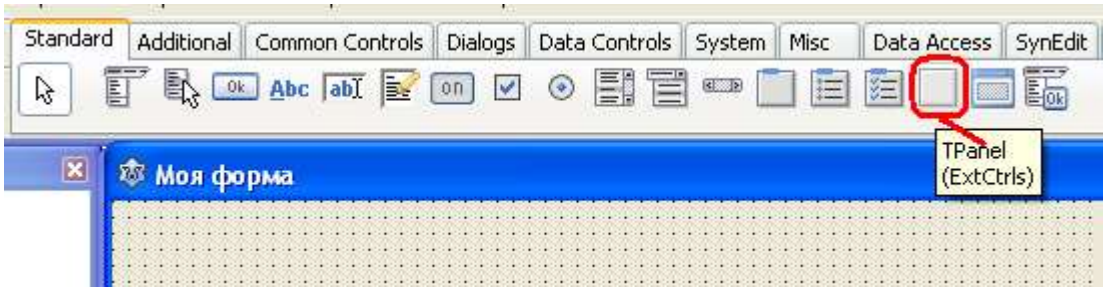


Рис. 3.4. Компонент TPanel

У вас на форме появится некое прямоугольное образование, в центре которого будет написано *Panel1* - имя компонента, которое **Lazarus** присвоил *по* умолчанию. Если мы будем устанавливать другие панели, **Lazarus** назовет их *Panel2, Panel3* и т.д. Если на форме у нас будет только одна панель, то цифру в названии (свойство Name) и вовсе можно будет убрать. В данном случае у нас будет несколько панелей, так что оставим панели имя, которое ей присвоил **Lazarus** *по* умолчанию. Рассмотрим некоторые свойства панелей, которыми нам часто придется пользоваться (*компонент* Panel1 при этом должен быть выделен).

Name - имя панели. Если будем ее переименовывать, то в этом свойстве.

Caption - текст в центре панели, в нашем случае, это *Panel1*. Этот текст очень редко используется, обычно его удаляют, чтобы панель была пустой. Мы тоже очистим этот текст.

Left, Top - расстояние в пикселях, соответственно, от левой и верхней границ формы до панели. Установим в обоих свойствах значение 10.

Height - высота панели в пикселях, установим ее в 320.

Width - ширина панели в пикселях, установим ее в 580.

Теперь сохраните изменения в проекте и запустите его. Вы увидите окно с выпуклой панелью, под которой осталось *место* для кнопки. Так обычно оформляют диалоговые окна. Однако выпуклость панели можно и убрать, превратив ее в аккуратную рамочку. Закройте окно и вернитесь к проекту. Рассмотрим три свойства, влияющих на выпуклость панели.

BevelInner - определяет стиль внутренней границы панели. Может иметь значения:

- bvNone - никакого стиля. (по умолчанию для BevelInner)
- bvSpace - пустая отрисовка границы, визуально не отличается от bvNone.
- bvRaised - приподнятая граница. (по умолчанию для BevelOuter)
- bvLowered - утопленная граница.

BevelOuter - определяет стиль внешней границы панели. Может иметь такие же значения, как и BevelInner.

BevelWidth - ширина окантовки в пикселях. *По* умолчанию, имеет *значение* 1, что обычно вполне достаточно. Чтобы еще больше подчеркнуть границу, можно установить *значение* 2, однако дальнейшее увеличение ширины не сделает панель красивей, скорее, наоборот. Впрочем, это зависит от вашего вкуса и конкретного дизайна.

Совет: чтобы панель сделать вогнутой, свойство BevelOuter *установите в* bvRaised, *а чтобы сделать красивую окантовку, сделайте* BevelInner = bvLowered, *а* BevelOuter = bvRaised. *Для ширины окантовки вполне хватит 1 пикселя.*

Попробуйте "поиграть" со свойствами панели: меняйте их, запускайте проект на выполнение, и смотрите, как будет меняться внешний вид панели. А когда наиграетесь и запустите программу в последний раз, разверните окно во весь экран. Вот это уже неприятно: окно программы увеличилось, а панель осталась прежней:

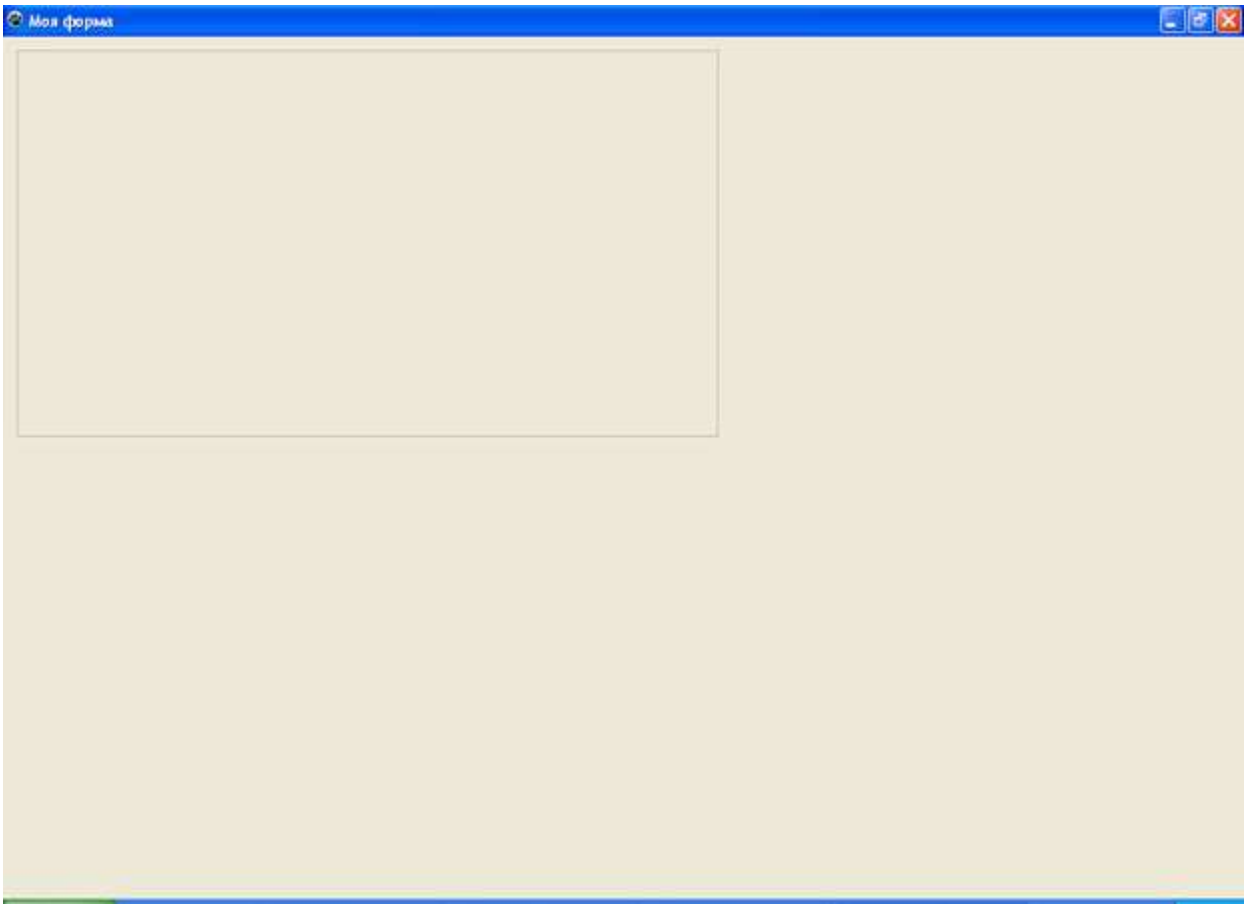


Рис. 3.5. Искажение дизайна окна при изменении размеров

Дело в том, что панель "привязывается" к верхней и левой границе формы. А при изменении размеров окна, правая и нижняя границы панели не изменяются. Чтобы "привязать" к форме все стороны панели, служат "якоря":

Anchors (англ. якоря) - свойство, разрешающее или запрещающее привязку сторон к внешнему контейнеру. Если панель располагается прямо на форме, то и привязывается она к форме. А если панель располагается на другой панели, то привязывается уже к ней. Подобное свойство имеет большинство визуальных

компонентов. Anchors - раскрывающееся свойство. Если щелкнуть *по* кнопке "+" слева от него, то свойство раскроется, показывая привязки сторон, а кнопка "+" превратится в "-":

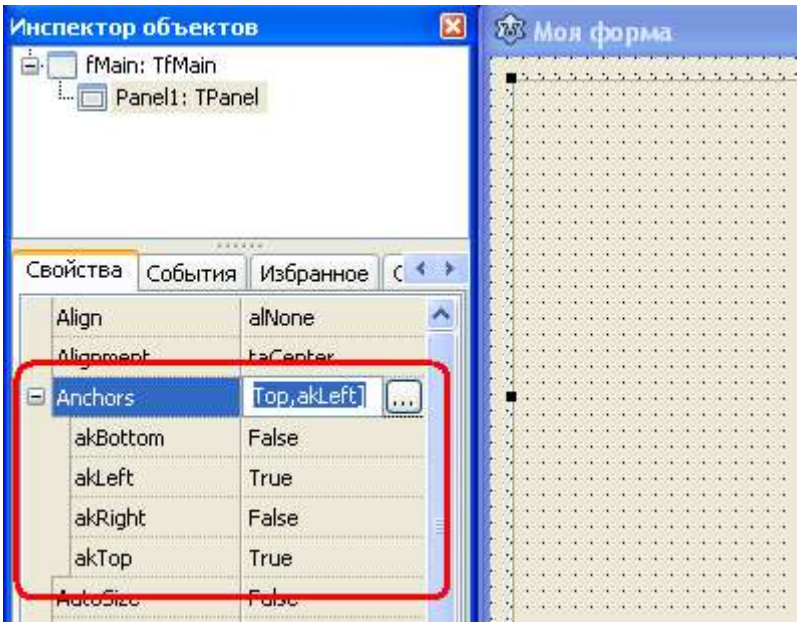


Рис. 3.6. Раскрытое свойство Anchors

Как мы видим, *по* умолчанию "привязаны" только левая и верхняя границы (akLeft=True, akTop=True). Чтобы разрешить привязку правой (akRight) и нижней (akBottom) границ панели, их тоже нужно установить в True. Сделайте это, запустите программу на выполнение и попробуйте изменять размеры окна. Теперь вместе с окном изменяет размеры и панель, что нам и требовалось.

Однако есть и другой способ привязки компонентов к различным сторонам формы или контейнера, на котором *компонент* находится. Поскольку этот способ применяется довольно часто, разберем его подробней. Для начала удалите панель - выделите ее на форме и нажмите **<Delete>**. Затем установите на форму новую панель. Таким образом, мы просто сбросили все изменения в настройках панели, ведь новая панель имеет все значения *по* умолчанию. А имя у нее такое же - **Panel1.** Очистим свойство Caption, и займемся выравниванием [3].

Align (англ. выравнивание) - свойство, позволяющее "прилепить" панель к одной из сторон внешнего контейнера, или ко всему контейнеру. Может иметь следующие значения:

- alBottom - Панель занимает весь низ контейнера (в нашем случае контейнером является форма).
- alClient - Панель занимает весь контейнер.
- alCustom - Пользовательское выравнивание, то же самое, что alNone.
- alLeft - Панель занимает всю левую часть контейнера.
- alNone - Нет выравнивания. Это значение по умолчанию.
- alRight - Панель занимает всю правую часть контейнера.
- alTop - Панель занимает весь верх контейнера.

Если мы установим это свойство в alClient, то наша панель займет всю форму. При изменении размеров формы будут изменяться и размеры панели. Если установим, к примеру, в alTop, то панель займет всю верхнюю часть формы. Если затем на этой панели установим кнопки, то получим **Панель инструментов**. А совместно с другим компонентом, TSplitter, который находится на вкладке **Additional Палитры компонентов** и представляет собой перемещаемую границу-разделитель, мы получим различные части окна, размеры которого *пользователь* сможет менять мышью во *время выполнения* программы [3]:

Рис. 3.7. Разделитель TSplitter

Давайте сделаем так. У нашей Panel1 установим Align = alTop. Затем добавим на форму разделитель TSplitter, и также установим у него Align = alTop.

Теперь добавим на свободное *место* формы еще одну панель. Очистим у нее свойство Caption, и установим Align = alLeft. Добавим еще один TSplitter, убедимся, что у него также Align = alLeft. Теперь весь верх и всю левую часть формы занимают панели, отгораживаясь от свободной части сплиттерами. На эту свободную часть формы мы установим третью панель. Очистим у нее свойство Caption и установим Align = alClient. Панель заняла всю оставшуюся часть формы. Теперь запустите проект на выполнение и убедитесь, что мышью можно перемещать границы-разделители, меняя размеры панелей. При изменении размеров окна изменяются и размеры панелей [3]:

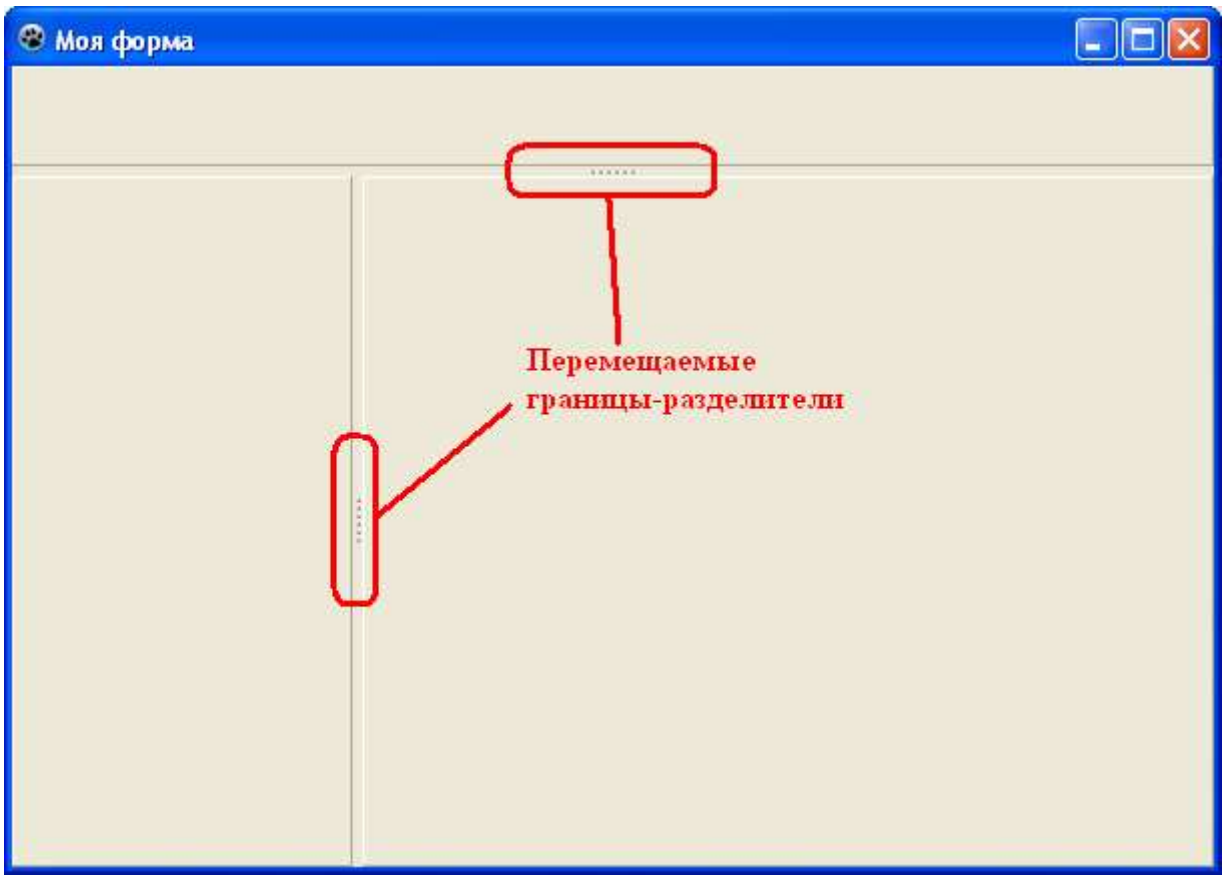


Рис. 3.8. Три панели и два разделителя на форме

Не забудьте сохранять проект время от времени.

Мы упустили еще одно полезное свойство панели - автоматическую настройку размера.

AutoSize - автоматическая настройка размера, логическое свойство. *По* умолчанию, установлено *значение* False - автонастройка запрещена. Если установить свойство в True, то автоматическая настройка будет разрешена.

