Основы объектно-ориентированного визуального программирования Visual Basic (VB)

1. Ведение. Этапы разработки проекта
2. Классы объектов. Семейство объектов
3. Свойства объектов
4. Методы объектов
5. События
6. Графический интерфейс
7. Событийные процедуры
8. Цвет объектов
9. Тип, имя и значение переменной. Присваивание
0. Арифметические, строковые и логические выражения.
1. Функции Функции VB
   - Функции преобразования типов
   - Математические функции
   - Строковые функции
   - Функции ввода/вывода
   - Функции даты и времени

12. Алгоритмическая структура «ветвление»
13. Алгоритмическая структура «выбор»
14. Алгоритмическая структура «цикл»
- цикл со счетчиком
- цикл с предусловием
- цикл с постусловием

15. Графические возможности языка 15. Графические возможности языка VB
16. Общие процедуры. Область видимости процедур
17. Массивы. Типы и объявление массивов

    Заполнение массива

    Поиск в массивах

    Сортировка массива

    Двумерные массивы и вложенные циклы

18. Решение логических задач

Visual Basic – **система программирования** т. к. позволяет кодировать алгоритмы на этом языке и **среда проектирования** т.к. позволяет осуществлять визуальное конструирование графического интерфейса.

Результатом процесса программирования и проектирования является **проект (Project)**, который объединяет в себе **программный код** и **графический интерфейс**.

VB содержит **программу-транслятор**, поэтому проекты могут выполнятся в системе программирования, а также м.б. преобразованы в **приложения**, выполняемые в ОС Windows.

Этапы разработки проекта:

1. Создание графического интерфейса проекта
2. Установка значений свойств объектов графического интерфейса
3. Создание и редактирование программного кода
4. Сохранение проекта

Программный объект – основная единица в объектно-ориентированном программировании.

Программные объекты имеют

имя ***(существительные. Что?)***

обладают свойствами ***(прилагательные. Какой?),***

могут использовать методы ***(глаголы. Что делает?)*** **и**

реагируют на события.

Классы объектов являются «шаблонами», определяющими наборы ***свойств, методов и событий.*** По шаблонам создаются объекты. Основными классами объектов являются объекты, реализующие графический интерфейс.

Пример. В MS Word существует класс объектов **«документ» (Document)**, который обладает определенными наборами:

свойств: *имя* **(Name),** *полное имя*(**FullName**) и т.д.

методов: *открыть документ* (**Open**), *напечатать документ* (**PrintOut**), *сохранить документ* (**Save**) и т.д.

событий: *открытие документа* (**Document_New**), *закрытие документа* (**Document_Close**) и т.д.

Объект, созданный по «шаблону» класса объектов, является **экземпляром класса** и наследует весь набор свойств, методов и событий данного класса. Каждый экземпляр класса объектов имеет уникальное для данного класса имя.

Document (Проба.txt)

Различные экземпляры класса обладают одинаковым набором свойств, но **значения** свойств у них могут отличаться.

Некоторые свойства экземпляра класса **Document**

Имя объекта	Свойства объекта и их значение	
	(FullName) полное имя	Path (путь)
proba.doc	C:\Документы\proba.doc	C:\Документы\
proba.txt	C:\proba.txt	C:\

Основой для создания графического интерфейса является объект «форма». На основании ***класса объектов*** **Form** можно создавать ***экземпляры объектов*** «форма», которые получат имена **Form1, Form2, Form3** и т.д.

Семейство объектов представляет собой объект, содержащий несколько объектов, экземпляров одного класса.

Пример.

1. Все открытые в текущий момент в приложении Word документы образуют семейство, которое обозначается: **Documents().** Обращение к объекту, входящему в семейство, производится по имени или индексу.

Documents(«Проба.doc»).

2. Все символы, входящие в **выделенный фрагмент документа** (**объект Selection**) входят в **семейство Characters().** Обращение к символу производится по его индексу **Characters(7).**

Свойства объектов (Properties)

Объект.Свойство = ЗначениеСвойства

Selection.Characters(1).Bold = True в выделенном фрагменте текста (объект Selection) для первого символа (объект Characters(1)) установлено начертание полужирный (свойство Bold).

С помощью конструкции **With … End With** можно задавать сразу несколько свойств объекта:

```
With Объект
    .Свойство1 = ЗначениеСвойства1
    .Свойство2 = ЗначениеСвойства2
    .Свойство3 = ЗначениеСвойства3
End With
```

Пример.

```
For I=1 To 10
    With
Selection.Characters(I)
            .Bold = True
            .Italic = True
    End With
Next I
```

Придать выделенному фрагменту текста из 10 символов начертание «***полужирный курсив***»

Методы объектов (Methods). Для того чтобы объект выполнил какую-либо операцию, необходимо применить метод, которым он обладает. Многие методы имеют аргументы, которые позволяют задать параметры выполняемых действий.

Обращение к методу объекта:

Объект.Метод арг1:=значение, арг2:=значение

Пример. 1. Сохранение на диске открытого в приложении Word документа реализуется методом Save без аргументов:
Documents(“Проба.doc”).Save

2. Открытие документа Проба.doc:

Documents().Open FileName:=“C:\Документы\ Проба.doc

3. Печать 3-х первых страниц документа **Проба.doc**:

Documents(“Проба.doc”).PrintOut

Range:=wdPrintFromTo, From:=“1”, To:=“3”

События (Events) – **действие, распознаваемое объектом**. Событие может создаваться пользователем (щелчок мышью или нажатие клавиши) или быть результатом воздействия других программных объектов. В качестве реакции на события вызывается определенная процедура, которая может изменять значения свойств объекта, вызывать его методы и т.д.

Пример.

1. Объект **Document (*Документ*)** *реагирует на события* **Open (*Открытие*), *New (Создание), Close (Закрытие).***
2. *Объект* **Selection (*Выделенный фрагмент документа*)** реагирует на события **Cut (*Вырезка*), Copy (*Копирование*), Paste (*Вставка*), Delete (*Удаление*)** и т.д.

Графический интерфейс проекта представляет собой **форму**, на которой размещены **управляющие элементы**.

Форма – объект, представляющий собой окно на экране.

Управляющие элементы – объекты, являющиеся элементами графического интерфейса приложения и реагирующие на события, произведенные пользователем или программными объектами.

Форма *и* ***управляющие элементы*** *обладают определенными наборами* ***свойств, методов и событий****.*

Визуальное проектирование графического интерфейса состоит в том, что на форму с помощью мыши перемещаются и «рисуются» управляющие элементы.

Классы управляющих элементов (Controls):

Для ввода/вывода данных: ***Текстовые поля (*****TextBox*****), метки (*****Label*****), списки (*****ListBox*****).**

Для вывода графики ***Графические окна*** **(PictureBox*****).***

Для организации диалога ***командные кнопки (*****CommandButton*****), переключатели*** **(OptionsButton).**

Некоторые классы объектов, их свойства, методы и события

Класс объектов	Свойства	Методы	События
Form(форма) **UserForm**	**Name (Имя)** **Caption** (Надпись) **Font (Шрифт)** **Height (Высота)** **Width (Ширина)**	**Show (Показать)**	**Load (Загрузка)**
CommandButton ***(командные кнопки)***	**Name (Имя)** **Caption (Надпись)** **Font (Шрифт)** **Height (Высота)** **Width (Ширина**	**Move (Переместить)**	
TextBox ***(Текстовые поля)***	**Name (Имя)** **Text(Текст)** **Font (Шрифт)** **Height (Высота)** **Width (Ширина)**		**DblClick (Двойной щелчок)**

Событийные процедуры. Для каждого объекта можно запрограммировать отклик (событийную процедуру) – реакцию объекта на произошедшее событие, например:

- **Click** *(*щелчок по объекту мышью),
- **Resize** (изменение размера объекта),
- **Load** (загрузка объекта) и т.д.

Событийная процедура – подпрограмма, которая начинает выполняться после реализации определенного события.

Каждая процедура – отдельный программный модуль реализующий определенный алгоритм.

Private Sub Объект(1)_Событие()

Объект(2).Свойство=ЗначениеСвойства

Объете(3).Метод арг1:=знач, арг2:=знач

….

End Sub

Выравнивание формы с использованием событийной процедуры и метода Move. Выравнивание формы по центру экрана монитора можно производить автоматически при запуске проекта. В этот момент производится загрузка формы и активируется событие **Form_Load().**

Объект Screen (экран монитора) обладает **свойствами Screen.Width (ширина экрана)** и **Screen.Height (высота экрана**).

Форма frm1 также обладает **свойствами frm1.Width и frm1.Height**.

Форма будет размещаться в центре экрана, если:

left (левая координата верхнего левого угла формы) будет равна:

(Screen.Width – frm1.Width)/2

top (верхняя координата верхнего левого угла формы) будет равна:

(Screen.Height – frm1.Height)/2

Для перемещения объектов в определенное место используется метод Move.

