МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»
Чистопольский филиал «Восток»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

Информационные технологии в приборостроении

Индекс по учебному плану: **Б1.В.05**

Направление подготовки: **12.03.01 Приборостроение**

Квалификация: **Бакалавр**

Профиль подготовки: **Приборостроение**

Типы задач профессиональной деятельности: **производственно-технологический, проектно-конструкторский**

Рекомендовано УМК ЧФ КНИТУ-КАИ

Чистополь
2023 г.

Введение в Visual Basic

Основные термины

1. ***Алгоритм*** - это четкое описание последовательности действий, которые необходимо выполнить для решения поставленной задачи.
2. ***Программа*** - это алгоритм, записанный на языке программирования.
3. ***Языком программирования*** называется специальный язык, понятный для компьютера.
4. ***Программирование*** - это процесс создания, отладки и тестирования программ.

Создание любой программы начинается с разработки алгоритма. Именно четкое описание последовательности действий позволяет мысленно представить будущую программу. Построив алгоритм, программист мыслит четко, последовательно, однозначно - так, как и будет впоследствии мыслить компьютер.

Этапы создания программы

1. ***Постановка задачи*** - составление точного и понятного словесного описания того, как должна работать программа, что должен делать пользователь в процессе ее работы.
2. ***Разработка интерфейса*** (интерфейс - способ общения) - создание экранной формы (окна программы).
3. ***Составление алгоритма.***
4. ***Программирование*** - создание программного кода на языке программирования.
5. ***Отладка программы*** - устранение ошибок.
6. ***Тестирование программы*** - проверка правильности ее работы.
7. ***Создание документации, помощи.***

Общие сведения о Visual Basic

Visual Basic - это универсальный язык программирования для начинающих.

В 1991 г. *Microsoft* представила *Visual* Basic. Первое признание серьёзными разработчиками *Visual Basic* получил после выхода версии 3 — VB3. Окончательное признание как полноценного средства программирования для Windows — при выходе версии 5 — VB5. Версию VB6, входящую в состав *Microsoft Visual Studio 6.0*, стала по-настоящему зрелым и функционально богатым продуктом. После этого разработчики из Microsoft существенно изменили направление развития данной технологии.

Visual Basic - один из первых языков, поддерживающий *событийно управляемое программирование (event-driven programming).*

В событийно-управляемом программировании вместо описания каждого шага вы лишь указываете, как реагировать на различные события (действия пользователя): выбор команды, щелчок кнопкой мыши в окне, перемещение мыши. На одни события можно предусмотреть реакцию, другие – просто игнорировать. Вы создаете не одну большую программу, а приложения *Windows*, состоящие из набора взаимодействующих микропрограмм (процедур), управляемых пользователем. С помощью *Visual Basic* такое приложение можно разработать достаточно быстро и без глубоких познаний и навыков.

Достоинства Visual Basic:

1. *Visual Basic* выгодно отличается от других языков программирования своей простотой и наглядностью.
2. *Visual Basic* динамично развивающийся язык.
3. *Visual Basic* встроен в такие программы как Word, Excel и др. С его помощью можно управлять этими программами из других программ.

Visual Basic - это объектно-ориентированный язык. Основой языка являются объекты. Например: окно, кнопка, поле со списком, с которыми работает программа.

Каждый объект имеет:

- свойства
- методы
- события

Свойства - это показатели, характеризующие объект.

Методы - это действия, которые можно произвести с объектом.

События - это действия, которые происходят с объектом.

Например: объект "телефон"

Свойства:	Методы:	События:
цвет	снять трубку	звонок
размер	набрать номер	длинный гудок
расположение	передвинуть	короткий гудок
	телефон	
вес		
объем		

В проектируемом приложении события будут возникать в основном в результате действий пользователя. Например, пользователь нажал на кнопку мыши или клавиатуры, запустил или завершил программу.

Среда программирования Visual Basic

Пуск - Программы - Microsoft Visual Basic 5.0, 6.0. - Microsoft Visual Basic 5.0, 6.0.

После запуска Visual Basic появляется окно создания проекта

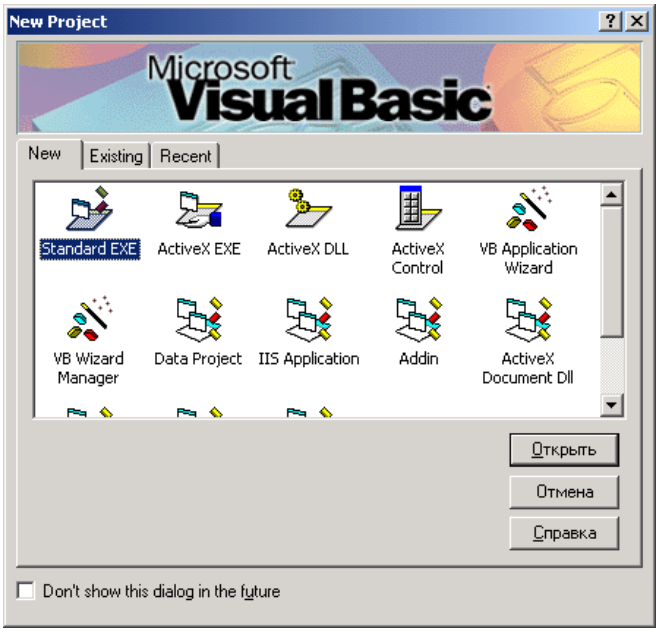


Рис.1.

Мы будем рассматривать только StandartEXE проекты

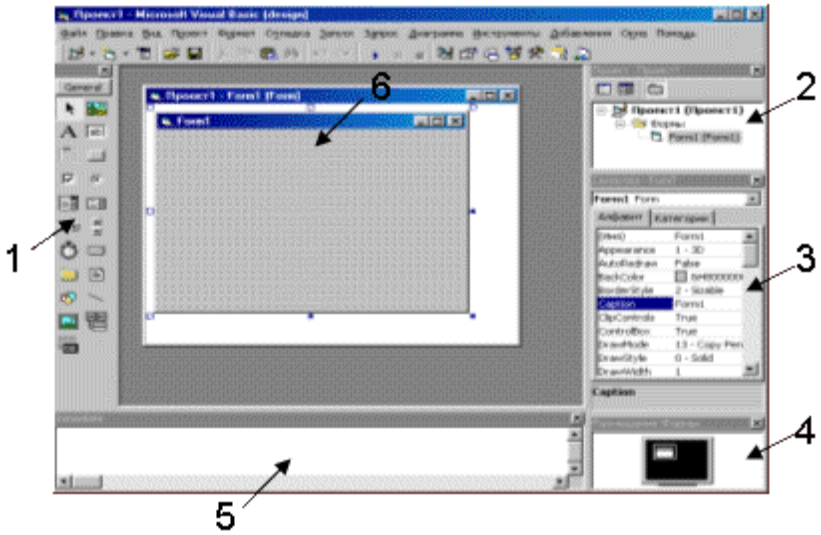


Рис.2.

Окно среды программирования Visual Basic.

1. панель элементов управления.
2. окно проводника проекта.
3. окно свойств текущего элемента управления или формы.
4. окно размещения формы на экране монитора.
5. окно для ввода команд. Команды выполняются сразу после ввода.
6. окна, содержащие формы, модули и другие элементы проекта.

Если какие-либо из перечисленных элементов не видны, то их можно вывести на экран (или скрыть) с помощью меню Вид (View).

Сохранение проекта

Проект приложения сохраняется в отдельном файле и также в отдельных файлах сохраняются элементы проекта.

При первом сохранении указываются имена файлов для всех элементов проекта. Поскольку проект состоит из нескольких файлов, то для него лучше создать отдельную папку.

Работа с элементами среды программирования

Элементы среды программирования - это небольшие окна, которые выводят различную информацию и позволяют управлять составными частями проекта.

1. **Проводник проекта** - отображает группы объектов [Группа] (например: Формы, Модули). В группах находятся непосредственно сами объекты: формы [Форма] , модули [Модуль] .

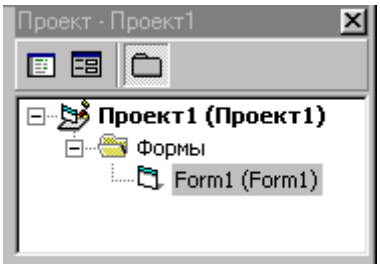


Рис.3.1.

2. **Окно свойств** - отображает свойства текущего объекта (формы или элементов управления: кнопок, списков, переключателей).

Чтобы сделать элемент текущим, необходимо щелкнуть по нему.

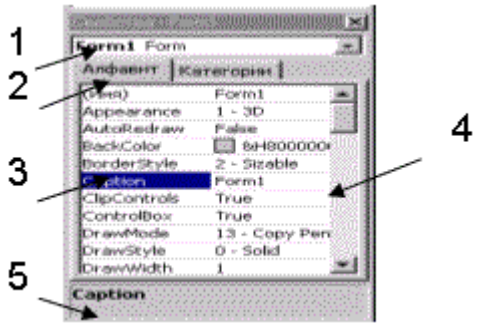


Рис.3.2.

Окно свойств.

1. Имя объекта, свойства которого отображаются.
2. Вкладки: Алфавит и Категории - изменяют порядок сортировки свойств: по алфавиту, по категориям (группам похожих свойств).
3. Графа: Название свойства.
4. Графа: Значение свойства.
5. Комментарий текущего (выбранного) свойства.

Значение свойства вписывается с клавиатуры, либо выбирается из списка. Список значений открывается кнопкой , кнопка с тремя точками открывает окно диалога, например, для выбора файлов или шрифтов.

Значения свойств могут быть логическими, т.е. иметь значения:

1. True - то есть Да, Истина, 1.

2. False - то есть Нет, Ложь, 0.

3. **Окно размещения формы на экране** - показывает, как будет расположена форма на экране после запуска программы.

В этом окне на изображенном экране монитора можно перетаскивать форму мышью.

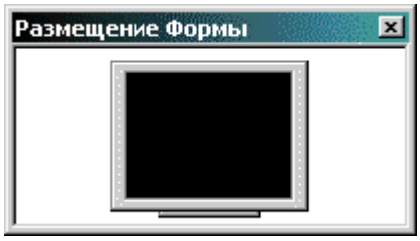


Рис.3.2.

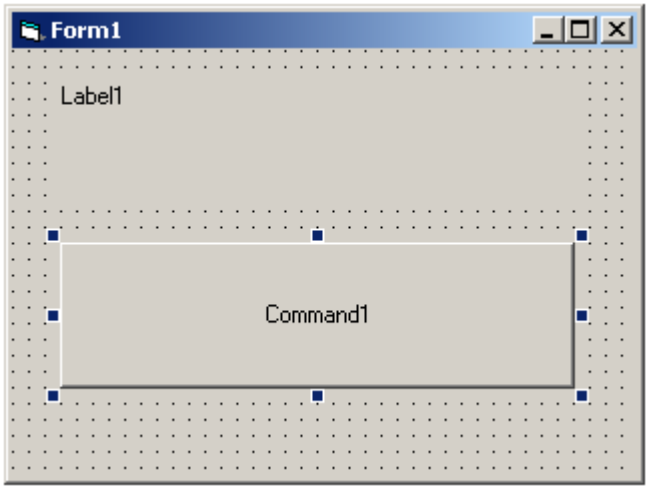
Проект 1. Первая программа.

Первый ваш проект будет построен с использованием двух элементов управления: *кнопки (command*) и *надписи (label*).

 *Command*

 *Label*

С помощью панели элементов управления разместить элементы как показано на рис.



Сделаем активной надпись на форме, щелкнув на ней мышкой. Теперь окно свойств показывает свойства, соответствующие надписи. Выберем свойство *Caption* и присвоим ему значение *“Моя первая программа ”*, выберем шрифт с

помощью свойства *Font*. Подгоним размеры метки так, чтобы текст свободно располагался на ней.

Сделаем активной кнопку и зададим свойство *Caption* в окне ее свойств: *“Жми сюда”*.

Теперь переходим к программной части программы. Кликаем два раза на кнопку, и появляется окно с кодом

В этом окне нужно написать один единственный оператор

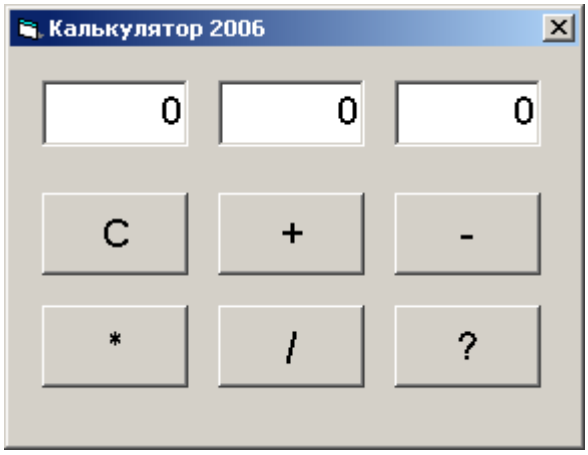
Label1.Caption = "Теперь я программист"

между строкой *Private Sub Command1_Click* и строкой *End Sub.*

Теперь, если пользователь щелкнет на командной кнопке (объект *Command1*, событие *Click*) во время работы проекта, меняется надпись.

Запустите свой проект, нажав клавишу *F5* и вы увидите свой первый проект в работе. Щелкните мышкой на кнопке.

Проект 2. Калькулятор v.1.



Form1

Caption – Калькулятор 2006

StartUpPosition – 2-CenterScreen

BorderStyle – 1-FixedSingle

TextBox1-3

Alingment – 1-RightJustify

Text – ~~Text~~

Font – см.

Command1-6

Caption – см.

Font – см.

```
Private Sub Command1_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = "0"
End Sub

Private Sub Command3_Click()
Text3.Text = Text1.Text – Text2.Text
End Sub

Private Sub Command6_Click()
MsgBox "Автор Иванов С.А."
End Sub
```

Свойства, методы и события формы и элементов управления.

Форма

Форма - это эскиз окна будущей программы. Форма является объектом, имеет свои свойства, методы, события. Форма является контейнером для других объектов, т.е. она может содержать другие элементы (кнопки, списки, текстовые поля и т.п.)

В проекте может содержаться несколько форм. При запуске программы появляется стартовая форма. Форма сохраняется в отдельном файле (точнее в 2-х файлах с разными расширениями).

Свойства формы

BackColor – цвет фона.

ForeColor – основной цвет. Все операторы Print выводят текст цветом, заданным в этом свойстве.

Цвета

Программно можно изменять цвет 3-мя способами:

1 способ) с помощью констант.

vbBlack, vbRed, vbGreen, vbYellow, vbBlue, vbMagenta, vbCyan, vbWhite

Например: Form1.BackColor = vbRed

2 способ) с помощью функции QBColor

QBColor(Color)

Значение функции QBColor	**Цвет**
0	**Черный**
1	**Синий**
2	**Зеленый**
3	**Фиолетовый**
4	**Красный**
5	**Малиновый**
6	**Желтый**
7	**Белый**
8	**Серый**
9	**Светло-синий**
10	**Светло-зеленый**
11	**Светло-фиолетовый**
12	**Светло-красный**
13	**Светло-малиновый**

14	**Светло-желтый**
15	**Ярко-белый**

Например: Form1.BackColor = QBColor(11)

3 способ) с помощью функции RGB

RGB(Red, Green, Blue)

Например: Form1.BackColor = RGB(200, 0, 0)

Caption – Текст, выводимый в строку заголовка

BorderStyle – Тип границ

Height – Высота формы

Width – Ширина формы

Top – Верхняя координата верхнего левого угла формы на экране

Left – Левая координата верхнего левого угла формы

Font Bold – Полужирный шрифт выводимого в форму текста

Font Italic – Курсивный шрифт выводимого в форму текста

Font Strikethru – Зачеркнутый шрифт выводимого в форму текста

Font Underline – Подчеркнутый шрифт выводимого в форму текста

Font Name – Имя шрифта выводимого в форму текста

Font Size – Размер шрифта выводимого текста

Enabled – Доступность или недоступность формы

Picture – Картинка, изображение на форме

Visible – Видимость или невидимость формы

Window State – Состояние окна (нормальное, свернутое, развернутое)

StartUpPosition –- Стартовая позиция формы

События формы

Click **–** Щелчок кнопкой мыши на форме

Dblclick **–** Двойной щелчок мыши на форме

Load **–** Загрузка формы

Key Press **–** Нажатие клавиши

Методы формы

Cls **–** Это***m*** метод очищает форму от всех изображений и текста

Print **–** Этот метод выводит текст на форму

Load Picture **–** Функция для установки у формы свойства Picture

Элементы управления

Элементы управления - это объекты, которые служат для организации интерфейса между пользователем и компьютером. Например: кнопки, списки, переключатели. *Элемент управления* - это объект, имеющий свои свойства, методы, события. Элементы управления добавляются с помощью панели элементов управления.

Рис.1.

1. *Надпись [A] Label* - служит для добавления текста на форму. Этот текст не может быть изменен пользователем, но может быть изменен программой.

Свойства:

* Caption - текст надписи.

* Font - шрифт, его размер, начертание.

* Alignment - выравнивание текста: Left (влево), Right (вправо), Center (по центру).

* ToolTipText - подсказка, появляющаяся при наведении указателя мыши на элемент управления.

* ForeColor - цвет текста.

* BackColor - цвет фона.

2. *Текстовое поле [] - TextBox*

Служит для того, чтобы пользователь мог ввести текст во время работы программы.

Свойства:

Text - содержит символы, которые ввел пользователь. Другие свойства аналогичны элементу "Надпись".

3. *Рамка [] - Frame* - используется для оформления, а также для группировки переключателей. Сверху на рамке можно сделать надпись с помощью свойства Caption. Если нужно создать элемент внутри рамки, то перед их рисованием рамку выделяют, тогда рамка может служить контейнером для группы переключателей.

4. *Кнопка [] Command Button.*

Свойства:

* Caption - надпись на кнопке.

* Enabled - доступность элемента. С помощью этого свойства блокируются элементы, которые пользователю нельзя использовать в данный момент. Заблокированные элементы отображаются серым цветом. В

заблокированное текстовое поле не получится ввести текст, а заблокированную кнопку нельзя нажать.

Выбирают из двух значений:

* True (Да) - элемент управления доступен пользователю.

* False (Нет) - элемент управления не доступен.

* Visible - видимость элемента управления:

* True (Да) - виден.

* False (Нет) - не виден.

5. *Флажок [] - Check Box*

Используется, когда пользователь должен ввести Да (флажок установлен [Check Box]) или Нет (флажок снят [Check Box])

Свойства:

Value - содержит значение элемента управления. Имеются следующие значения:

0. - нет, флажок снят;

1. - есть, флажок установлен;

2. - флажок недоступен.

6. *Переключатель [] - Option Button*

Позволяет пользователю выбрать один вариант из нескольких.

Свойства:

Value - показывает выбрана опция (1) или нет (0).

7. *Поле со списком [] ComboBox*

В это поле пользователь может вводить текст так же, как и в TextBox, а кроме этого, если нажать , то откроется список, из которого можно выбрать нужную строку.

Свойства:

* Text - содержимое строки, введенное пользователем или выбранное из списка.

* List - строки списка (многострочное свойство).

* ListIndex - номер выбранной пользователем строки (нумерация начинается с нуля, если никакая строка не была выбрана, то свойство равно –

8. *Список [] ListBox*

Содержит список строк, в котором пользователь может выбрать одну или несколько строк.

Свойства элемента аналогичны элементу ComboBox, за исключением свойства Text, которое здесь отсутствует.

9. *Рамка для рисунка [] PictureBox*

Содержит рисунок. Рисунок выбирается с помощью свойства Picture, в которое вводится имя файла или выбирается с помощью кнопки .

Понятие программного кода

Программный код - это набор слов и символов языка программирования.

Слова и символы должны быть записаны строго по правилам языка, без орфографических и пунктуационных ошибок. Именно точное написание позволит компьютеру однозначно понять и выполнить программу.

Окно программного кода

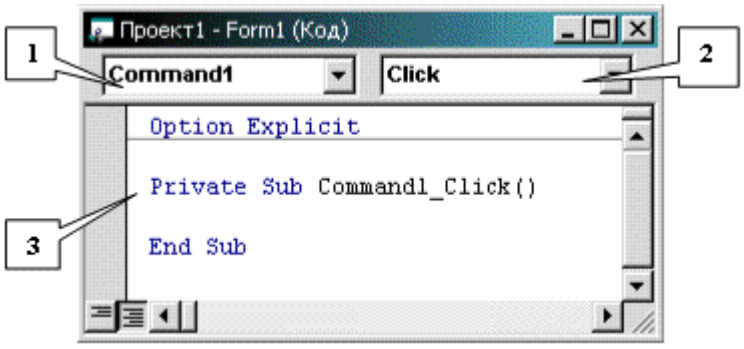
Программный код записывается в окне кода. Такое окно имеется у каждой формы.

Открыть окно кода можно:

1 способ - в окне Проводник Проекта щелкнуть правой кнопкой по нужной форме и в открывшемся меню выбрать Показать код (View Code).

2 способ - дважды щелкнуть по элементу управления на форме или по самой форме в окне формы. Примечание: при этом не только открывается окно кода, но и создается процедура обработки события (см. ниже).

Структура окна кода:



1. Список элементов управления
2. Список событий элементов управления
3. Процедура (код)

Процедуры

Процедура - это обособленный фрагмент программного кода, с помощью которого решается обычно небольшая задача.

Процедуры бывают:

1. **Процедуры обработки событий**. Выполняются при возникновении какого-либо события в каком-либо элементе управления (или форме).

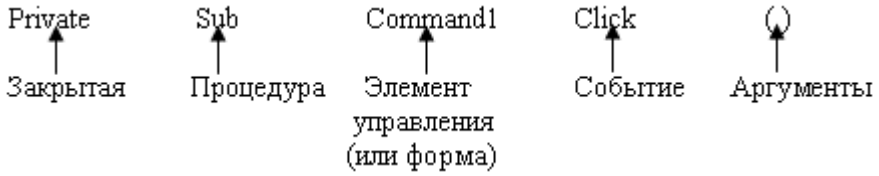
2. **Произвольные процедуры**. Она не связаны с событиями и могут быть вызваны из любой другой процедуры и выполнены в любое время.

Структура процедуры

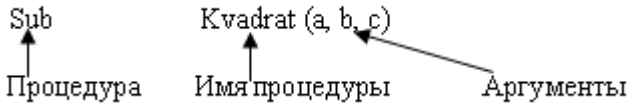
Процедура состоит из следующих элементов:

1. **Заголовок процедуры** - отмечает начало процедуры, ее тип, назначение (событие).

Пример заголовка процедуры, которая выполняется при щелчке мышью по кнопке с именем Command1.



У произвольных процедур заголовок следующий:



Имя процедуры должно быть уникально, должно начинаться с буквы, не должно содержать пробелов и других знаков, кроме знака подчеркивания. По имени происходит вызов процедуры, когда необходимо ее выполнить.

2. **Окончание процедуры** - заканчивает программный код процедуры.

End Sub

Примечание: у функций: End Function

3. **Тело процедуры** - это строки между заголовком и окончанием. Их количество неограниченно. Строки содержат предписания, которые должны выполниться при вызове процедуры (возникновении события).