Установите на верхнюю панель кнопку **TButton**. Свойства Left и Top кнопки установите в 1. Теперь выделите панель, на которой находится кнопка, и переведите свойство AutoSize в True. Верхняя панель *по*-прежнему занимает всю верхнюю часть формы, но теперь ее *высота* уменьшилась *по* высоте кнопки! Таким способом можно создавать простые панели инструментов для приложения. Правда, если вы запустите программу на выполнение, то увидите, что теперь верхним разделителем невозможно менять высоту верхней панели, ведь она у нас автоматически принимает размер под расположенную на ней кнопку. Так что и надобность в первом сплиттере в этом случае отпадает.

Кнопка TButton

Кнопки, пожалуй, самый используемый *компонент* в программах *Windows*. Они повсюду: в диалоговых окнах, в панелях инструментов, в сообщениях. Познакомимся поближе с этими элементами.

Итак, простая кнопка **TButton**. Находится на вкладке **Standard Палитры компонентов**. Мы уже использовали несколько раз этот *компонент*, и сейчас в проекте одна кнопка находится на верхней панели. Выделите кнопку, и посмотрим

на ее свойства. С большинством из них вы уже знакомы, познакомимся с некоторыми другими.

Cancel - логическое свойство, *по* умолчанию равно False. Если равно True, то нажатие на кнопку **<Esc>** будет равносильно нажатию на эту кнопку.

Default - логическое свойство, *по* умолчанию равно False. Если равно True, то нажатие на кнопку **<Enter>** будет равносильно нажатию на эту кнопку. Однако если фокус находится на какой-нибудь другой кнопке, то нажатие на **<Enter>** нажмет ее.

ModalResult - полезное в некоторых случаях свойство, особенно в диалоговых окнах. Может иметь значения [3]:

- mrNone (по умолчанию) - нет результата
- mrOk - OK
- mrCancel - Отмена
- mrAbort - Прервать
- mrRetry - Повторить
- mrIgnore - Игнорировать
- mrYes - Да
- mrNo - Нет
- mrAll - Все
- mrNoToAll - Нет для всех
- mrYesToAll - Да для всех
- mrClose - Закрыть

Любая форма также имеет свойство ModalResult. Если у кнопки в этом свойстве присвоить одно из этих значений, и этой кнопкой закрыть форму, то и у формы свойство ModalResult станет таким, как у кнопки. Предположим, у вас имеется *диалоговое окно* с тремя кнопками: "**Да**", "**Нет**" и "**Отмена**". И каждой из этих кнопок в свойстве ModalResult вы присвоили соответственное *значение*. Если каждая из этих кнопок будет закрывать окно, то при его закрытии можно посмотреть свойство ModalResult формы и определить, какой кнопкой *пользователь* закрыл окно. И в соответствии с результатом, выполнять дальнейшие действия.

Примечание. Просто присваивание свойству ModalResult *кнопки какого то значения не закроет окно, это действие нужно программировать отдельно.*

Hint - текст всплывающей подсказки. Эта подсказка всплывет, когда *пользователь* подведет *указатель* мыши к кнопке. Но подсказка появится, только если ShowHint = True.

Теперь будем разбираться с событиями. Однако прежде измените свойство Caption кнопки, и впишите туда *слово* Выход. Свойство Name оставьте *по* умолчанию.

Двойной щелчок по кнопке приведет к тому, что **Lazarus** сгенерирует событие - нажатие на эту кнопку. Сразу же **Редактор кода** окажется наверху, и вы увидите мигающий *курсор* там, где нужно запрограммировать это событие. Код, который мы туда впишем, будет выполняться всякий раз, когда *пользователь* нажмет на эту кнопку. Впишем туда короткое

```
Close;
```

Именно так, с точкой с запятой в конце. Это оператор закрытия окна. А если это окно главное, то значит, закроется вся *программа*:

```
35  { TfMain }

    procedure TfMain.Button1Click(Sender: TObject);
    begin
39    Close;
40  end;

    end.
43
```

Рис. 3.9. Программирование события нажатия на кнопку

С процедурами мы будем разбираться позже, пока обратите внимание на имя процедуры: Button1Click. Button1 - это имя нашей кнопки, а событие, которое мы программировали, называется OnClick - событие нажатия на кнопку. В **Инспекторе объектов** перейдите на вкладку **События**, и найдите событие OnClick. Как видите, там уже прописана именно эта процедура [3]:

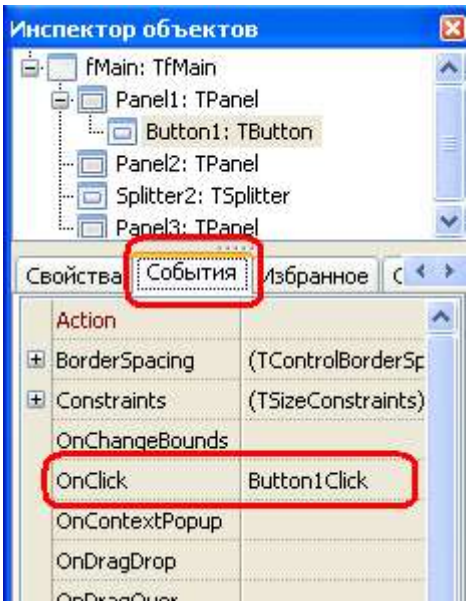


Рис. 3.10. Событие OnClick кнопки

Если мы выделим это событие, то справа от него появится кнопка "**…**", нажатие на которую приведет к генерированию этого события. Поскольку у нас оно уже запрограммировано, то просто откроется эта процедура и внутрь нее установится *курсор*. Вот вам второй способ генерировать нужное событие, выбирайте, что вам удобней. Нам не раз придется переключаться в Инспекторе объектов между вкладками **Свойства** и **События**.

Другие события кнопок нам пока не нужны. Сохраните проект и запустите его. Убедитесь, что нажатие на кнопку приводит к закрытию программы.

Кнопкой **<F12>** верните на передний план **Редактор форм**, а в **Инспекторе объектов** верните вкладку **Свойства**. В **Палитре компонентов** откройте

вкладку **Additional**. Первые два компонента в этой вкладке - это кнопки **TBitBtn** и **TSpeedButton**:

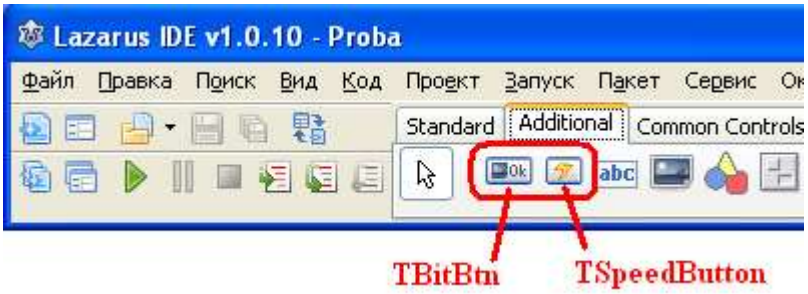


Рис. 3.11. Кнопки TBitBtn и TSpeedBtn

Поставьте две кнопки **TBitBtn** и одну **TSpeedButton** куда-нибудь на нашу форму, например, на верхнюю панель, правее нашей **Button1**. Будем разбираться, чем они отличаются от обычной кнопки.

Кнопка TBitBtn

Это более продвинутая кнопка, имеющая некоторые дополнительные свойства. Прежде всего, вы заметили, что она несколько более высокая, чем обычная кнопка. **TBitBtn** помимо текста может содержать и изображение. Для этого используются несколько свойств.

Glyph - Изображение. С помощью этого свойства можно найти и загрузить изображение, которое будет отображаться на кнопке.

Примечание: Большинство изображений хранится в папке **Images** *там, куда вы установили Lazarus. Если вы оставили адрес по умолчанию, это будет*

C:\lazarus\images

Там хранится несколько папок, где изображения рассортированы по тематикам. Однако необязательно использовать только эти изображения. В Интернете можно найти огромное количество бесплатных коллекций изображений, которые можно использовать в своих проектах.

Выделите свойство Glyph кнопки **BitBtn1** и щелкните *по* кнопке "**…**" справа от него. Откроется диалог выбора изображения. В этом диалоге нажмите кнопку "**Загрузить**" и найдите папку

C:\lazarus\images\menu

(если вы указывали другой *путь* при установке **Lazarus**, то укажите его). Откроется *список* с картинками, которые хранятся в этой папке. При выборе она будет отображаться в правой части окна. Найдите картинку **menu_exit** и нажмите кнопку "**Открыть**", затем "**ОК**". Выбранное изображение окажется на кнопке, слева от надписи. Кстати, чтобы уж придать кнопке соответствующий вид, измените свойство Caption на Выход. А заодно, сгенерируйте событие нажатия на эту кнопку, и впишите код закрытия, как это делали для простой кнопки. Сохраните проект и запустите его; убедитесь, что нажатие на эту кнопку также закрывает программу.

Однако с этой кнопкой мы еще не закончили. Выделите ее в **Редакторе форм** и обратите внимание на свойства кнопки.

GlyphShowMode - политика показа или сокрытия изображения. Может быть:

- gsmAlways - всегда показывать изображение.

- gsmNever - никогда не показывать изображение.
- gsmApplication - изображение будет показано во время выполнения программы. Значение по умолчанию для этого свойства.
- gsmSystem - будет показано изображение или нет, зависит от текущих настроек системы.

Данное свойство лучше не изменять.

Layout - *место* расположения изображения. Может быть:

- blGlyphBottom - изображение ниже текста
- blGlyphLeft - изображение слева от текста (значение по умолчанию)
- blGlyphRight - изображение справа от текста
- blGlyphTop - изображение над текстом

Данные настройки целиком зависят от конкретного стиля интерфейса, который вы выбрали для вашей программы. Но в большинстве случаев, *значение по* умолчанию - то, что нужно.

Spacing - расстояние в пикселях от изображения до текста. Это свойство меняют также редко.

TabOrder - порядковый номер выделения компонентов. Имеется у многих визуальных компонентов. Определяет, в каком порядке компоненты будут получать фокус ввода при нажатии клавиши **<Tab>**. Нумерация начинается с 0, установить одинаковый номер для двух компонентов невозможно, если они находятся на одном контейнере.

Enabled - доступность компонента. Также присутствует у многих визуальных компонентов. Логическое свойство. Если True, то компонент доступен (кнопку можно нажимать), если False, то недоступен (кнопка серого цвета, нажать на нее невозможно). Измените это свойство, запустите программу и посмотрите, как изменилась кнопка. Затем верните значение по умолчанию.

Visible - видимость компонента. Если True, то компонент видимый, иначе компонент спрятан. "Поиграйте" и с этим свойством, не забудьте вернуть его в конце в значение по умолчанию.

Мы не зря установили на форму две кнопки **TBitBtn**. Для изучения следующего свойства выделите BitBtn2.

Kind - разновидность кнопки. При выборе какого-то значения на кнопке появляется соответствующие текст и картинка, а кнопка сразу обладает нужными свойствами, зависящими от ОС - закрывает окно (в случае если Kind = bkClose), присваивает форме нужное *значение* ModalResult. Правда, в работающей программе (*по* крайней мере, для *Windows* XP *Professional* SP3) текст на кнопке будет на английском языке. Может быть:

- **bkAbort** - Прервать
- **bkAll** - Все
- **bkCancel** - Отмена
- **bkClose** - Закрыть
- **bkCustom** - Пользовательские настройки - текст и надпись устанавливает программист. Это значение по умолчанию.
- **bkHelp** - Справка

- **bkIgnore** - Пропустить
- **bkNo** - Нет
- **bkNoToAll** - Нет для всех
- **bkOK** - OK
- **bkRetry** - Повтор
- **bkYes** - Да
- **bkYesToAll** - Да для всех

Установите Kind = bkClose, сохраните проект и запустите его. Убедитесь, что теперь и кнопка **BitBtn2** закрывает окно. Для этого нам даже не пришлось ее программировать!

Кнопка TSpeedButton

Напоследок поработаем с этим типом кнопок. Они чаще всего применяются для организации панелей инструментов. Дело в том, что эта кнопка не имеет фокуса ввода! Что я имею в виду? Запустите программу на выполнение и клавишей **<Tab>** перемещайтесь *по* кнопкам. Вы увидите, что можно выделить любую кнопку, кроме **SpeedButton1**. На такую кнопку можно нажать только мышью, при этом, если окно не закроется, фокус вернется к тому компоненту, который был выделен ранее [3].

Как видите, кнопка квадратная, имеет малый размер и рассчитана только на то, чтобы с помощью свойства Glyph в нее загрузили изображение. Однако она имеет такие же свойства, как TBitBtn: Caption, Layout, Space и т.д., поэтому в данной кнопке также можно использовать текст совместно с изображением, или только текст, или только изображение.

Поскольку кнопка не имеет фокуса ввода, свойства TabOrder у нее также нет. Отсутствует и свойство Kind. Но в остальном она похожа на **TBitBtn**.

С помощью свойства Glyph загрузите картинку **menu_exit**, как в прошлом примере. Однако программировать событие OnClickмы не будем - зачем писать один и тот же код? Можно поступить проще. Перейдите на вкладку "**События**" **Инспектора объектов**, и, не нажимая в событии **OnClick** на кнопку "**…**", откройте доступный *список* [3]:

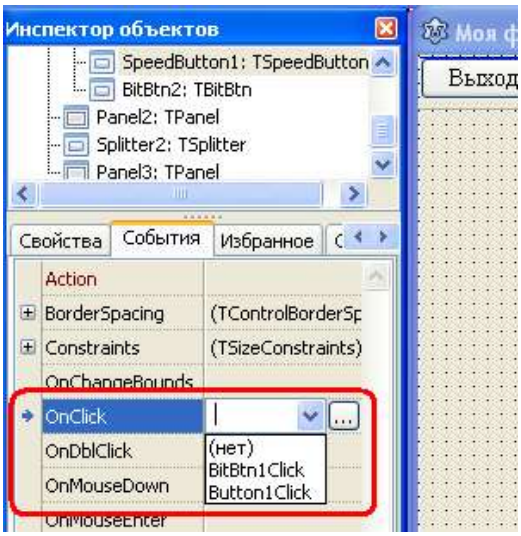


Рис. 3.12. Выбор события OnClick

Выберите в списке Button1Click, сохраните проект и запустите его на выполнение. Убедитесь, что наша SpeedButton1закрывает программу - это срабатывает событие первой кнопки.

Напоследок исправим один недостаток. Если вы обратили внимание, у нас на форме 4 кнопки, но при установке указателя на любую из них всплывает та самая подсказка, которую мы писали в свойстве Hint формы: "Наша *всплывающая подсказка*". А это не очень хорошо. Нужно в свойстве Hint кнопок написать соответствующие тексты, например:

- Кнопка Button1
- Кнопка BitBtn1
- Кнопка BitBtn2
- Кнопка SpeedButton

Конечно, в реальных приложениях в свойстве Hint вы будете писать подходящий текст подсказки, например, "*Выход* из программы". Но у нас все кнопки выполняют *выход* из программы, и чтобы отличить их друг от друга, мы укажем соответствующий текст. Сохраните проект и запустите его на выполнение; убедитесь, что текст всплывающих подсказок меняется.

Контрольные вопросы:

1. Дизайнер Форм (Form Designer).
2. Окно Редактора Исходного Текста (Editor Window).
3. Палитра Компонент (Component Palette).
4. Инспектор Объектов (Object Inspector).
5. Справочник (On-line help).
6. Консольное приложение.

Лабораторная работа 13. *Ввод/вывод данных. Программная реализация линейных алгоритмов в интегрированной среде разработки. Отладка (тестирование) программы.*

Задание: Решить следующие задачи по образцу:

Теоретический материал

При линейном программировании одни операторы идут строго за другими. На практике редко встречаются задачи, алгоритм решения которых является линейным. Часто оказывается, что алгоритм решения даже элементарной задачи не является линейным. Например, пусть надо вычислить по формуле ток в электрической цепи. Если предположить, что пользователь всегда будет вводить верные данные, то алгоритм решения этой задачи действительно является линейным. Однако полагаться на то, что пользователь будет вести себя так, как надо программе, не следует. Формула расчета предполагает, что величина сопротивления не равна нулю. А что будет, если пользователь введет 0? Ответ простой: возникнет ошибка "Деление на ноль", и программа аварийно завершит работу. Можно, конечно, возложить ответственность за это на пользователя, но лучше внести изменения в алгоритм решения, чтобы расчет выполнялся только в том случае, если введены верные данные.

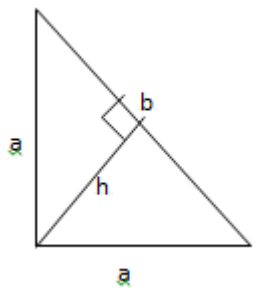
Задача № 1

Постановка задачи:

В равнобедренном прямоугольном треугольнике известна высота h, опущенная на гипотенузу. Найти стороны треугольника.