Формат метода: object.Move left,top

Код событийной процедуры выравнивания формы по центру экрана монитора:

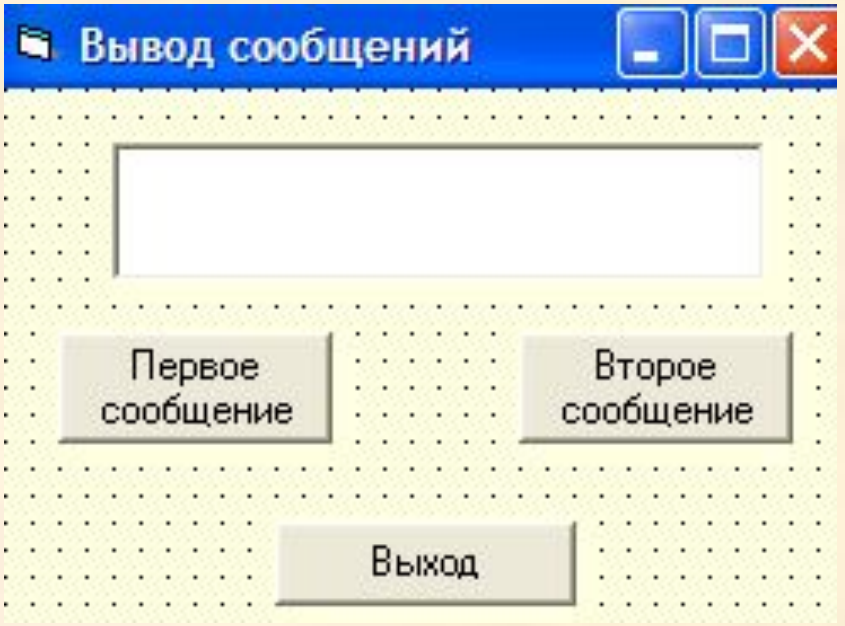
```
Private Sub Form_Load()
frm1.Move(Screen.Width – frm1.Width)/2, (Screen.Height – frm1.Height)/2
End Sub
```

Задание 4.7. Создать проект «Вывод сообщения», в котором на форму выводится текстовое сообщение «Первое задание выполнено!» с помощью метки и текстового поля. Выход из программы щелчком по *Exit.*

```
Private Sub Form_Load()
frm1.Move (Screen.Width - frm1.Width) / 2, (Screen.Height - frm1.Height) / 2
End Sub
Private Sub cmd1_Click()
End
End Sub
```

Задание 4.8 **Создать проект «Вывод сообщений», в котором каждый из двух различных вариантов текста выводится в текстовое поле TextBox щелчком по одной из двух кнопок . Выход из программы щелчком на третьей кнопке.**

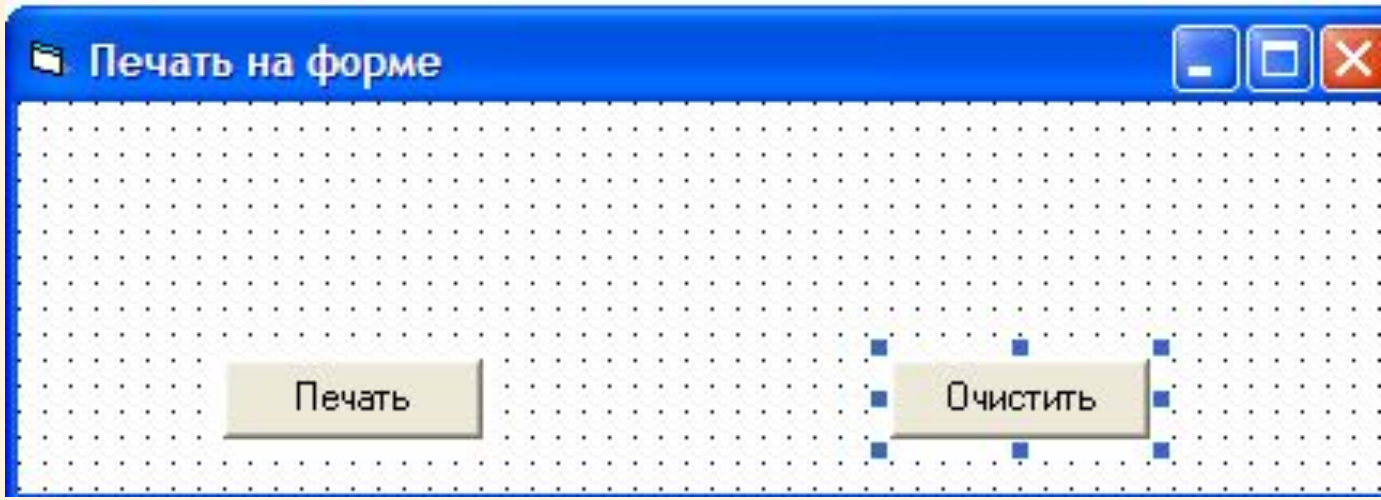
```
Private Sub Cmd1_Click()
Txt1.Text = «Первое сообщение"
End Sub

Private Sub Cmd2_Click()
Txt1.Text = «Второе сообщение"
End Sub

Private Sub Cmd3_Click()
End
End Sub
```




```
Private Sub Form_Load()
Frm1.Move (Screen.Width - Frm1.Width) / 2, (Screen.Height - Frm1.Height) / 2
End Sub
```

Задание 4.9. «Печать по форме»

```
Private Sub cmd1_Click()
With Font
.Name = "Times New Roman"
.Size = 18
.Italic = True
End With
frm1.ForeColor = QBColor(12)
frm1.Print "Times New Roman, 18, курсив, красный"

With Font
.Name = "Arial"
.Size = 14
.Italic = False
.Underline = True
End With
frm1.ForeColor = QBColor(9)
frm1.Print "Arial, 14, подчеркнутый, синий"
```



```
With Font
.Name = "Courier New"
.Size = 12
.Bold = True
.Underline = False
End With
frm1.ForeColor = QBColor(2)
frm1.Print "Courier New, 12, полужирный, зеленый"
End Sub
Private Sub cmd2_Click()
frm1.Cls
End Sub
```

Тип, имя и значение переменной

Переменные используются для хранения и обработки данных в программах.

Переменная представлена **именем** и служит для обращения к данным определенного **типа**, конкретные **значения** которых хранятся в ячейках оперативной памяти.

Имя переменной (идентификатор) состоит из лат. и рус. Букв, начинается с буквы и не должно включать знак «.». Длина имени ≤ 255 символов.

Тип переменных определяется типом данных, которые могут быть значением переменных.

Тип переменной	Возможные значения	Объем памяти	Приставка к имени
Byte	целые числа от 0 до 255	1 байт	**byt**
Integer	целые числа [-32768; +32767]	2 байта	**int**
Long	целые числа от [-2147483648; +2147483647]	4 байта	**lng**
Single	Десятичные числа одинарной точности (7 знач. цифр) $[-1{,}4\cdot10^{-45}; 3{,}4\cdot10^{38}]$	4 байта	**sng**
Double	Десятичные числа двойной точности (15 знач. цифр) $[-5\cdot10^{-324}; 1.7\cdot10^{308}]$	8 байтов	**dbl**

Тип переменной	Возможные значения	Объем памяти	Приставка к имени
Boolean	Логические значения - True и False	2 байта	**bln**
String	Строка символов	1 символ – 1 байт	**str**
Currency	Число в денежном формате	8 байтов	**cur**
Date	дата 1 января 100 г. до 31 декабря 9999 года	8 байтов	**dtm**
Object	Ссылка на любой объект	4 байта	**obj**
Variant	Любые значения	>16 байтов	**vnt**

Объявление типа переменных:

Dim Имя переменной [As ТипПеременной]

По умолчанию – универсальный тип Variant

Переменная может получит или изменить значение с помощью **оператора присваивания:**

[Let] Имя переменной = Выражение

Пример.

Dim A As Byte, **B** As Integer, **C** As Single, **D** As String, **G** As Boolean

bytA=255

intB=-32768

sngC=3.14

strD=“информатика”

blnG= True

Переменные, значения которых не меняются в процессе выполнения программы, называются константами.

Const ИмяКонстанты [As Тип] =ЗначениеКонстанты

Пример. Const PI = 3.1415

Const PI **As Long** = 3 ' PI = 3, PI имеет тип Long

Упр. 1. Определите тип переменной

bytЧисло - целое число от 0 до 255

intC - целое число [-32768; +32767]

lngЧисло1- целое число [-2147483648; +2147483647]

sngC - Десятичные числа одинарной точности (7 знач. цифр) [-1,4·10-45; 3,4·1038]

blnA - Десятичные числа одинарной точности (7 знач. цифр) [-1,4·10-45; 3,4·1038]

strСтрока1 - Строка символов

dtmTime - Логическое значение

vntD - Любые значение

Арифметические выражения содержат числовые переменные, числовые функции, знаки арифметических операций, скобки ().

Арифметические операции

Операция	Пояснение	Пример	Приоритет операций:
+	Сложение	2+3	^
-	Вычитание	5-2	/
*	Умножение	2*3	\
/	Деление	7/2 (результат 3.5)	MOD
\	Деление нацело	7\2 (результат 3)	*
MOD	Остаток от деления	7 MOD 2 (остаток 1)	+
^	Возведение в степень	2^3 (результат 8)	-

Строковые выражения включают переменные строкового типа, строки (последовательности символов, строковые функции).