Создание процедуры

Для создания процедуры выполните следующее:

1 способ - дважды щелкните по нужному элементу управления или форме. Откроется окно кода, а в нем появится заголовок и окончание процедуры.

Если необходимо другое событие, то его выбирают с помощью списка в верхнем правом углу окна кода.

2 способ - откройте окно кода, выполните Инструменты - Добавить процедуру - укажите имя и параметры процедуры - Ok.

Вызов процедур на исполнение

1. Чтобы выполнилась процедура обработки события, это событие должно произойти.

2. Для выполнения произвольной процедуры в теле другой процедуры указывают имя этой процедуры.

Private Sub Command1_Click( )

Kvadrat

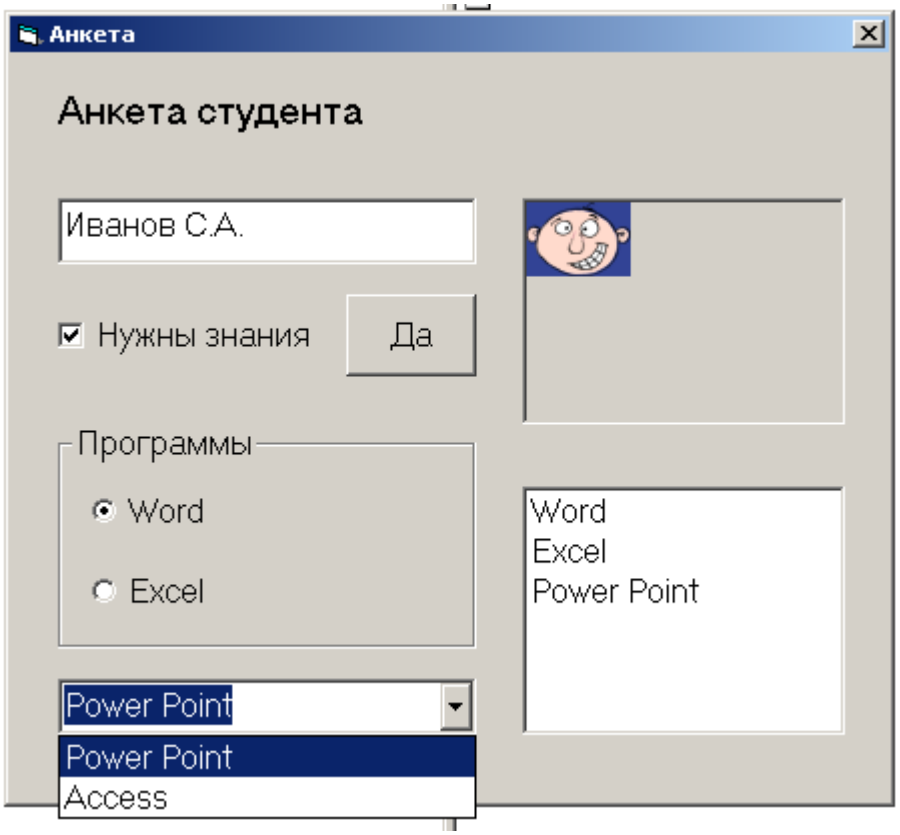
End Sub

Здесь при нажатии на кнопку Command1 возникает событие Click (щелчок мышью) и вызывается и выполняется процедура Kvadrat.

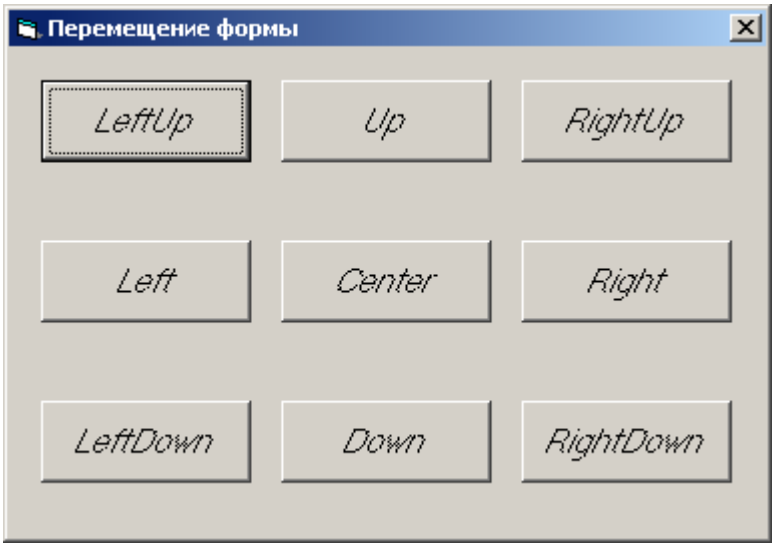
Код процедуры выполняется построчно и сверху вниз.

Проект 1. Анкета Студента.

Создать интерфейс.



Проект 2. Перемещение формы.



Form1

Caption – Перемещение формы

StartUpPosition – 2-CenterScreen

BorderStyle – 1-FixedSingle

Command1 – 9

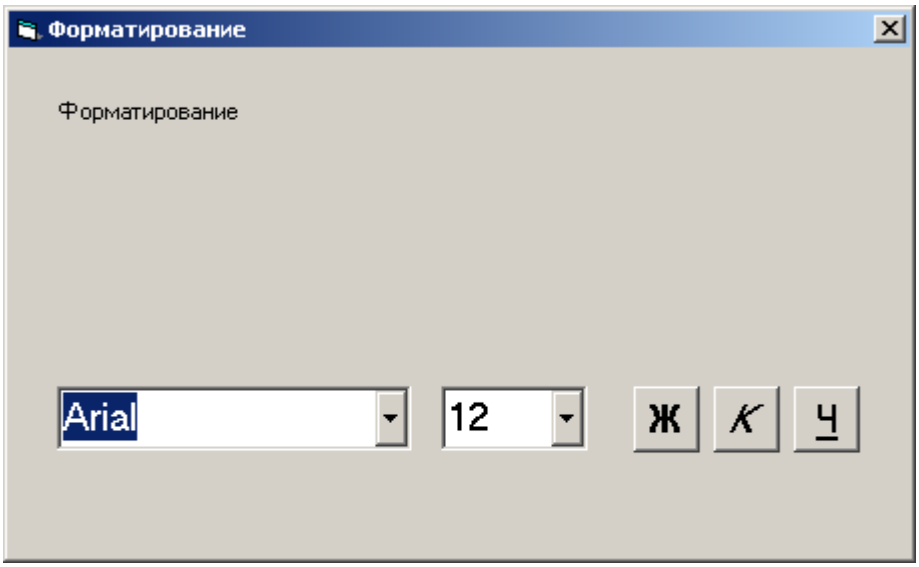
Caption – см. рис.

Font – см.

```
Private Sub Command1_Click()
Form1.Left = 0
Form1.Top = 0
End Sub

Private Sub Command8_Click()
Form1.Top = Screen.Height - Form1.Height
End Sub
```

Проект 3. Форматирование.



Form1, Label1, Combo1-2, Command1-3 – см.

Чтобы работало. (Для Combo1-2 нужно использовать событие Click)

Переменные и функции.

Переменная - эта ячейка памяти, хранящая какое-либо одно значение (одно число, один фрагмент текста). Переменная имеет имя и значение. Переменные служат для хранения исходных данных, используемых в программе, а также результатов вычислений.

Примечание: свойства объектов также являются переменными, т.к. тоже хранят определенные числовые или текстовые значения.

Имя переменной - это строка символов, которая отличает ее от других переменных и объектов программы (элементов управления). Таким образом, имена переменных должны быть уникальны.

Правила написания имен переменных:

1. Имя переменной должно начинаться с буквы.

2. Остальными символами могут быть буквы (строчные или прописные), цифры и символ подчеркивания. Пробел, точка, запятая и другие специальные знаки - недопустимы.

3. Длина имени не должна превышать 255 символов.

4. Имя переменной не должно совпадать с ключевыми символами языка Visual Basic (например: If, Then, For, To, Next, Print и другими).

Правильные имена: a, a1, a_1, СуммаИтогоПоГрафе, if2

Неправильные имена: 1, 1a, a 1, Сумма: итого по графе, if.

Значение переменной - это данные, которые в ней хранятся.

Тип данных (тип переменной) обуславливает то, как хранятся и обрабатываются данные.

Например: 2 + 3 = 5 - числа, а "2" + "3" = "23" – текст

Тип данных	**Объем занимаемой памяти**	**Краткая запись**
Целые типы		

Integer (целое)	**2 байта**	**%**
Long (длинное целое)	**4 байта**	**&**
Byte (Байт)	**1 байт**	
Boolean (булево)	**2 байта**	
Плавающие типы		
Single Десятичные числа	**4 байта**	**!**
Double Десятичные числа	**8 байт**	**#**
Строковые типы		
String Текстовая информация	**1 байт на каждый символ**	**$**
Объектные типы		
Object Рисунки или ссылки на любой другой объект	**4 байта**	
Variant-типы		
Variant (числовые типы)	**16 байт**	
Variant	**22 байта**	

(строковые типы)	**+длина строки**	
Прочие типы		
Currency Число в денежном формате	**8 байт**	**@**
Date Дата	**8 байт**	

Переменная типа Variant может принимать любой тип в зависимости от хранящегося в ней значения, однако, занимает больше памяти.

Если переменная не была объявлена, то она имеет тип Variant.

Присвоение значения переменной

Для присвоения значения служит оператор присваивания, которым является знак равно (=).

Оператор - это слово или знак, выполняющий одно предписание (одну операцию).

В математических выражениях используются знаки арифметических операций:

^ возведение в степень * умножение

/ деление \ целочисленное деление

Mod остаток от деления + сложение

– вычитание

Для изменения порядка операций используются только круглые скобки ( ).

Математическое выражение сначала обрабатывается и вычисляется результат, который затем присваивается переменной (помещается в переменную). При написании чисел в коде программы целая и дробная часть числа отделяются точкой.

Например:

1. Переменной присваивается конкретное значение:

a = 5 b = "Мир"

2. Переменной присваивается результат выражения:

a = 2 * 3 + 10/2 b = "Миру" + "" + "Мир!"

3. Переменной присваивается значение выражения, в котором используются другие переменные:

a = (i + 5) * 2.5 b = c + d + "май" c = b

Аналогично присваиваются значения свойствам. Это записывается так:

Объект.Свойство = Значение

Например: Command1.Caption = "Вася"

Эта строка изменяет надпись на кнопке с именем Command1 на Вася.

Свойства могут быть как слева, так и справа от знака равно. Подумайте, что произойдет после выполнения следующей строки:

Command1.Caption = Form1.Caption

Объявление типа переменной

Поскольку Visual Basic может сам определить тип переменной по ее значению (см. Variant), то тип переменной можно не объявлять (не указывать). Однако, объявление переменных является правилом хорошего тона программирования а так же позволяет избежать некоторых ошибок в программе, таких как неодинаковое написание имени переменной в разных местах программы, несоблюдения типов данных в вычислениях.

Переменную объявляют в начале окна кода или в начале процедуры с помощью оператора Dim такой записью:

Dim Имя_переменной As Тип_переменной

Например: Dim a As Byte

Объявлена переменная a типа Byte. В такую переменную можно поместить число от 0 до 255. При попытке присвоения переменной большего числа возникнет ошибка, а число, имеющее дробную часть будет округлено.

Dim b As String

Объявлена переменная b типа String. Эта переменная будет содержать текст (хотя текст может состоять и из цифр, но 2 плюс 3 будет 23).

При объявлении нескольких переменных можно перечислять их через запятую: Dim a As Byte, b As String

После строки объявления можно присваивать значение переменной.

Например:

```
Dim a As Byte
a = 5
```

Область видимости переменной

Оператор, объявляющий переменную, сообщает Visual Basic, что будет храниться в этой переменной, и где эту переменную можно использовать. Область, где используется переменная, называется областью видимости переменной. Если переменная доступна, то она существует, не наоборот. Переменная может существовать в памяти и быть доступной для некоторых частей программного кода и при этом быть недоступной «невидимой» для других.

Вы можете объявить переменную для работы в пределах одной процедуры, в любой процедуре данной формы или для работы во всей программе.

Когда вы объявляете переменную, область ее видимости задается одним из ключевых слов: Dim, Private, Public. Однако область видимости переменной зависит и от того, где она объявлена.

Dim. Таким образом объявляют локальные переменные, которые существуют только во время вызова процедуры, где они объявлены. Но если переменная с помощью Dim объявлена в разделе глобальных объявлений формы или модуля, то она будет доступна во всех процедурах этой формы или модуля, но для других форм и модулей такая переменная будет «невидимой».

Private. Отличается от Dim тем, что не может объявлять переменные внутри процедуры или функции. При объявлении же в разделе глобальных объявлений формы или модуля Dim и Private равнозначны.

Public. Если переменная объявлена с использованием этого ключевого слова, то она является глобальной и доступна из всех форм и модулей проекта.

Если переменная объявлена как Public в коде формы, то из других форм и модулей доступ к ней должен осуществляться через следующую конструкцию:

ИмяФормы . ИмяПеременной

Если переменная объявлена как Public в разделе объявлений программного модуля, то доступ к ней возможен просто через ее имя.

Static. Переменные Static объявляются внутри процедур и функций и вне их недоступны, но после завершения работы процедуры или функции, в которой объявлены, сохраняют свое значение.

По умолчанию объявленная переменная является локальной, действующей только в пределах той процедуры, в которой она была создана.

Статические переменные

Большинство локальных переменных, созданных внутри процедуры, становятся не нужны после завершения процедуры. Однако бывают случаи, когда необходимо сохранить значение переменной после того, как процедура выполнена. Это бывает, когда процедура вызывается несколько раз и ее значение зависит от ее предыдущего значения. Для создания такой локальной переменной, сохраняющей свое значение после завершения процедуры, в объявлении этой переменной необходимо использовать ключевое слово Static. Оно сообщает Visual Basic, что область действия переменной ограничивается процедурой, но переменная должна после выполнения процедуры сохранить свое значение.

Статические переменные обычно используются в событиях таймера.

Создание своих собственных констант

Часто бывает разумным вместо неоднократно повторяющихся строк или чисел использовать имя константы. В этом случае вы должны создать свою собственную константу. Для того, чтобы использовать константу, ее в программе сначала нужно определить. Определяются константы с помощью ключевого слова Const, которое присваивает константе имя и значение:

В именах констант обычно используют только прописные буквы, соединяя слова, из которых образовано имя константы, знаком подчеркивания (_). В

последнее время стало привычным для имен констант использование комбинации прописных и строчных букв и префикса, составленного из строчных букв.

Приведем пример объявления константы

Const numSentense As String = “В лесу растут грибы”

Const CUR_SUMMA As Integer = 255

Список функций по категориям

- **Математические функции**
  - Abs, Exp, Fix, Int, IsNumeric, Rnd, Round, Sgn, Sqr
- **Функции обработки массивов**
  - Array, IsArray, LBound, UBound
- **Функции обработки строк**
  - Asc, Chr, Filter, InStr, InStrRev, Join, LCase, Left, Len, LTrim, Mid, Partition
  - Replace, Right, RTrim, Space, Spc, Split, Str, StrComp, StrConv, String
  - StrReverse, Tab, Trim, TypeName, UCase, Val,
- **Тригонометрические функции**
  - Atn, Cos, Log, Sin, Tan
- **Функции преобразования типа данных**
  - CBool CByte CCur CDate CDbl CDec CInt CLgn CSgn CStr CVar CVDate
- **Функции загрузки данных**
  - Choose IIf InputBox LoadPicture LoadResData LoadResPicture LoadResString
  - MsgBox
- **Функции работы с файлами**
  - CurDir Dir EOF FileAttr FileDateTime FileLen FreeFile GetAttr Input Loc
  - LOF Seek
- **Функции обработки системных параметров**
  - CallByName Command CVErr DoEvents Environ Erl Error GetAllSettings
  - GetSetting IMEStatus IsEmpty IsError IsMissing IsNull Shell Switch
- **Функции обработки цвета**
  - QBColor RGB

- **Функции дат и времени**
  - Date DateAdd DateDiff DatePart DateSerial DateValue Day Hour IsDate Minute
  - Month MonthName Now Second Time Timer TimeSerial TimeValue Weekday
  - WeekdayName Year
- **Функции преобразования чисел в разные системы счисления**
  - Hex Oct VarType
- **Функции работы с объектами**
  - CreateObject GetAutoServerSettings GetObject IsObject
- **Финансовые функции**
  - DDB FV IPmt IRR MIRR NPer NPV Pmt PPmt PV Rate SLN SYD
- **Функции форматирования**
  - Format FormatCurrency FormatDateTime FormatNumber FormatPercent
- **Функции работы с указателями**
  - ObjPtr StrPtr VarPtr

Функция MsgBox

```
MsgBox(Prompt, [Buttons], [Title], [HelpFile], [Context])
```

Функция **MsgBox** выводит на экран диалоговое окно, содержащее сообщение, устанавливая режим ожидания нажатия кнопки пользователем

Возвращаемое значение

Возвращает значение типа *Integer*, указывающее, какая кнопка была нажата

Ниже перечислены значения, содержащие код нажатой кнопки:

Константа	Значение	Нажатая кнопка
vbOK	1	OK
vbCancel	2	Отмена(Cancel)

vbAbort	3	Прервать (Abort)
vbRetry	4	Повторить (Retry)
vbIgnore	5	Пропустить (Ignore)
vbYes	6	Да (Yes)
vbNo	7	Нет (No)

Параметры

Функция содержит именованные аргументы

Часть	Описание
Prompt	Обязательный. Строковое выражение, отображаемое как сообщение в диалоговом окне. Максимальная длина строки составляет приблизительно 1024 символа. Длинный текст разбивается автоматически, но можно задавать разбиение строки явно, используя символы возврата каретки и перевода строки(vbCrLf)
Buttons	Необязательный аргумент - целочисленная константа, которая является суммой VB-констант, определяющих ряд характеристик диалогового окна - число и тип кнопок, тип значка, основная кнопка, модальность окна сообщения(см.ниже). Значение по умолчанию равно 0
Title	Необязательный. Строковое выражение, отображаемое в строке заголовка диалогового окна. Максимальное число символов для заголовка около 50. Если этот параметр опущен, в строку заголовка помещается имя приложения
HelpFile	Необязательный. Строковое выражение, определяющее имя

	файла Справки, содержащего контекстно-зависимую Справку о данном диалоговом окне. Если этот параметр указан, то необходимо задать также и параметр *Context*
Context	Необязательный. Числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот параметр указан, то необходимо задать также и параметр *HelpFile*

Константы, используемые в аргументе Buttons для задания вида выводимых кнопок, пиктограмм

Константа	**Значение**	**Описание**
vbOKOnly	0	Отображается только кнопка "OK"
vbOKCancel	1	Отображаются кнопки "OK" и "Отмена" (Cancel)
vbAbortRetryIgnore	2	Отображаются кнопки "Прервать" (Abort), "Повторить" (Retry) и "Пропустить" (Ignore)
vbYesNoCancel	3	Отображаются кнопки "Да" (Yes), "Нет" (No) и "Отмена" (Cancel)
vbYesNo	4	Отображаются кнопки "Да" (Yes) и "Нет" (No)
vbRetryCancel	5	Отображаются кнопки "Повторить" (Retry) и "Отмена" (Cancel)
vbCritical	16	Используется значок "Критическое сообщение"
vbQuestion	32	Используется значок

		"Предупреждающий запрос"
vbExclamation	48	Используется значок "Предупреждение"
vbInformation	64	Используется значок "Информационное сообщение"
vbDefaultButton1	0	Основной является первая кнопка
vbDefaultButton2	256	Основной является вторая кнопка
vbDefaultButton3	512	Основной является третья кнопка
vbDefaultButton4	768	Основной является четвертая кнопка
vbApplicationModal	0	Модальное окно на уровне приложения: чтобы продолжить работу с текущим приложением, необходимо ответить на данное сообщение
vbSystemModal	4096	Модальное окно на уровне системы: все приложения будут недоступны до тех пор, пока пользователь не ответит на данное сообщение
vbMsgBoxHelpButton	16384	Добавляется кнопка Справка(Help)
VbMsgBoxSetForeground	65536	Аналогично параметру vbApplicationModal
vbMsgBoxRight	524288	Текст выравнивается по правому краю

vbMsgBoxRtlReading	1048576	Задает порядок вывода текста справа налево для арабской системы и иврит

Примечание

Если используется кнопка Отмена(Cancel), то можно вместо нее нажимать на клавишу Esc

В отличие от InputBox окно **MsgBox** не позиционируется на экране, а всегда располагается в центре экрана

Функция InputBox

```
InputBox(Prompt[,Title] [,Default] [,XPos] [,YPos] [,HelpFile,Context])
```

Функция выводит на экран диалоговое(модальное) окно с кнопкой закрытия, содержащее заданное сообщение, поле ввода, кнопки OK, Cancel и опционально заголовок и/или кнопку Help, ожидая от пользователя ввода текста или щелчка кнопки. При задании не только одного первого параметра, необходимо использовать функцию **InputBox** в выражении. Для пропуска некоторых параметров нужно включить соответствующие разделители в виде запятых

Возвращаемое значение

Возвращает значение типа *String*, включающее содержимое окна текста

Параметры

Часть	Описание
Prompt	Обязательный. Строковое выражение, отображаемое как

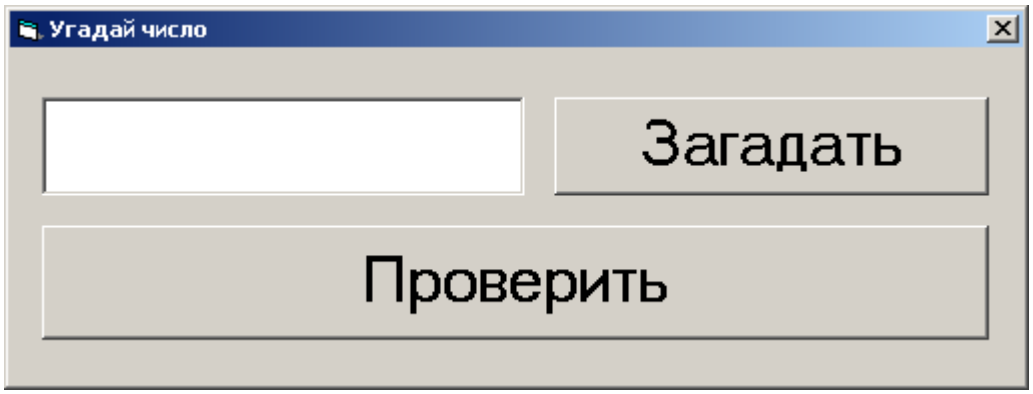
	сообщение в диалоговом окне. Максимальная длина параметра *Prompt* составляет приблизительно 1024 символа и зависит от ширины используемых символов. Строковое значение *Prompt* может содержать нескольких физических строк. Для разделения строк допускается использование символа возврата каретки, символа перевода строки или комбинации этих символов
Title	Необязательный. Строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот параметр опущен, в строку заголовка помещается имя приложения. Максимальное число символов заголовка около 50
Default	Необязательный. Строковое выражение, отображаемое в окне текста как ответ, используемый по умолчанию, если пользователь не введет другую строку. Если этот параметр опущен, окно текста отображается пустым
XPos	Необязательный. Числовое выражение, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана (в твипах). Если этот параметр опущен, то диалоговое окно выравнивается по центру экрана по горизонтали
YPos	Необязательный. Числовое выражение, задающее расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана (в твипах). Если этот параметр опущен, то диалоговое окно помещается по вертикали на расстоянии примерно на одну треть высоты экрана
HelpFile	Необязательный. Строковое выражение, определяющее имя файла Справки, содержащего контекстно-зависимую Справку о данном диалоговом окне. Если этот параметр указан, то

	необходимо задать также и параметр *Context*
Context	Необязательный. Числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот параметр указан, то необходимо задать также и параметр *HelpFile*

Примечание

Если указаны оба параметра, *HelpFile* и *Context*, то пользователь имеет возможность нажатием клавиши F1 вызвать соответствующую контекстную справку. Некоторые главные приложения, например, Microsoft Excel, также автоматически добавляют в диалоговое окно кнопку «Справка». Если пользователь щелкает кнопку **OK** или нажимает ENTER, функция **InputBox** возвращает содержимое поля ввода. Если пользователь щелкает кнопку **Cancel**, то функция возвращает пустую строку (« »)

Проект 1. Угадай число v.1




```
Dim Digit As Integer
Dim Attempt As Integer

Private Sub Command1_Click()
```

```
MsgBox "Коварный компьютер загадал число", vbOKOnly + vbExclamation
Randomize
Digit = Int(Rnd * 100)
Attempt = 0
Text1.SetFocus
End Sub

Private Sub Command2_Click()
Attempt = Attempt + 1
If Text1.Text > Digit Then MsgBox "Загаданное число меньше"
If Text1.Text < Digit Then MsgBox "Загаданное число больше"
If Text1.Text = Digit Then MsgBox "Вы угадали за " & Attempt & " попыток",
vbOKOnly + vbExclamation
Text1.SetFocus
End Sub
```

Проект 2. Калькулятор v.2

Кнопки 0,1,2,3,4,5,6,7,8,9, +,-,*,/,=, C

Для кнопки 2

```
Text1.Text = Text1.Text + “2”
```

Для кнопки +

```
x1 = Text1.Text
Oper = 1
Text1.Text = “”
```

Предварительно в области глобальных переменных

```
Dim x1 As Integer, x2 As Integer, Oper As Integer
```

x1 – первое число, x2 – второе число, Oper – номер операции

Oper = 1 – операция +

Oper = 2 – операция – и.т.д

Для кнопки =

```
x2 = Text1.Text
```

If Oper = 1 Then Text1.Text = x1 + x2

…

Для кнопки C

Text1.Text = “”

Приложение. Описание функций.