Математическая модель:

В равнобедренно прямоугольном треугольнике высота, опущенная на гипотенузу, делит треугольник на два равных равнобедренных прямоугольных треугольника.



b = a * 2 ^ 0.5

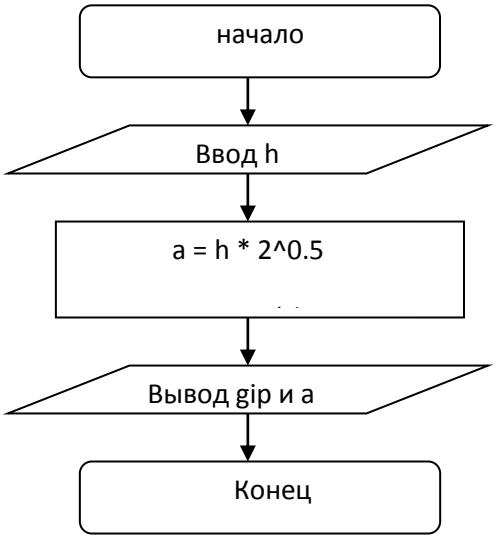
b = 2 * h

По теореме Пифагора получаем:

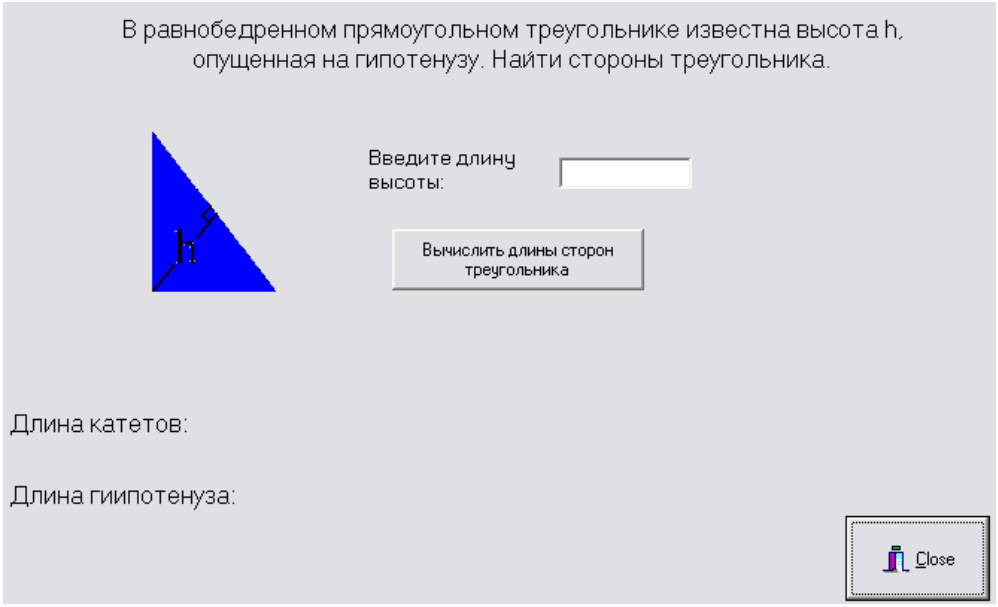
a^2 = h^2 + h^2

По данной формуле находим длины катетов треугольника, а по формуле b = 2 * h находим длину гипотенузы [3].

Блок-схема:

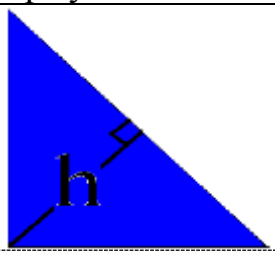


Описание интерфейса:



В таблице показаны свойства объектов

Объекты и их свойства

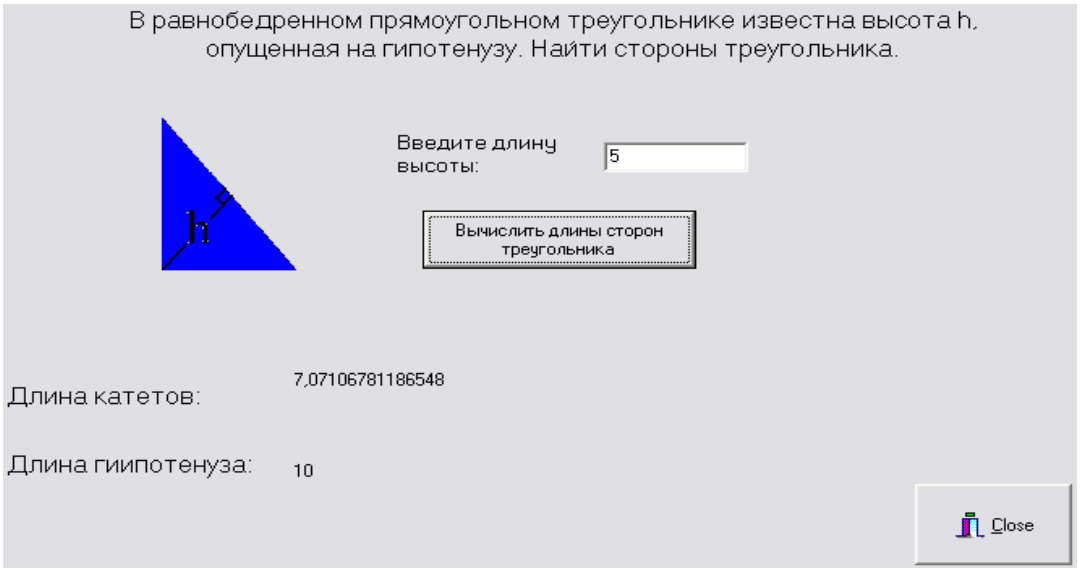
Объект	Свойство	Устанавливаемое значение
Label1	Caption	В равнобедренном прямоугольном треугольнике известна высота h, опущенная на гипотенузу. Найти стороны треугольника.
Image1	Picture	
Label2	Caption	Введите длину высоты:
Edit1	Text	“ ”
Button1	Caption	Вычислить длины сторон треугольника
	WordWrap	True
Label3	Caption	**Длина катетов:**
Label4	Caption	**Длина гиипотенуза:**
BitBtn1	Kind	bkClose

Текст программы:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Buttons;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Image1: TImage;
    BitBtn1: TBitBtn;
    Label2: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Label3: TLabel;
    Label4: TLabel;
```

```
    Label5: TLabel;
    Label6: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
var h, a, gip : real;
var a1,gip1: string;
   procedure TForm1.Button1Click(Sender: TObject); // Щелчек по командной кнопке
begin //Начать работу
    h := StrToFloat(Form1.Edit1.Text); //Высоту h берем из текстового поля и переводим в вещественный тип
    a := h*sqrt(2); //Вычисляем длину катета
    gip:= 2*h; // ычисляем длину гипотенузы
    a1:=FloatToStr(a); // Переводим вещественные величины в строковые
    gip1:=FloatToStr(gip); // Переводим вещественные величины в строковые
    Form1.Label5.Caption := a1; //Выводи результат на форму
    Form1.Label6.Caption := gip1 //Выводи результат на форму
end;
end.
```

Тестирование:



Задача № 2

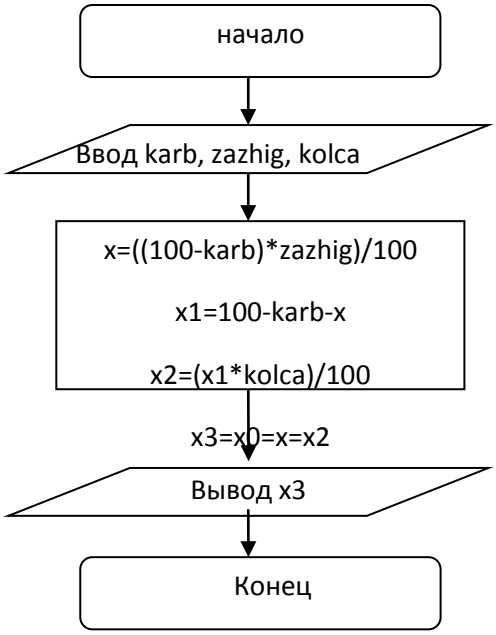
Постановка задачи:

Владелец автомобиля приобрел новый карбюратор, который экономит 50% топлива, новую систему зажигания, которая экономит 30 % топлива, и поршневые кольца, экономящие 20% топлива. Верно ли, что его автомобиль теперь сможет обходится совсем без топлива? Найти фактическую экономию для произвольно заданных сэкономленных процентов [3].

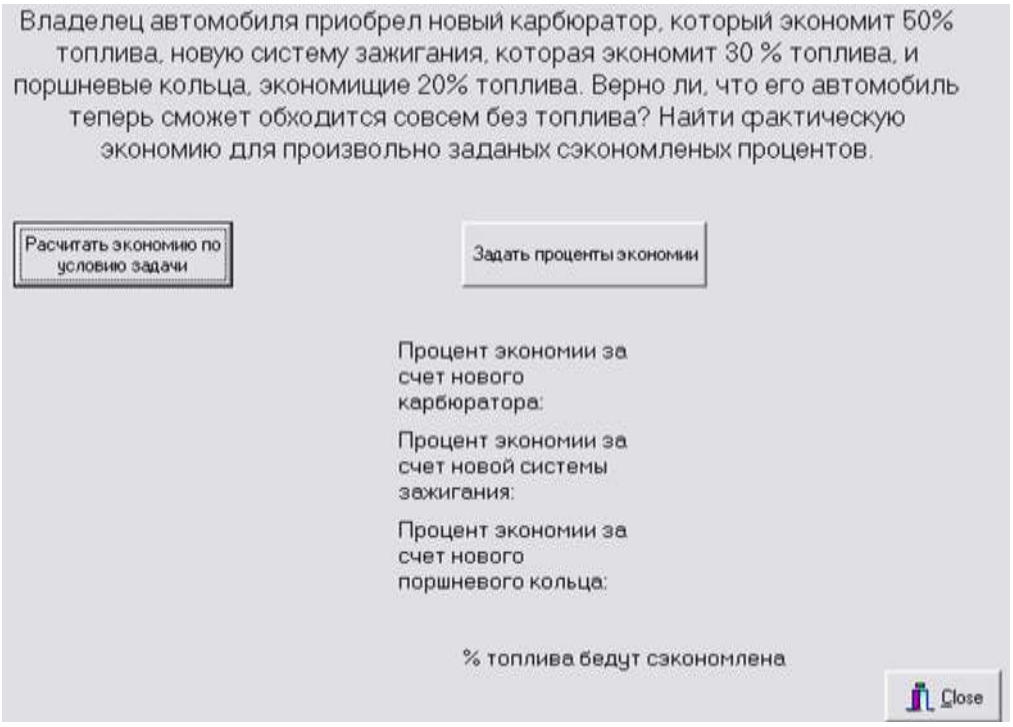
Математическая модель:

1.От 100% отнять процент экономии за счет нового карбюратора
2.Оставшейся процент взять за 100%, и решить пропорцию
3.От оставшегося процента полученного в п. 1 отнять результат полученный в п.2
4.повторить пункты 2 и 3

Блок-схема:



Описание интерфейса:



В таблице показаны свойства объектов

Объекты и их свойства

Объект	Свойство	Устанавливаемое значение
Label1	Caption	Владелец автомобиля приобрел новый карбюратор, который экономит 50% топлива, новую систему зажигания, которая экономит 30 % топлива, и поршневые кольца, экономищие 20% топлива. Верно ли, что его автомобиль теперь сможет обходится совсем без топлива? Найти фактическую экономию для произвольно заданных сэкономленых процентов.
Label3	Caption	% топлива будет фактически сэкономленно
Label4	Caption	Процент экономии за счет нового карбюратора::
Button2	Caption	Ввести проценты
Button1	Caption	Расчитать экономию по условию задачи
Label5	Caption	Процент экономии за счет новой системы зажигания:
Label4	Caption	Процент экономии за счет нового карбюратора:
Label6	Caption	Процент экономии за счет нового поршневого кольца:
Button1	Caption	Расчитать экономию
Label11	Caption	% топлива бедут сэкономлена
BitBtn1	Kind	bkClose

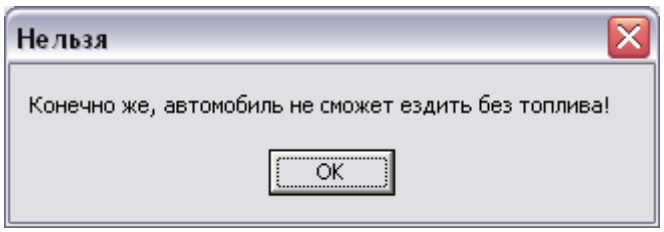
Текст программы:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    BitBtn1: TBitBtn;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
  var Karb, Zazhig, Kolca,x4: string;
  var x,x1,x2,x3, x0 : Real;
procedure TForm1.Button1Click(Sender: TObject);
begin
```

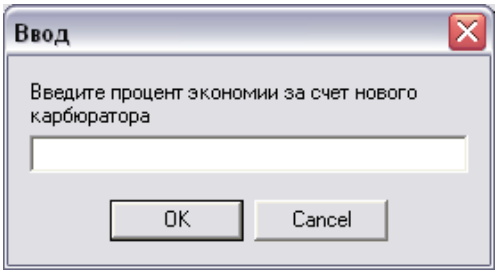
```
            ShowMessage('Конечно же, автомобиль не сможет ездить без топлива!');
//Выводим сообщение
            // Вычисляем результат по условию задачи
            x0:=50;
            x:= (50*30)/100;
            x1:= 50-x  ;
            x2:= 35*20/100  ;
            x3:=x0+x+x2  ;
            x4:=  FloatToStr(x3);
            // Выводим результат не форму
            Form1.Label2.Visible:= true;
            Form1.Label3.Visible:= true;
            Form1.Label2.Caption := x4
        end;
        procedure TForm1.Button2Click(Sender: TObject);
        begin
            // Вводим произвольные проценты
            Karb:=inputbox('Ввод', 'Введите процент экономии за счет нового
карбюратора', '');
            Zazhig:=InputBox('Ввод','Введите процент экономии за счет новой системы
зажигания','');
            Kolca:=InputBox('Ввод','Введите процент экономии за счет новых
поршневых колец','');
            //Записываем их на форму
            Form1.Label7.Caption:=karb+'%';
            Form1.Label8.Caption :=Zazhig+'%';
            Form1.Label9.Caption := Kolca+'%';
            // Делаем расчеты
            x0:=StrToFloat(karb);
            x:=((100-strToFloat(karb))*StrToFloat(zazhig))/100;
            x1:=100-StrToFloat(karb)-x;
            x2:=(x1*StrToFloat(kolca))/100;
            x3:=x0+x+x2;
            //Выводим результат на форму
            Form1.Label10.Caption := FloatToStr(x3);
        end;
        end.
```

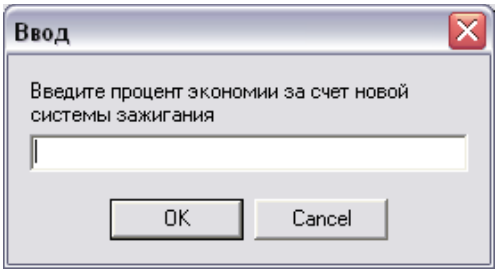
Тестирование:

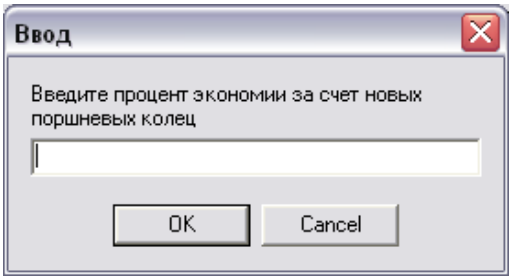
При нажатии на кнопку «рассчитать экономию по условию задачи» появляется сообщение [3]:



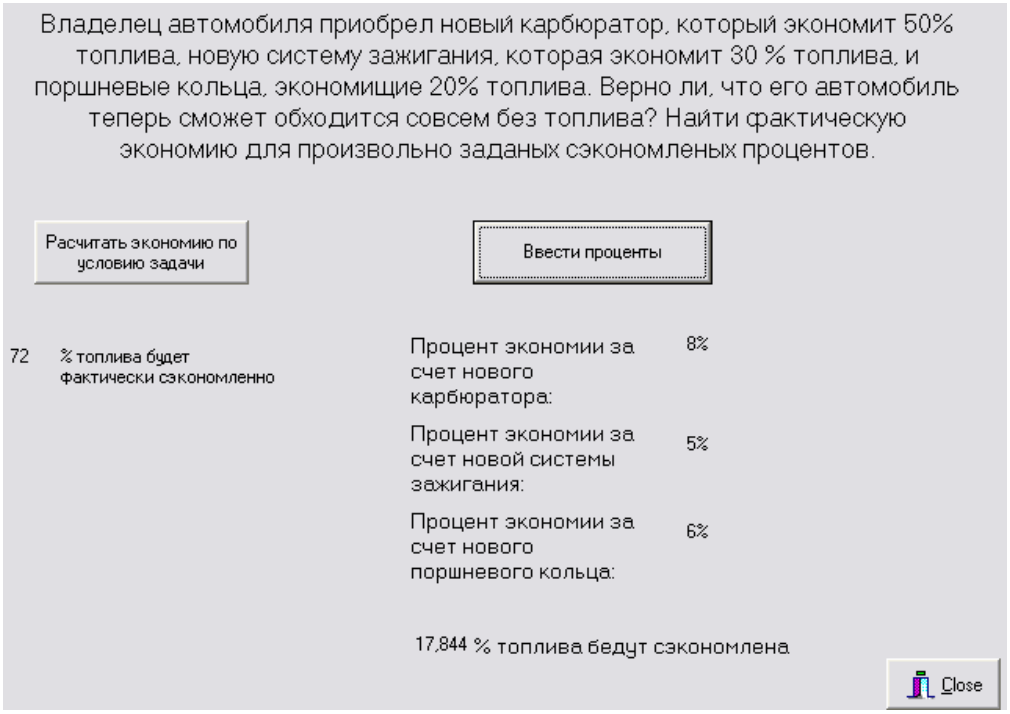
При нажатии на кнопку «Ввести проценты» последовательно появляются окна которые необходимо заполнить:







Введем в каждое окно поочередно цифры, например: 8, 5, 6. Получаем результат:



Задача № 3

Постановка задачи:

Заданы уравнения двух пересекающихся прямых на плоскости: y=k1*x+b1; y=k2*x+b2. Найти угол между ними [3].