Конкатенация (+ или &) – операция сложения строк или строковых переменных

Примеры:

```
Dim Number, Var1, Var2
Var1 = 34
Var2 = 6
Number = Var1 + Var2 ' Возвратит 40.
Var1 = "34"
Var2 = "6" ' Инициализируем переменные со строками
Number = Var1 + Var2 ' Возвратит "346" (произошла
' конкатенация, а не сложение!).
```

Логические выражения включают **логические переменные (blnA As Bolean)**, **логические значения (True, False)**, результаты операций сравнения чисел и строк, а также **логических операций (AND, OR, NOT)/**

Пример. 5>3= True

"A"="B"= False

(5>3) And ("A"="B") = False

(5>3) Or ("A"="B") = True

Not (5>3) = False

Операции сравнения:

< меньше
> больше
<= меньше или равно
>= больше или равно
= равно
<> не равно

Логические операции:

And - логическое умножение
Eqv - эквивалентность
Imp - импликация
Not - логическое отрицание
Or - логическое сложение
Xor - логическое исключающее сложение

Функции в языке Visual Basic

ИмяФункции (СписокАргументов)

I. Функции преобразования типов данных

1. Val(Строка$) - Превращает строку в число. **Value** - величина

Пример:

```
Dim V
V = Val("2457") ' Возвратит 2457.
V = Val(" 2 45 7") ' Возвратит 2457.
V = Val("24 and 57") ' Возвратит 24.
V = Val("") ' Возвратит 0.
V = Val("laja") ' Возвратит 0.
V = Val(«&O3720") ' Возвратит 2000.
V = Val("&H7D0") ' Возвратит 2000.
```

Применяется для преобразования строкового значения свойств **Text** текстовых полей в число, которое затем используется в арифметических выражениях

```
Private Sub CmndMinus_Click()
Txt3.Text = Val(Txt1.Text) - Val(Txt2.Text)
End Sub
```

2. Str (Число) – преобразует десятичное число в строку

Пример:

```
Dim S
S = Str(459) ' Возвратит "459".
S = Str(-459.65) ' Возвратит "-459.65".
S = Str(459.001) ' Возвратит "459.001".
```

3. Hex(Число) -. преобразует десятичное число в шестнадцатеричное в строковой форме.

Пример:

```
Dim H
H = Hex(5) ' Возвратит 5.
H = Hex(10) ' Возвратит A.
H = Hex(459) ' Возвратит 1CB.
```

4. Oct(Число) - преобразует десятичное число в восьмеричное в строковой форме.

```
Dim O
O = Oct(4) ' Возвратит 4.
O = Oct(8) ' Возвратит 10.
O = Oct(459) ' Возвратит 713.
```

5. Asc(Строка$) – преобразует строку в числовой код первого символа

Пример:

```
Dim N
N = Asc("A") ' Возвратит 65.
N = Asc("a") ' Возвратит 97.
N = Asc("Apple") ' Возвратит 65.
```

6. Chr (ЧислоКод) - возвращает символ, соответствующий определённому коду. Эта функция является обратной Asc.

Пример:

```
Dim C
C = Chr(65) ' Возвратит A.
C = Chr(97) ' Возвратит a.
C = Chr(62) ' Возвратит >.
C = Chr(37) ' Возвратит %.
```

II. Математические функции

1. **Sin(Число) - вычисляет синус Числа**
2. **Cos(Число) - вычисляет косинус Числа.**
3. **Tan(Число) - вычисляет тангенс Числа**
4. **Atn (Число) - вычисляет арктангенс Числа.**

Пример:

```
Dim A, C, S, D, pi
A = 1.3 ' Определяем угол в радианах
C = 1 / Sin(A) ' Вычисляем косеконс
S = 1 / Cos(A) ' Вычисляем секонс
C = 1 / Tan(A) ' Вычисляем котангенс
pi = 4 * Atn(1) ' Вычисляет значение числа pi.
```

5. Sqr(Число) - Возвращает корень Числа

Пример:

```
Dim S
S = Sqr(4) 'Возвратит 2.
S = Sqr(23) 'Возвратит 4.79583152331272.
S = Sqr(0) 'Возвратит 0.
S = Sqr(-4) 'Генерирует ошибку (корень из отрицательного числа).
```

6. Log(Число) - вычисляет натуральный логарифм Числа (по основанию **e**). (Возвращает тип Double). **e=2.71**

Для того, чтобы получить логарифм по основанию n нужно произвести следующее вычисление:

Logn(x) = Log(x) / Log(n)

7. Exp (Число) - Возвращает экспоненту Числа.

Пример:

```
Form1.Caption = Exp(1) 'Отобразит на Caption число e (т.е. e в степени 1)
```

8. **Rnd[(Число)] - Генерирует случайное число от 0 до 1.**

Для генерации случайного числа **X** в интервале **[A,B]** используют формулу:
X=RND*(B-A) +A или
X=RND*(B-A+1) +A (включает крайние знач. интервала [A,B])

Каждый раз при запуске программы, если не переустанавливается база генератора случайных чисел, формируется одна и та же последовательность чисел.
RANDOMIZE (*база*) - переустанавливаем базу генератора случайных чисел.

Пример:

```
Dim V
RANDOMIZE TIMER
V = Int((6 * Rnd) + 1) ' Генерирует случайное число от 1 до 6
```

III. Строковые функции

1. **LEN(Строка$)** – **определяет длину Строки$**
2. **Left(Строка$,n)** - **вырезает n символов, начиная с первого символа до указанного номера**
3. **Right(Строка$,n)** - **вырезает n символов из Строки$, начиная справа**
4. **Mid(Строка$,n,k)** - **вырезка из Строки$ с n-ой позиции k символов**

Пример:

```
Dim strA, strL, strR, strS As String, intN As Integtr
strA = “Школа" ' Определяем строку
intN=Len(strA) ' Определяем длину строки
strL = Left(«Школа», 1) ' Возвратит «Ш»
strL = Left(strA, 3) ' Возвратит «Шко»
strL = Left(«Школа», 20) ' Возвратит «Школа»
strR= Right(strA, 1) ' Возвратит «а»
strR = Right(«Школа», 3) ' Возвратит «ола»
strS=Mid(«Школа»,2,3) ' Возвратит «кол»
```

IV. Функции ввода и вывода

1. InputBox (Приглашение$, Заголовок$,[ПоУмолчанию$]) – вводит данные с помощью диалоговой панели ввода (Окно Ввода)**.**

Пример:

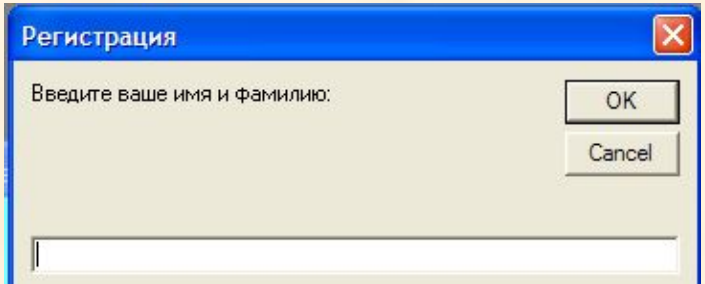
Выводим окно с заголовком "ВНИМАНИЕ", запросом "Введите пароль", в окошке будет выделенный текст "Значение_по_умолчанию". Координаты появления окна - 100,100pix.

```
Form1.Caption = InputBox("Введите пароль", _
"ВНИМАНИЕ", "Значение_по_умолчанию", 100, 100)
```

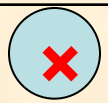
2. **MsgBox(Сообщение$[,ЧисКод1+ЧисКод2][, Заголовок$]) - выводит на экран окно сообщения (Message Box) которое будет ждать клика на одной из кнопок. Возвращает Число, по которому можно определить какую кнопку нажал пользователь.**
Значение ЧисКод1 определяет вид пиктограммы, которая помещается в панель сообщений.
Значение ЧисКод2 определяет набор кнопок, размещаемых на панели

С помощью одного числа, являющегося суммой чисел **ЧисКод1+ЧисКод2,** можно одновременно устанавливать определенную пиктограмму и определенную комбинацию кнопок, размещенных на панели сообщений.

Пример. Число **36=32** (код пиктограммы ***«Вопрос»*****+4**(код комбинации кнопок ***Да,Нет***).