Abs (функция)	возвращает абсолютное значение числа
And (операция)	логическое И
AppActivate (оператор)	активизирует окно приложения
Array (функция)	создает массив из параметров и возвращает его какпеременную типа Variant
Asc (функция)	возвращает числовой код первого символа строки аргумента
Atn (функция)	возвращает арктангенс числа в радианах
Beep (оператор)	проигрывает звуковой сигнал через динамик компьютера
Call (оператор)	передает управление процедуре модуля (Sub), функциимодуля (Function) или подпрограмме DLL
CBool (функция)	приводит выражение к типу Boolean
CByte (функция)	преобразует выражение к типу Byte
CCur (функция)	преобразование выражения к типу Currency
CDate	преобразование выражения к типу Date

(функция)	
CDbl (функция)	преобразование к типу Double
ChDir (оператор)	изменяет текущий каталог на устройстве
ChDrive (оператор)	изменяет текущее устройство
Choose (функция)	возвращает значение из списка аргументов с определеннымпорядковым номером
Chr (функция)	возвращает символ, связанный с определенным числовымкодом
CInt (функция)	преобразование выражения к типу Integer
CLng (функция)	преобразование выражения к типу Long
Close (оператор)	закрывает файл, открытый оператором Open
Command (функция)	возвращает командную строку, используемую длязапуска Visual Basic или приложения на Visual Basic
Const	оператор объявления констант
Cos (функция)	возвращает косинус числа
Create Object (функция)	создать OLE Automation объект
CSng (функция)	преобразование выражения к типу Single
CStr (функция)	преобразование выражения к типу String
CurDir (функция)	возвращает текущий каталог логического устройства

CVar (функция)	преобразование выражения к типу Variant
CVErr (функция)	возвращает подтип ошибки, для определенного пользователемномера ошибки
Date (оператор)	устанавливает значение системной даты
Date (функция)	возвращает значение системной даты
DateAdd (функция)	возвращает переменную типа Variant, содержащуюдату, отличающуюся от заданной на определенный интервал времени
DateDiff (функция)	возвращает число временных интервалов между двумядатами
DatePart (функция)	возвращает определенную часть заданной даты
DateSerial (функция)	возвращает дату для заданного года, месяцаи дня
DateValue (функция)	возвращает дату
Day (функция)	возвращает число от 1 до 31, соответствующее текущемудню месяца
DDB (функция)	возвращает значение амортизационных потерь за определенныйпериод
Declare (оператор)	на уровне модуля объявляет ссылки ко внешним подпрограммамв DLL
Deftype (операторы)	устанавливает тип данных по умолчанию на уровнемодуля для переменных, параметров подпрограмм, а также возвращаемыхзначений для функций и операторов Property Get, начинающихся

	с определенныхсимволов
Dim (оператор)	объявляет переменные и выделяет память под них
Dir (функция)	возвращает имя файла или каталог, подходящий дляданного шаблона или атрибута файла, или метку тома устройства
DoEvents (функция)	прерывает выполнение приложения
Do... Loop (оператор)	повторяет блок команд до тех пор, пока условиеверно или до тех пор, пока условие не станет верным
End (оператор)	заканчивает подпрограмму или блок команд
Environ (функция)	возвращает строку, связанную с переменной окруженияоперационной системы
EOF (функция)	возвращает значение, указывающее, достигнут ликонец файла
Eqv (оператор)	проверяет логическое равенство двух выражений
Erase (оператор)	повторно инициализирует элементы массивов фиксированногоразмера и перераспределяет память под динамические массивы
Error (оператор)	эмулирует возникновение ошибки
Error (функция)	возвращает текст сообщения данного номера ошибки
Exit (операторы)	осуществляет выход из циклов Do ... Loop, For... Next, функции и процедур

Exp (функция)	возвращает экспоненту числа
FileAttr (функция)	возвращает режим открытия или номер (handle) файла
FileCopy(операмор)	копирует файл
FileDateTime (функция)	возвращает дату и время создания или последнеймодификации файла
FileLen (функция)	возвращает длину файла в байтах
Fix (функция)	возвращает целую часть числа
For Each...Next (оператор)	повторяет одну и ту же последовательностькоманд для каждого элемента массива или коллекции
For...Next (оператор)	повторяет последовательность команд определенноечисло раз
Format (функция)	форматирует выражение в соответствии с заданнымформатом
FreeFile (функция)	возвращает следующий не занятый номер файла дляиспользования в операторе Open
Function (оператор)	объявляет имя, аргументы и код подпрограммы,возвращающей значение (функции)
FV (функция)	возвращает значение ренты, основываясь на периодическихвзносах и постоянной норме капиталовложений
Get (оператор)	читает данные из открытого файла в переменную

GetAttr (функция)	возвращает атрибуты файла, каталога или меткитома
GetObject (функция)	возвращает OLE Automation объект для файла сданным расширением
GoSub... Return (оператор)	выполняет подпрограмму
GoTo (оператор)	передает управление определенной строке подпрограммыбез возврата контроля
Hex (функция)	возвращает строку, представляющую шестнадцатеричноезначение числа
Hour (функция)	возвращает целое число в диапазоне 0 - 23 включительно,представляющее определенный час дня
If...Then... Else (оператор)	выполнение групп команд в зависимостиот значения выражения
Iff (функция)	возвращает одно из двух значении в зависимости от значениявыражения
Imp (операция)	импликация двух выражений
Input (функция)	возвращает символы из файла, открытого для последовательногодоступа или как двоичный файл
Input # (оператор)	считывает данные из открытого файла в переменные
InputBox (функция)	показывает диалоговое окно ввода, ожидает вводатекста и возвращает содержимое введенного текста, после закрытия окна
InStr (функция)	возвращает позицию первой найденной

	подстроки в строке
Int (функция)	возвращает целую часть числа
Is (операция)	сравнение двух ссылок на объекты
IsArray (функция)	возвращает булево значение, указывающее, являетсяли данная переменная массивом
IsDate (функция)	возвращает булево значение, указывающее, можетли выражение быть преобразовано к типу Date
IsEmpty (функция)	возвращает булево значение, указывающее, инициализированоли значение данной переменной
IsError (функция)	возвращает булево значение, указывающее, являетсяли выражение значением кода ошибки
IsMissing (функция)	возвращает булево значение, указывающее,был ли передан данный необязательный параметр в подпрограмму
IsNull (функция)	возвращает булево значение, указывающее, несодержит ли выражение недопустимое (Null) значение
IsNumeric (функция)	возвращает булево значение, указывающее,может ли данное выражение рассматриваться как число
IsObject (функция)	возвращает булево значение, указывающее, являетсяли выражение объектом OLE Automation
Kill (оператор)	удаляет файл
LBound (функция)	возвращает значение нижней границы индекса массива

LCase (функция)	возвращает строку в нижнем регистре
Left (функция)	возвращает определенное число символов с начала строки
Len (функция)	возвращает число символов строки или число байт,необходимых для хранения переменной
Let (оператор)	присваивает значение выражения переменной или свойству
Like (операция)	сравнение двух строк
Line Input # (оператор)	считывает строку из файла в переменную
Load (оператор)	загружает в память форму или элемент управления
LoadPicture (функция)	загружает графический образ в объекты: Form,
Picture Box u Image	
Loc (функция)	возвращает текущую позицию чтения/записи в открытомфайле
Lock (оператор)	контролирует доступ других процессов ко всему иличасти
открытого файла	
LOF (функция)	возвращает размер в байтах открытого файла
Log (функция)	возвращает натуральный логарифм числа

LSet (оператор)	копирует строку в строковую переменную, а такжекопирует значение переменной одного специализированного типа в переменнуюдругого специализированного типа
LTrim (функция)	возвращает копию строки без лидирующих пробелов
Mid (оператор)	замещает определенное число символов в строке на символыиз другой строки
Mid (функция)	возвращает определенное число символов с определеннойпозиции строки
Minute (функция)	возвращает целое число в диапазоне 0 - 59, представляющееминуту часа
MkDir (оператор)	создает новый каталог
Mod (операция)	возвращает остаток от деления двух чисел
Month (функция)	возвращает целое число в диапазоне 1 - 12, представляющееномер месяца
MsgBox (функция)	показывает сообщение в диалоговом окне, ожидаетвыбор одной из кнопок пользователем и возвращает значение, указывающее,какая кнопка была выбрана
Name (оператор)	переименовывает файл или каталог
Not (операция)	логическое отрицание
Now (функция)	возвращает текущие значения даты и времени
Oct (функция)	возвращает строку, представляющую

	восьмеричное представлениечисла
On Error (оператор)	устанавливает обработчик ошибок и задает местоположениеподпрограммы обработки; используется также для отмены обработки ошибокподпрограммой обработчика
On..GoSub, On...GoTo (операторы)	передача управления на одну изнескольких определенных строк (меток), в зависимости от значения выражения
Open (оператор)	скрывает файл для ввода/вывода
Option Base (оператор)	используется для объявления значения нижнейграницы размерности индексов массивов по умолчанию
Option Compare (оператор)	используется на уровне модуля для объявленияметода сравнения по умолчанию при сравнении строк
Option Explicit (оператор)	используется на уровне модуля для установкипроверки наличия объявлений для всех переменных в данном модуле
Option Private (оператор)	используется на уровне модуля для указания,что весь модуль является Private
Or (операция)	логическое ИЛИ
Partition (функция)	возвращает строку, указывающую, сколько развстретились числа из заданного диапазона
Print # (оператор)	записывает форматированные данные в файл

Private (оператор)	используется на уровне модуля для объявления Privateпеременных и выделяет место в памяти для их хранения
Property Get (оператор)	объявляет имя, аргументы и код подпрограммыполучения значения свойства
Property Let (оператор)	объявляет имя, аргументы и код процедурыустановки значения свойства
Property Set (оператор)	объявляет имя, аргументы и код процедурыустановки ссылки на объект
Public (оператор)	используется на уровне модуля для объявления Publicпеременных и выделяет место в памяти для их хранения
Put (оператор)	записывает переменную в файл
QSColor (функция)	возвращает RGB код, соответствующий номеру цвета
Randomize (оператор)	инициализирует генератор случайных чисел
RGB (функция)	возвращает целое число, представляющее значение RGBкода
ReDim (оператор)	используется на уровне подпрограммы для переопределенияразмера динамических массивов и выделения под них места в памяти
Rem (оператор)	вставка комментариев в программу
Reset (оператор)	закрывает все открытые программой файлы
Resume	продолжает выполнение программы после

(оператор)	завершенияпроцедуры обработчика ошибок
Right (функция)	возвращает определенное число символов с правой стороныстроки
RmDir (оператор)	удаляет каталог
Rnd (функция)	возвращает случайное число
RSet (оператор)	копирует правую часть строки в строковую переменную
RTrim (функция)	возвращает копию строки без конечных пробелов
SavePicture (оператор)	сохраняет в файл графический образ объектаForm, элементов управления Picture Box или Image
Second (функция)	возвращает целое значение в диапазоне 0 - 59,представляющее секунду в минуте
Seek (оператор)	устанавливает позицию для следующей операции чтенияили записи в открытый файл
Seek (функция)	возвращает текущую позицию чтения/записи открытогофайла
Select Case (оператор)	выполняет одну или несколько команд, в зависимостиот значения выражения
SendKeys (оператор)	посылает одно или несколько нажатий клавиш активномуокну, как если бы они были введены пользователем с клавиатуры
Set (оператор)	связывает ссылку на объект с переменной или свойством

SetAttr (оператор)	устанавливает атрибуты файла
Sgn (функция)	возвращает знак числа
Shell (функция)	запускает внешнюю программу на выполнение
Sin (функция)	возвращает значение синуса угла
Space (функция)	возвращает строку, содержащую определенное числопробелов
Spc (функция)	позиционирование в строке вывода
Sqr (функция)	подсчет значения квадратного корня числа
Static (оператор)	используется на уровне модуля для объявления переменныхи выделяет место в памяти для их хранения. Переменные сохраняют значениядо завершения программы
Stop (оператор)	приостанавливает выполнение программы
Str (функция)	возвращает строковое представление числа
StrComp (функция)	возвращает результат сравнения строк
StrConv (функция)	возвращает преобразованную строку
String (функция)	возвращает строку заданной длины из одинаковых символов
Sub (оператор)	объявляет имя, параметры и тело процедуры
Switch (функция)	подсчитывает значения списка выражении и возвращаетзначение или выражение, связанное с выражением из списка, значениекоторого равно

	True
Tab (функция)	позиционирование в строке вывода
Tan (функция)	возвращает значение тангенса угла
Time (оператор)	устанавливает значение системных часов
Time (функция)	возвращает значение типа Date, указывающее текущеесистемное время
Timer (функция)	возвращает число секунд, прошедших после полуночи
TimeSerial (функция)	возвращает значение типа Date, содержащеевремя для заданного часа, минуты и секунды
Time Value (функция)	возвращает значение типа Date, содержащеевремя суток
Trim (функция)	возвращает копию строки без начальных и конечных пробелов
Type (оператор)	объявляет на уровне модуля специализированный типданных
TypeName (функция)	возвращает строку информации о заданной переменной
UBound (функция)	возвращает значение наибольшего индекса для даннойразмерности массива
UCase (функция)	возвращает строку, преобразованную в верхний регистр
Unload (оператор)	выгружает форму или элемент управления из памяти

Unlock (оператор)	контролирует доступ других процессов ко всему иличасти открытого файла
Val (функция)	возвращает числовое представление строки
VarType (функция)	возвращает значение, указывающее тип переменной
Weekday (функция)	возвращает целое число, представляющее день недели
While...Wend (оператор)	выполняет в цикле последовательность команддо тех пор, пока верно условие
Width # (оператор)	назначает ширину строки вывода для операции записив открытый файл
With (оператор)	выполняет последовательность команд для конкретногообъекта или переменной специализированного типа
Write # (оператор)	записывает данные в файл
Xor (операция)	исключающее ИЛИ
Year (функция)	возвращает целое число, представляющее год

Операторы управления

Оператор условия If...Then

Наиболее часто используется оператор If.. .Then, который может иметь простую однострочную или блочную структуру.

Однострочный синтаксис

If Условие Then *Оператор* [Else *Оператор]*

Функционирование оператора такой структуры относительно просто. Если условие после if истинно, т.е. результат равен True, выполняется оператор, указанный за Then. Если же результат равен False, то выполняется оператор, следующий за ключевым словом Else, если такое имеется:

```
If A = 7 Then Beep
If X < 9 Then Print "False!" Else Print "True!"
```

В первом примере выдается звуковой сигнал, если переменная A равна 7. Во втором примере выводится текст False!, если значение переменной X меньше 9;

в противном случае выводится текст True!.

Блочный или многострочный синтаксис

```
If Условие Then
Операторы 1
[Elself Условием Then]
    [Операторы 2]
[Else]
    [Операторы 3]
    [End If]
End If
```

В принципе блочная запись предоставляет такие же возможности, как и однострочная. Но если в зависимости от условия необходимо выполнить не простую команду, а группу операторов, следует использовать блочный синтаксис.

Это относится и к ветви Else. Кроме того, блочная структура с Elself позволяет анализировать несколько условий:

```
If A > 5 Then
Print "Ждите"
End If
If Name = "Иванов" Then . .
Print "Ваша карточка удерживается!"
Else
Print "Деньги, пожалуйста!"
End If
If Обращение = 1 Then
Print "Глубокоуважаемый господин"
Elself Обращение = 2 Then
Print "Глубокоуважаемая госпожа"
Elself Обращение = 3 Then
Print "Глубокоуважаемые дамы и господа"
Else
Print "Здравствуйте, люди"
End If
```

При формировании более сложных условий блочная запись удобнее. Использование в этом случае блочного синтаксиса улучшает читабельность программы.

В качестве условия может быть использовано любое логическое выражение со знаками >, <, >=, <=, =, <> (неравно) и логическими операциями And (и), Or (или), Not (не).

```
a>b And a>c
```

Оператор выбора Select Case

Еще одним оператором ветвления Visual Basic является Select Case, который позволяет выполнить одну из нескольких групп операторов в зависимости от значения условия.

Инструкция Select Case имеет следующий синтаксис:

Select Case *Проверочное_выражент*

[Case *Значение1*]

[Операторы 1]

[Case *Значение 2*]

[*Операторы2*]

[Case Else]

[*Операторы 3*]

End Select

```
Private Sub Command_Click()
n = Int (Rnd * 10) +1 'случайное число от 1 до 10
Select Case n
Case 1
    Print "Равно 1"
Case 2, 3
    Print "Равно 2 или 3"
Case 4 To 6
    Print "Больше или равно 4 и меньше или равно 6"
Case Is >= 9
    Print "Больше или равно 9"
Case Else
    Print "Ни одно из предшествующих"
End Select
End Sub
```

В качестве значения для блока Case можно указывать не только одно значение (1), но и несколько, разделенных запятой (2, 3). Можно определять

также области сравнения (4 то 6) или воспользоваться относительным сравнением (Is >= 9) Вместо непосредственного проверочного выражения можно использовать ключевое слово Is Блок Case Else выполняется, если ни одно из предыдущих условий не является истинным. Если условию Select Case соответствует несколько блоков Case, то выполняется первый из них

```
Private Sub Commandl_Click()
Select Case n
Case 0
    Print "Равно О"
Case 0 To 10
    Print "Между 0 и 10, кроме О"
End Select
End Sub
```

Во втором блоке Case обрабатываются значения от 0 до 10, однако значение 0 перехватывается первым блоком Case. Поэтому операторы второго блока Case будут выполняться, если значение условия больше или равно 1 и меньше или равно 10.

Циклы

Для многократного выполнения одного или нескольких операторов предназначены циклы Visual Basic предлагает две конструкции: цикл For. . .Next

дает возможность устанавливать число проходов цикла, а цикл Do... Loop завершается при выполнении заданного условия.

Оператор цикла For…Next

Цикл For. . .Next является самой старой и самой простой конструкцией:

For *Счетчик = Начальное_значение* To *Конечное_значение* [Step Шаг]

Операторы

Next [Счетчик]

В начале выполнения цикла значение Счетчик устанавливается в Начальное значение. При каждом проходе переменная Счетчик увеличивается на 1 или на величину шаг. Если она достигает или становится больше (меньше, при

отрицательном шаге) Конечное значение, то цикл завершается и выполняются следующие операторы. Разность между начальным и конечным значением, деленная на величину шага, составляет число проходов:

```
For I = 1 To 10
Print I * 100
Next I
For L = 100 To 5 Step – 0.5
X = Y * L
Next
For I1 = 1 To 5
For 12 = 10 To 20
Print I1 + I2
Next I2
Next I1 'Или ..(Next I2,I1 вместо 2-х Next)
```

В этом примере представлены разные конструкции циклов For.. .Next. Часто для вычислений внутри цикла используются числовые переменные. Для безусловного выхода из цикла используется оператор Exit For.

Оператор цикла Do... Loop

Если количество проходов должно зависеть от условия, используют цикл Do... Loop. В зависимости от позиции условия различают два варианта цикла Do...Loop.

Цикл, управляемый в начале

```
Do [(While | Until) Условие]
[Операторы]
[Exit Do]
[Операторы]
Loop
```

Цикл, управляемый в конце

```
Do
[Операторы]
```

[Bxit Do) *[Операторы]*

Loop [(While | Until} Условие]

Если условие проверяется в начале цикла, то он никогда не выполняется в случае невыполнения условия. Если же проверка происходит в конце, цикл выполняется как минимум один раз, независимо от того, выполнено условие или нет.

Тело цикла выполняется неопределенное число раз, пока условие не вызовет выход из цикла:

```
Do Until EOF(Файл)
Input #1, SomeData
Loop
Do
X = X + 1
Print "Hello"
Loop While X < 9
```

Рассмотренные варианты циклов Do... Loop предоставляют разработчику большие возможности организации повторяющихся вычислений.

Оператор цикла While…Wend

В Visual Basic цикл while. .. Wend играет второстепенную роль. Он используется только для совместимости с другими диалектами Basic, а также для совместимости с более ранними версиями Visual Basic, в которых не было оператора Do. . .Loop.