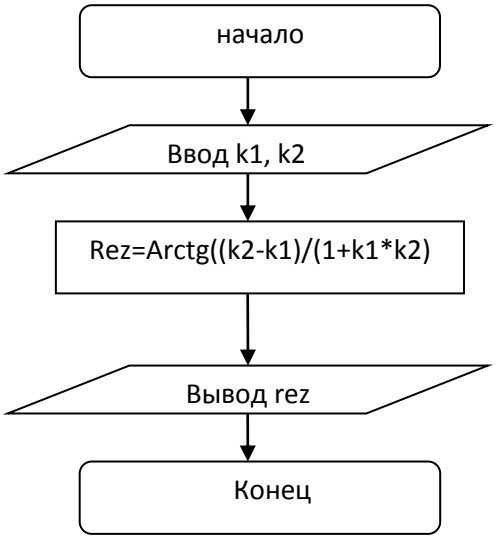
Математическая модель:

Что бы найти угол между двумя прямыми надо найти тангенс угла по формуле:

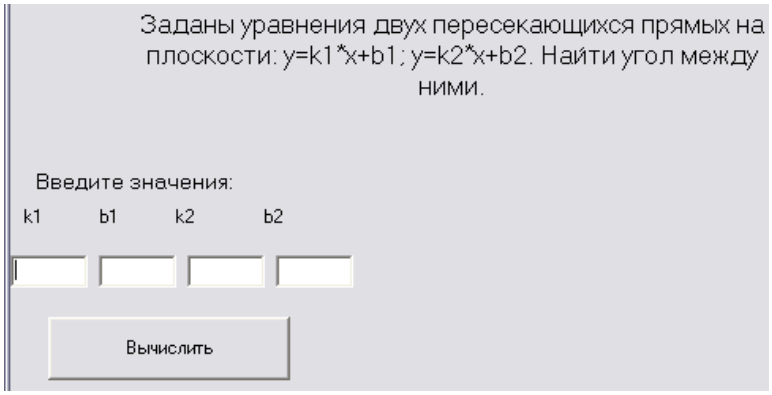
$$\mathrm{tg}(\psi) := \frac{k2 - k1}{1 + k1 \cdot k2}$$

Для того, что бы найти значение угла ψ, надо найти значение arctg (ψ).

Блок-схема:



Описание интерфейса:



В таблице показаны свойства объектов

Объекты и их свойства

Объект	Свойство	Устанавливаемое значение
Label1	Caption	Заданы уравнения двух пересекающихся прямых на плоскости: y=k1*x+b1; y=k2*x+b2. Найти угол между ними.
Label2	Caption	Введите значения:
Label3	Caption	k1
Button1	Caption	Вычислить
Label5	Caption	k2
Label4	Caption	b1
Label6	Caption	b2
Button1	Caption	Расчитать экономию

Edit1	Text	
Edit2	Text	
Edit3	Text	
Edit4	Text	
BitBtn1	Kind	bkClose

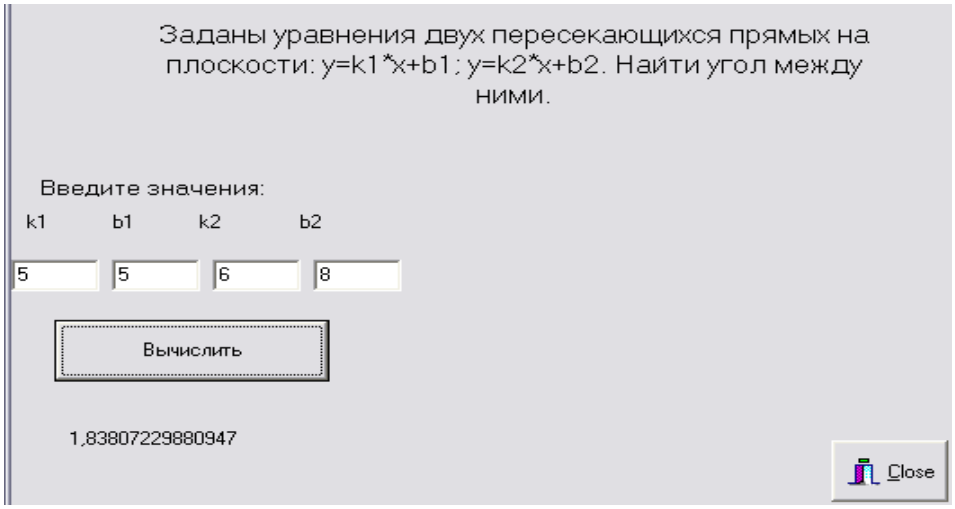
Текст программы:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    BitBtn1: TBitBtn;
    Button1: TButton;
    Label7: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
  var k1,k2,b1,b2, rez: real;
```