Значения ЧисКод1 и ЧисКод2, определяющие вид панели сообщений

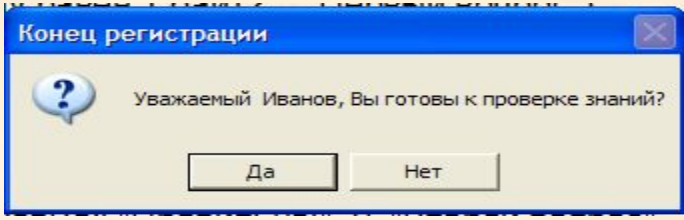
ЧисКод1	Пиктограмма
16	
32	
48	
64	

ЧисКод2	Набор кнопок
0	*ОК*
1	*ОК, Отмена*
2	*Стоп, Повтор, Пропустить*
3	*Да, Нет, Отмена*
4	*Да, Нет*
5	*Повтор, Отмена*

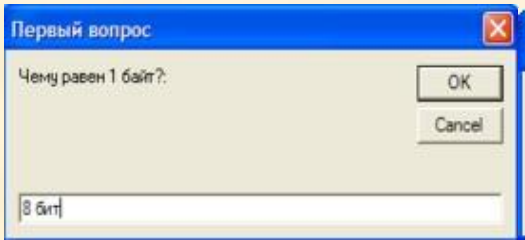
Значения функции MsgBox	
Нажатая кнопка	**Знач. ф-и**
ОК	**1**
Отмена	**2**
Стоп	**3**
Повтор	**4**
Пропустить	**5**
Да	**6**
Нет	**7**

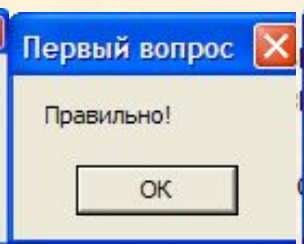
Проект «Проверка знаний»

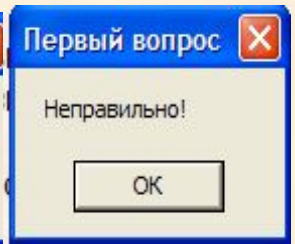
```
Dim bytB, bytN As Byte, strA, strB As String
Sub cmd1_Click()
strA = InputBox("Введите ваше имя и фамилию:", "Регистрация")
bytB = MsgBox("Уважаемый  " + strA + ", Вы готовы к проверке знаний?", 36,
"Конец регистрации")
```

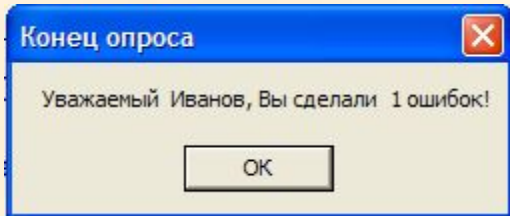



```
If bytB = 7 Then End

strC = InputBox("Чему равен 1 байт?:", "Первый вопрос")

If strC = "8 бит" Then MsgBox "Правильно!", 0, "Первый вопрос" _
Else MsgBox "Неправильно!", 0, "Первый вопрос": bytN = bytN + 1

strC = InputBox("Переведите десятичное число 5 в двоичную систему
счисления:", "Второй вопрос")

If strC = "101" Then MsgBox "Правильно!", 0, "Второй вопрос" _
Else MsgBox "Неправильно!", 0, "Первый вопрос": bytN = bytN + 1
MsgBox "Уважаемый  " + strA + ", Вы сделали " + Str(bytN) + " ошибок!", 0, "Конец
опроса"

End Sub
```









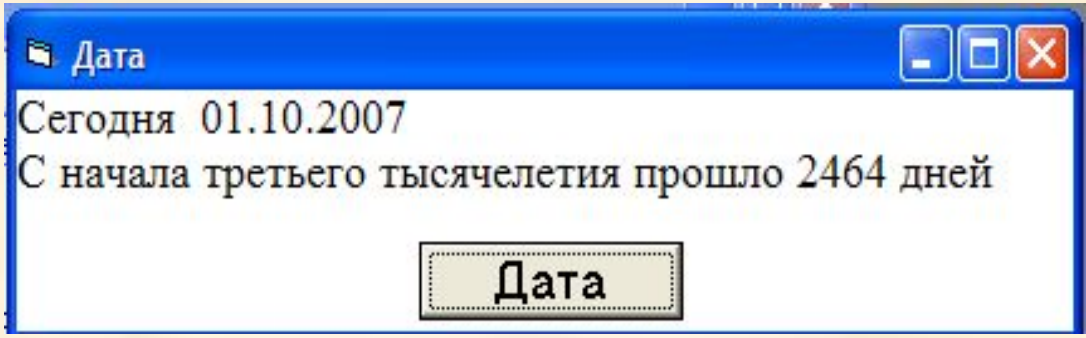
Функции даты и времени

Date - возвращает текущую дату.

Пример:

```
Dim MyDate
MyDate = Date ' MyDate содержит текущую системную дату.
```

Проект «Дата»




```
Dim dtmA, dtmB As Date
Sub cmd1_Click()
dtmA = Date
dtmB = #1/1/2001#
Print "Сегодня  "; dtmA
Print "С начала третьего тысячелетия прошло";
dtmA - dtmB; "дней"
End Sub
```

Функция Time$ - Возвращает String значение, содержащее текущее системное время, которое можно вывести в текстовое поле.

Значение времени представляется #Часы:Минуты: Секунды#

Пример:

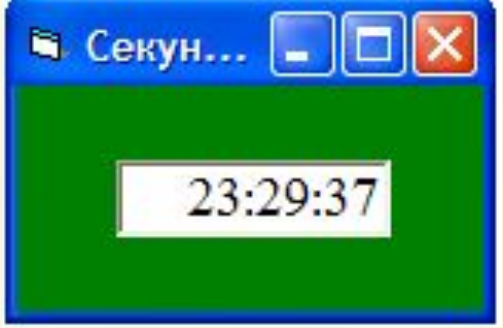
```
Dim MyTime
MyTime = Time ' Возвращает текущее системное время.
```

Timer - Возвращает Single, содержащее количество секунд, прошедших после полуночи. Не отображается на форме в процессе выполнения проекта, проверяет показание системных часов по событию Timer.

Свойство **Interval** (значения [0;65536] задает периодичность события Timer. Чтобы событие Timer происходило каждую секунду, свойству Interval небходимо присвоить значение 1000.

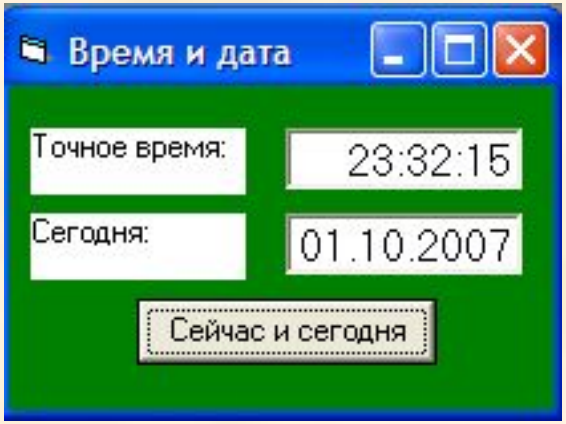
Проект «Секундомер»

```
Sub tmr1_Timer()
txtTime.Text = Time$
End Sub
```



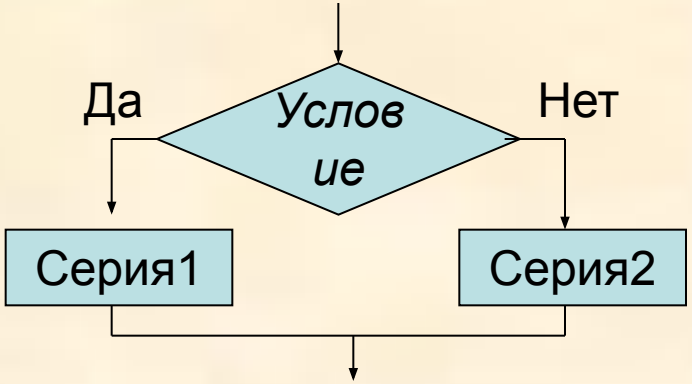
4.21.

```
Sub cmdTD_Click()
txtTime.Text = Time$
txtData.Text = Date
End Sub
```



Алгоритмическая структура «ветвление» - в зависимости от истинности или *ложности Условия* выполняется одна или другая *Серия* команд

Блок-схема:



Условие –логическое выражение

Серия1, Серия2 – набор команд

Языки программирования VB и VBA:

Многострочная запись:

```
If УсловиеThen
    Серия1
[Else
    Серия2 ]
End If
```

Однострочная запись:

1) If *Условие*Then Серия1 [Else Серия2]

2) If *Условие*_
Then Серия1_
[Else Серия2]

Пример 1:

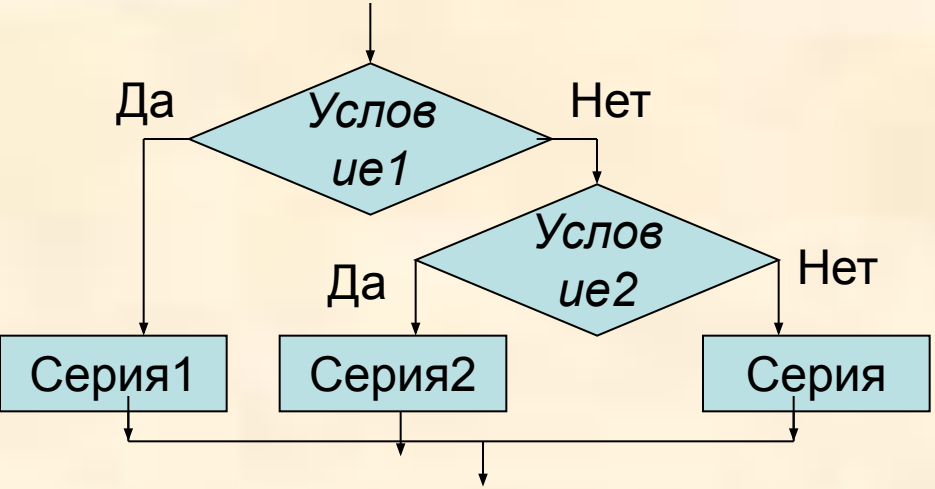
```
If (a = b) And (c <> d) Then
    b = d
    a = 20
End If
```

Пример 2:

```
If (a = b) Or (c <> d) Then
    b = d
    a = 20
Else
    c = d
End If
```

Алгоритмическая структура «выбор» - выполняется одна из нескольких последовательностей команд при истинности соответствующего условия

Блок-схема:



Языки программирования VB и VBA:

```
Select Case Выражение
Case Условие 1
    Серия 1
Case Условие 2
    Серия 2
Case Else
    Серия
End Select
```

Пример: В зависимости от значения переменной iTest, строковой переменной strResult присваиваются различные значения

```
Select Case iTest
Case 1
    strResult = "iTest = 1"
Case 2, 3, 4
    strResult = "iTest = 2, 3 или 4"
Case 5 To 9
    strResult = "iTest находится в диапазоне от 5 до 9"
Case iTest < 0
    strResult = "iTest меньше 0"
Case Is > 9
    strResult = "iTest больше 9"
Case Else
    strResult = "iTest равно 0"
End Select
```

Алгоритмическая структура «цикл» - организация повторения действий, пока верно некоторое условие.

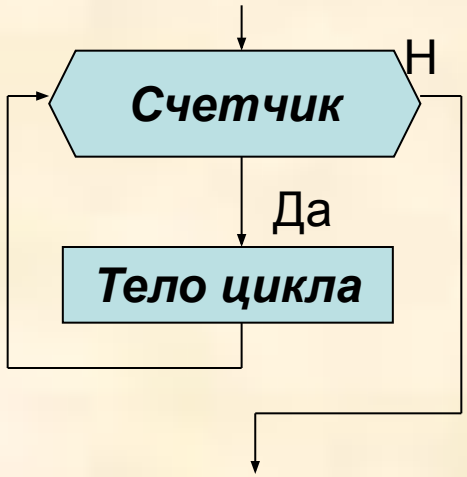
Тело цикла – набор повторяемых действий.

Типы циклов:

1. **Цикл со счетчиком**
2. **Цикл по условию**

1. **Цикл со счетчиком –** число повторений заранее известно

Блок-схема:



Языки программирования VB и VBA:

```
For Счетчик=НачЗнач To КонЗнач [Step шаг]
Тело цикла
Next
```

Пример:

В этом примере всем элементам массива iArray присваивается значение 5.

```
Dim c As Integer
Dim iArray(10) As Integer
For c = 0 To 10
    iArray(c) = 5
Next c
```

Что напечатает программа?

1) FOR X=0 TO 10 STEP 2

   PRINT X

   NEXT X

Ответ:

0

2

4

6

8

10

2) FOR I=10 TO 1 STEP -1

   PRINT I;

   NEXT

Ответ: 10 9 8 7 6 5 4 3 2 1

3) FOR I=1 TO 100 : NEXT **Ответ:** пустой цикл

4) X=2

   Y=3

   FOR K=X+Y TO X*Y

   PRINT K

   NEXT K

ОТВЕТ:

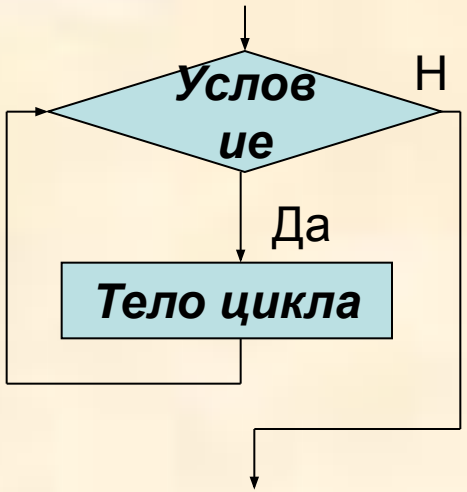
5

6

2. Циклы по условию:

а) Цикл с предусловием

Блок-схема:



Языки программирования VB и VBA:

Пока *Условие* (условие продолжения цикла) *верно* – повторяй:

```
Do While Условие
Тело цикла
Loop
```

Пример 1.

```
Do While 2 > 1
    Degub.Print "Вечный цикл"
Loop
```

Пока *Условие* (условие завершения цикла) *ложно* – повторяй:

```
Do Until Условие
Тело цикла
Loop
```

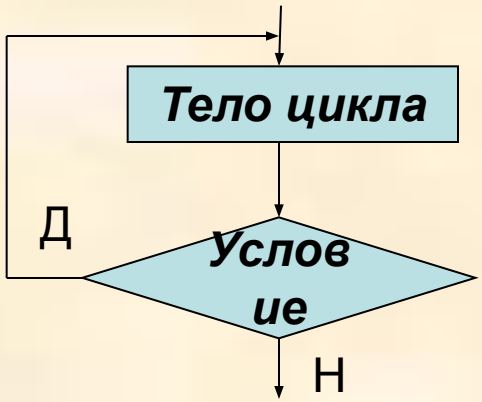
Пример 2.

Если у вас случаянно получился такой цикл, то выйти из него можно при нажатии Ctrl+Break. Но это работает только в среде разработки.

```
Dim n As Integer
n = 100
Do While n >= 0
    n = n - 1
    Debug.Print n
Loop
```

а) Цикл с постусловием (выполняется всегда, хотя бы раз)

Блок-схема:



Языки программирования VB и VBA:

Пока *Условие* (условие продолжения цикла) *верно* – повторяй:

```
Do
Тело цикла
Loop While Условие
```

Пример1:

```
Dim n As Integer
n = 100
Do
    n = n - 1
    Debug.Print n
Loop Until n < 11
```

Пока *Условие* (условие завершения цикла) *ложно* – повторяй:

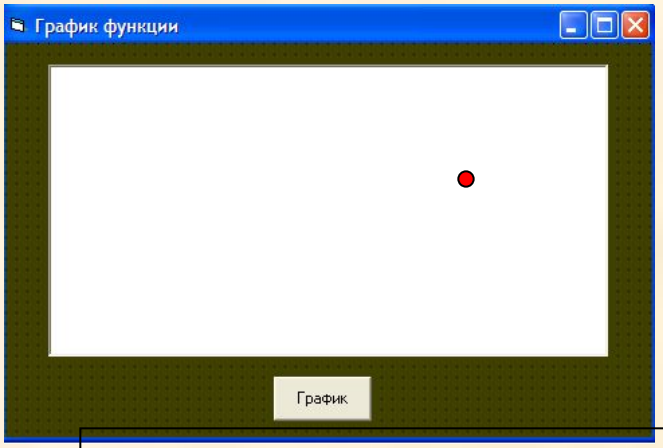
```
Do
Тело цикла
Loop Until Условие
```

Графические возможности языка VB

На формах (Form) или в **графических окнах (PictureBox)** можно **рисовать графические примитивы** с использованием **графических методов:**

Обращение к методу объекта:
Объект.Метод арг1:=значение, арг2:=значение

1. Scale – задает систему координат и масштаб для формы или графического окна:
object.Scale (x1,y1)-(x2,y2)
Пример. 'Задание масштаба
picGraph.Scale (-10, 2)-(10, -2)



2. Pset – Установка точки с заданными координатами и цветом:
object.Pset (x,y) [,Color]
Пример. picGraph.Pset (5, 1) vbRed
По умолчанию – черный цвет

Если графический метод применяется к объекту «форма», то при его записи имя объекта можно опустить

Цвет объектов (Color)

1. QBColor(color) - Возвращает значение Long, содержащее цвет, заданный номером от 0 до 15.

Пример:

```
MyForm.BackColor = QBColor(6)
```

Таблица цветов:

Номер	Цвет	Номер	Цвет
0	Чёрный	8	Серый
1	Синий	9	Светло-синий
2	Зелёный	10	Светло-зелёный
3	Циан	11	Светлый циан
4	Красный	12	Светло-красный
5	Мажента	13	Светлая мажента
6	Жёлтый	14	Светло-жёлтый
7	Белый	15	Ярко-белый

2. RGB(Red, Green, Blue) - возвращает Long значение, содержащее цвет, заданный тремя компонентами **R,G, B**.

Цвет задается числом по формуле:

$bytRed+256 \cdot bytGreen+256^2 \cdot bytBlue$

Пример:

```
MyForm.BackColor = RGB(255,0,0) 'цвет формы изменится на красный
MyForm.BackColor = RGB(255,255,255) 'цвет формы изменится на белый
```