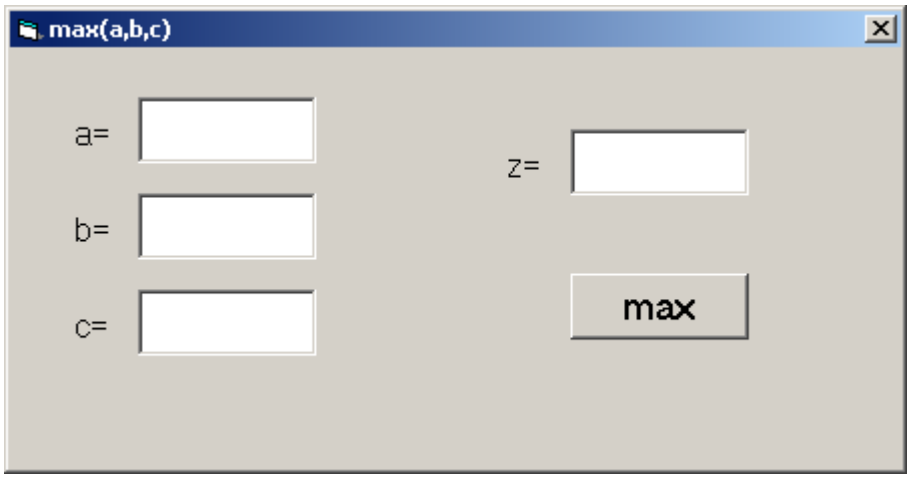
```
While Условие
[Операторы]
Wend
```

Принцип его действия такой же, как и цикла Do While.. . Loop. Поэтому вместо него проще использовать Do.. .Loop. Кроме того, для цикла while. . .Wend нет оператора досрочного выхода типа Exit:

```
While X = True
Print Time
```

Wend

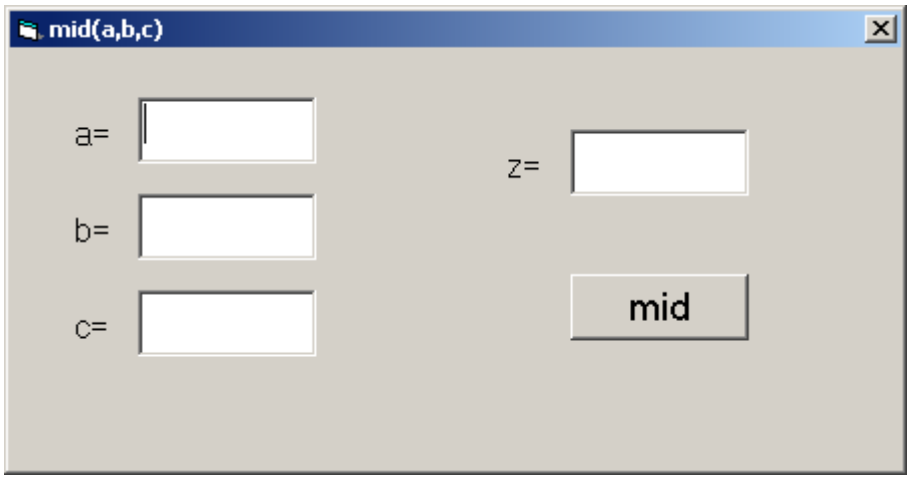
Проект 1. Максимум.




```
Private Sub Command1_Click()
Dim a As Integer, b As Integer, c As Integer, z As Integer
a = Text1.Text: b = Text2.Text: c = Text3.Text
If a > b And a > c Then
   z = a
Else
   If b > c Then
      z = b
   Else
      z = c
   End If
End If
Text4.Text = z
End Sub
```

Проект 2. «Среднее».

Необходимо определить «среднее» трех чисел т.е. найти то число которое расположено на числовой оси между двумя оставшимися. Например: 3, 7, 4. Среднее 4.



Проект 3. Треугольник.

Интерфейс аналогичный. a, b, c – длины сторон треугольника. Определить какой это треугольник.

1) можно ли построить на этих отрезках треугольник
2) треугольник – равносторонний
3) треугольник – равнобедренный
4) треугольник – прямоугольный
5) треугольник – простой

Проект 4. 2 в степени n.

Код программы может быть таким.

```
Private Sub Command1_Click()
Dim n As Integer, i As Integer, p As Long
n = Text1.Text
p = 1
For i = 1 To n
   p = p * 2
Next i
Label2.Caption = p
End Sub
```

Проект 5. n! (факториал).

Вычислить n!. Аналогично 2 в степени.

Проект 6. e (основание натурального логарифма).

Вычислить e с точностью 0,0001 (цикл Do)

События мышки и клавиатуры.

События мышки:

MouseDown – происходит при нажатии любой кнопки

MouseUp – происходит при отпускании нажатой кнопки мышки

MouseMove – происходит при перемещении указателя в новую точку

Эти события распознаются большинством элементов управления.

Все события используют следующие параметры

Button, Shift, X, Y

Button – битовое поле, младшие биты дают состояние кнопок мыши

Shift – младшие биты дают состояние клавиш Shift, Ctrl, Alt

X, Y – положение указателя мыши системе координат объекта (лучше использовать пиксельную систему ScaleMode – 3-Pixel)

MouseDown

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

Command1.Move X,Y

End Sub

При нажатии Command1 перемещается в положение указанное курсором мыши.

Лучше: Command1.Move X – Command1.Width/2, Y – Command1.Height/2

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

Form1.Line – (X, Y)

End Sub

Происходит рисование линии при нажатии кнопки мыши

MouseMove

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
Form1.Line – (X, Y)
End Sub
```

Происходит рисование линии при перемещении указателя мыши

Проект 1. Paint v.1.

```
Dim DrawNow As Boolean

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single,
Y As Single)
DrawNow = True
Form1.CurrentX = X
Form1.CurrentY = Y
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
If DrawNow Then
   Form1.Line -(X, Y)
End If
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y
As Single)
DrawNow = False
End Sub
```

Состояние кнопок мыши

За состояние кнопок мыши отвечает параметр Button

2-е представление	10-е представление	Const VB	Описание
000	0		Не нажата ни одна кнопка
001	1	vbLeftButton	Нажата левая кнопка
010	2	vbRightButton	Нажата правая кнопка
011	3	vbLeftButton + vbRightButton	Нажаты левая и правая кнопки
…	…	…	…
111	7	…	…

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = vbLeftButton Then
        Print “нажата левая кнопка”
    End If
    If Button = vbRightButton Then
        Print “нажата правая кнопка”
    End If
    End Sub
```

Состояние клавиш Shift, Ctrl, Alt

За состояние кнопок мыши отвечает параметр Shift

2-е представление	10-е представление	Const VB	Описание
001	1	vbShiftMask	Нажата Shift

010	2	vbCtrlMask	Нажата Crtl
100	4	vbAltMask	Нажата Alt

События клавиатуры:

KeyDown – возникает при нажатии клавиши

KeyUp – возникает при отпускании нажатой клавиши

KeyPress – когда нажата клавиша соответствующая ASCII

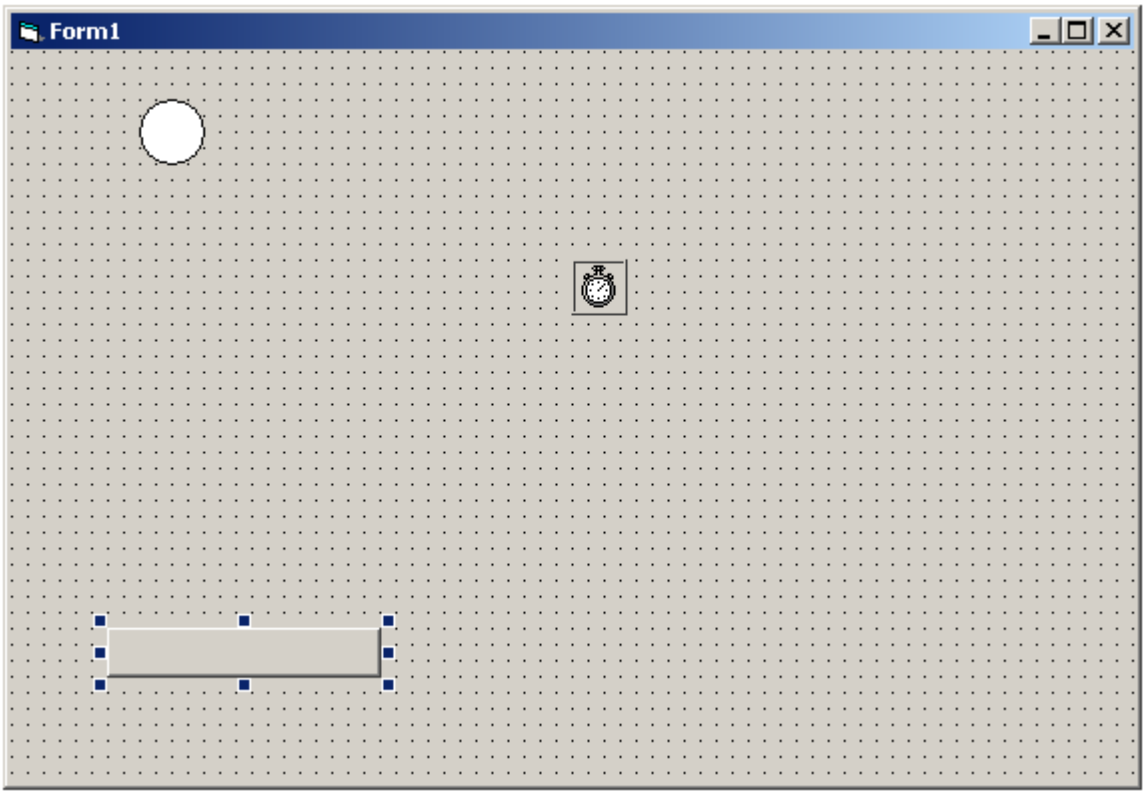
В качестве параметров к этим событиям используются KeyCode, Shift

Для работы всех событий клавиатуры необходимо свойство KeyPreView

KeyPreView – True

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = vbKeyA Then Print “нажата клавиша A”
If KeyCode = vbKeyHome Then Print “нажата клавиша Home”
If KeyCode >= vbKey0 And KeyCode <= vbKey9 Then Print “нажата цифровая клавиша (0 – 9)”
…
End Sub
```

Проект 2. PinBall.



Form1

ScaleMode – 3-Pixel

KeyPreView – True

Shape1

BackStyle – 1-Opaque

Shape – 3-Cirle

Coomand1

Caption – ~~Command1~~

Enabled – False

Timer1

Interval – 200

```
Dim BallX, BallY, StepX, StepY, RocketX As Integer

Private Sub Form_Load()
BallX = Shape1.Left
BallY = Shape1.Top
```

```
StepX = 10
StepY = 10
RocketX = Command1.Left
End Sub

Private Sub Timer1_Timer()
If BallX + StepX < 0 Or BallX + StepX > Form1.ScaleWidth Then StepX = -StepX
If BallY + StepY < 0 Or BallY + StepY > Form1.ScaleHeight Then StepY = -StepY
BallX = BallX + StepX
BallY = BallY + StepY
Shape1.Move BallX, BallY
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = vbKeyLeft And RocketX >= 10 Then
   RocketX = RocketX - 10
   Command1.Left = RocketX
End If
If KeyCode = vbKeyRight And RocketX <= Form1.Width - 10 Then
   RocketX = RocketX + 10
   Command1.Left = RocketX
End If
End Sub
```

1) Доделать проект 1. Левой кнопкой рисовать, правой стирать (стирать лучше прямоугольником Line () – (), Color, BF)

2) Доделать проект2. учесть размеры шарика и ракетки, сделать отражение от ракетки.

Графические методы и элементы управления.

Графические методы

1) Print – используется для отображения текста, значения переменных на формах и графических окнах (PictureBox), причем элементы разделяются ; и ,.

; - VB печатает элементы один за другим без пробелов

, - при печати VB переходит на следующую позицию табуляции

a = 5

Print “Значение переменной a = ”; a

a = 2

b = 5

Print “a = ”; a, “b = ”; b

Метод Print позволяет задавать координаты вывода с помощью свойств CurrentX, CurrentY

2) Cls – удаляет весь текст созданный методом Print и устанавливает CurrentX = 0, CurrentY = 0

3) Pset – устанавливает цвет точки

Pset (x, y), color

4) Point – возвращает цвет точки

Point (x, y)

5) Line – рисует линию или прямоугольник

Line (x1, y1) – (x2, y2), color - линия

Line (x1, y1) – (x2, y2), color, B – прямоугольник

Line (x1, y1) – (x2, y2), color, BF – закрашенный прямоугольник

6) Circle – рисует окружность или эллипс

Cirlce (x, y), radius, color - окружность

Cirlce (x, y), radius, color, start, end, aspect – эллипс или дуга

7) PaintPicture – рисует графику в произвольном месте

Графические свойства

1) DrawWidth – толщина линий

2) DrawStyle – стиль линий

Проект 1. Звездное небо.

Form1

Caption – Звездное небо

ScaleMode – 3-Pixel

KeyPreView – True

Command1

Caption – Start Demo

```
Private Sub Command1_Click()
Command1.Visible = False
Randomize
DrawWidth = 5
While True
   x = Int(Rnd * Form1.ScaleWidth)
   y = Int(Rnd * Form1.ScaleHeight)
   c = QBColor(Int(Rnd * 16))
   c = RGB(Rnd * 256, Rnd * 256, Rnd * 256)
   Form1.PSet (x, y), c
   DoEvents
Wend
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
End
End Sub
```

DoEvents – Используется для передачи управления Windows для обработки событий из очереди

Графические элементы управления

 - Image (образ)

 - Line (линия)

 - Shape (контур)

 - PictureBox (графическое окно)

PictureBox может являться контейнером для других элементов управления т.е. содержать другие элементы

Добавление графики в элемент Image во время выполнения программы

1) Загрузка из файла

Image1.Picture = LoadPicture(“filename”)

2) Загрузка из файла ресурсов

Image1.Picture = LoadResPicture(Id, ResType)

3) Загрузка из другого Image

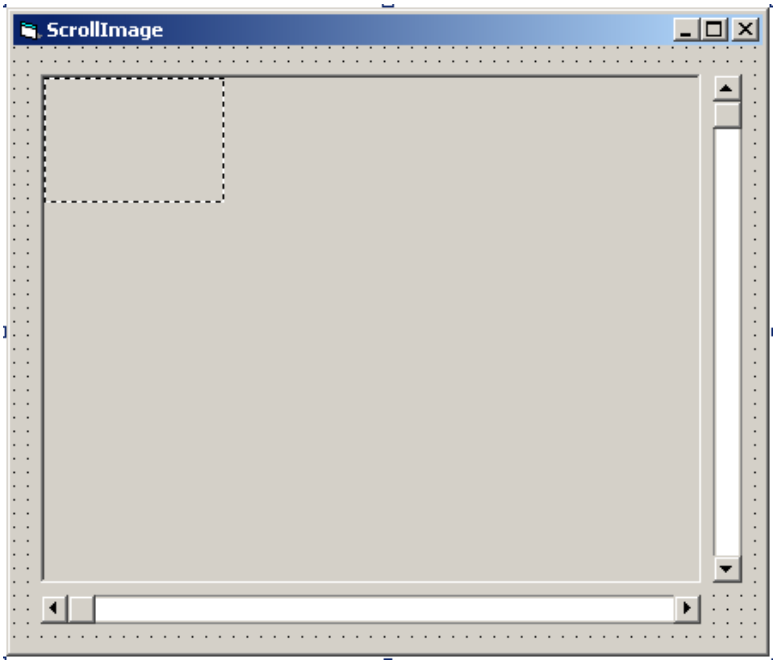
Image1.Picture = Image2.Picture

Наиболее быстрый способ загрузка из другого Image

Удаление графики из элемента Image

Image1.Picture = LoadPicture(“”)

Проект 2. Scroll Image.



Сначала добавить PictureBox, далее выделив PictureBox Picture1, добавить на форму Image Image1 так как показано на рисунке (Left = 0, Top = 0, затем добавить VScroll1, HScroll1

В Paint нарисовать большой рисунок 1000x1000 и сохранить.

VScroll1

SmallChange – 20

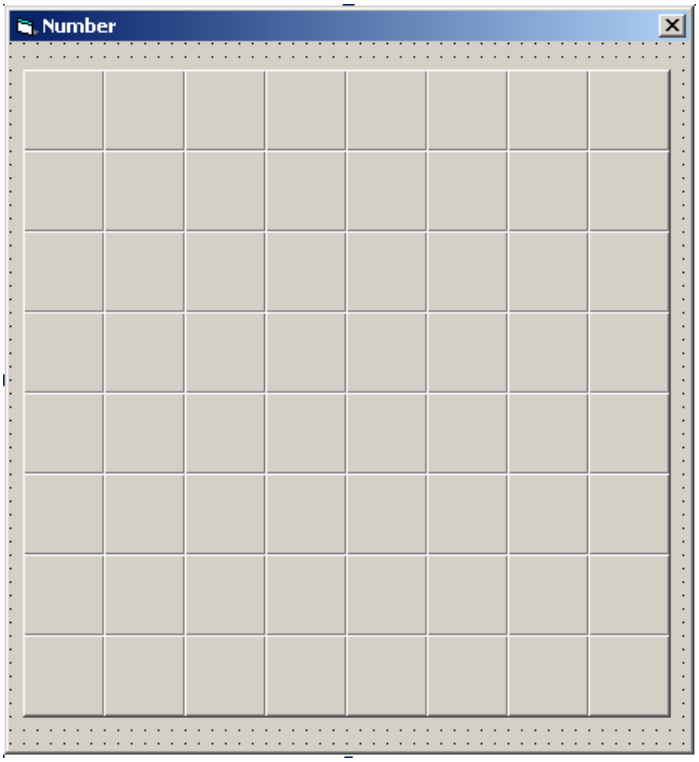
LargeChange – 100

```
Private Sub Form_Load()
Image1.Picture = LoadPicture (…)
VScroll1.Max = Image1.Height – Vscroll1.Height
End Sub

Private Sub VScroll1_Change()
Image1.Top = – VScroll1.Value
End Sub
```

Аналогично с элементом HScoll1

Проект 2. Угадай число v.2.



Вначале добавить PictureBox, далее на PictureBox скопировать Command (при копировании на запрос ответить да)

Picture1

AutoReDraw - True

Для добавления процедуры NewNumber (см.ниже) Tolls – Add Procedure, набрать NewNumber, Ok

```
Dim Number As String
Dim Att As Integer

Private Sub Command1_Click(Index As Integer)
Dim NumberTmp As String
Command1(Index).Visible = False
Att = Att + 1
NumberTmp = InputBox("Введите число:", , , 5000, 20)
If NumberTmp = Number Then
   For i = 0 To 63: Command1(i).Visible = False: Next i
```

```
        MsgBox vbTab + "Победа!!!" + vbCrLf + "Вы угадали! Кол-во попыток:" +
Str$(Att)
        NewNumber
    End If
    End Sub

    Private Sub Form_Load()
    Form1.Visible = True
    NewNumber
    End Sub

    Public Sub NewNumber()
    Att = 0
    For i = 0 To 63
        Command1(i).Visible = True
    Next i
    Randomize
    Number = Str$(Int(Rnd * 90) + 10)
    Number = Mid$(Number, 2)
    Picture1.Cls
    Picture1.FontSize = Int(Rnd * 90) + 160
    Picture1.CurrentX = Int(Rnd * (230 - Picture1.FontSize))
    Picture1.CurrentY = Int(Rnd * (400 - Picture1.FontSize)) - 60
    Picture1.Print Number
    MsgBox "Компьтер загадал число – угадай его !!!"
    End Sub
```

Меню и панель инструментов.

Меню

Меню это удобный способ объединения команд в группы и наиболее легкий способ представления пользователю доступа к ним.

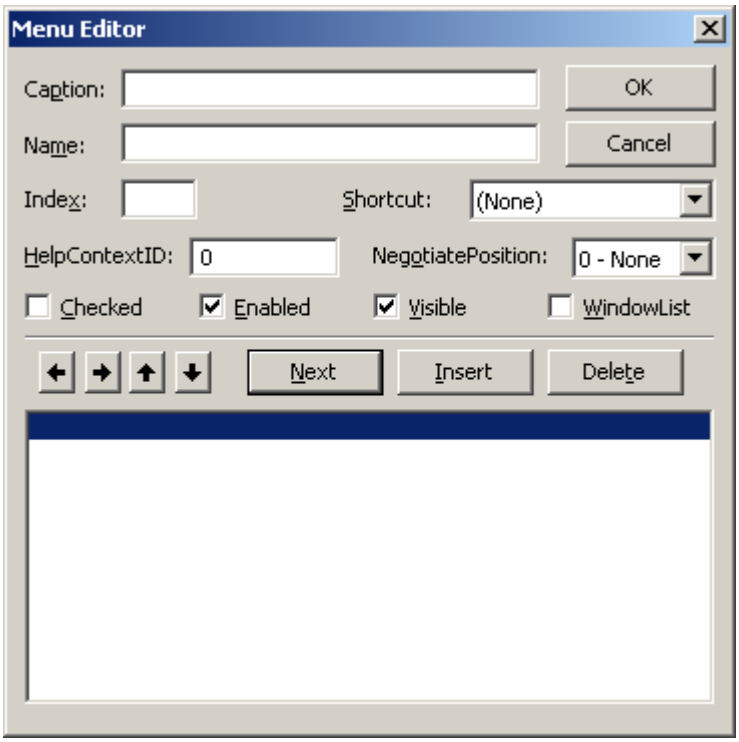
Строка menu bar расположено сразу под строкой заголовка и содержит одно или более меню.

При выборе меню раскрывается список пунктов меню menu items.

Пункты могут быть командами, разделителями и подменю.

Элемент управления меню является объектом и поэтому имеет свои свойства и события.

Меню создается с помощью Menu Editor (Tools – Menu Editor).



Свойства элемента меню

Caption – текст который появляется в пункте меню;

Name – значение этого свойства используется для ссылки на электронное меню;

Visible – видимость

Checked – установлено или неустановленно;

Enabled – разрешено неразрешено;

Чтобы создать элемент меню следует:

1. Выбрать форму;
2. Tools – Menu Editor;
3. Caption – набрать текст для одного заголовка меню (Open);
4. Name – набрать имя с помощью которого можно будет ссылаться на электронное меню (Open или mnuOpen);
5. Для уменьшения уровня смещения меню нажать  ;
6. Установить при желании другие свойства элемента меню;
7. Нажать Next для создания другого элемента меню или Insert для добавления элемента меню. С помощью кнопок  ,  можно передвигать меню между существующими элементами меню.
8. нажать Ok и выйти из редактора.

Разделители меню отображаются в виде горизонтальных линий между командами меню, для создания разделителя нужно:

1. Если он добавляется, то нажать Insert;
2. При необходимости сместить разделитель  ;
3. Caption – набрать – (минус)
4. Заполнить Name (Sep1 или mnuSep1)
5. нажать Ok.

Назначение клавиш доступа и быстрых клавиш

С помощью клавиш доступа AccessKeys пользователь может открыть меню удерживая Alt и набрать на клавиатуре назначения символ для данного меню.

Для назначения клавиш доступа:

1. Выбрать элемент меню;
2. В поле Caption набрать & непосредственно перед той буквой которая будет клавишей доступа.

Быстрые клавиши ShortcutKeys позволяет выполнить команду меню не открывая его. Чтобы назначить быструю клавишу

1. Выбрать команду меню
2. Выбрать функциональную клавишу или комбинированные клавиши в списке ShortCut.

Контекстное меню Pop – up menu

Это меню отображается в любом месте формы обычно при нажатии правой кнопки мыши для отображения контекстного меню используется метод Popup menu.