```
  var rez1: string;
procedure TForm1.Button1Click(Sender: TObject);
begin
   //Переводим необходимые для расчета величины в вещественный тип
   k1:=StrToFloat(Form1.Edit1.Text);
   k2:=StrToFloat(Form1.Edit3.Text);
   //расчет
   rez:= ((k2-k1)/(1+k1*k2));
   rez:= Arctan(rez);
   //переводим результат в градусы
   rez:=57*rez;
   //Переводим необходимые величины в строковый тип
   rez1:=floatToStr(rez);
    //выводим результат на форму
   Form1.Label7.Caption := rez1
end;
end.
```

Тестирование:

Введем произвольные числа в ячейки, нажмем на кнопку «Вычислить» получаем результат [3]:



Контрольные вопросы:

1. Что такое ввод данных?
2. Что такое вывод данных?
3. Что такое область значений данных?
4. Что такое тип данных?
5. Что такое отладка программ?
6. Что такое контрольный вариант расчета?

Лабораторная работа 14. *Программная реализация алгоритмов разветвляющейся структуры в интегрированной среде разработки.*

В среде Lazarus разработать программу с графическим интерфейсом реализующую одну из задач соответствующую своему варианту.

Задания.

1. Вычислить значение

$$Z=\begin{cases} X^2+3, \text{ если } x<0, \\ X-0{,}8, \text{ если } x>\pi/2, \\ \cos X, \text{ если } 0\le X\le \Pi/2. \end{cases}$$

2. Вычислить значение

$$Z=\begin{cases} a-b, \text{ если } a>0, \\ a+b, \text{ если } a=0, \\ ab, \text{ если } a<0 \end{cases}$$

16. Вычислить значение

$$Z=\begin{cases} a/b, \text{ если } a>0, \\ a*b, \text{ если } a=0, \\ a-b, \text{ если } a<0 \end{cases}$$

17. $Z=a+b \sin x/ x^2-bx+0{,}5$, если $x\ge 0$

18. Вычислить значение

$$Z=\begin{cases} X^2+Y^2, \text{ если } X^2+Y^2< 1, \\ X+Y, \text{ если } X^2+Y^2>1, \\ 0{,}8, \text{ если } X^2+Y^2=1. \end{cases}$$

19. Вычислить значение

$$Y=\begin{cases} a+ bx+cx^2, \text{ если К=1,} \\ d+cx+fx^2, \text{ если К=2,} \\ g+ hx+tx^2, \text{ в остальных случаях} \end{cases}$$

20. Вычислить значение

$Y= 18,9X^2-1,6X+1,97$, если $X<0$ и $Z=9,75X^2-0,35\text{arctg } x$, если $X\geq 0$.

21. Вычислить значение $Z=a+b \sin x/ x^2-bx+0,5$, если$x\geq 0$.

22. Вычислить значение $Z= \sin x/ x^2 -\sqrt{1+3x}$, если $x\leq 0,5$

23. Вычислить значение

$$Z=\begin{cases} \text{Arctg}\,x+y/1-xy \text{ , если } xy<1, \\ 3,14, \text{ если } xy=1, \\ 3,14+ (x+y/1-xy), \text{ если } xy>1. \end{cases}$$

24. Вычислить значение $Y^2-0,3$, если $y<0$,

$$Z=\begin{cases} 1, \text{ если } 0\leq y\leq 1, \\ \\ Y^2+0,3, \text{ если } y>1. \end{cases}$$

25. Вычислить значение $A_i=X^I/(2i)$, $i=1,2,\ldots,30$.

26. Вычислить значение $A_i=(X^{2i}\ \text{SIN}(X^i))/i^2$, $i=1,2,\ldots,30$.

27. Вычислить значение $A_i=((-1)^i*X^i)/i(i+1)(i+2)$, $i=1,2,\ldots,30$.

28. Вычислить значение $A_I=(X^{2i}*(i+3))/2i$, $i=1,2,\ldots,30$.

16. Вычислить значение $A_I=(X^I*(i+2))/i+1$, $i=1,2,\ldots,30$.

17. Даны вещественные числа У, С и целое n($1\leq n\leq 5$). Вычислить Z=3 COS(X)/X+C, где Хвычисляется по формулам:

1) $X=1,2\ Y^2$ 2)X=3,5Y 3)X=1,5 4)X=ARCTG(Y) 5) $X=6+0,5Y^2$, а значение птрактуется как номер для вычисления X.

18. Дано целочисленное значение n и вещественные значения X,У и Z. При $n<0$,$n=0$ или $n>0$ изменить соответственно значения X на X-0,5, У на У/3,5, а Z на Z^2.

19. Вычислить значение $У_i=X_i+\sin x_i$ для всех $0\le X_i\le 1$, если $X_i=X_{i-1}+h$, $X_0=0$ и $h=0,1$

20. Все отрицательные компоненты вещественного вектора $a(a_1,a_2,\ldots,a_{10})$ заменить их квадратами.

21. Дан вещественный вектор X $(X_1,X_2,\ldots,X_{10})$. Умножить на 2 компонента с нечетными номерами.

22. Дан вещественный вектор b $(b_1,b_2,\ldots,b_{10})$. Все компоненты с четными номерами уменьшить на 0,5.

23. Все компоненты вещественного вектора C $(C_1,C_2,\ldots, C_{10})$, начиная с первой по порядку положительной компоненты увеличить в два раза.

24. Вычислить n! для n=7.

25. Присвоить переменной Тзначение “ 3”, если числа X,У,Z равны “2”, если числа равны; “1”, если все числа различны.

26. Определить количество тех членов последовательности

$A_k=SIN(K^2+C)$, K=1,2,….,10, которые удовлетворяют условию $0\le A_K\le 1/2$.

27. Определить значения последовательности $\{x_i\}$ из условия $X_{i+1}=0,5X_{i-1}-X_i^2$, $X_1=X_i=0,65$ для i=3,4….10.

28. Вычислить значение $Y=X^n/3!$ Для n=1,2,…10

29. Вычислить значение $Y=(1)^n/2n(2n-1)$ для n=1,2,….10.

30. Вычислить значение $Y=2n/n^2\sqrt{x}+n$ для n=1,2,….,10.

Контрольные вопросы:
1. Особенности программной реализации разветвляющихся алгоритмов.
2. Что такое отладка программ?
3. Что такое контрольный вариант расчета?

Лабораторная работа 15. *Программная реализация алгоритмов циклической и смешанной структуры в интегрированной среде разработки.*

Для своего варианта составить программу, реализующую указанные действия, и вывести информацию графически в любом выбранном самостоятельно варианте.

1. Даны действительные числа A, B, C, D. Выяснить, можно ли уместить прямоугольник со сторонами A, B внутри прямоугольника со сторонами C,D.
2. Даны действительные числа x, y, z. Найти минимальное из них.
3. Даны действительные положительные числа A, B, C. Выяснить, пройдет ли кирпич с ребрами A,B,C в прямоугольное отверстие со сторонами x,y.
4. Определить, лежит ли точка D(c,b), где $c = \sqrt{a1 + a2}$, $b = a1 + 0{,}7a3$, внутри прямоугольника, ограниченного осями координат, а также прямыми y = 5 и x = 10. a1, a2, a3 - произвольные числа.
5. Выяснить, существует ли треугольник с координатами вершин A(x1,y1), B(x2,y2), C(x3,y3), если да, то найти его площадь.
6. Даны действительные числа A, B, C. Проверить, выполняются ли неравенства A<B<C, если да, то присвоить A = B+C, иначе A = C-B.
7. Даны действительные числа x, y. Вычислить значение функции z = log(x-y)-x/y. Проверить принадлежность x и y области допустимых значений функции.
8. На плоскости расположена окружность радиуса R с центром в начале координат. Определить положение точки x с координатами (A,B) относительно окружности (лежит ли точка внутри окружности, на окружности или вне ее).
9. Даны круг радиуса R и квадрат со стороной A. Определить их взаимное положение при условии, что их центры совпадают.
10. Вывести на печать переменные A, B, C в порядке их возрастания.
11. Проверить, какие из чисел A, B, C принадлежат интервалу (1;25) и не являются четными.
12. Даны действительные числа A, B. Если они оба отрицательные, то заменить каждое из них его квадратом, иначе - положительные из них увеличить в два раза.
13. Выяснить, существует ли треугольник с координатами вершин A(x1;y1), B(x2;y2), C(x3;y3).
14. Даны действительные числа x,y. Вычислить значение функции z = log(x/y)-1/x. Проверить, принадлежат ли x и y области допустимых значений функции.
15. Даны действительные числа A, B. Если они оба неотрицательные, то заменить каждое из них его кубом, иначе отрицательные из них заменить их модулями.

Даны площадь квадрата S1 и круга S2. Определить поместится ли круг в квадрат и наоборот.

16. На плоскости расположена окружность радиуса R с центром в начале координат. Определить, лежат ли точки A(x1;y1) и B(x2;y2) на окружности.

17. Составить программу вычисления корней системы уравнений с двумя неизвестными $\begin{cases} a1x + b1y = c1 \\ a2x + b2y = c2 \end{cases}$ методом Крамера. Убедиться, что главный определитель не равен 0.

18. Даны действительные числа A, B, C, D. Выяснить, можно ли уместить прямоугольник со сторонами A, B внутри прямоугольника со сторонами C,D.

19. Вывести на печать переменные A, B, C в порядке их убывания.

20. Даны действительные числа x, y, z. Найти максимальное из них.

21. Проверить, какие из чисел A, B, C, D не принадлежат интервалу (3,15).

22. Даны действительные числа x, y. Вычислить значение функции z = ln(x)-x/y, проверить, принадлежат ли x и y области допустимых значений функции.

23. Даны действительные числа A, B. Если они имеют разные знаки, то напечатать их произведение, иначе напечатать их квадраты.

24. Выяснить, существует ли треугольник с длинами сторон A,B,C. Если да, то найти его площадь.

25. Даны действительные числа x, y. Вычислить значение функции z = arcsin(x)-y.

26. Даны действительные числа x, y, z. Получить максимальное из них по модулю.

27. Даны действительные числа x, y. Вычислить значение функции z = arcsin(x+y).

28. На каком из интервалов $(-\infty; k1), (k1; k2), (k2; +\infty)$ лежит точка с координатой x? k1, k2, x - произвольные числа причем k1<k2.

29. Лежат ли обе точки D(a1;b1) и C(a2;b2) внутри круга радиуса R с центром в начале координат? Если такой точки нет, выдать соответствующее сообщение.

Контрольные вопросы:

1. Особенности программной алгоритмов реализации циклической и смешанной структуры.

2. Что такое отладка программ?

3. Что такое контрольный вариант расчета?

Лабораторная работа 16. *Изучение построения и компонент вычислительных сетей.*

Теоретические сведения к лабораторной работе

Основными устройствами для быстрой передачи информации на большие расстояния в настоящее время являются телеграф, радио, телефон, телевизионный передатчик, телекоммуникационные сети на базе вычислительных систем.

Передача информации между компьютерами существует с самого момента возникновения ЭВМ. Она позволяет организовать совместную работу отдельных компьютеров, решать одну задачу с помощью нескольких компьютеров, совместно использовать ресурсы и решать множество других проблем.

Под ***компьютерной сетью*** понимают комплекс аппаратных и программных средств, предназначенных для обмена информацией и доступа пользователей к единым ресурсам сети [6,7].

Основное назначение компьютерных сетей - обеспечить совместный доступ пользователей к информации (базам данных, документам и т.д.) и ресурсам (жесткие диски, принтеры, накопители CD-ROM, модемы, выход в глобальную сеть и т.д.).

Абоненты сети – объекты, генерирующие или потребляющие информацию.

Абонентами сети могут быть отдельные ЭВМ, промышленные роботы, станки с ЧПУ (станки с числовым программным управлением) и т.д. Любой абонент сети подключён к станции.

Станция – аппаратура, которая выполняет функции, связанные с передачей и приёмом информации.

Для организации взаимодействия абонентов и станции необходима физическая передающая среда.

Физическая передающая среда – линии связи или пространство, в котором распространяются электрические сигналы, и аппаратура передачи данных.

Одной из основных характеристик линий или каналов связи является скорость передачи данных (пропускная способность) [6,7].

Скорость передачи данных – количество бит информации, передаваемой за единицу времени.

Обычно скорость передачи данных измеряется в битах в секунду (бит/с) и кратных единицах Кбит/с и Мбит/с.

Соотношения между единицами измерения: 1 Кбит/с =1024 бит/с; 1 Мбит/с =1024 Кбит/с; 1 Гбит/с =1024 Мбит/с.

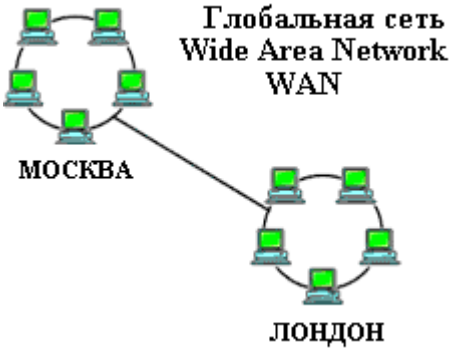
На базе физической передающей среды строится коммуникационная сеть. Таким образом, компьютерная сеть – это совокупность абонентских систем и коммуникационной сети.

Виды сетей. По типу используемых ЭВМ выделяют **однородные** и **неоднородные** сети. В неоднородных сетях содержатся программно несовместимые компьютеры.

По территориальному признаку сети делят на **локальные** и **глобальные.**

Локальные сети **(LAN, Local Area Network)** объединяют абонентов, расположенных в пределах небольшой территории, обычно не более 2–2.5 км.

Локальные компьютерные сети позволят организовать работу отдельных предприятий и учреждений, в том числе и образовательных, решить задачу организации доступа к общим техническим и информационным ресурсам [6,7].



Глобальные сети (WAN, Wide Area Network) **объединяют абонентов, расположенных друг от друга на значительных расстояниях: в разных районах города, в разных городах, странах, на** разных континентах (например, сеть Интернет) [6,7].

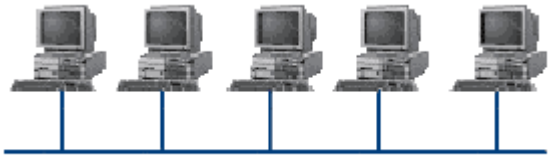
Взаимодействие между абонентами такой сети может осуществляться на базе телефонных линий связи, радиосвязи и систем спутниковой связи. Глобальные компьютерные сети позволят решить проблему объединения информационных ресурсов всего человечества и организации доступа к этим ресурсам.

Основные *компоненты коммуникационной сети:*

- передатчик;
- приёмник;
- сообщения (цифровые данные определённого формата: файл базы данных, таблица, ответ на запрос, текст или изображение);
- средства передачи (физическая передающая среда и специальная аппаратура, обеспечивающая передачу информации).

Топология локальных сетей. Под топологией компьютерной сети обычно понимают физическое расположение компьютеров сети относительно друг друга и способ соединения их линиями.

Топология определяет требования к оборудованию, тип используемого кабеля, методы управления обменом, надежность работы, возможность расширения сети. **Существует три основных вида топологии сети: шина, звезда и кольцо.**



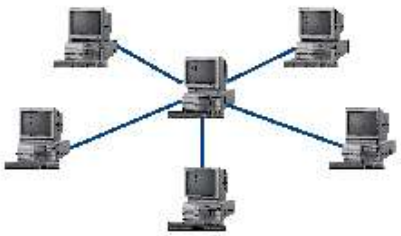
Шина (bus), при которой все компьютеры параллельно подключаются к одной линии связи, и информация от каждого компьютера одновременно передается ко всем остальным компьютерам. Согласно этой топологии создается одноранговая сеть. При таком соединении компьютеры могут передавать информацию только по очереди, так как линия связи единственная [6,7].

Достоинства:

- ✓ простота добавления новых узлов в сеть (это возможно даже во время работы сети);
- ✓ сеть продолжает функционировать, даже если отдельные компьютеры вышли из строя;
- ✓ недорогое сетевое оборудование за счет широкого распространения такой топологии.

Недостатки:

- ✓ сложность сетевого оборудования;
- ✓ сложность диагностики неисправности сетевого оборудования из-за того, что все адаптеры включены параллельно;
- ✓ обрыв кабеля влечет за собой выход из строя всей сети;
- ✓ ограничение на максимальную длину линий связи из-за того, что сигналы при передаче ослабляются и никак не восстанавливаются.



Звезда (star), при которой к одному центральному компьютеру присоединяются остальные периферийные компьютеры, причем каждый из них использует свою отдельную линию связи. Весь обмен информацией идет исключительно через центральный компьютер, на который ложится очень большая нагрузка, поэтому он предназначен только для обслуживания сети [6,7].

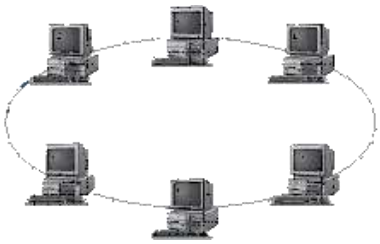
Достоинства:

- ✓ выход из строя периферийного компьютера никак не отражается на функционировании оставшейся части сети;
- ✓ простота используемого сетевого оборудования;
- ✓ все точки подключения собраны в одном месте, что позволяет легко контролировать работу сети, локализовать неисправности сети путем отключения от центра тех или иных периферийных устройств;
- ✓ не происходит затухания сигналов.

Недостатки:

- ✓ выход из строя центрального компьютера делает сеть полностью неработоспособной;
- ✓ жесткое ограничение количества периферийных компьютеров;

- ✓ значительный расход кабеля.



Кольцо (ring), при котором каждый компьютер передает информацию всегда только одному компьютеру, следующему в цепочке, а получает информацию только от предыдущего в цепочке компьютера, и эта цепочка замкнута. Особенностью кольца является то, что каждый компьютер восстанавливает приходящий к нему сигнал, поэтому затухание сигнала во всем кольце не имеет никакого значения, важно только затухание между соседними компьютерами.

Достоинства:

- ✓ легко подключить новые узлы, хотя для этого нужно приостановить работу сети;
- ✓ большое количество узлов, которое можно подключить к сети (более 1000);
- ✓ высокая устойчивость к перегрузкам.

Недостатки:

- ✓ выход из строя хотя бы одного компьютера нарушает работу сети;
- ✓ обрыв кабеля хотя бы в одном месте нарушает работу сети.

В отдельных случаях при конструировании сети используют комбинированную топологию. Например, **дерево** (tree) – комбинация нескольких звезд.