3. С помощью 8 констант, определяющих цвет:

vbBlack - черный vbRed - красный

vbBlue - синий vbMagenta - сиреневый

vbGreen - зеленый vbYellow - желтый

vbWhite - белый

3. Line – рисование линии, прямоугольника или закрашенного прямоугольника заданного цвета:

Object.Line (x1,y1) – (x2,y2) [,color] [,B][F]

(x1,y1), (x2,y2) – координаты концов линии (левого верхнего и правого нижнего угла прямоугольника)

B(Box) – рисует прямоугольник

F(Fill) – закрашивает прямоугольник

4. Circle - рисование окружности, овала или дуги

Object.Circle (X,Y), radius [,color, start, end, aspect]

(X,Y) – координаты центра окружности

radius – радиус

Color – цвет окружности

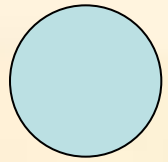
start, end – начальный и конечный угол дуги

Aspect – коэффициент сжатия

Пример:

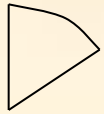
P=3.14

CIRCLE (250,120),50 - окружность с центром (250,120) радиусом 50



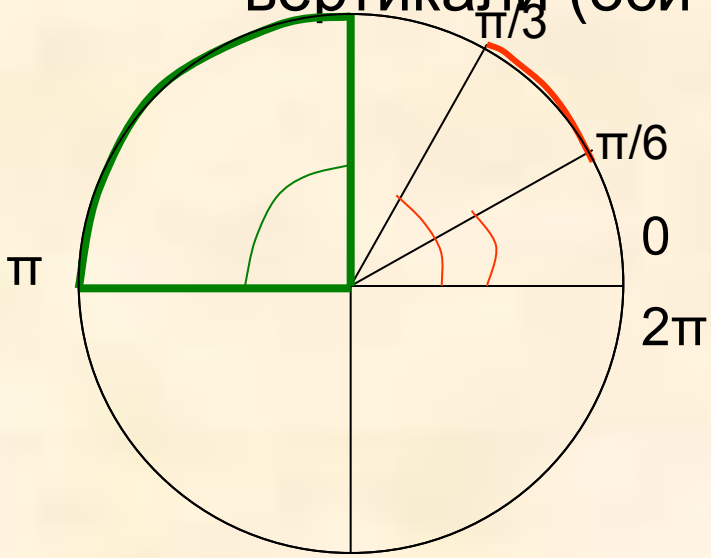
CIRCLE (250,120),60,4,P/6,P/3 - дуга(часть окружности между указанными углами)

CIRCLE (250,120),70,2, - P/2,-P - Сектор



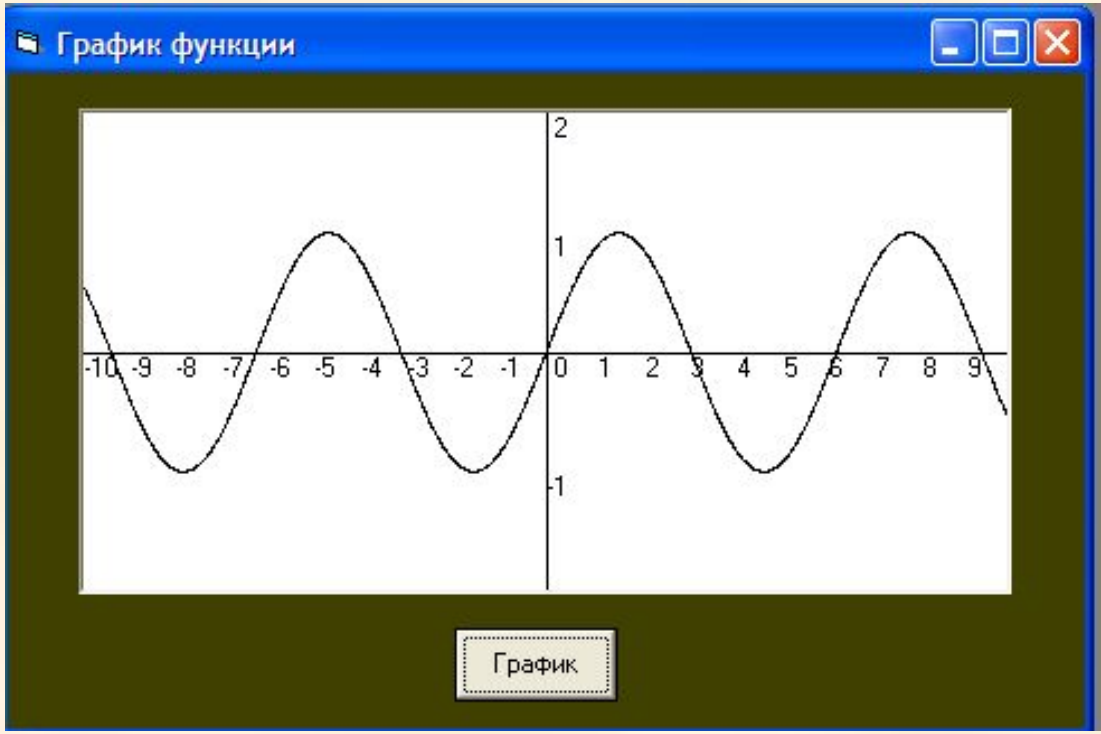
CIRCLE (250,120),80,,,,3 - окружность, сжатая по горизонтали (оси X)

CIRCLE (250,120),80,,,,0.6 - окружность, сжатая по вертикали (оси Y)



Проект «Построение графика функции»

```
Dim sngX As Single, intI As Integer
Private Sub cmd1_Click()
picGraph.Scale (-10, 2)-(10, -2) 'Задание масштаба

For sngX = -10 To 10 Step 0.01 'Построение графика
picGraph.PSet (sngX, Sin(sngX))
Next sngX

picGraph.Line (-10, 0)-(10, 0) 'Ось X
For intI = -10 To 10
picGraph.PSet (intI, 0)
picGraph.Print intI
Next intI
'Ось Y
picGraph.Line (0, 2)-(0, -2)
For intI = -2 To 2
picGraph.PSet (0, intI)
picGraph.Print intI
Next intI
End Sub
```



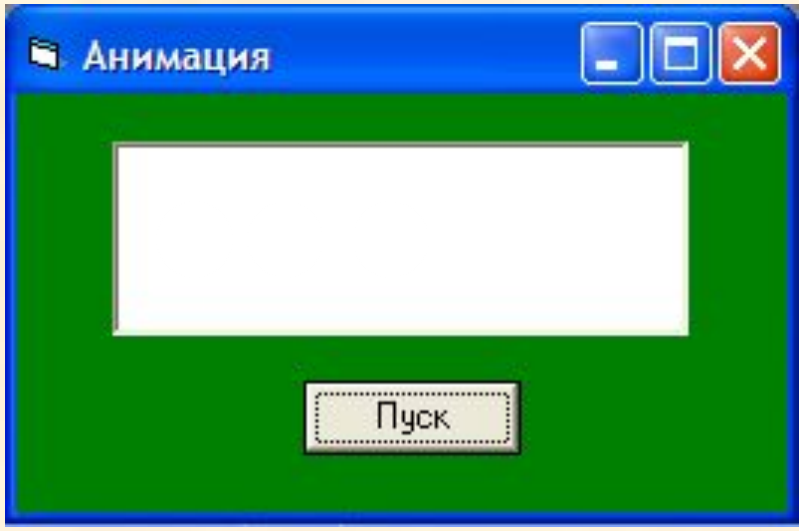
Анимация – иллюзия движения объекта на экране

Алгоритм:

1. Создание изображения на экране в точке (X,Y)
2. Пауза для фиксации изображения
3. Стереть объект (нарисовать цветом фона)
4. Изменить положение объекта
5. Повторить пункты 1-4