```
Edit (Cut, Copy, Paste)
Visible - False
Private Sub Form_MouseDown
If Button = vbRightButton Then
        Popupmenu Edit
End If
End Sub
```

Элемент управления CommonDialog.

CommonDialog – это стандартный набор окон диалога: открытия и сохранения файла, установка принтера, установка цветов шрифта. Для добавления CommonDialog используется Project – Components – Microsoft Common Dialog Control

Методы CommonDialog

ShowOpen – отображает окно открытого файла

ShowSave – окно сохранения файла

ShowColor – окно выбора цвета

ShowFont – окно выбора шрифта

ShowPrinter – окно настройки принтера

ShowHelp – система помощи Windows

Open и Save позволяет задать диск, папку, расширение и имя файла. Свойство FileName используется для получения имени выбранного файла.

Установка фильтров:

CommonDialog1.Filter = “All files (*.*) | *.* | Text files (*.txt) | *.txt”

Задание фильтра по умолчанию:

CommonDialog1.FilterIndex = 1

Отображение окна диалога Open:

CommonDialog1.ShowOpen

Вызов процедуры открытия файла (этого мы еще не знаем)

OpenFile (CommonDialog1.FileName)

В окне **Color** пользователь может выбрать цвет из палитры или создать собственный цвет. Можно использовать свойство Color для получения выбранного цвета.

CommonDialog1.ShowColor

Form1.BackColor = CommonDialog1.Color

В окне **Font** пользователь может выбрать шрифт, его размер, цвет и стиль. Выбранные свойства: Color, FontBold, FontItalic, FontUnderline, FontName, FontSize.

CommonDialog1.ShowFont

Text1.FontName = CommonDialog1.FontName

Text1.FontSize = CommonDialog1.FontSize

Text1.FontBold = CommonDialog1.FontBold

Text1.FontItalic = CommonDialog1.FontItalic

Text1.FontUnderline = CommonDialog1.FontUnderline

Text1.ForeColor = CommonDialog1.Color

В окне **Print** пользователь может задать параметры печати. Свойства Copies, FromPage, ToPage, hDC, Orientation.

Панель инструментов ToolBar

Элемент управления Microsoft Windows Common Controls

InageList – элемент хранения списков графическое изображения для других элементов управления.

ProgressBar – элемент индикации процесса выполнения длительных операций

Slider – Скользящий указатель или элемент установки значения из диапазона

StatusBar – панель состояния

TabStrip – Элемент для отображения многостраничных вкладок.

TreeView – элемент отображения иерархических структур

UpDown – элемент изменения значений

ImageList никогда не используется самостоятельно. Он предоставляет другим элементам управления список графических изображений.

Index – другие элементы выбираю изображения ImageList. При этом на вкладыше General нужно установить размер изображения. Поддерживающие форматы gif, jpg ,cur, bmp.

ToolBar – панель с различными кнопками свойства которых определяет разработчик. Key – свойство используется для определения выбранной пользователем кнопки