Каждый компьютер, который функционирует в локальной сети, должен иметь **сетевой адаптер (сетевую карту).** Функцией сетевого адаптера является передача и прием сигналов, распространяемых по кабелям связи. Кроме того, компьютер должен быть оснащен сетевой операционной системой.

При конструировании сетей используют следующие виды кабелей:

неэкранированная витая пара. Максимальное расстояние, на котором могут быть расположены компьютеры, соединенные этим кабелем, достигает 90 м. Скорость передачи информации - от 10 до 155 Мбит/с; *экранированная витая пара.* Скорость передачи информации - 16 Мбит/с на расстояние до 300 м.

коаксиальный кабель. Отличается более высокой механической прочностью, помехозащищённостью и позволяет передавать информацию на расстояние до 2000 м со скоростью 2-44 Мбит/с;

волоконно-оптический кабель. Идеальная передающая среда, он не подвержен действию электромагнитных полей, позволяет передавать информацию на расстояние до 10 000 м со скоростью до 10 Гбит/с.

Понятие о глобальных сетях. *Глобальная сеть* – это объединения компьютеров, расположенных на удаленном расстоянии, для общего использования мировых информационных ресурсов. На сегодняшний день их насчитывается в мире более 200. Из них наиболее известной и самой популярной является сеть Интернет.

В отличие от локальных сетей в глобальных сетях нет какого-либо единого центра управления. Основу сети составляют десятки и сотни тысяч компьютеров,

соединенных теми или иными каналами связи. Каждый компьютер имеет уникальный идентификатор, что позволяет "проложить к нему маршрут" для доставки информации. Обычно в глобальной сети объединяются компьютеры, работающие по разным правилам (имеющие различную архитектуру, системное программное обеспечение и т.д.). Поэтому для передачи информации из одного вида сетей в другой используются шлюзы [6,7].

Шлюзы (gateway)– это устройства (компьютеры), служащие для объединения сетей с совершенно различными протоколами обмена.

Протокол обмена– это набор правил (соглашение, стандарт), определяющий принципы обмена данными между различными компьютерами в сети.

Протоколы условно делятся на базовые (более низкого уровня), отвечающие за передачу информации любого типа, и прикладные (более высокого уровня), отвечающие за функционирование специализированных служб.

Главный компьютер сети, который предоставляет доступ к общей базе данных, обеспечивает совместное использование устройств ввода-вывода и взаимодействия пользователей называется ***сервером.***

Компьютер сети, который только использует сетевые ресурсы, но сам свои ресурсы в сеть не отдает, называется *клиентом* (часто его еще называют *рабочей станцией*).

Для работы в глобальной сети пользователю необходимо иметь соответствующее аппаратное и программное обеспечение.

Программное обеспечение можно разделить на два класса:

- ✓ программы-серверы, которые размещаются на узле сети, обслуживающем компьютер пользователя;
- ✓ программы-клиенты, размещенные на компьютере пользователя и пользующиеся услугами сервера.

Глобальные сети предоставляют пользователям разнообразные услуги: электронная почта, удаленный доступ к любому компьютеру сети, поиск данных и программ и так далее [6,7].

Выполнение работы:

Задание №1.

1. Дать определение понятию «Компьютерная сеть».
2. Укажите основное назначение компьютерной сети.
3. Дать определение понятию «Сетевой адаптер».
4. Дать определение понятию «Модем».
5. Дать определение понятию «Протокол».
6. Укажите объект, который является абонентом сети.
7. Укажите основную характеристику каналов связи.
8. Что такое локальная сеть, глобальная сеть?
9. Что понимается под топологией локальной сети?
10. Какие существуют виды топологии локальной сети?
11. Охарактеризуйте кратко топологию «шина», «звезда», «кольцо».

Задание №2.

Заполнить таблицу «Характеристика каналов связи».

Канал связи	Физические характеристики	Пропускная способность	Помехоустойчивость
Телефонная связь			
Радиосвязь			
Оптическое волокно			
Спутниковая связь			

Задание №3.

Решите задачу.

Максимальная скорость передачи данных в локальной сети 100 Мбит/с. Сколько страниц текста можно передать за 1 сек, если 1 страница текста содержит 50 строк и на каждой строке - 70 символов.

Написать вывод по проделанной работе.

Контрольные вопросы:

1. Что такое вычислительная сеть?
2. Что такое компоненты вычислительных сетей?
3. Что такое построение вычислительных сетей?

Лабораторная работа 17. *Изучение методов защиты информации в компьютерных сетях. Электронная подпись.*

Механизмы защиты информации

Одним из условий безопасной работы в информационной системе является соблюдение пользователем ряда правил, которые проверены на практике и показали свою высокую эффективность. Их несколько:

1. Использование программных продуктов, полученных законным официальным путем. Вероятность наличия вируса в пиратской копии во много раз выше, чем в официально полученном программном обеспечении.
2. Дублирование информации. Прежде всего, необходимо сохранять дистрибутивные носители программного обеспечения. При этом запись на носители, допускающие выполнение этой операции, должна быть, по возможности, заблокирована. Следует особо позаботиться о сохранении рабочей информации. Предпочтительнее регулярно создавать копии рабочих файлов на съемных машинных носителях информации с защитой от записи. Копируется либо весь файл, либо только вносимые изменения. Последний вариант применим, например, при работе с базами данных.
3. Регулярное обновление системного ПО. Операционную систему необходимо регулярно обновлять и устанавливать все исправления безопасности от Microsoft и других производителей, чтобы устранить существующие уязвимости программного обеспечения.
4. Ограничение доступа пользователей к настройкам операционной системы и системным данным. Для обеспечения стабильной работы системы довольно часто требуется ограничивать возможности пользователей, что можно сделать либо с помощью встроенных средств Windows, либо с помощью специализированных программ, предназначенных для управления доступом к компьютеру.

В корпоративных сетях возможно применение групповых политик в сети домена Windows.

5. Для максимально эффективного использования сетевых ресурсов необходимо вводить ограничения доступа авторизованных пользователей к внутренним и внешним сетевым ресурсам и блокировать доступ неавторизованных пользователей.
6. Регулярное использование антивирусных средств. Перед началом работы целесообразно выполнять программы-сканеры и программы-ревизоры. Антивирусные базы должны регулярно обновляться. Кроме того, необходимо проводить антивирусный контроль сетевого трафика.
7. Защита от сетевых вторжений обеспечивается применением программно-аппаратных средств, в том числе: использованием межсетевых экранов, систем обнаружения/предотвращения вторжений IDS/IPS (Intrusion Detection/Prevention System), реализацией технологий VPN (Virtual Private Network).
8. Применение средств аутентификации и криптографии – использование паролей (простых/сложных/неповторяющихся) и методов шифрования. Не рекомендуется использовать один и тот же пароль на разных ресурсах и разглашать

сведения о паролях. При написании пароля на сайтах следует быть особенно внимательным, чтобы не допустить ввода своего пароля на мошенническом сайте-двойнике.

9. Особую осторожность следует проявлять при использовании новых (не известных) съемных носителей информации и новых файлов. Новые съёмные носители обязательно должны быть проверены на отсутствие загрузочных и файловых вирусов, а полученные файлы – на наличие файловых вирусов. При работе в распределенных системах или в системах коллективного пользования новые сменные носители информации и вводимые в систему файлы целесообразно проверять на специально выделенных для этой цели компьютерах, не подключенных к локальной сети. Только после всесторонней антивирусной проверки дисков и файлов они могут передаваться пользователям системы.

10. При работе с полученными (например, посредством электронной почты) документами и таблицами целесообразно запретить выполнение макрокоманд средствами, встроенными в текстовые и табличные редакторы (MS Word, MS Excel), до завершения полной проверки этих файлов.

11. Если не предполагается осуществлять запись информации на внешние носители, то необходимо заблокировать выполнение этой операции, например, программно отключив USB-порты.

12. При работе с общими ресурсами в открытых сетях (например, Интернет) использовать только проверенные сетевые ресурсы, не имеющие вредоносного контента. Не следует доверять всей поступающей на компьютер информации – электронным письмам, ссылкам на Web-сайты, сообщениям на Интернет-пейджеры. Категорически не рекомендуется открывать файлы и ссылки, приходящие из неизвестного источника.

Постоянное следование приведенным рекомендациям позволяет значительно уменьшить вероятность заражения программными вирусами и защищает пользователя от безвозвратных потерь информации. Однако даже при скрупулезном выполнении всех правил профилактики возможность заражения ПК компьютерными вирусами полностью исключить нельзя, поэтому методы и средства противодействия вредоносному ПО необходимо постоянно совершенствовать и поддерживать в работоспособном состоянии.

Антивирусные средства защиты информации

Массовое распространение вредоносного программного обеспечения, серьезность последствий его воздействия на информационные системы и сети вызвали необходимость разработки и использования специальных ***антивирусных средств*** и методов их применения.

Нужно отметить, что не существует антивирусных средств, гарантирующих обнаружение всех возможных программ-вирусов.

Антивирусные средства применяются для решения следующих задач:

- обнаружение вредоносного ПО в информационных системах;
- блокирование работы вредоносного ПО;
- устранение последствий воздействия вредоносного ПО.

Обнаружение вредоносного ПО желательно осуществлять на стадии его внедрения в систему или, по крайней мере, до начала осуществления им деструктивных действий. При обнаружении такого программного обеспечения или его деятельности необходимо сразу же прекратить работу программы-вируса в целях минимизации ущерба от ее воздействия на систему.

Устранение последствий воздействия вирусов ведется в двух направлениях:

- удаление вирусов;
- восстановление (при необходимости) файлов, областей памяти.

Процедуру удаления обнаруженного вредоносного кода из зараженной системы необходимо выполнять крайне аккуратно. Часто вирусы и троянские программы предпринимают специальные действия, чтобы скрыть факт своего присутствия в системе, или встраиваются в нее так глубоко, что задача его уничтожения становится достаточно нетривиальной.

Восстановление системы зависит от типа вируса, а также от времени его обнаружения по отношению к началу деструктивных действий. В том случае, когда программа-вирус уже запущена в системе и ее деятельность предусматривает изменение или удаление данных, восстановление информации (особенно, если она не продублирована) может быть невыполнимо.Для борьбы с вирусами используются программные и программно-аппаратные средства, которые применяются в определенной последовательности и комбинации, образуя методы защиты от вредоносного ПО.

Известны следующие методы обнаружения вирусов, активно применяемые современными антивирусными средствами:

- сканирование;
- обнаружение изменений;
- эвристический анализ;
- использование резидентных сторожей;
- использование программно-аппаратной защиты от вирусов.

Сканирование – один из самых простых методов обнаружения вирусов, осуществляется программой-сканером, которая просматривает файлы в поисках опознавательной части вируса – ***сигнатуры***. Под сигнатурой понимается уникальная последовательность байтов, принадлежащая конкретному вирусу и не встречающаяся в других программах.

Программа фиксирует наличие уже известных вирусов, для которых сигнатура определена. Для эффективного применения антивирусных программ, использующих метод сканирования, необходимо регулярное обновление сведений о новых вирусах.

Метод ***обнаружения изменений*** базируется на использовании программ-ревизоров, которые следят за изменениями файлов и дисковых секторов на компьютере. Любой вирус каким-либо образом изменяет систему данных на диске. Например, может измениться загрузочный сектор, появиться новый исполняемый файл или измениться уже существующий, и т.п.

Как правило, антивирусные программы-ревизоры определяют и запоминают в специальных файлах образы главной загрузочной записи, загрузочных секторов логических дисков, характеристики всех контролируемых файлов, каталогов и номера дефектных кластеров диска. Периодически ревизор проверяет текущее

состояние областей дисков и файловой системы, сравнивает с предыдущим состоянием и немедленно выдает сообщения обо всех подозрительных изменениях.

Главным достоинством метода является возможность обнаружения вирусов всех типов, а также новых неизвестных вирусов.

Имеются у этого метода и недостатки. С помощью программ-ревизоров невозможно определить вирус в файлах, которые поступают в систему уже зараженными. Вирусы будут обнаружены только после размножения в системе.

Эвристический анализ, как и метод обнаружения изменений, позволяет определять неизвестные вирусы, но не требует предварительного сбора, обработки и хранения информации о файловой системе.

Эвристический анализ в антивирусных программах основан на сигнатурах и эвристическом алгоритме, призван улучшить способность программ-сканеров применять сигнатуры и распознавать модифицированные версии вирусов в тех случаях, когда код неизвестной программы совпадает с сигнатурой не полностью, но в подозрительной программе явно выражены более общие признаки вируса либо его поведенческая модель. При обнаружении подобных кодов, выдается сообщение о возможном заражении. После получения таких сообщений необходимо тщательно проверить предположительно зараженные файлы и загрузочные сектора всеми имеющимися антивирусными средствами.

Недостатком данного метода является большое количество ложных срабатываний антивирусных средств в тех случаях, когда в легальной программе присутствуют фрагменты кода, выполняющего действия и/или последовательности, свойственные некоторым вирусам.

Метод ***использования резидентных сторожей*** основан на применении программ, которые постоянно находятся в оперативной памяти устройства (компьютера) и отслеживают все действия, выполняемые остальными программами. В случае выполнения какой-либо программой подозрительных действий, свойственных вирусам (обращение для записи в загрузочные сектора, помещение в оперативную память резидентных модулей, попытки перехвата прерываний и т.п.), резидентный сторож выдает сообщение пользователю.

Применение антивирусных программ с резидентным сторожем снижает вероятность запуска вирусов на компьютере, но следует учитывать, что постоянное использование ресурсов оперативной памяти под резидентные программы уменьшает объем памяти, доступной для других программ.

На сегодняшний день одним из самых надежных механизмов защиты информационных систем и сетей являются *программно-аппаратные средства*, как правило, включающие в себя не только антивирусные системы, но и обеспечивающие дополнительный сервис. Эта тема подробно рассмотрена в разделе "Программно-аппаратные средства обеспечения безопасности информационных сетей".

Криптографические методы защиты информации

Криптографические методы защиты информации – это мощное оружие в борьбе за информационную безопасность.

Криптография (от древне-греч. κρυπτος – скрытый и γραφω – пишу) – наука о методах обеспечения конфиденциальности и аутентичности информации.

Криптография представляет собой совокупность методов преобразования данных, направленных на то, чтобы сделать эти данные бесполезными для злоумышленника. Такие преобразования позволяют решить два главных вопроса, касающихся безопасности информации:

- защиту конфиденциальности;
- защиту целостности.

Проблемы защиты конфиденциальности и целостности информации тесно связаны между собой, поэтому методы решения одной из них часто применимы для решения другой.

Известны различные подходы к классификации методов криптографического преобразования информации. По виду воздействия на исходную информацию методы криптографического преобразования информации могут быть разделены на четыре группы:

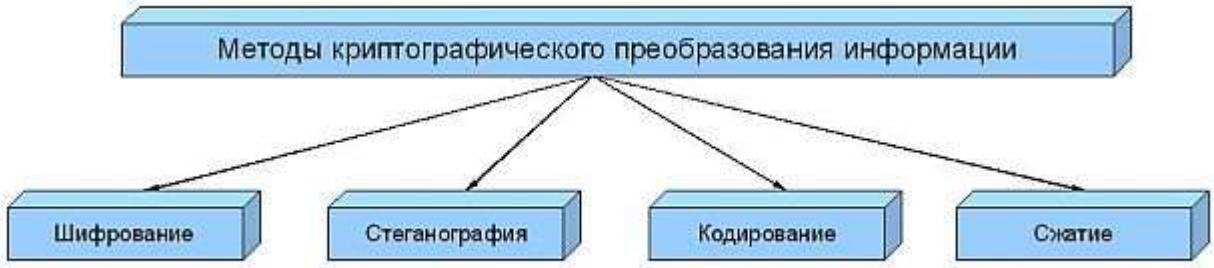


Рис. 2.1. Классификация методов криптографического преобразования информации

Процесс ***шифрования*** заключается в проведении обратимых математических, логических, комбинаторных и других преобразований исходной информации, в результате которых зашифрованная информация представляет собой хаотический набор букв, цифр, других символов и двоичных кодов.

Для шифрования информации используются **алгоритм преобразования** и **ключ**. Как правило, алгоритм для определенного метода шифрования является неизменным. Исходными данными для алгоритма шифрования служит информация, подлежащая зашифрованию, и ключ шифрования. Ключ содержит управляющую информацию, которая определяет выбор преобразования на определенных шагах алгоритма и величины операндов, используемых при реализации алгоритма шифрования. Операнд – это константа, переменная, функция, выражение и другой объект языка программирования, над которым производятся операции.

В отличие от других методов криптографического преобразования информации, методы ***стеганографии*** позволяют скрыть не только смысл хранящейся или передаваемой информации, но и сам факт хранения или передачи закрытой информации. В основе всех методов стеганографии лежит маскирование закрытой информации среди открытых файлов, т.е. скрываются секретные данные, при этом создаются реалистичные данные, которые невозможно отличить от настоящих. Обработка мультимедийных файлов в информационных системах открыла практически неограниченные возможности перед стеганографией.

Графическая и звуковая информация представляются в числовом виде. Так, в графических объектах наименьший элемент изображения может кодироваться одним байтом. В младшие разряды определенных байтов изображения в соответствии с алгоритмом криптографического преобразования помещаются биты скрытого файла. Если правильно подобрать алгоритм преобразования и изображение, на фоне которого помещается скрытый файл, то человеческому глазу практически невозможно отличить полученное изображение от исходного. С помощью средств стеганографии могут маскироваться текст, изображение, речь, цифровая подпись, зашифрованное сообщение.

Скрытый файл также может быть зашифрован. Если кто-то случайно обнаружит скрытый файл, то зашифрованная информация будет воспринята как сбой в работе системы. Комплексное использование стеганографии и шифрования многократно повышает сложность решения задачи обнаружения и раскрытия конфиденциальной информации.

Содержанием процесса ***кодирование*** информации является замена исходного смысла сообщения (слов, предложений) кодами. В качестве кодов могут использоваться сочетания букв, цифр, знаков. При кодировании и обратном преобразовании используются специальные таблицы или словари. В информационных сетях кодирование исходного сообщения (или сигнала) программно-аппаратными средствами применяется для повышения достоверности передаваемой информации.

Часто кодирование и шифрование ошибочно принимают за одно и тоже, забыв о том, что для восстановления закодированного сообщения, достаточно знать правило замены, в то время как для расшифровки сообщения помимо знания правил шифрования, требуется ключ к шифру.

Сжатие информации может быть отнесено к методам криптографического преобразования информации с определенными оговорками. Целью сжатия является сокращение объема информации. В то же время сжатая информация не может быть прочитана или использована без обратного преобразования. Учитывая доступность средств сжатия и обратного преобразования, эти методы нельзя рассматривать как надежные средства криптографического преобразования информации. Даже если держать в секрете алгоритмы, то они могут быть сравнительно легко раскрыты статистическими методами обработки. Поэтому сжатые файлы конфиденциальной информации подвергаются последующему шифрованию. Для сокращения времени передачи данных целесообразно совмещать процесс сжатия и шифрования информации.

Основным видом криптографического преобразования информации в компьютерных сетях является ***шифрование***. Под шифрованием понимается процесс преобразования открытой информации в зашифрованную информацию (шифртекст) или процесс обратного преобразования зашифрованной информации в открытую. Процесс преобразования открытой информации в закрытую получил название зашифрование, а процесс преобразования закрытой информации в открытую – расшифрование.

За многовековую историю использования шифрования информации человечеством изобретено множество методов шифрования или шифров. ***Методом***

шифрования (шифром) называется совокупность обратимых преобразований открытой информации в закрытую информацию в соответствии с алгоритмом шифрования. Большинство методов шифрования не выдержали проверку временем, а некоторые используются и до сих пор. Появление компьютеров и компьютерных сетей инициировало процесс разработки новых шифров, учитывающих возможности использования компьютерной техники как для зашифрования/расшифрования информации, так и для атак на шифр. ***Атака на шифр (криптоанализ, криптоатака)*** – это процесс расшифрования закрытой информации без знания ключа и, возможно, при отсутствии сведений об алгоритме шифрования.

Современные методы шифрования должны отвечать следующим требованиям:

- стойкость шифра противостоять криптоанализу (криптостойкость) должна быть такой, чтобы вскрытие его могло быть осуществлено только путем решения задачи полного перебора ключей;
- криптостойкость обеспечивается не секретностью алгоритма шифрования, а секретностью ключа;
- шифртекст не должен существенно превосходить по объему исходную информацию;
- ошибки, возникающие при шифровании, не должны приводить к искажениям и потерям информации;
- время шифрования не должно быть большим;
- стоимость шифрования должна быть согласована со стоимостью закрываемой информации.

Криптостойкость шифра является его основным показателем эффективности. Она измеряется временем или стоимостью средств, необходимых криптоаналитику для получения исходной информации по шифртексту, при условии, что ему неизвестен ключ.

Сохранить в секрете широко используемый алгоритм шифрования практически невозможно. Поэтому алгоритм не должен иметь скрытых слабых мест, которыми могли бы воспользоваться криптоаналитики. Если это условие выполняется, то криптостойкость шифра определяется длиной ключа, так как единственный путь вскрытия зашифрованной информации – перебор комбинаций ключа и выполнение алгоритма расшифрования. Таким образом, время и средства, затрачиваемые на криптоанализ, зависят от длины ключа и сложности алгоритма шифрования.

Работа простой криптосистемы проиллюстрирована на рис. 2.2.

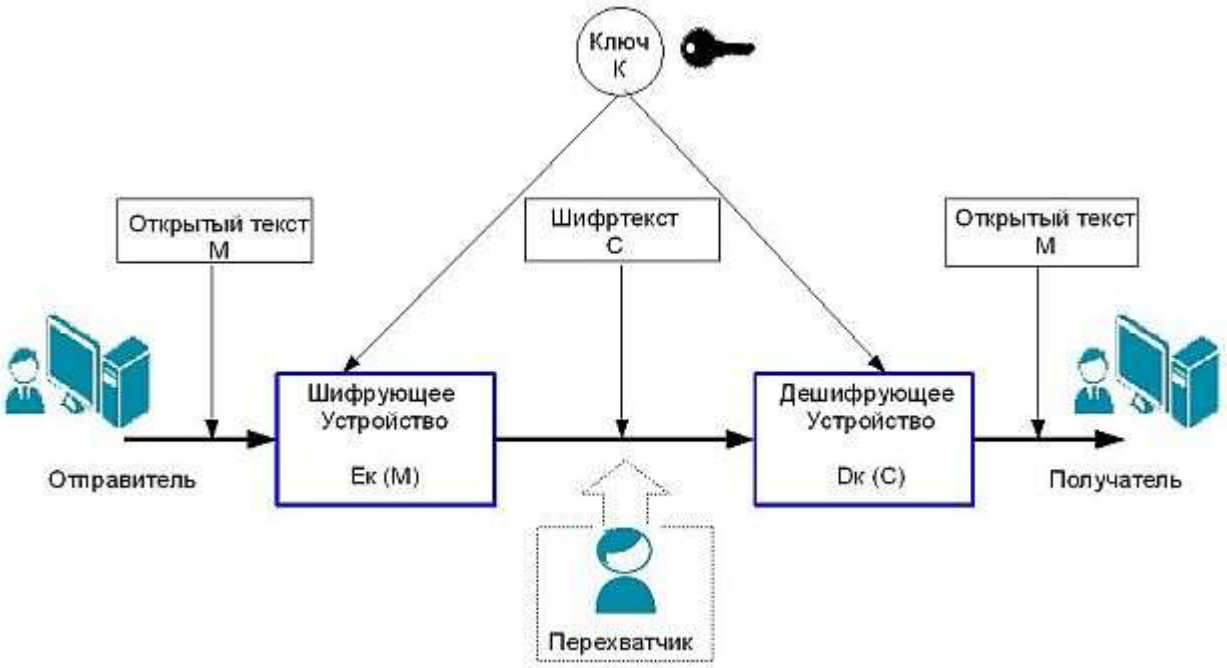


Рис. 2.2. Обобщённая схема криптографической системы

Отправитель генерирует открытый текст исходного сообщения **М**, которое должно быть передано законному получателю по незащищённому каналу. За каналом следит перехватчик с целью перехватить и раскрыть передаваемое сообщение. Для того чтобы перехватчик не смог узнать содержание сообщения **М**, отправитель шифрует его с помощью обратимого преобразования **Ек** и получает шифртекст (или криптограмму) **С=Ек(М)** , который отправляет получателю.

Законный получатель, приняв шифртекст **С**, расшифровывает его с помощью обратного преобразования **Dк(С)** и получает исходное сообщение в виде открытого текста **М**.

Преобразование **Ек** выбирается из семейства криптографических преобразований, называемых криптоалгоритмами. Параметр, с помощью которого выбирается отдельное преобразование, называется криптографическим ключом **К**.

Криптосистема имеет разные варианты реализации: набор инструкций, аппаратные средства, комплекс программ, которые позволяют зашифровать открытый текст и расшифровать шифртекст различными способами, один из которых выбирается с помощью конкретного ключа **К**.

Преобразование шифрования может быть симметричным и асимметричным относительно преобразования расшифрования. Это важное свойство определяет два класса криптосистем:

- симметричные (одноключевые) криптосистемы;
- асимметричные (двухключевые) криптосистемы (с открытым ключом).

Симметричное шифрование

Симметричное шифрование, которое часто называют шифрованием с помощью секретных ключей, в основном используется для обеспечения конфиденциальности данных. Для того чтобы обеспечить конфиденциальность данных, пользователи должны совместно выбрать единый математический

алгоритм, который будет использоваться для шифрования и расшифровки данных. Кроме того, им нужно выбрать общий (секретный) ключ, который будет использоваться с принятым ими алгоритмом шифрования/дешифрования, т.е. один и тот же ключ используется и для зашифрования, и для расшифрования (слово "симметричный" означает одинаковый для обеих сторон).

Пример симметричного шифрования показан на рис. 2.2.

Сегодня широко используются такие алгоритмы шифрования, как Data Encryption Standard (DES), 3DES (или "тройной DES") и International Data Encryption Algorithm (IDEA). Эти алгоритмы шифруют сообщения блоками по 64 бита. Если объем сообщения превышает 64 бита (как это обычно и бывает), необходимо разбить его на блоки по 64 бита в каждом, а затем каким-то образом свести их воедино. Такое объединение, как правило, происходит одним из следующих четырех методов:

- электронной кодовой книги (Electronic Code Book, ECB);
- цепочки зашифрованных блоков (Cipher Block Changing, CBC);
- x-битовой зашифрованной обратной связи (Cipher FeedBack, CFB-x);
- выходной обратной связи (Output FeedBack, OFB).

Triple DES (3DES) – симметричный блочный шифр, созданный на основе алгоритма DES, с целью устранения главного недостатка последнего – малой длины ключа (56 бит), который может быть взломан методом полного перебора ключа. Скорость работы 3DES в 3 раза ниже, чем у DES, но криптостойкость намного выше. Время, требуемое для криптоанализа 3DES, может быть намного больше, чем время, нужное для вскрытия DES.

Алгоритм ***AES*** (Advanced Encryption Standard), также известный как Rijndael – симметричный алгоритм блочного шифрования – шифрует сообщения блоками по 128 бит, использует ключ 128/192/256 бит.

Шифрование с помощью секретного ключа часто используется для поддержки конфиденциальности данных и очень эффективно реализуется с помощью неизменяемых "вшитых" программ (firmware). Этот метод можно использовать для аутентификации и поддержания целостности данных.

С методом симметричного шифрования связаны следующие проблемы:

- необходимо часто менять секретные ключи, поскольку всегда существует риск их случайного раскрытия (компрометации);
- достаточно сложно обеспечить безопасность секретных ключей при их генерировании, распространении и хранении.

Асимметричное шифрование

- Асимметричное шифрование часто называют шифрованием с помощью **открытого ключа**, при котором используются разные, но взаимно дополняющие друг друга ключи и алгоритмы шифрования и расшифровки. Отношение между ключами является математическим – один ключ зашифровывает информацию, а другой ее расшифровывает.
- Асимметричное шифрование – система шифрования и/или электронной цифровой подписи (ЭЦП), при которой открытый ключ передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу, и используется для

проверки ЭЦП и для шифрования сообщения. Для генерации цифровой подписи и для расшифрования сообщения используется секретный ключ. Криптографические системы с открытым ключом в настоящее время широко применяются в различных сетевых протоколах, в частности, в защищенных протоколах передачи данных **TLS** и его предшественнике **SSL** (лежащих в основе протокола **HTTPS**), в протоколе безопасного удаленного управления **SSH**. Также используется в протоколах шифрования электронной почты **PGP** и **S/MIME.**

- Для того чтобы установить связь с использованием шифрования через открытый ключ, обеим сторонам нужно получить два ключа: открытый $К_1$ и частный (секретный) $К_2$ (рис. 2.3). Для шифрования и расшифровки данных обе стороны будут пользоваться разными ключами.
- В симметричной криптосистеме секретный ключ надо передавать отправителю и получателю по защищённому схеме распространения ключей (например из рук в руки или с помощью поверенного курьера). В асимметричной криптосистеме по подобной схеме передается только открытый ключ, а частный (секретный) ключ хранится на месте его генерации (у владельца).

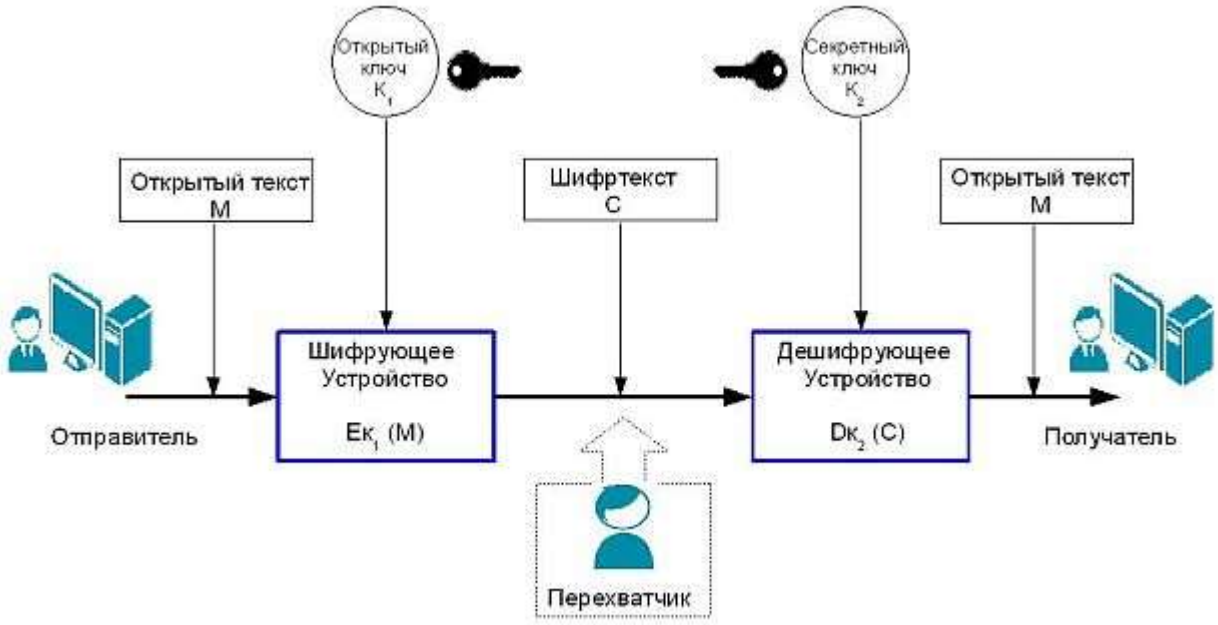


Рис. 2.3. Обобщенная схема асимметричной криптосистемы

- Механизмы генерирования пар открытых/частных ключей являются достаточно сложными, но в результате получаются пары очень больших случайных чисел, одно из которых становится открытым ключом, а другое – секретным. Генерирование таких чисел требует больших процессорных мощностей, поскольку эти числа, а также их произведения должны отвечать строгим математическим критериям. Однако этот процесс генерирования абсолютно необходим для обеспечения уникальности каждой пары открытых/частных ключей. Алгоритмы шифрования с использованием открытых ключей часто используют в приложениях, где аутентификация проводится с помощью цифровой подписи и управления ключами.

- Среди наиболее известных алгоритмов открытых ключей можно назвать **RSA** (Rivest-Shamir-Adleman) и **DSA** (Digital Signature Algorithm). Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм DSA применяется для создания цифровой подписи, но не для шифрования
- При использовании алгоритма RSA можно не только использовать открытый ключ для зашифрования, а секретный для расшифрования, но и наоборот: данные можно зашифровать с помощью секретного ключа, а открытый ключ применять при расшифровывании данных. Конечно, такой способ не дает возможность сохранять секреты, поскольку открытый ключ свободно доступен и любой может расшифровать информацию, но это дает способ поручиться за целостность содержимого сообщения: если открытый ключ правильно расшифровывает данные, значит, они были зашифрованы с помощью секретного ключа. Такой метод называется ***электронной цифровой подписью***. Идея применения цифровой подписи строится на двух основных предположениях: во-первых, что секретный ключ уникален и защищен (только владелец ключа имеет к нему доступ), и, во-вторых, единственным способом поставить цифровую подпись является применение этого секретного ключа.

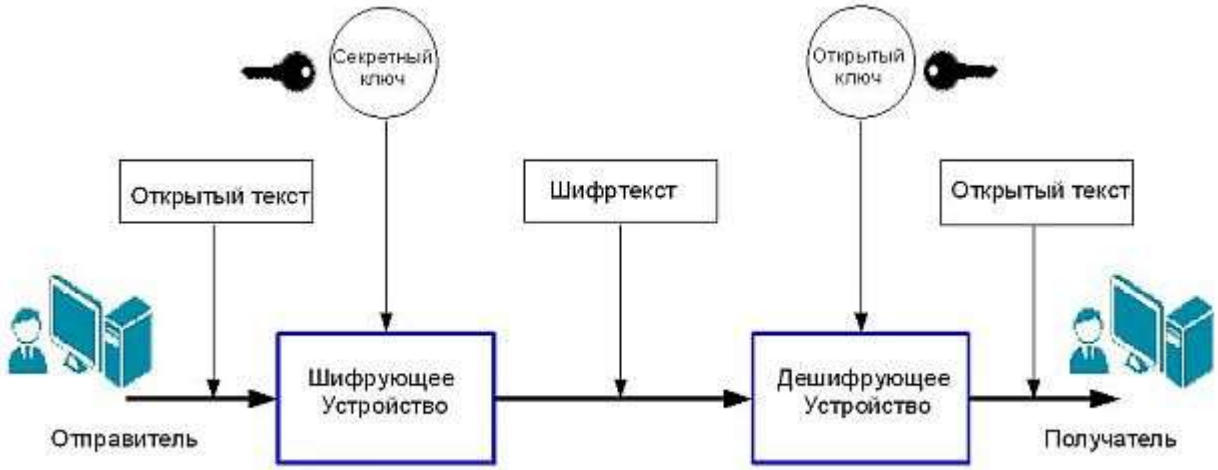


Рис. 2.4. Применение цифровой подписи

- Если данные зашифрованы с помощью не секретного ключа отправителя, а, например, открытого ключа или какого-либо другого, то получатель, расшифровав данные с использованием открытого ключа, получит не открытый текст сообщения, а бессмыслицу.
- Как правило, при создании цифровой подписи шифруется не весь открытый текст, а определенный фрагмент, так называемый ***дайджест сообщения*** (message digest), который генерируется (на основе вычислений) из исходного текста сообщения. К нему добавляется информация о том, кто подписывает документ. Получившаяся строка далее зашифровывается секретным ключом подписывающего с использованием того или иного алгоритма. Получившийся зашифрованный набор бит и представляет собой подпись.