```
Private Sub Toolbarl_ButtonClick(ByVal Button As Button)
If Button.Key = "tbLoad" Then
MsgBox "Теперь загружается"
End If
End Sub
```

StatusBar — это важный элемент, который должен быть в каждом приложении, отображающий различную информацию, например дату, время и многое другое.

TabStrip — диалоговые окна с вкладками. TabStrip содержит семейство объектов Tab, но не является контейнером, поэтому для отображения в нем элементов управления необходим элемент контейнер, например Frame.

ProgressBar отображает, насколько продвинулся процесс копирования, перемещения, загрузки или сохранения файлов. Важнейшими свойствами являются Min (нижняя граница). Max (верхняя граница) и Value (текущее значение). Если необходимо отображать процесс от 0 до 100%, то устанавливается Min = 0, Max = 100.

Буфер обмена

Для работы с буфером обмена используется объект ClipBoard. Нкоторые методы ClipBoard

Clear – очистка буфера

SetText – копирование текста в буфер

GetText – вставка текста из буфера

SetData – копирование графики в буфер

GetData – вставка графики из буфера

Clipboard.SetData Picturel.Picture

Picture2.Picture = Clipboard.GetData

Проект 1. Блокнот.

Форма содержит TextBox, CommonDialog, Menu, ImageList, ToolBar, StatusBar, PopupMenu

CommonDialog

Filter – …

Menu:

Файл (Новый, Открыть, Сохранить, разделитель, Выход)

Редактирование (Вырезать, Скопировать, Вставить)

Формат (Шрифт)

Справка (О программе …)

PopupMenu

Вырезать, Скопировать, Вставить, разделитель, Шрифт

ToolBar:

Новый, Открыть, Сохранить, Вырезать, Скопировать, Вставить, О программе … (для этого сначала нарисовать в Paint 7 кнопок, соответствующих командам, внести их в ImageList1, установить у ToolBar ImageList – ImageList1)

Для Новый

Text1.Text = “”

Для Открыть

CommonDialog1.ShowOpen

Form1.Caption = CommonDialog1.FileName

Для Вырезать

ClipBoard.Clear

ClipBoard.SetText Text1.SelText

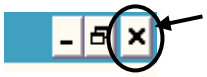
Text1.SelText = “”

Запуск и закрытие программы.

Запуск программы осуществляется несколькими способами:

1. **Пуск –> Программы –> Microsoft Access**
2. дважды щелкнуть левой кнопкой мыши по ярлыку  на рабочем столе

Закрыть программу можно:

1. нажав вместе клавиши **[Alt] + [F4]**
2. щелкнув по кнопке **X** в правом верхнем углу окна 

Создание новой базы данных.

При запуске программы перед вами открывается следующее окно:

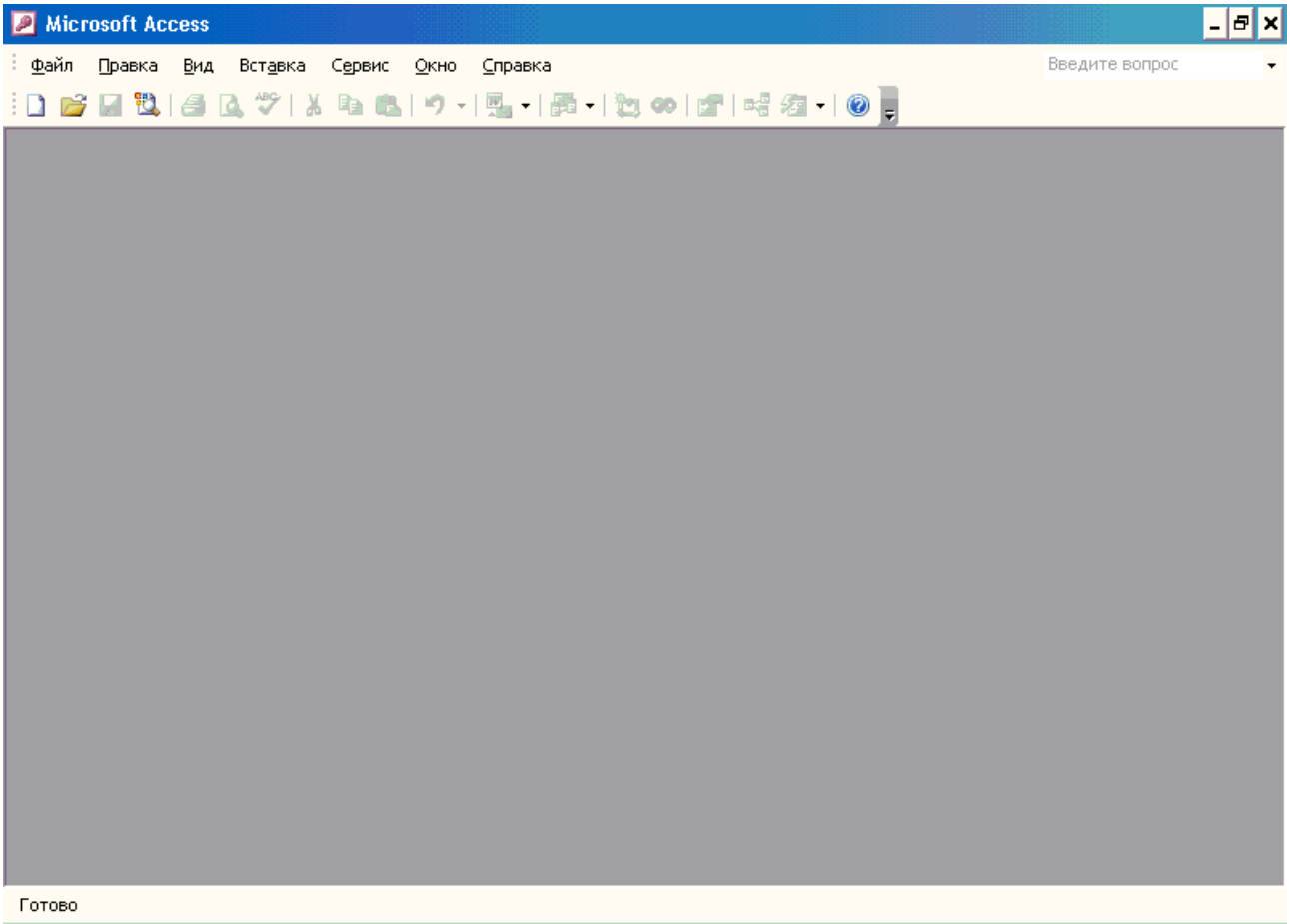
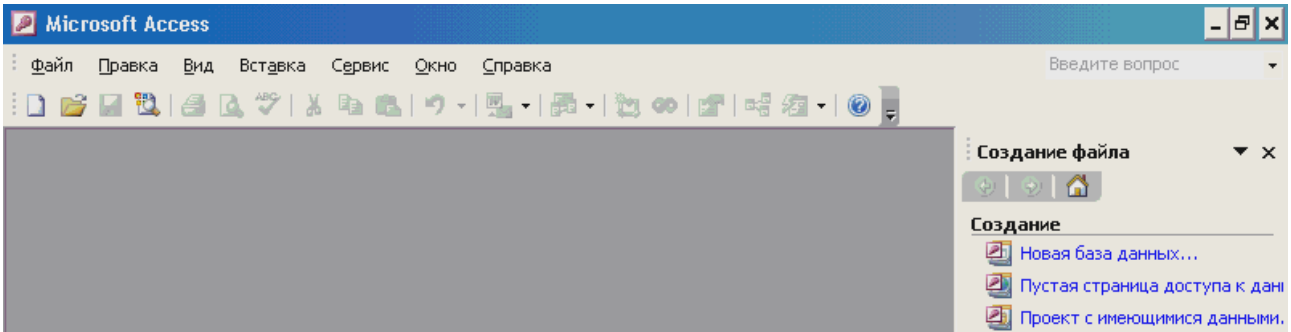


Рис. 1. Окно программы Microsoft Access

Для создания новой базы данных выбираем **Файл –> Создать**. Справа в окне программы появляется новая панель (Рис. 2).



В этой панели выбираем **Новая база данных…** На экране появляется следующее окно:

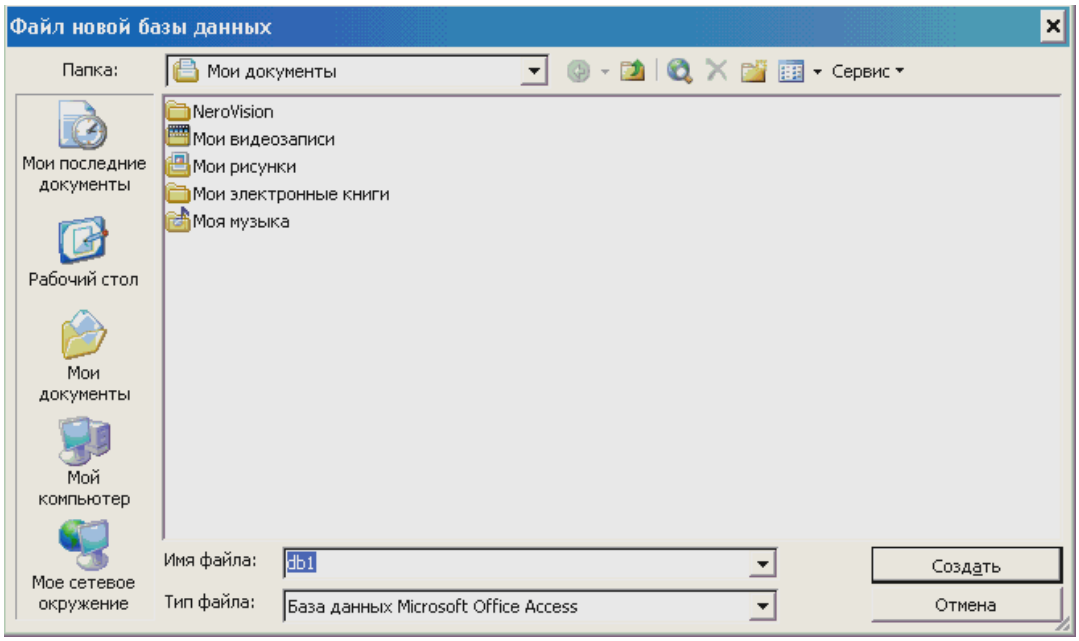


Рис.3. Окно создания файла новой базы данных

В поле **Имя файла** вводим новое имя файла базы данных и нажимаем кнопку **Создать**. Открывается окно новой базы данных (Рис.4).

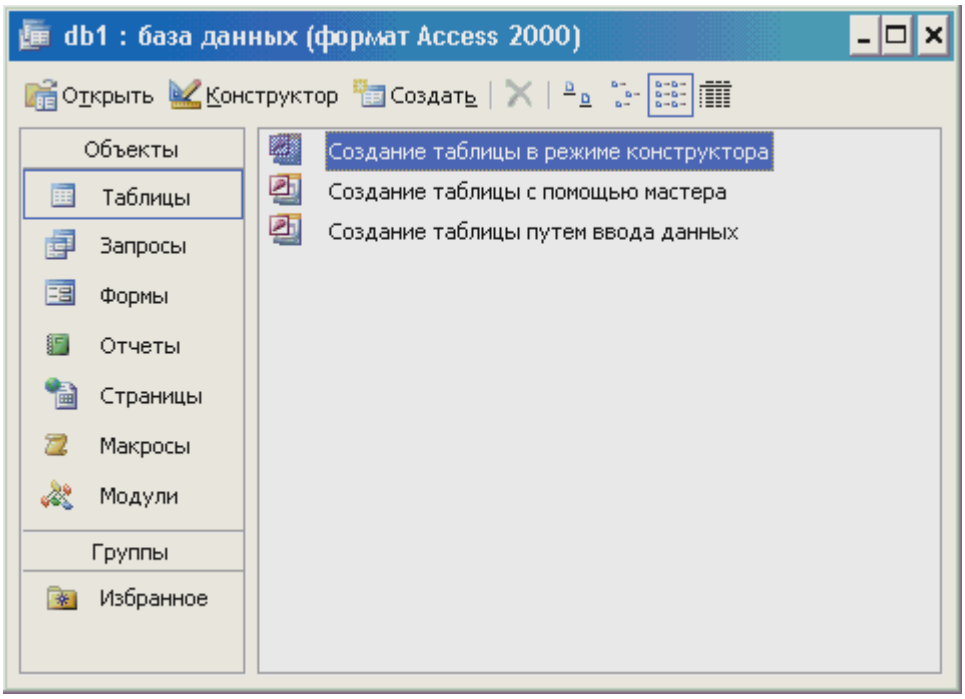


Рис. 4. Окно базы данных

В базу данных (БД) Access могут входить разнородные объекты. Различают следующие типы (см. Рис. 4):

Таблицы. В Access все данные хранятся в виде таблиц.

Запросы позволяют выбрать из БД только ту информацию, которая соответствует определенному условию.

Форма представляет бланк, подлежащий заполнению или маску-формуляр, позволяющую ограничить объем информации, доступной пользователю.

Отчет предназначен для печати любого набора специально оформленных данных.

Макрос автоматизирует выполнение конкретной операции БД без программирования.

Модуль содержит программы на языке Visual Basic, применяемые для настройки и расширения БД.

Т а б л и ц ы .

Любая база данных в Microsoft Access состоит из таблиц.

****

***

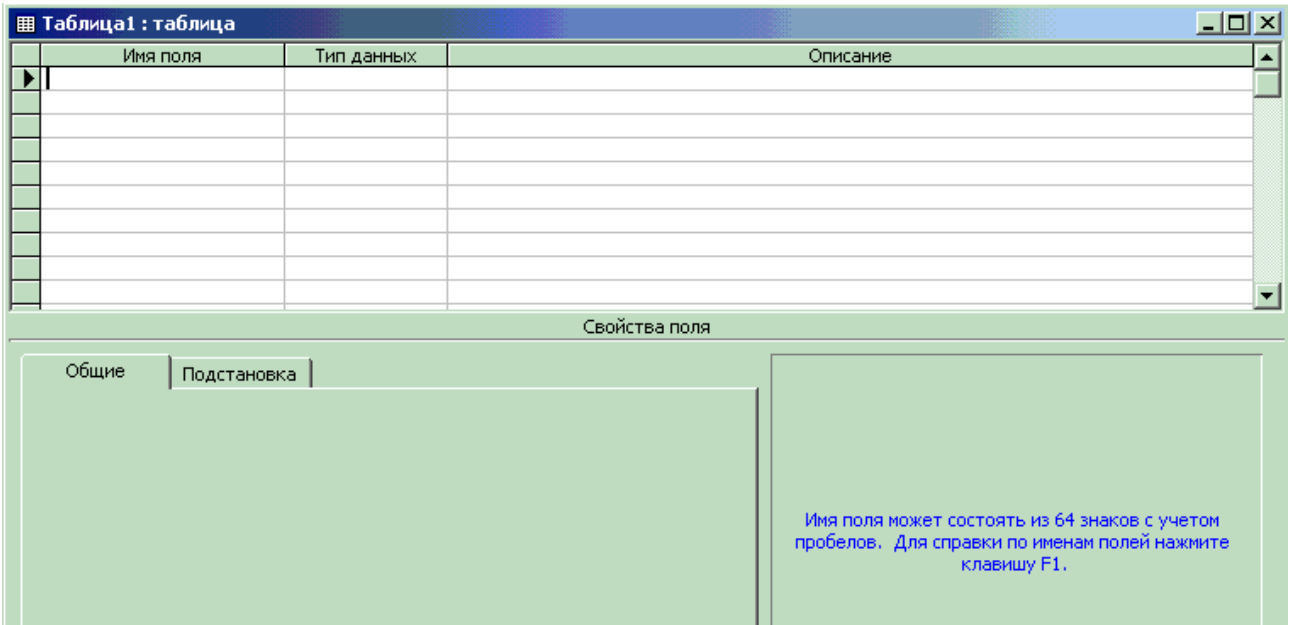
Первый пункт контрольного задания – создать БД из таблиц.

	Товар	Металл	Вес	Цена	Код производителя
	Кольцо	Золото	4	1200	1001
	Серьги	Золото	3	950	1002
	Кольцо	Серебро	3	300	1003
	Цепочка	Золото	2	900	1003
	Серьги	Платина	2	1650	1002
	Цепочка	Серебро	3	400	1003
	Брошь	Золото	5	1300	1001

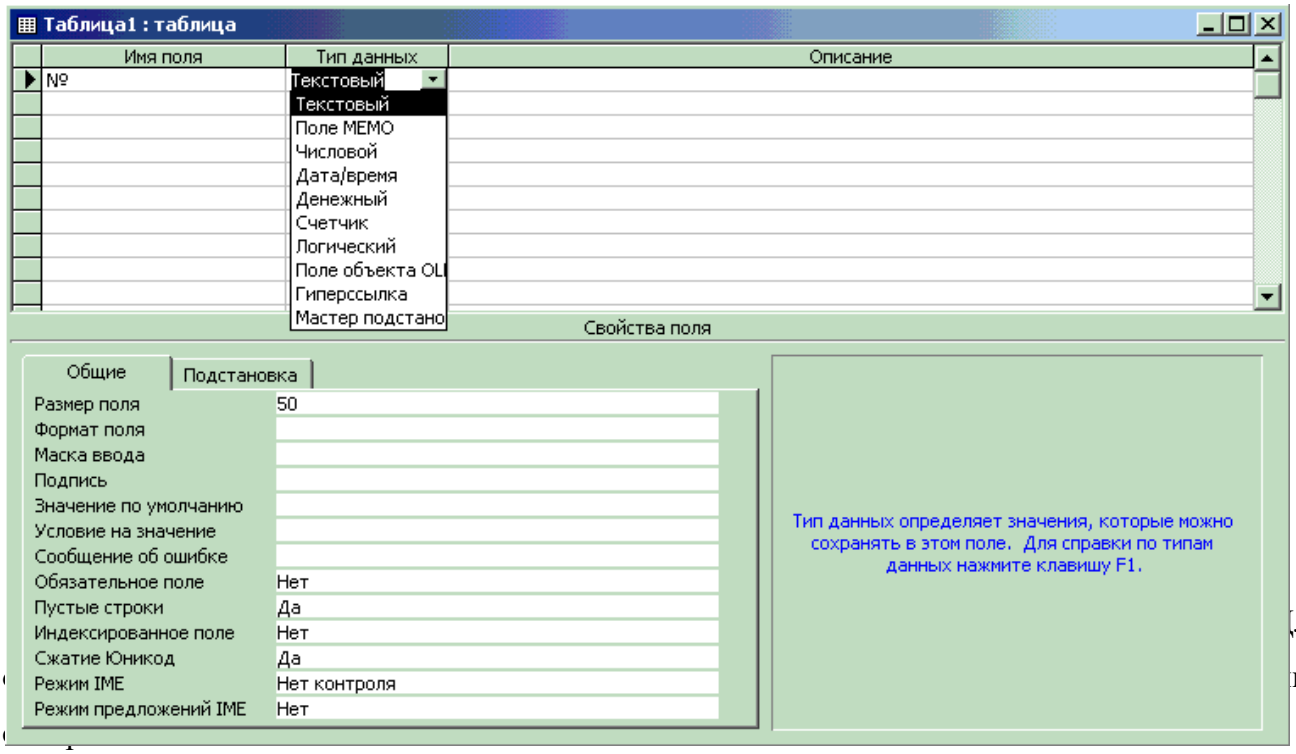
Код производителя	Фирма	Город
1001	ООО «Алмаз-Холдинг»	Казань
1002	ЗАО «Бриллиант»	Москва
1003	ООО «Ютта»	Новосибирск

****

В окне базы данных (Рис.4) дважды щелкаем мышкой **Создание таблицы в режиме конструктора**. Открывается следующее окно (Рис. 5)



В столбец **Имя поля** вводим заголовки столбцов таблицы и в соответствии с данными этих столбцов выбираем **Тип данных** (Рис.6).



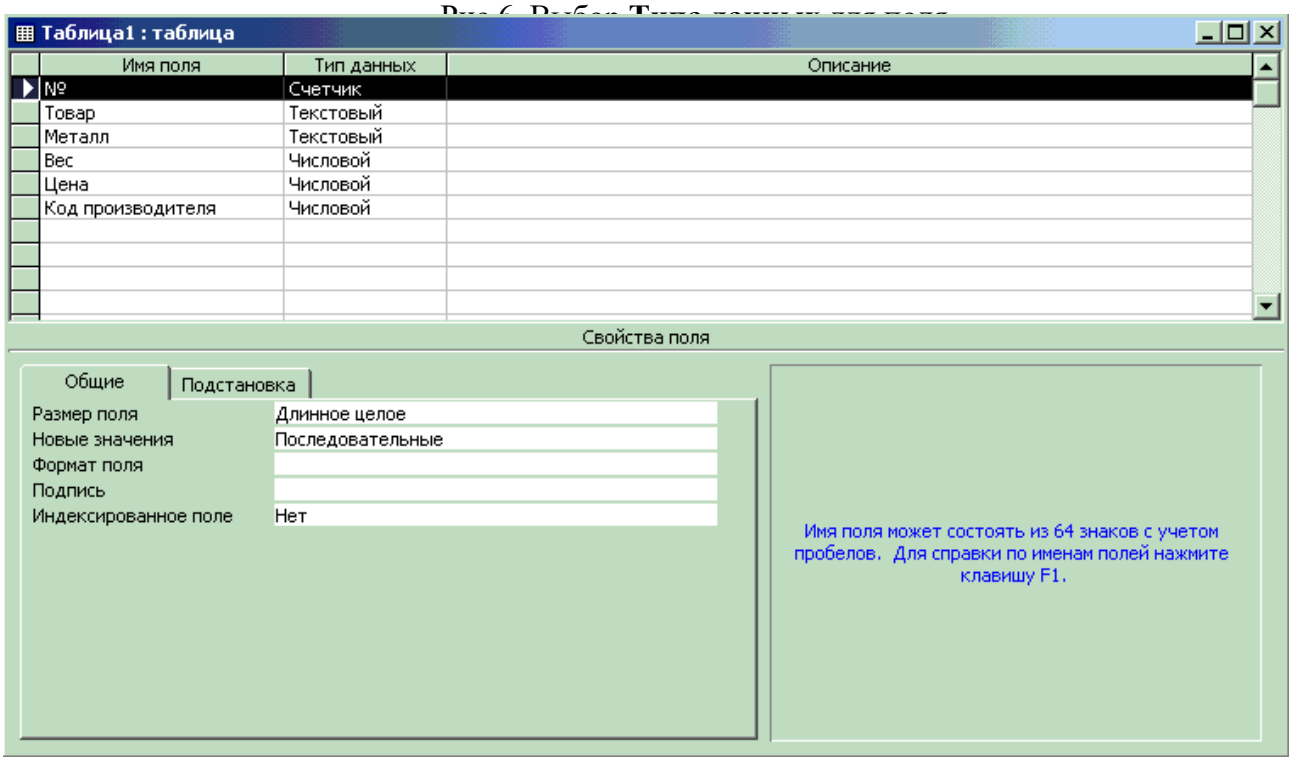


Рис.7. Заполненная таблица Конструктора

Далее обязательно делаем ключевое поле (обычно, но не всегда, это поле **Счетчик**). Выделяем мышкой строку и на панели инструментов нажимаем кнопку **Ключевое поле** 

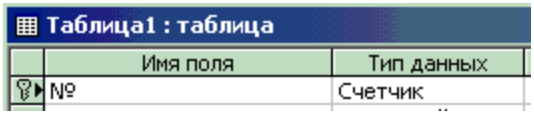


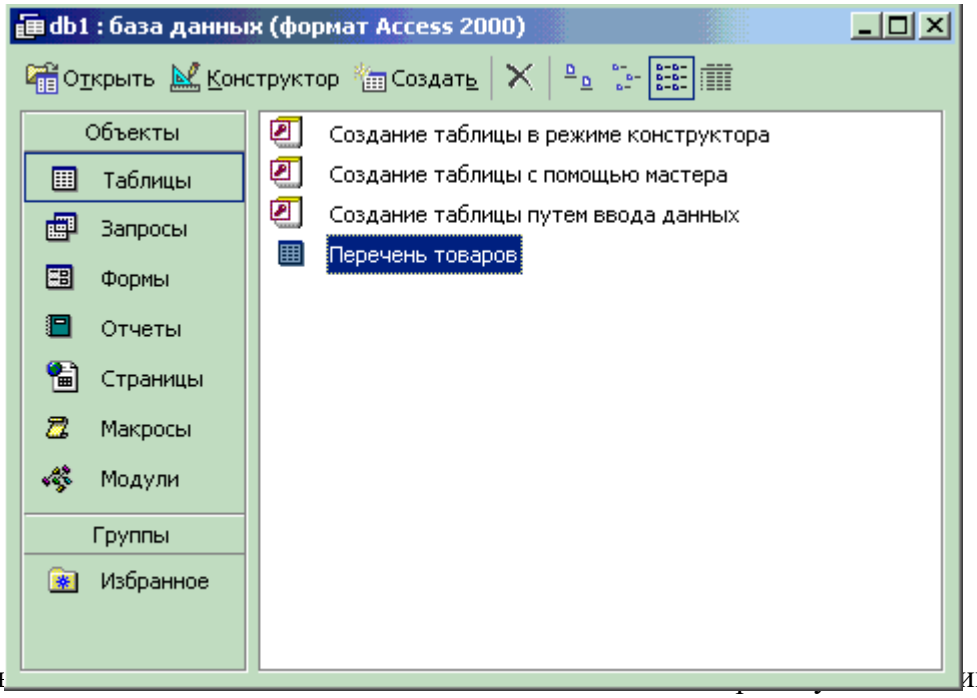
Рис.8. Ключевое поле по №

Закрываем окно Конструктора.

На вопрос: ***«Сохранить изменения макета или структуры объекта "таблицы Таблица1"?»*** отвечаем **ДА**

В окне **Сохранение** вводим новое имя файла (напр, Перечень товаров) и нажимаем **ОК**.

В окне базы данных появляется таблица **«Перечень товаров»** (Рис.9)



Дважды щелкаем по таблице и начинаем вводить данные (счетчик ставится сам) – Рис.10.

Рис.9. Окно базы данных с таблицей перечня товаров

Перечень товаров : таблица

№	Товар	Металл	Вес	Цена	Код производи
1	Кольцо	Золото	4	1200	1001
2	Серьги	Золото	3	950	1002
3	Кольцо	Серебро	3	300	1003
4	Цепочка	Золото	2	900	1003
5	Серьги	Платина	2	1450	1002
6	Цепочка	Серебро	3	400	1003
7	Брошь	Золото	5	1300	1001
(Счетчик)			0	0	0

Рис.10. Таблица с данными

Аналогично добавляем другую таблицу и называем ее **«Производители»** (ключевое поле в этой таблице – **Код производителя**, Тип ключевого поля – **Числовой**, т.к. тип счетчик начинает нумерацию только с 1, а у нас 1001). Далее заполняем ее данными и закрываем.

В разделе таблицы в окне базы данных появились 2 таблицы: **«Перечень товаров»** и **«Производители».**

Теперь данные этих таблиц нужно связать. Для этого заходим **Сервис –> Схема данных…**

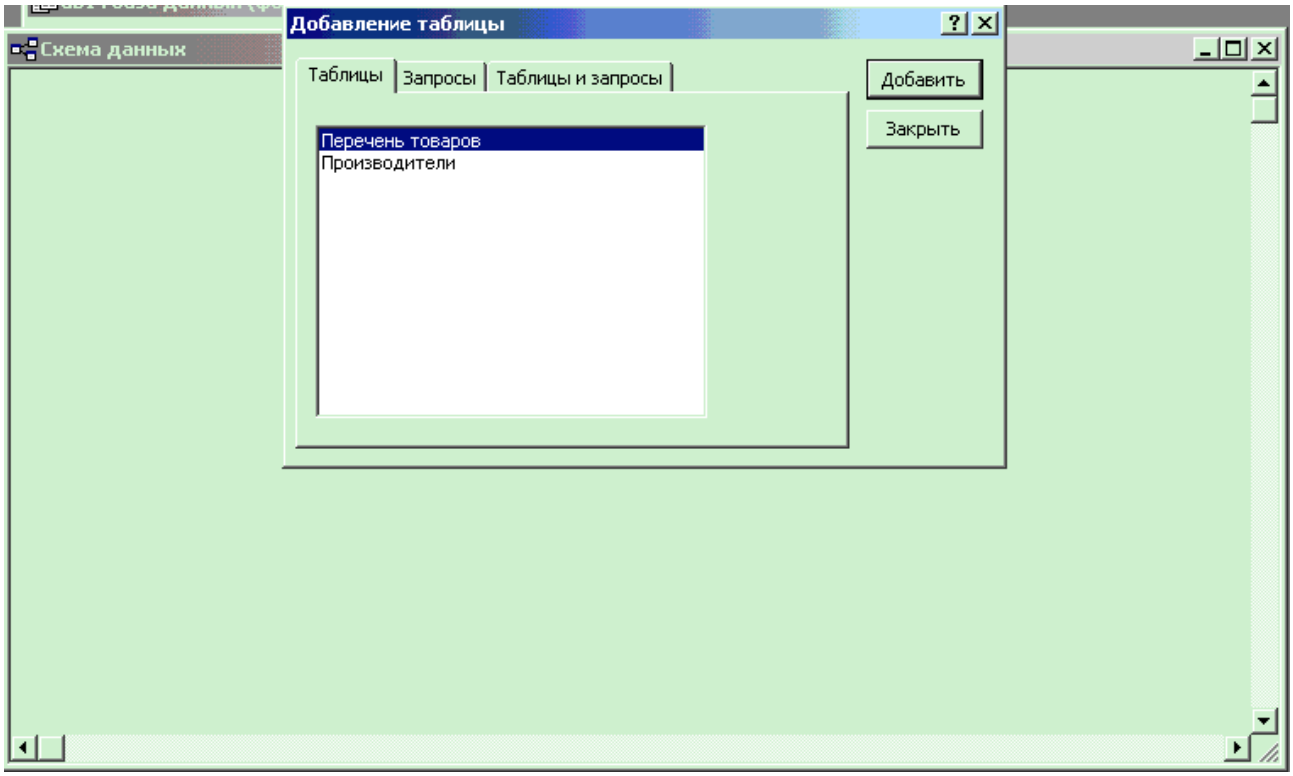


Рис.11. Добавление таблиц в окно Схемы данных

В окне **«Добавление таблицы»** добавляем обе таблицы (по очереди их выделяя мышкой и нажимая кнопку **Добавить**), затем закрываем это окно. В окне **Схема данных** появляется два списка (Рис.12).

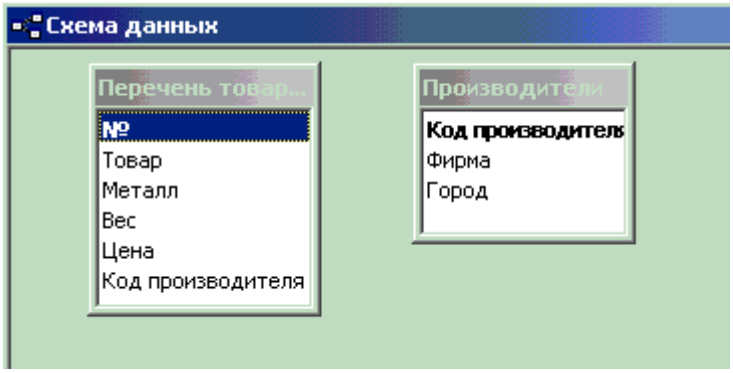


Рис.12. Таблицы в окне Схема данных

Цепляем мышкой **Код производителя** в первом окне и тащим в другое окно на это же название.

В появившемся окне ставим переключатель **Обеспечение целостности данных** и нажимаем **Создать.**

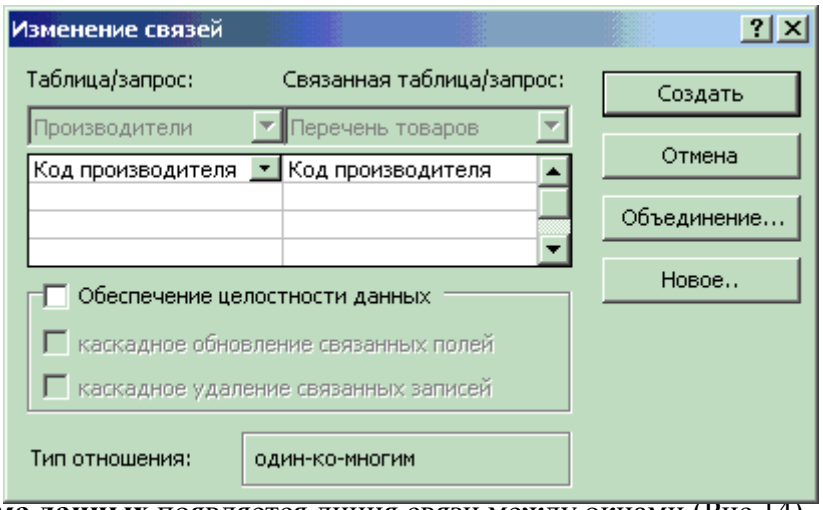
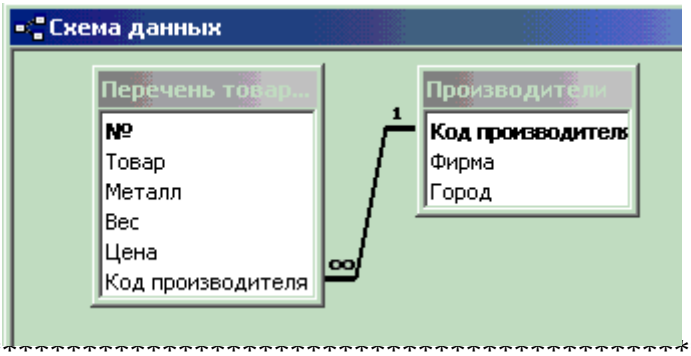


Рис.13. Окно изменения связей между таблицами

В окне **Схема данных** появляется линия связи между окнами (Рис.14).



****

Рис.14. Связи между таблицами

***

Второй пункт задания – создать Запросы.

Для нашей БД создадим следующий запрос по товарам

- не из Серебра
- по цене от 800 до 1500 руб.
- от Казанского производителя.

****

***

В окне базы данных (Рис.9) переходим на вкладку **Запросы**. Выбираем **Создание запроса в режиме конструктора.**

Добавление таблиц в запрос аналогично Рис.11.

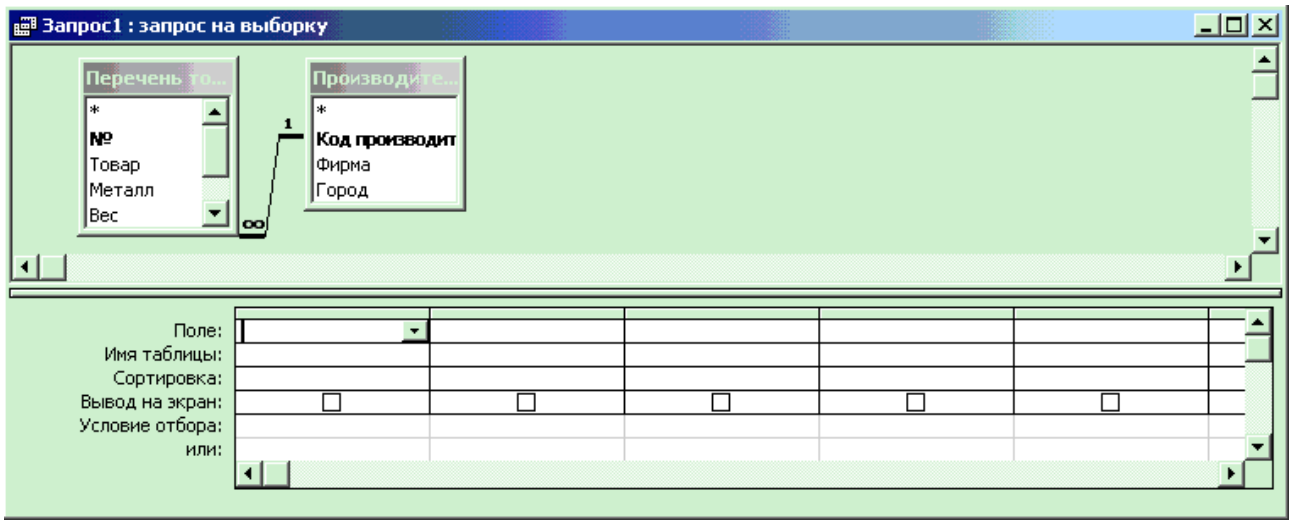


Рис.15. Окно Запроса

В окне Запроса выбираем **Поля** (первая строка), которые хотим вывести (№, Товар, Металл, Вес, Цена, Фирма, Город) (см. Рис.16).

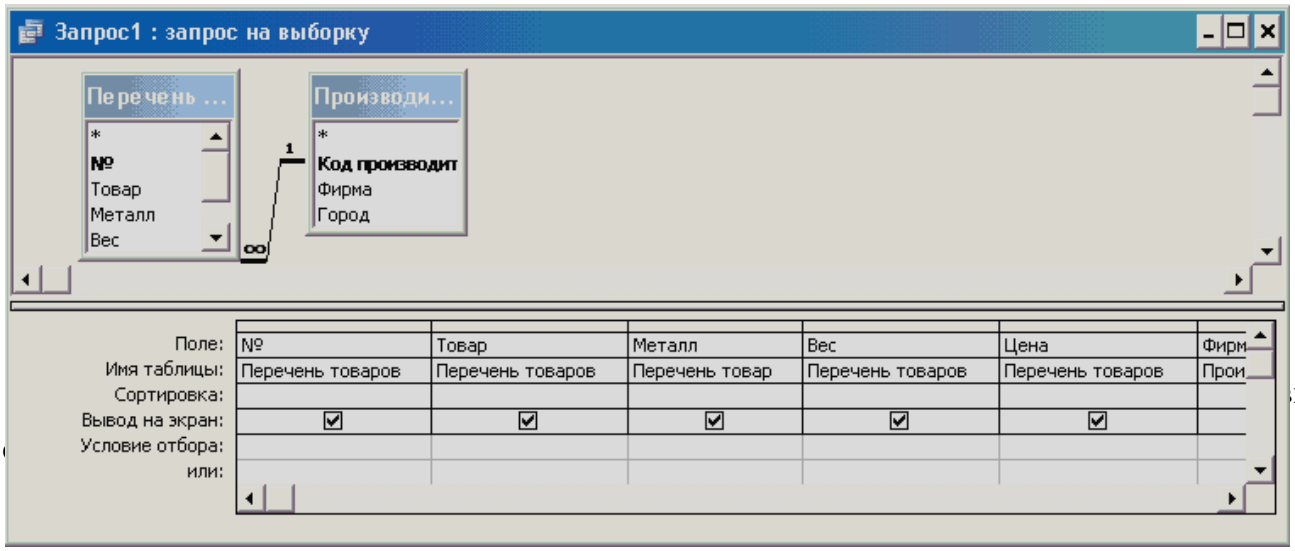
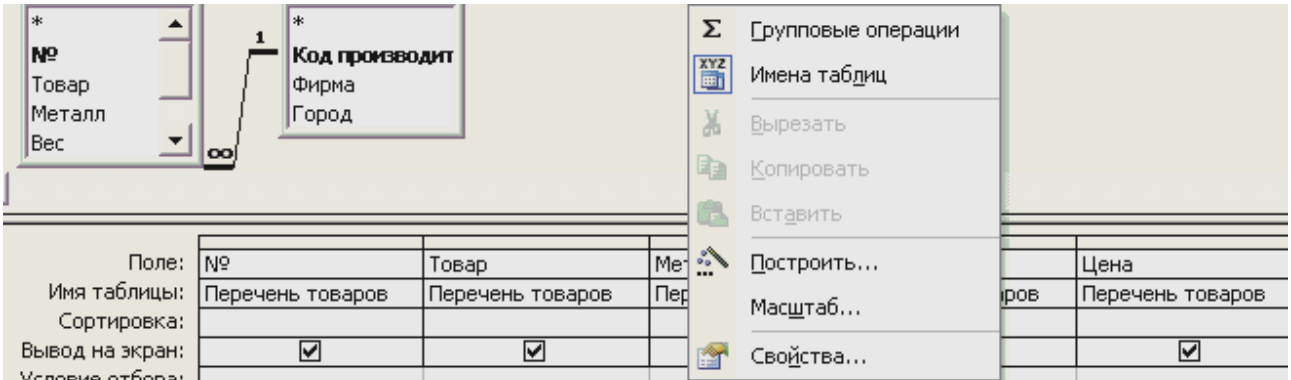


Рис.16. Заполненные поля в окне Запроса



В открывшемся окне в левом списке раскрываем папку **Таблицы**, заходим в **Перечень товаров**, в среднем списке двойным щелчком выбираем **Металл**. Металл добавляется в верхнее окно. Далее нажимаем кнопку **“<>”**(не равно) и пишем **Серебро** (Рис.18). **ОК.**

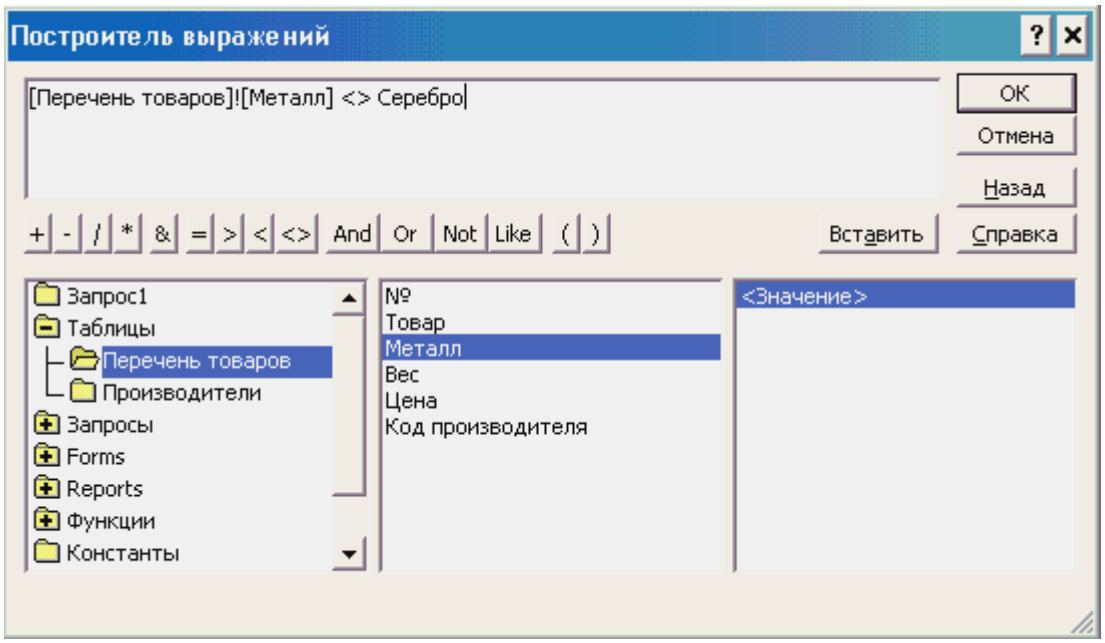


Рис. 18 Ввод условия

Аналогично вводим условия для цены (должна получиться строка: **[Перечень товаров]![Цена] > 800 And [Перечень товаров]![Цена] < 1500**) и условие для города (**[Город] = Казань**). Закроем Конструктор Запросов и сохраним его как **Запрос 1**. Затем в окне БД два раза щелкнем мышкой **Запрос1**. Если Вы все сделали правильно, то получиться как на Рис.19.

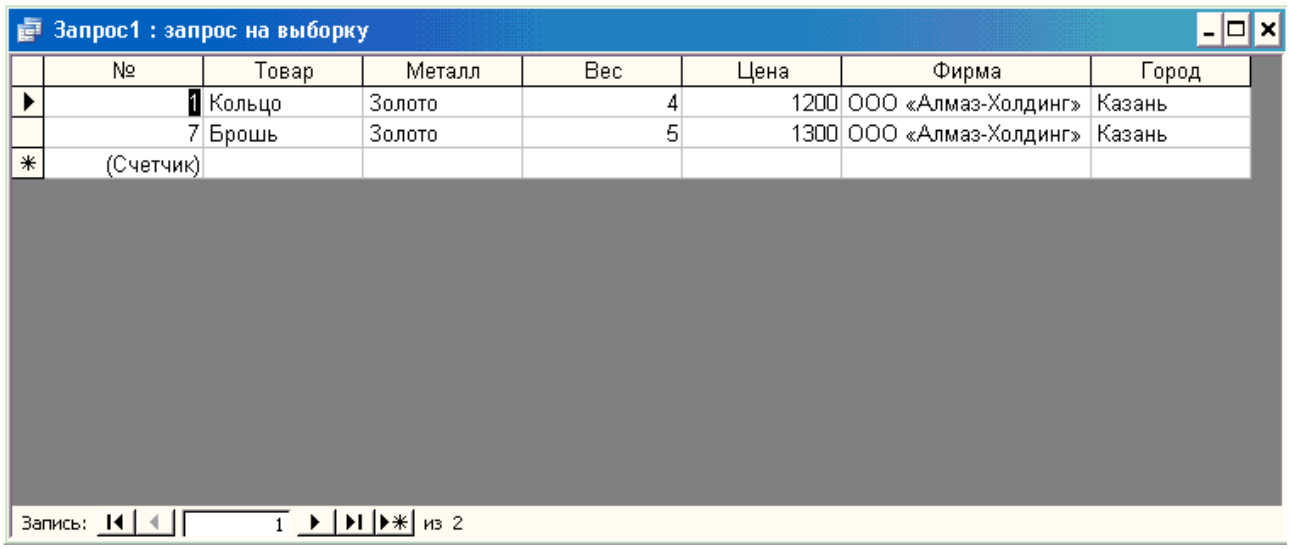


Рис.19 Результат запроса

Закроем **Запрос1** и перейдем на вкладку **Формы**.

Ф о р м ы .

****

***

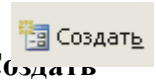
Третий пункт задания – создание Форм

****

***

Создадим форму по таблице **Перечень товаров**.

В окне БД нажимаем кнопку **Создать**



В окне **Новая форма** выбираем **Автоформа: в столбец**, ниже в раскрывающемся списке выбираем таблицу **Перечень товаров**. **ОК**. Получаем Форму по первой таблице (Рис.20).

Закроем форму (с

Рис.20. Форма

О т ч е т ы .

****

***

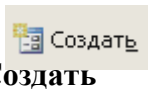
Четвертый пункт задания – создание отчетов.

****

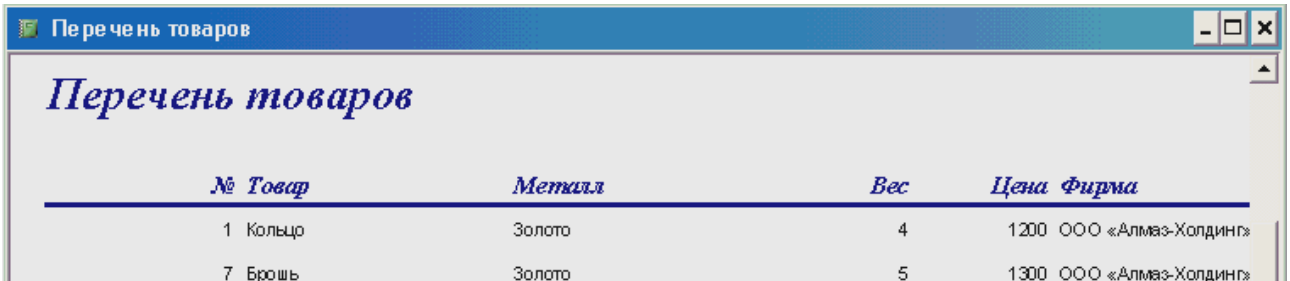
***

Создадим отчет по **Запросу1**.

В окне БД нажимаем кнопку **Создать**



В окне **Новый отчет** выбираем **Автоотчет: ленточный**, ниже в раскрывающемся списке выбираем **Запрос1**. **ОК**. Получаем отчет по Запросу1 (Рис.21).



Закроем отчет и сохраним его.

Создание новой панели для текущей базы данных

****
***

Пятый пункт задания – создать новую панель для автоматизации работы с новой базой данных

****
***

Нужно автоматизировать работу с текущей БД, необходимо, чтобы все ее элементы (таблицы, формы, отчеты и запросы) отображались на экране в виде отдельной панели инструментов.

Для этого заходим **Сервис –> Настройка**, вкладка **Панели Инструментов**, **Создать**. В окне **Создание панели инструментов** вводим название новой панели (напр, Ювелир), нажимаем ОК. Рядом с окном **Настройка** появляется маленькая панель инструментов **Ювелир**.

Переходим в окне **Настройка** на вкладку **Команды**, в левом окне выделяем категорию **Все таблицы**, а в правом хватаем кнопку таблицы **Перечень товаров** (наша первая таблица) и мышкой тащим на новую панель **Ювелир**. Также поступаем с остальными **Таблицами, Запросами, Формами и Отчетами**. Закрываем окно **Настройка**.

Программа POWERPOINT – простое и доступное средство для создания презентаций.

POWER – в переводе с англ. мощь, сила; POINT – точка, указатель, т.е. мощно указать или представить.

Презентация (от английского presentation - представление) в электронном виде – форма объединения различных видов информации в одном документе, предназначенном для показа на экране.

Основные возможности программы:

- Ввод текста;
- Выбор фона слайда;
- Вставка на слайд рисунков, фото и других объектов: видео и аудио;
- Редактирование текста и изображений;
- Вставка новых слайдов и их копии;
- Настройка переходов слайдов;
- Настройка эффектов анимации слайдов;
- Вставка гиперссылок и управляющих кнопок;
- Настройка демонстрации презентации;
- Вывод слайдов на печать.

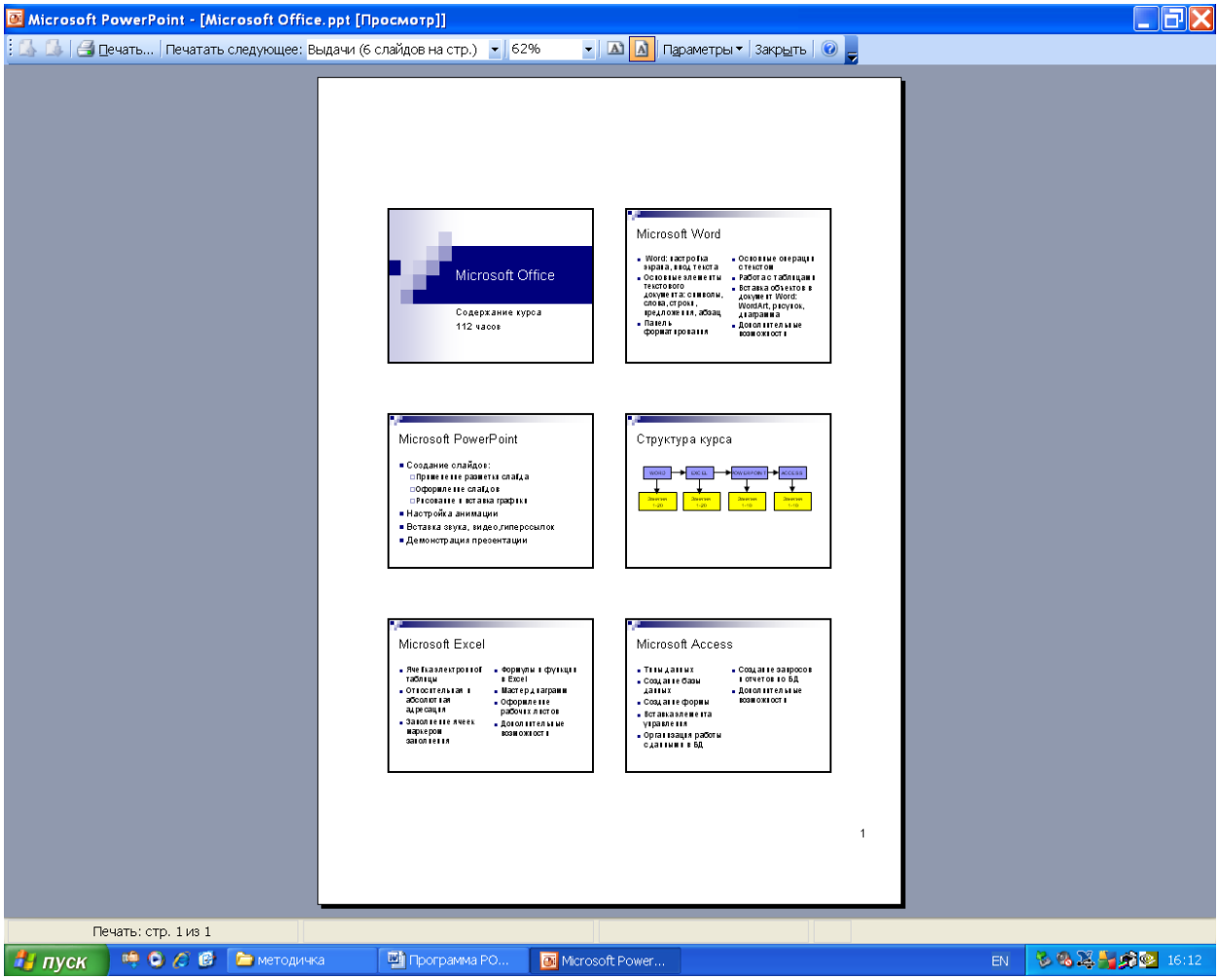
Чтобы приступить к созданию презентации, надо подготовить материал: скомпоновать и разбить его на отдельные слайды. Не старайтесь вместить на слайд как можно больше информации – на экране плохо видно мелкий текст и вообще много читать с экрана никто не любит. Не утомляйте зрителей излишними анимационными эффектами. А вот разметку и дизайн слайда выбирайте спокойную и в одном стиле.

Документом POWERPOINT является файл с произвольным именем с расширением .ppt (презентация), или .pps (демонстрация презентации).

Приступим к практическим занятиям.

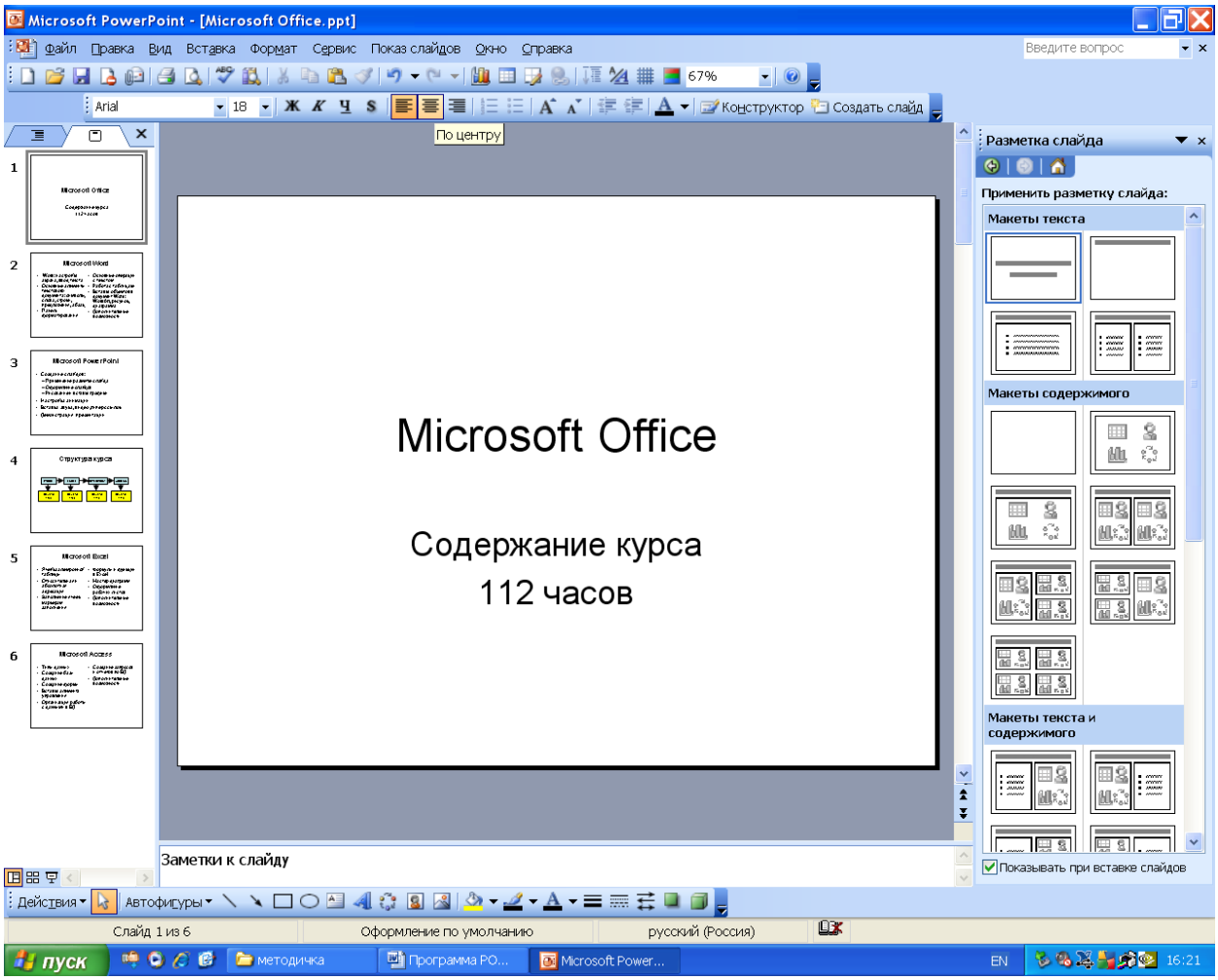
Практическая работа №1.

Создать презентацию «Microsoft Office», состоящую из 6 слайдов.



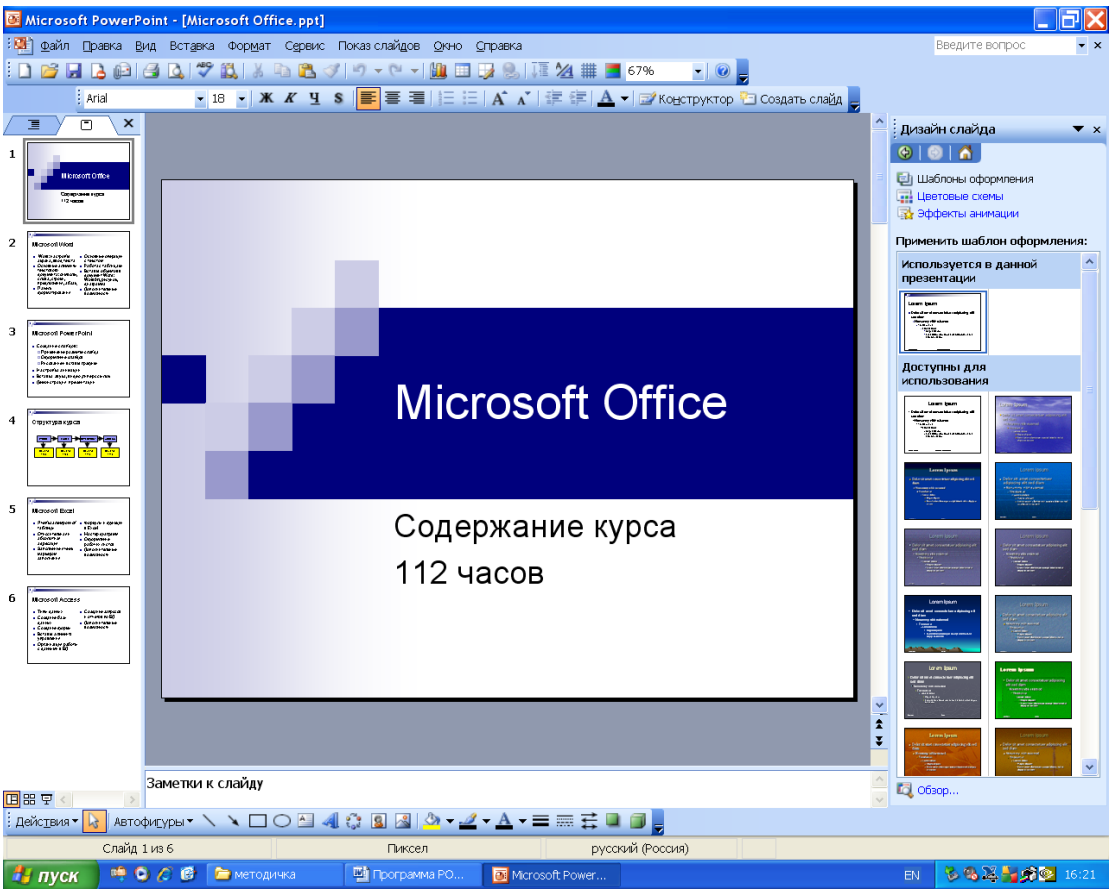
1.При запуске программы POWERPOINT создаётся презентация с именем Презентация 1, состоящая всего из одного слайда.

Заходим в меню ФОРМАТ-РАЗМЕТКА СЛАЙДОВ и в области задач выбираем нужную разметку:



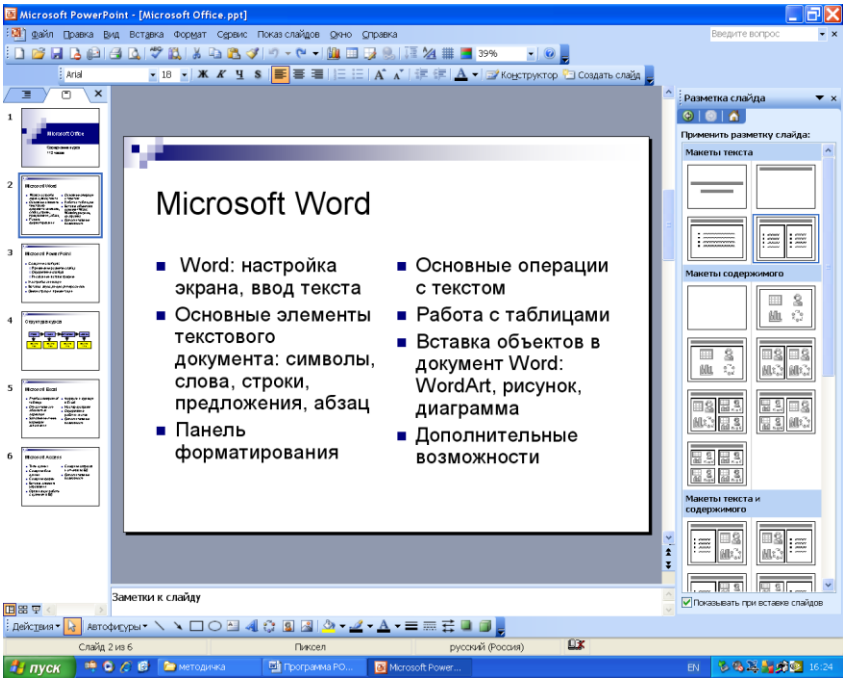
Щёлкнув мышкой в первой строке, набираем *Microsoft Office*, а во второй: *Содержание курса 112 часов.*

Теперь выберем дизайн нашей презентации: заходим в меню *формат – оформление слайда* и выбираем подходящий дизайн.



Переходим ко второму слайду: в главном меню **вставка – создать слайд,** либо на панели форматирования воспользуйтесь кнопкой **создать слайд.**

Второй слайд имеет другую разметку: заголовок и два столбца текста:



Далее всё аналогично, лишь

меняем разметку слайда на каждом слайде.

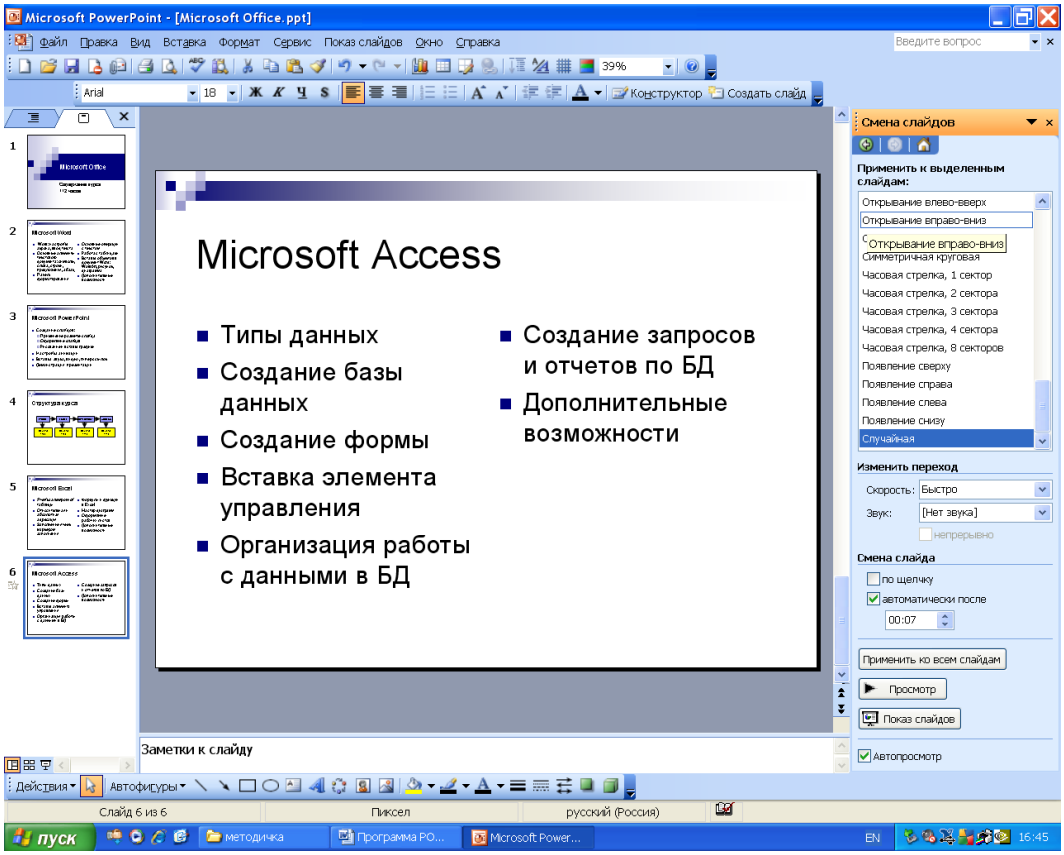
На слайде №3, где встречается встроенный список, надо воспользоваться кнопками панели форматирования «уменьшить» или «увеличить отступ».

На слайде №4 надо выбрать пустой слайд или с одной строкой названия. Далее воспользоваться панелью рисования и выбрать прямоугольник и стрелку. Построить схему. Текст на прямоугольниках пишется с помощью кнопки «надпись» также с панели рисования.

Презентация готова, просмотрим её на экране. Для этого в меню «Показ слайдов» выбираем «начать показ». Чтобы просмотреть презентацию, нам приходиться нажимать клавишу или мышку при переходе от слайда к слайду. В конце просмотра нажмите клавишу ESC, чтобы вернуться в программу.