- Отправитель посылает дайджест сообщения вместе с этим сообщением. При приеме получателем производятся такие же вычисления дайджеста. Если в сообщение были внесены изменения, результат вычисления будет отличаться от полученного, что свидетельствует о том, что целостность сообщения нарушена.
- Алгоритм вычисления дайджеста принимает входные данные любой длины и преобразует их, чтобы получить псевдослучайный результат фиксированной длины. Другой термин, часто использующийся для дайджеста сообщений, – это хэш.
- ***Хэш (hash)*** – это результат преобразования входных данных произвольной длины в данные фиксированной длины. Функция, с помощью которой реализуется это преобразование, называется *функция хэширования* или *хэш-функция*.
- В криптографии принято выделять *криптографически стойкие* хэш-функции, удовлетворяющие следующим условиям: во-первых, необратимость (т.е. невозможность восстановления исходного текста по результатам вычислений) и, во-вторых, стойкость к коллизиям. Коллизия – это ситуация, когда двум сообщениям соответствует один и тот же хэш. Наиболее часто в качестве алгоритма хэширования используется алгоритм **MD5** (генерируется 128-битное значение) или **SHA-1** (генерируется 160-битное значение).
- Хэш-функции широко применяются при создании пользовательских паролей, когда строка произвольной длины (пароль) преобразуется в указанный ключ заранее заданной длины, и для проверки целостности данных, когда данные отправляются вместе с контрольным значением.

Сертификаты открытых ключей

Криптография с открытым ключом предоставляет не только мощный механизм для шифрования, но и средства идентификации и аутентификации пользователей и устройств. Однако, если между отправителем и получателем нет конфиденциальной схемы передачи асимметричных ключей, то возникает серьезная опасность появления злоумышленника-посредника. Ручное распространение ключей становится непрактичным и создает бреши в системе безопасности. По этим причинам было разработано другое решение – применение ***сертификатов открытых ключей***.

В своей основной форме сертификатом называется информационный пакет, содержащий открытый ключ, электронную подпись, подтверждающую этот ключ, и имя доверенной третьей стороны, удостоверяющей достоверность этих фактов. Третья сторона, которой доверяют пользователи и которая подтверждает идентичность пользователя и его права, называется ***удостоверяющим центром CA (Certificate Authority)*** или ***центром сертификации*** . CA создает сертификаты пользователей, включающие следующую информацию: имя пользователя (сюда же относится пароль и другие дополнительные сведения, однозначно идентифицирующие этого пользователя), открытый ключ пользователя, время действия сертификата, конкретные операции, в которых этот ключ может быть использован (идентификация, шифрование). Удостоверяющий центр полностью ответственен за аутентичность своих конечных пользователей.

Удостоверяющие центры можно разделить на 2 категории: открытые и частные. Открытые CA действуют через Интернет, предоставляя услуги

сертификации всем желающим. Частные CA, как правило, принадлежат организациям или закрытым сетям и выдают сертификаты только пользователям локальных сетей.

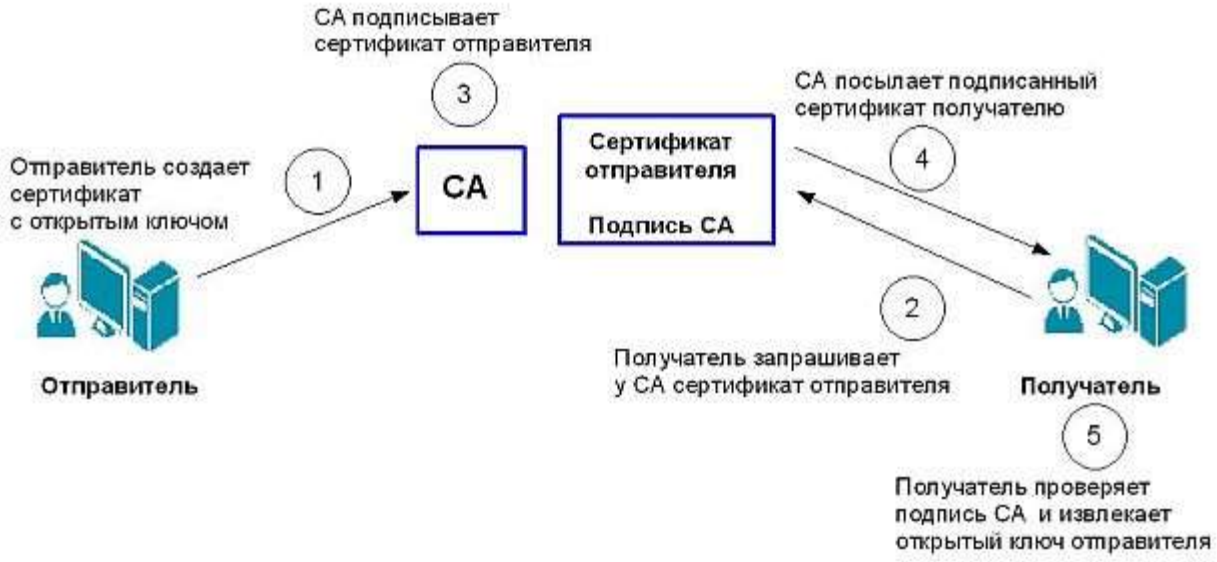


Рис. 2.5. Выдача сертификатов удостоверяющим центром CA

Наиболее широко распространенным форматом сертификатов, принятым для технологии открытых ключей, является *стандарт X.509 v.3*, описанный в документах RFC 2459 и RFC 5280.

Надежной системой распространения открытого ключа CA является *инфраструктура открытых ключей PKI (Public Key Infrastructure)*, которая определяет взаимодействие между конечными пользователями и доверяющими сторонами, предоставляя возможность безопасно выпускать и работать с сертификатами.

Криптографические средства широко применяются в различных программно-аппаратных элементах системы защиты информации, таких как межсетевые экраны, Интернет-маршрутизаторы, концентраторы доступа. Один из примеров реализации программно-аппаратной криптографической системы представлен комплексом "КРИПТОН" фирмы "Анкад". Программный комплекс *Crypton IPMobile* предназначен для организации виртуальной частной сети (VPN).

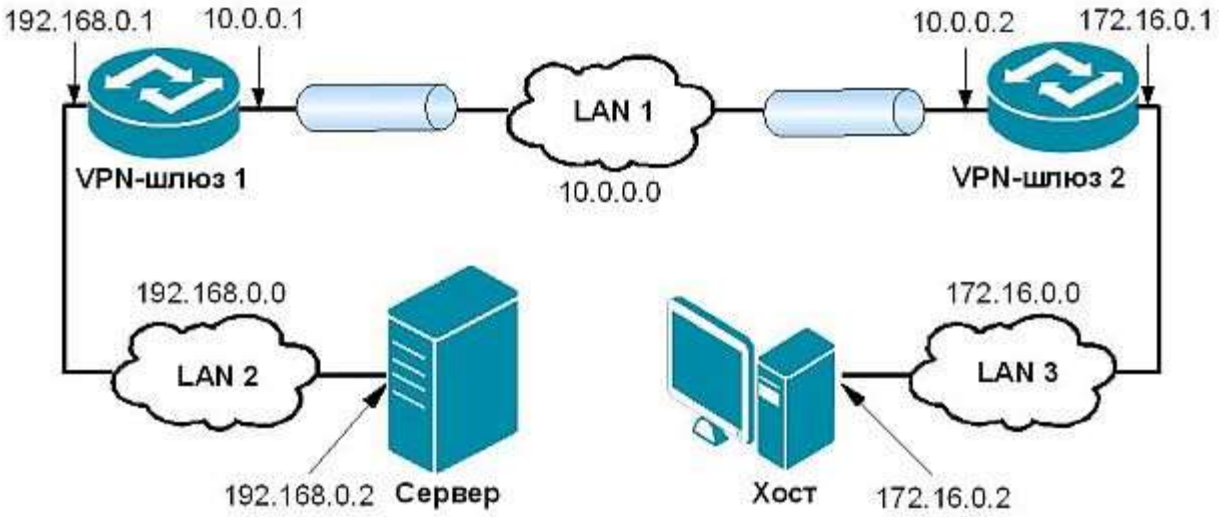


Рис. 2.6. Типичная схема организации частной виртуальной сети с использованием VPN-шлюзов

Crypton IPMobile может работать на основе концентратора доступа **D-Link DSA-3110**.

Рис. 2.7. Crypton-VPN на базе концентратора доступа D-Link DSA-3110

Программно-аппаратный комплекс **Crypton-VPN DSA-3110** обладает функциональностью криптографического маршрутизатора с применением шифрования. Концентратор доступа DSA-3110 представляет собой систему обеспечения доступа к сети с использованием технологии VPN, которая интеллектуально управляет аутентификацией, авторизацией и учетом подключающихся пользователей. Это устройство обеспечивает сеть полным набором функций, включая управление учетными записями и выдачу статистики по трафику с использованием технологии NetFlow.

Безопасность сетевого доступа

Концентратор доступа DSA-3110 обеспечивает клиентским компьютерам удобный и безопасный способ предоставления доступа к сети оператора/глобальной сети. Система гарантирует, что только зарегистрированные пользователи смогут использовать ресурсы сети. Используемые технологии доступа VPN (PPTP или PPPoE) позволяют надежно авторизовать пользователей и обеспечить защищенное и безопасное подключение. Активирование режима криптографического маршрутизатора и инициализация ключей и политики осуществляется с помощью информации, записанной на сменный носитель Touch Memory (TM).

Выполнение функций AAA

DSA-3110 выполняет 3 основные функции – аутентификацию, авторизацию и учет (Authentication, Authorization and Accounting (AAA)), которые часто встречаются во многих сетевых сервисах. Примером этих сервисов является доступ в Интернет через телефонную линию, электронная коммерция, печать через Интернет. Аутентификация выполняет проверку идентичности для авторизации доступа к сетевому ресурсу. Для учета использования ресурсов выдаются данные статистики подключений и соответствующая информация о трафике с целью анализа тенденций, планирования пропускной способности, биллинга, аудита и распределения затрат.

Использование в сетях операторов

DSA-3110 также может использоваться в коммерческих сетях, предоставляя сервисы клиентам (рис. 2.8). Для того чтобы удостовериться, что пользователь является именно тем, кем он себя заявил, необходима аутентификация. После того, как пользователь будет аутентифицирован, необходимо удостовериться, что он авторизован на выполнение тех операций, которые запрашивает. Авторизация обычно обеспечивается путем использования списков доступа или политик безопасности.

Использование в компаниях

Зачастую при предоставлении сотрудникам доступа в глобальную сеть предприятию необходимо разграничить доступ своих сотрудников к различным службам (рис. 2.9). DSA-3110 позволяет инициировать и использовать различные гибкие правила доступа к сети оператора/глобальной сети на основании множества критериев, таких как: адрес/служба/протокол/порт. Использование режима криптомаршрутизатора базируется на статических IP-адресах клиентов. При этом DSA-3110 является аппаратным межсетевым экраном (Firewall), что позволяет организовать защиту сети предприятия от внешних атак. Так же устройство позволяет сотрудникам, находящимся за пределами компании, получить защищенный доступ в локальную сеть предприятия.

Рис. 2.8. Схема применения DSA-3110 в сети оператора

Рис. 2.9. Схема применения DSA-3110 в сети компании

Цифровая подпись

Обычная подпись включена в документ; это - часть документа. Когда мы пишем чек, подпись находится на чеке; это - не отдельный документ. Но когда мы подписываем документ в цифровой форме, мы передаем подпись как отдельный документ. Передатчик передает два документа: сообщение и подпись. Получатель получает оба документа и проверяет, что подпись принадлежит предполагаемому передатчику. Если это доказано, сообщение сохраняется; иначе оно отклоняется.

Метод проверки

Второе отличие между двумя типами подписей - метод подтверждения подписи. В случае обычной подписи, когда получатель получает документ, он сравнивает подпись на документе с подписью в архиве. Если они одинаковые, документ подлинный. Получатель должен иметь в архиве копию для сравнения этой подписи. При цифровой подписи - получатель получает сообщение и подпись, но копия подписи не хранится нигде. Получатель должен применить методику проверки комбинации сообщения и подписи, чтобы проверить подлинность.

Отношения

В случае обычной подписи есть отношения "один ко многим" между подписью и документами. Человек использует ту же самую подпись, чтобы подписать много документов. Для цифровой подписи есть непосредственные отношения между подписью и сообщением. Каждое сообщение имеет свою собственную подпись. Подпись одного сообщения не может использоваться в другом сообщении. Если Боб получает два сообщения, один за другим, от Алисы, он не может использовать подпись первого сообщения, чтобы проверить второе. Каждое сообщение нуждается в новой подписи.

Резервное копирование

Другая разность между двумя типами подписей - качество, названное *резервное копирование*. При обычной подписи ее копия на поступившем документе может отличаться от копии, сохраненной в архиве. В цифровой подписи нет такого различия, если нет указателя времени (такого как метка времени) на документе. Например, предположите, что Алиса передает документ, инструктирующий Боба заплатить Еве. Если Ева перехватывает документ и сохраняет подпись, она может воспользоваться этим позже (скопировать подпись), чтобы снова получить деньги от Боба.

3.2. Процесс

Рис.3.1 показывает процесс цифровой подписи. Передатчик использует алгоритм подписания, чтобы подписать сообщение. Сообщение и подпись передают приемнику, приемник получает сообщение и подпись и применяет алгоритм подтверждения к комбинации "сообщение - подпись". Если результат истинен, сообщение принято; иначе - отклонено.

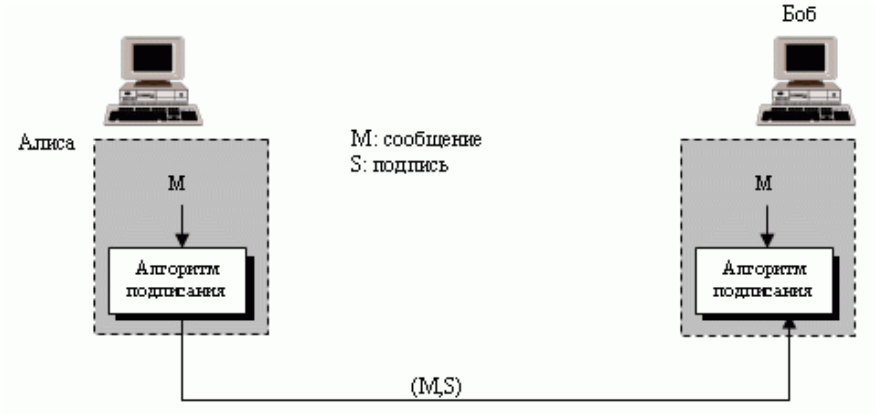


Рис. 3.1. Процесс передачи цифровой подписи

Потребность в ключах

Обычная подпись походит на частный "*ключ*", принадлежащий лицу, которое подписывает документ. Подписывающее лицо использует его, чтобы подписать документы; никто другой не имеет эту подпись. Копия подписи находится в архиве подобно открытому ключу; любой может использовать его, чтобы проверить документ, сравнить подпись с первоначальной подписью.

В цифровой подписи подписывающее лицо использует свой секретный *ключ* - прикладной *алгоритм подписания* - для подписания документа. С другой стороны проверяющий использует открытый *ключ* подписывающего лица - прикладной *алгоритм* подтверждения - для того, чтобы проверить документ.

Мы можем добавить секретный и открытый ключи к рис. 3.1, чтобы дать законченное понятие цифровой подписи (см. рис. 3.2). Обратите внимание, что когда документ подписан, любой, включая Боба, может проверить это, потому что каждый имеет *доступ* к открытому ключу Алисы. Алиса не должна использовать свой открытый *ключ* для того, чтобы подписывать документ, потому что тогда любой сможет подделать ее подпись.

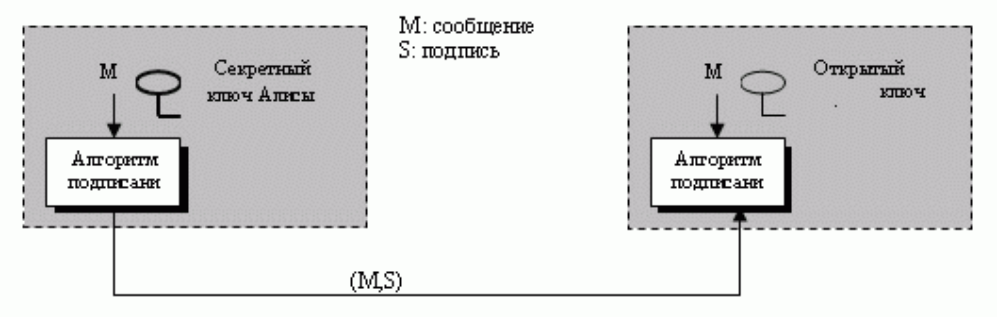


Рис. 3.2. Дополнение передачи цифровой подписи ключом Алисы

Можем ли мы использовать секретный (симметричный) *ключ*, чтобы и подписать и проверить подпись? Ответ отрицателен по нескольким причинам.

Первая: *ключ* засекречивания известен только двум объектам (например, Алисе и Бобу). Так, если Алиса должна подписать другой документ и передать его Тэду, она должна использовать другой *ключ* засекречивания.

Вторая: как мы увидим, создавая *ключ* засекречивания для сеанса, в него включают признаки установления подлинности, которые используют цифровую подпись. Мы получим порочный круг.

Третье: Боб может использовать *ключ* засекречивания, который он использует между собой и Алисой, подписать документ, передать его Тэду и имитировать, что он прибыл от Алисы.

Цифровая подпись **нуждается в применении системы открытого ключа. Подписывающее лицо подписывается своим секретным ключом; принимающий проверяет его общедоступным ключом подписывающего лица.**

Мы должны понимать различие между секретным и открытым ключами, используемыми в цифровых подписях, и секретным и открытым ключами, которые применяются в *криптографической системе* для конфиденциальности. В последних секретный и открытый ключи приемника используются в процессе. Передатчик применяет открытый *ключ* приемника, чтобы зашифровать сообщение; приемник использует свой собственный секретный *ключ*, чтобы расшифровать сообщение. В цифровой подписи нужен и секретный и открытый ключи передатчика. Передатчик использует свой секретный *ключ*, приемник использует открытый *ключ* передатчика.

Криптографическая система использует секретный и открытый ключи приемника, *цифровая подпись* **- секретный и открытый ключи передатчика.**

Подписание дайджеста

"B. Стандарты и организации по стандартизации" мы узнали, что криптосистемы с асимметричными ключами очень неэффективны, когда имеют дело с длинными сообщениями. В системе цифровой подписи сообщения обычно длинные, но мы должны использовать асимметрично-ключевые схемы. Решение состоит в том, что подпись дайджеста сообщения - намного короче, чем сообщение. Как мы узнали в "C. Набор протоколов TCP/IP" , тщательно выбранный дайджест сообщения имеет непосредственное отношение к сообщению. Передатчик может подписать дайджест сообщения, а приемник может проверить дайджест сообщения, - эффект тот же самый.

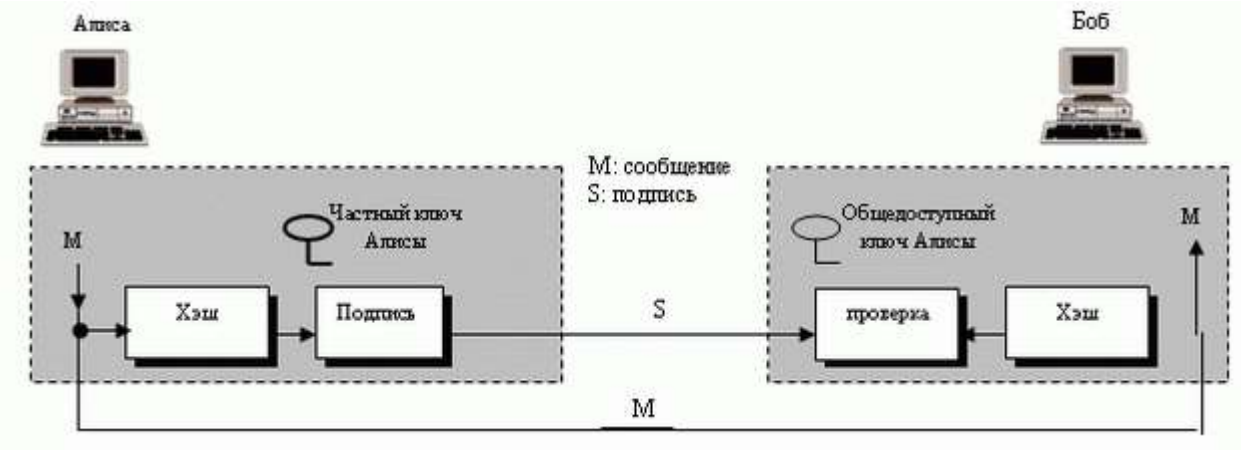


Рис. 3.3. Подписание дайджеста

Дайджест получается из сообщения на стороне Алисы и проходит процесс подписания, используя секретный ключ Алисы. Алиса затем передает сообщение и подпись Бобу. Как мы увидим позже в этой лекции, есть варианты в процессе подписания, которые зависят от системы. Например, могут быть проведены дополнительные вычисления, прежде чем получается дайджест, или может быть использована другая система засекречивания. В некоторых системах подпись имеет множество значений.

На стороне Боба используется та же самая общедоступная хэш-функция, и сначала из полученного сообщения создается дайджест. Далее вычисляются подпись и дайджест. Процесс подтверждения также проверяет прикладные критерии по результатам вычисления, чтобы определить подлинность подписи. Если подпись подлинна, то сообщение принимается; в противном случае она отклоняется.

Контрольные вопросы:

1. Что такое методы защиты информации в компьютерных сетях?
2. Симметричное и асимметричное кодирование.
3. Открытый и закрытый коды.
4. Что такое электронная подпись?
5. Виды электронной подписи.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Грошев А. С. Информатика: лабораторный практикум / А.С. Грошев. – Архангельск, Арханг. гос. техн. ун-т, 2012. – 148 с.
2. Грошев А. С. Информатика: Учебник для вузов – Архангельск, Арханг. гос. техн. ун-т, 2010. – 470 с.
3. http://www.intuit.ru – Сайт Национального открытого университета ИНТУИТ
4. Сергеев А. П. Microsoft Office 2010. Самоучитель. 2010 г. С – 624.
5. http://pascalabc.net – Сайт системного программиста на языке Pascal.
6. Симонович С.В. Информатика. Базовый курс: Учебник для вузов. 3-е изд. Стандарт третьего поколения. СПб.: Питер, 2015.
7. Уткин В.Б., Балдин К.В., Рукосуев А.В. Математика и информатика. Учебное пособие. Дашков и К, Москва, 2014 г., С-470.