Чтобы настроить автоматически просмотр презентации, заходим в меню «показ слайдов» - «смена слайдов», убираем галочку с окошка «по щелчку» и ставим галочку в окошке «автоматически после», проставляя нужное количество секунд для прочтения слайда, и не забудем нажать кнопку «применить ко всем слайдам»,

если это необходимо:



Как настроить анимацию?

Выделяем объект, например первая строка первого кадра:

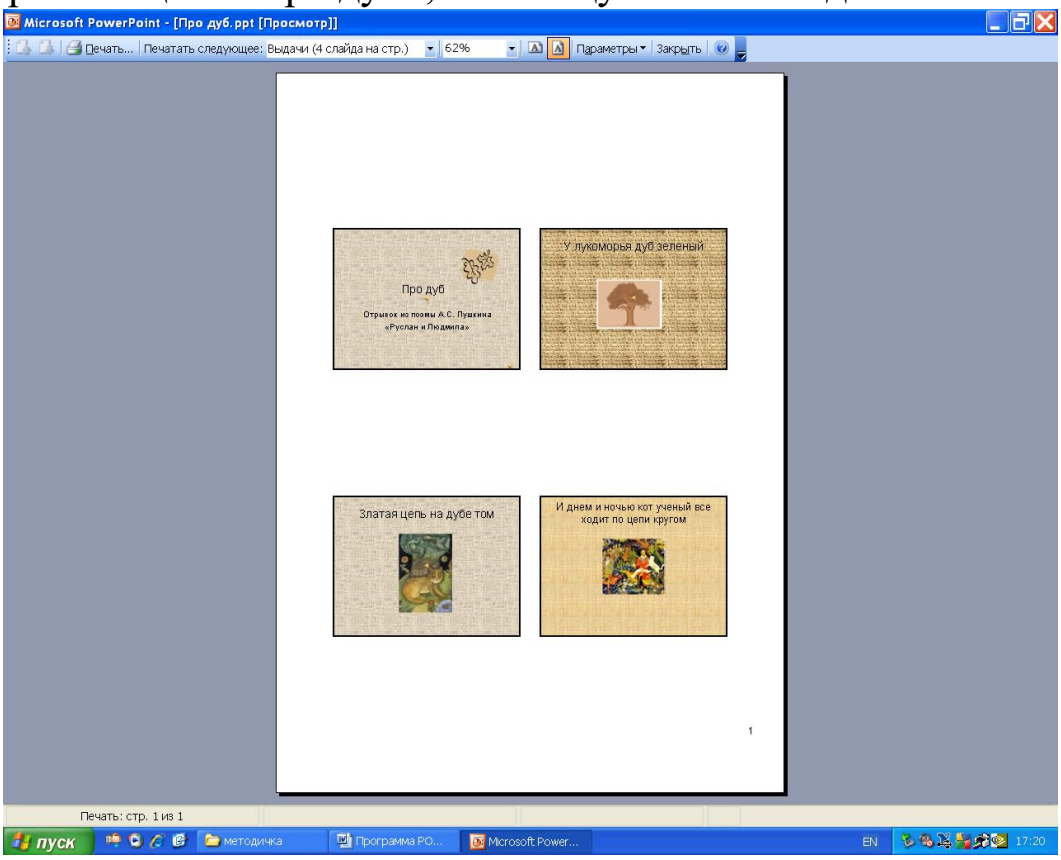
Microsoft Office.

Далее заходим в меню *Показ слайдов – настройка анимации*. Справа на панели задач нажмём кнопку *добавить эффект – на вход – и выберем (например,) кнут.* Далее, справа вместо мышки выбираем «с предыдущим», скорость и переходим к анимации следующего объекта - **Содержание курса 112 часов.** Выделяем, справа нажмём кнопку *добавить эффект – на вход – и выберем (опять, чтобы в одном стиле) кнут.* Далее, справа вместо мышки выбираем «после предыдущего» и, если нужно, меняем скорость.

Так можно настроить всю презентацию.

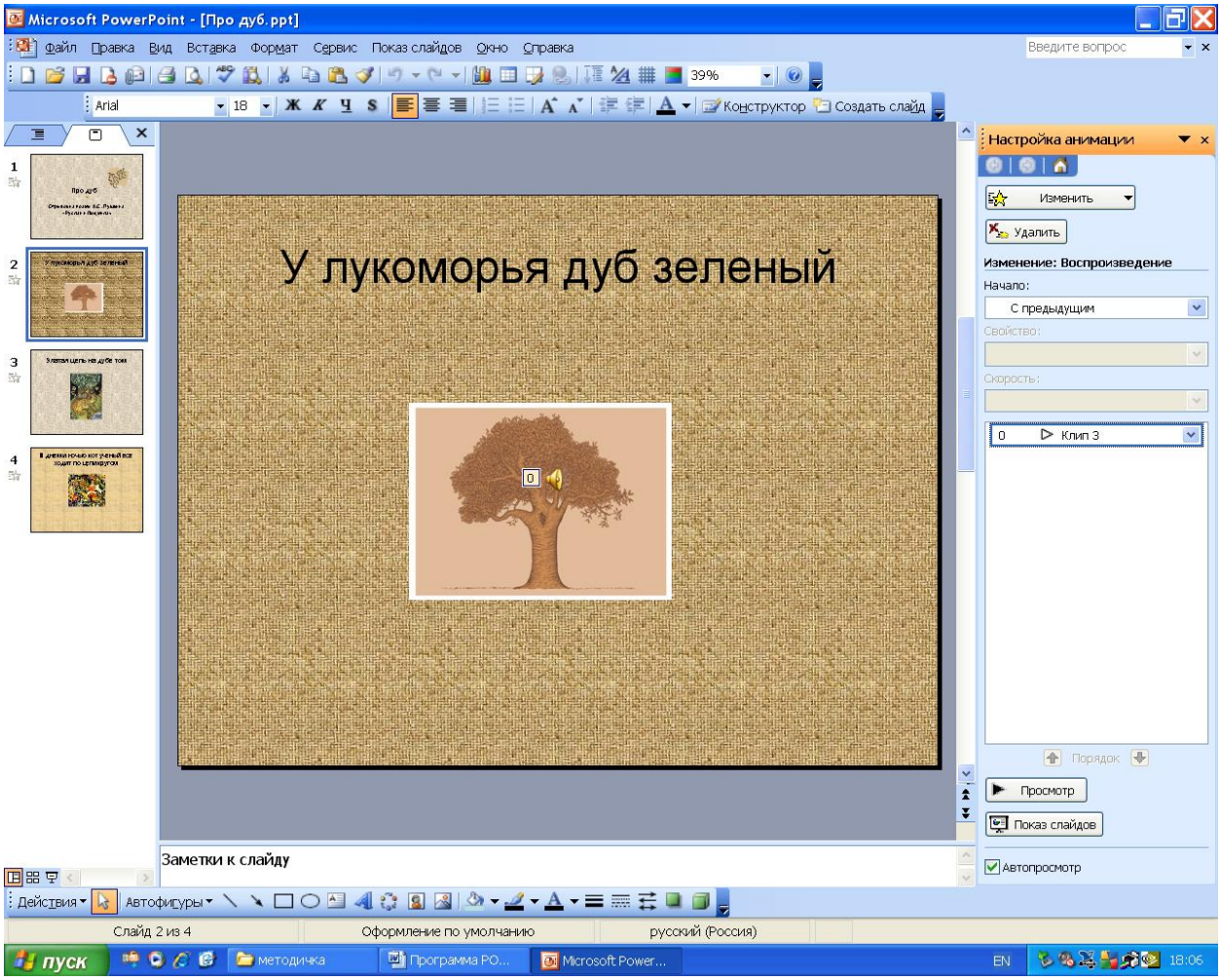
Практическая работа №2.

Создать презентацию «Про дуб», состоящую из 4 слайдов.



оздавать слайды и анимировать мы научились. Вставить текст и картинку на слайд – аналогично работе в WORD. Теперь озвучим наше творение. С первого слайда у нас зазвучит музыка, а со второго пойдет текст на фоне музыки.

Итак, встаём на 2-й слайд. Заходим в меню:



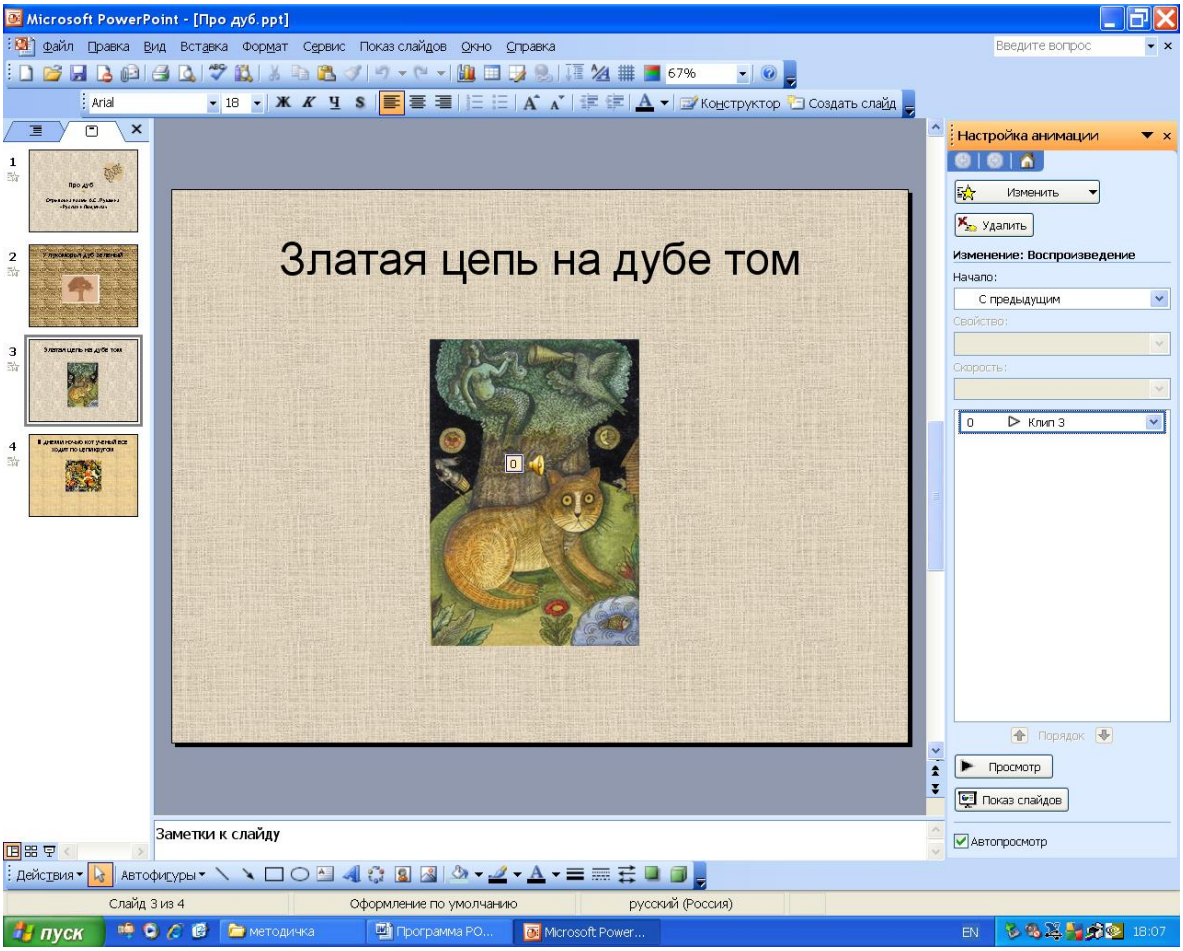
вставка – фильмы и звук – записать звук

в открывшемся окне «ЗВУКОЗАПИСЬ» нажмём на красный кружок, как только подготовим микрофон и текст. Нажимаем и говорим в микрофон: У ЛУКОМОРЬЯ ДУБ ЗЕЛЁНЫЙ… нажали на синий прямоугольник, а, нажав на треугольник, прослушали. Если всё в порядке, нажимаем ОК и эта запись будет уже на нашем слайде. Если не понравилась запись, удаляем её и снова записываем.

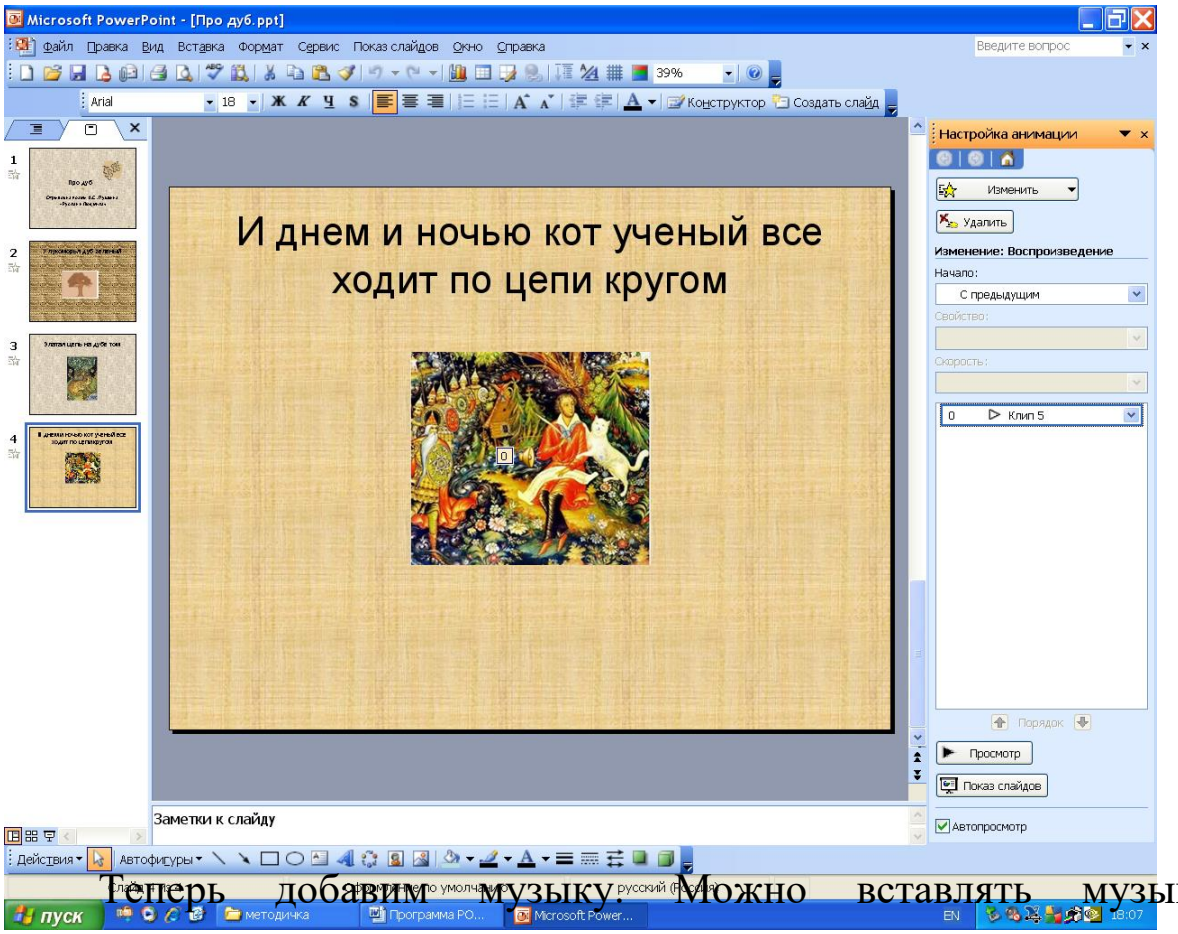
Теперь настроим наше звуковое сопровождение слайда. Заходим в меню ПОКАЗ СЛАЙДОВ – НАСТРОЙКА АНИМАЦИИ и справа, щёлкнув по клипу (выделив его), вместо мышки ставим «с предыдущим», а в параметрах эффектов клипа (нажать на галочку

справа слова КЛИП) поставить С НАЧАЛА и в зоне ЗАКОНЧИТЬ отметить «ПОСЛЕ ТЕКУЩЕГО СЛАЙДА».

Тоже самое проделать со слайдами №3 (записать звук: ЗЛАТАЯ ЦЕПЬ НА ДУБЕ ТОМ…)



и №4(записать звук: И ДНЁМ И НОЧЬЮ КОТ УЧЁНЫЙ ВСЁ ХОДИТ ПО ЦЕПИ КРУГОМ…).



Теперь добавим музыку. Можно вставлять музыкальные произведения в формате MP3 или WAV. Этот музыкальный файл должен быть обязательно в той же папке, где находится Ваша презентация.

Встаём на 1-й слайд, далее ВСТАВКА – ФИЛЬМЫ И ЗВУК – ЗВУК ИЗ ФАЙЛА, и указываем музыкальный файл (он может быть и с диска, но потом обязательно перепишите его в папку с презентацией). На возникшей табличке укажите АВТОМАТИЧЕСКИ.

Настроим музыкальное сопровождение: заходим в меню

ПОКАЗ СЛАЙДОВ – НАСТРОЙКА АНИМАЦИИ и справа щёлкаем мышкой по появившемуся клипу, выше вместо мышки выбираем с предыдущим, а в параметрах эффектов клипа указать с какого по какой номер слайда будет звучать музыка.

Переход слайдов настраиваем так же, как в предыдущей работе.

Практическая работа №3.

Создать презентацию «Прогноз погоды», состоящую из 6 слайдов.

На первом слайде – название презентации и пять кнопок с названием городов.

Затем создаём 5 слайдов: каждый слайд – это название города с картинкой – видом этого города и погодные условия: снег, дождь, солнце и значение температуры.

На примере этой презентации отработаем приёмы работы с гиперссылками и управляющими кнопками.

Вернёмся на первый слайд:

Рядом с названием города вставим кнопку. Для этого заходим в меню ПОКАЗ СЛАЙДОВ – УПРАВЛЯЮЩИЕ КНОПКИ и выбираем первую без рисунка. Как только мы растянем прямоугольник, возникнет табличка «настройка действия» и в ней мы должны указать, куда перейти по этой кнопке:

Можно не управляющую кнопку выбрать, а нарисовать фигурку из автофигур (как на рисунке), тогда выделить её и зайти в меню ПОКАЗ СЛАЙДОВ – НАСТРОЙКА ДЕЙСТВИЯ и далее также её настроить.

Аналогично настраиваем все кнопки.

Теперь перейдём на второй слайд и настроим кнопку возвращения на 1-й слайд (слайд – меню).

Внизу с помощью кнопки НАДПИСЬ (на панели рисования)напишем НАЧАЛО. Выделим надпись и настроим её как гиперссылку, т.е. зайдём в меню ПОКАЗ СЛАЙДОВ – НАСТРОЙКА ДЕЙСТВИЯ и выберем команду «перейти на первый слайд». Для удобства можно ещё добавить кнопочки (из автофигур) по краям слайда. Слева настроить её как «перейти на предыдущий слайд», а правую - «перейти на следующий слайд».

И так каждый слайд.

В конце запустите презентацию и проверьте работу всех кнопок.

Теперь подитожим полученные знания:

1) Разметка и дизайн слайдов: ФОРМАТ-РАЗМЕТКА СЛАЙДОВ или ФОРМАТ – ОФОРМЛЕНИЕ СЛАЙДА;

2) как добавлять слайды:

* из главного меню ВСТАВКА – СОЗДАТЬ СДАЙД

*из файлов: ВСТАВКА – СЛАЙДЫ ИЗ ФАЙЛОВ – ОБЗОР – указать какие файлы;

3) 3 режима (внизу слева 3 кнопки)

Обычный, сортировщик (можно менять местами слайды, копировать, удалять, добавлять), показ слайдов (показ во весь экран);

4) упаковка презентации: ФАЙЛ-ПОДГОТОВИТЬ ДЛЯ КОМПАКТ ДИСКА; такую презентацию можно показывать даже не имея РР на компьютере;

5) ввести текст на слайд: на панели рисование кнопка «НАДПИСЬ»;

6) фон слайда: ФОРМАТ – ФОН – ЦВЕТ или способы заливки, текстура и т.д.

7) переместить рисунок или объект: выделить объект – порядок – на задний план и т.п.

8) картинка: ВСТАВКА – РИСУНОК – ИЗ ФАЙЛА;

9) перекрасит картинку: прав. Кн. – ФОРМАТ ОБЪЕКТ – ПЕРЕКРАСИТЬ;

10) есть несколько рисунков, а фон отличается: вызываем панель»настройка изображений» и выбираем ПРОЗРАЧНЫЙ ЦВЕТ и щёлкаем по рисунку;

11) вставит видео фрагмент: ВСТАВКА – ФИЛЬМ ИЗ ФАЙЛА, отметить АВТОМАТИЧЕСКИ;

12) озвучивание: ВСТАВКА – ФИЛЬМ И ЗВУК – с *компакт диска* или *из файла* или *записать звук (голосовое озвучивание);*

13) настройка звука: ПОКАЗ СЛАЙДОВ - НАСТРОЙКА АНИМАЦИИ, справа название клипа – параметры эффектов-воспроизведение звука: сначала – после N-го слайда;

14) ссылка с картинки (или объекта):

Выделить картинку – прав.кнопка – изменить гиперссылку – связать с местом в документе – выбрать N-й слайд – можно добавить подсказку;

15) управляющие кнопки: выбрать кнопку – указать действие – указать СЛАЙД… ОК

У кнопки можно менять цвет, объем, тень (панель настройка изображения);

16) анимация объекта:

Выделить объект – ПОКАЗ СЛАЙДОВ – НАСТРОЙКА АНИМАЦИИ - ДОБАВИТЬ ЭФФЕКТ – НА ВХОД – выбираем эффект или пути перемещения.

17) переход анимированных слайдов:

В режиме сортировщик выбрать кнопку СМЕНА , выбрать эффект и настроить;

18) настройка показа слайдов:

ПОКАЗ СЛАЙДОВ – НАСТРОЙКА ПРЕЗЕНТАЦИИ – с какого по какой и можно зациклить.

19) распечатка презентации:

ФАЙЛ – ПЕЧАТЬ – СЛАЙДЫ либо ВЫДАЧА СЛАЙДОВ на странице.