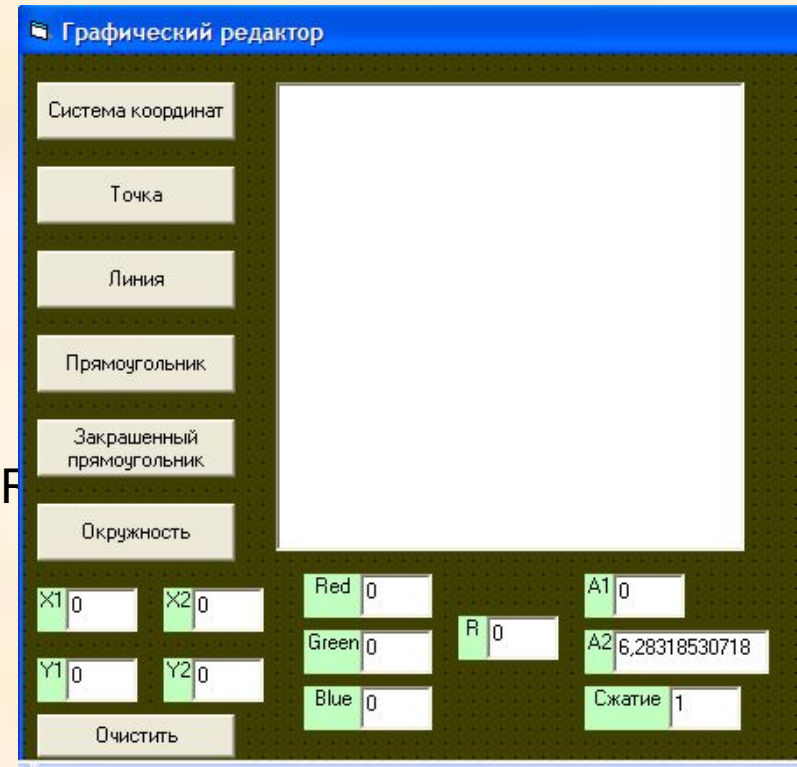
Проект «Анимация»

```
Dim intX As Integer, lngI As Long
Private Sub cmdStart_Click()
‘Масштаб
picAnim.Scale (-10, 10)-(10, -10)
‘Анимация
For intX = -10 To 10
    ‘Рисование
     picAnim.FillColor = vbBlue
     picAnim.Circle (intX, 0), 1, vbBlue
      ‘Задержка стирания
           For lngI = 1 To 10000000
           Next lngI
     ‘Стирание
     picAnim.FillColor = vbWhite
     picAnim.Circle (intX, 0), 1, vbWhite
Next intX
End Sub
```



4.2 Проект «Графический редактор»

```
Dim X1, X2, Y1, Y2, Rad, R, G, B  As Byte, A1, A2, K As Single, C As Long
Private Sub cmdScale_Click()
‘ Система координат
picPaint.Scale (0, 100)-(100, 0)
End Sub
Private Sub cmdPoint_Click()
‘ Точка
X1 = Val(txtX1)
Y1 = Val(txtY1)
R = Val(txtR): G = Val(txtG): B = Val(txtB): C = RGB(R, G, B)
picPaint.PSet (X1, Y1), C
End Sub
Private Sub cmdLine_Click()
‘ Линия
X1 = Val(txtX1)
Y1 = Val(txtY1)
X2 = Val(txtX2)
Y2 = Val(txtY2)
R = Val(txtR): G = Val(txtG): B = Val(txtB): C = RGB(R
picPaint.Line (X1, Y1)-(X2, Y2), C
End Sub
```



```
Private Sub cmdLineB_Click()
‘ Прямоугольник
X1 = Val(txtX1)
Y1 = Val(txtY1)
X2 = Val(txtX2)
Y2 = Val(txtY2)
R = Val(txtR): G = Val(txtG): B = Val(txtB): C = RGB(R, G, B)
picPaint.Line (X1, Y1)-(X2, Y2), C, B
End Sub
Private Sub cmdLineBF_Click()
‘ Закрашенный прямоугольник
X1 = Val(txtX1)
Y1 = Val(txtY1)
X2 = Val(txtX2)
Y2 = Val(txtY2)
R = Val(txtR): G = Val(txtG): B = Val(txtB): C = RGB(R, G, B)
picPaint.Line (X1, Y1)-(X2, Y2), C, BF
End Sub
```

```
Private Sub cmdCircle_Click()
‘ Окружность
X1 = Val(txtX1)
Y1 = Val(txtY1)
Rad = Val(txtRad)
A1 = Val(txtA1)
A2 = Val(txtA2)
K = Val(txtK)
R = Val(txtR): G = Val(txtG): B = Val(txtB): C = RGB(R, G, B)
picPaint.Circle (X1, Y1), Rad, C, A1, A2, K
End Sub
Private Sub txtCls_Click()
‘ Очистка
picPaint.Cls
End Sub
```

Вспомогательные алгоритмы. Подпрограммы

Алгоритм «Дежурство»

1. Прийти 7 часам
2. Накрыть столы к завтраку
3. Дождаться окончания завтрака
4. Убрать со столов посуду
5. Вытереть столы
6. Расставить стулья
7. Уйти
8. Прийти к 13 часам
9. Накрыть столы к обеду
10. Дождаться окончания обеда
11. Убрать со столов посуду
12. Вытереть столы
13. Расставить стулья
14. Уйти
15. Прийти 18 часам
16. Накрыть столы к ужину
17. Дождаться окончания ужина
18. Убрать со столов посуду
19. Вытереть столы
20. Расставить стулья
21. Уйти
22. Конец

Алгоритм «Дежурство»

(основной)

1. Прийти 7 часам
2. Накрыть столы к завтраку
3. Дождаться окончания завтрака
4. Выполнить «Обязанности»
5. Прийти к 13 часам
6. Накрыть столы к обеду
7. Дождаться окончания обеда
8. Выполнить «Обязанности»
9. Прийти 18 часам
10. Накрыть столы к ужину
11. Дождаться окончания ужина
12. Выполнить «Обязанности»
13. Конец

Алгоритм «Обязанности»

(вспомогательный)

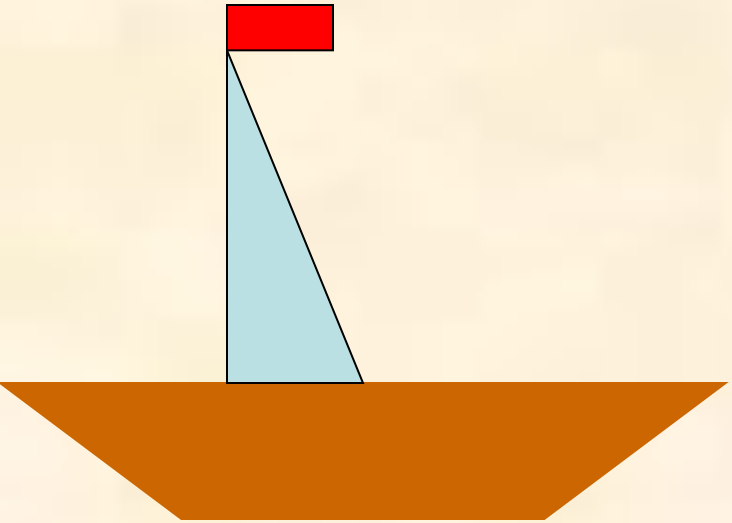
1. Убрать со столов посуду
2. Вытереть столы
3. Расставить стулья
4. Уйти

В процедурных языках:

Вспомогательный алгоритм – алгоритм, снабженный заголовком, позволяющим вызывать его из других алгоритмов

Подпрограмма – вспомогательный алгоритм, записанный на языке программирования.

Вспомогательные алгоритмы и подпрограммы позволяют конструировать сложные алгоритмы и программы из более простых алгоритмов.



```
REM Корабль
Gosub 100
Gosub 200
Gosub 300
END
100 REM Лодка
……..
RETURN
200 REM Парус
……..
RETURN
300 REM Флаг
……..
RETURN
```

Общие процедуры. Область видимости процедур

В объектно-ориентированных языках программирования вспомогательные алгоритмы реализуются с помощью **общих процедур,** которые создаются тогда, когда в программном модуле можно выделить многократно повторяющиеся последовательности действий (алгоритмы).

Общая процедура - ***подпрограмма, которая начинает выполняться после ее вызова из другой процедуры***.

Каждой общей процедуре дается уникальное ***Имя процедуры*** и устанавливается ***список входных*** и ***выходных параметров.***

Список входных параметров – набор переменных, значение которых д. б. установлено до начала выполнения процедуры.

Список выходных параметров – набор переменных, значение которых д. б. установлено после окончания выполнения процедуры

Синтаксис: Sub ИмяПроцедуры (СписокПараметров)

Программный код

End Sub

Вызов общей процедуры:

1. **С помощью оператора Call**

   Call ИмяПроцедуры (СписокПараметров)

1. **По имени:**

   ИмяПроцедуры СписокПараметров

Размещение общей процедуры:

1. В составе программного модуля одной из форм проекта (файл с расширение **frm**)
2. В стандартном программном модуле (файле с расширением **bas**)

Область видимости процедуры

Общие и событийные различают:

1. ***Локальные*** – доступные только внутри одного программного модуля и не м.б. вызваны из других модулей. Задаются с помощью ключевого слова **Private**:

   Private Sub ИмяПроцедуры
   Программный код
   End Sub

2. ***Глобальные*** – м.б вызваны из всех программных модулей проекта. Задаются с помощью ключевого слова **Public**:

   Public Sub ИмяПроцедуры Программный код
   End Sub

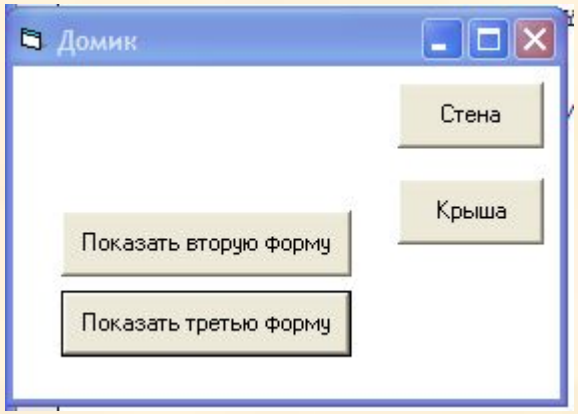
Область видимости переменной

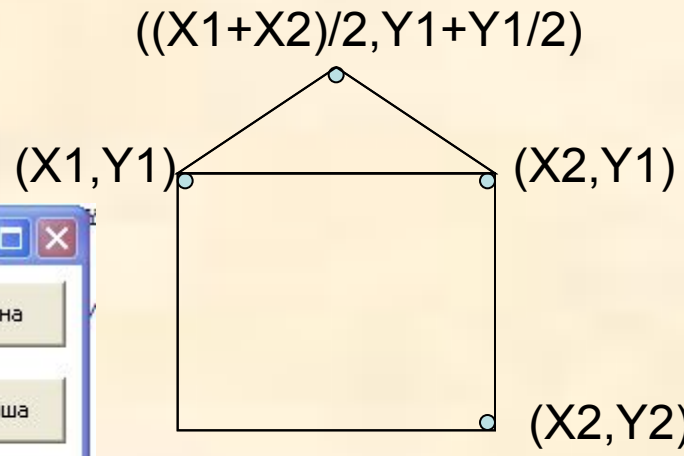
Различают переменные:

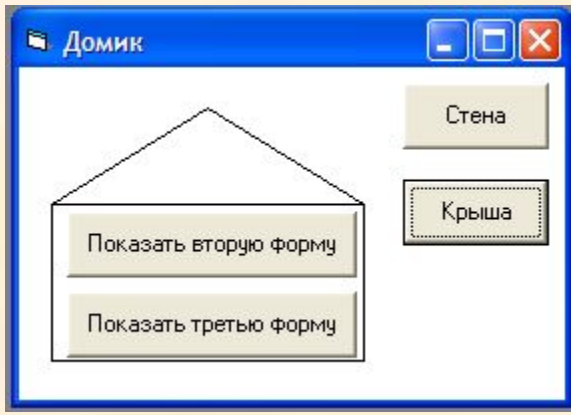
1. ***Локальные*** – доступные только внутри процедуры или программного модуля и к ней невозможно обращение из другой процедуры или модуля. Определяются с помощью ключевого слова **Dim**.
2. Если переменная определена перед процедурой, то она м. б. только в этой процедуре; если перед программным модулем в области ***(General) (Declaration)*** программного кода, то она м.б. вызвана только в этом модуле.
3. ***Глобальные*** – м.б. вызваны из всех программных модулей проекта. Определяются с помощью ключевого слова **Global** в области ***(General) (Declaration)*** программного кода.

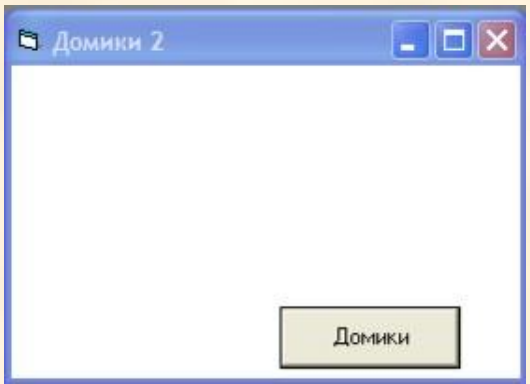
PrjVB13. Проект «Рисование домика»

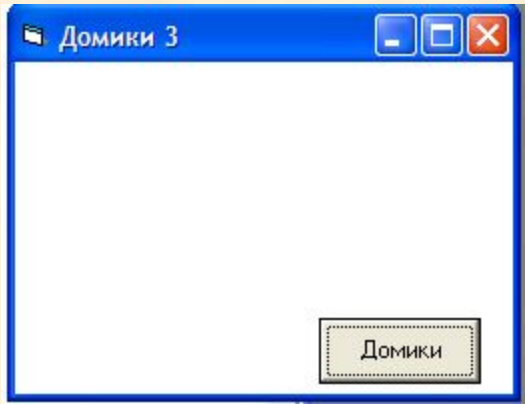
```
‘ Первая форма
Private Sub cmdСтена_Click()
Scale (0, 170)-(350, 0)
frm1.Line (20, 100)-(220, 20), , B
End Sub
Private Sub cmdКрыша_Click()
Scale (0, 170)-(350, 0)
frm1.Line (20, 100)-(220, 100)
frm1.Line (20, 100)-(120, 150)
frm1.Line (120, 150)-(220, 100)
End Sub
Public Sub Домик2(X1, X2, Y1, Y2 As Single)
frm2.Line (X1, Y1)-(X2, Y2), , B
frm2.Line (X1, Y1)-(X2, Y1)
frm2.Line (X1, Y1)-((X1 + X2) / 2, Y1 + Y1 / 2)
frm2.Line ((X1 + X2) / 2, Y1 + Y1 / 2)-(X2, Y1)
End Sub
Private Sub cmd2_Click()
frm2.Show
End Sub
Private Sub cmd3_Click()
frm3.Show
End Sub
```



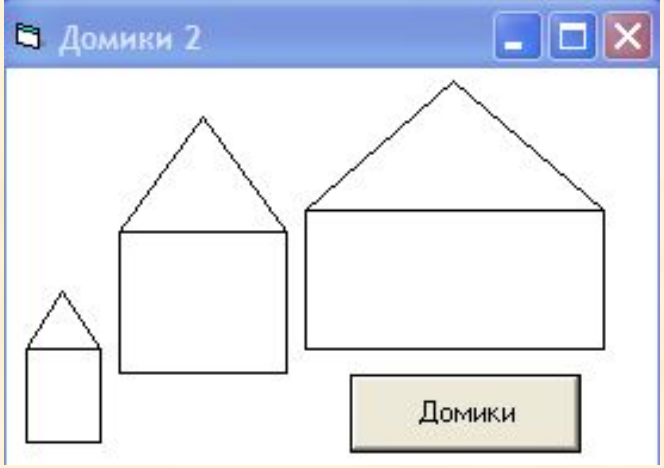




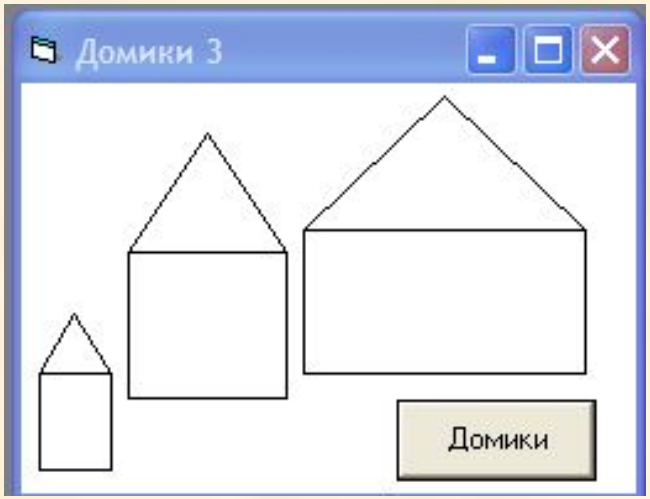




```
‘ Вторая форма
Private Sub cmdДомики2_Click()
frm2.Scale (0, 170)-(350, 0)
Call frm1.Домик2(10, 50, 50, 10)
Call frm1.Домик2(60, 150, 100, 40)
Call frm1.Домик2(160, 320, 110, 50)
End Sub
```




```
‘ Стандартный программный модуль
Global X1, X2, Y1, Y2 As Single
Public Sub Домик3(X1, X2, Y1, Y2 As Single)
frm3.Line (X1, Y1)-(X2, Y2), , B
frm3.Line (X1, Y1)-(X2, Y1)
frm3.Line (X1, Y1)-((X1 + X2) / 2, Y1 + Y1 / 2)
frm3.Line ((X1 + X2) / 2, Y1 + Y1 / 2)-(X2, Y1)
End Sub
‘ Третья форма
Private Sub cmdДомики3_Click()
frm3.Scale (0, 170)-(350, 0)
Module1.Домик3 10, 50, 50, 10
Module1.Домик3 60, 150, 100, 40
Module1.Домик3 160, 320, 110, 50
End Sub
```



Модульный принцип построения проекта и программного кода

Проект «Рисование домика»

Программный модуль формы frm1.frm:

Глобальная общая процедура Домик2(X1, X2, Y1, Y2 As Single)

Локальная событийная процедура cmdСтена_Click()

Локальная событийная процедура cmdКрыша_Click()

Локальная событийная процедура cmd2_Click()

Локальная событийная процедура cmd3_Click()

Программный модуль формы frm2.frm:

Локальная событийная процедура cmdДомики2_Click()

Программный модуль формы frm3.frm:

Локальная событийная процедура cmdДомики3_Click()

Стандартный программный модуль Module1.bas

Глобальная общая процедура Домик3(X1, X2, Y1, Y2 As Single)

Удобной формой для организации данных являются таблицы.

Различают таблицы:

1. Линейные. Данные располагаются в одну строку **R=[165,170,171,158,160]**

R(1)	R(2)	R(3)	R(4)	R(5)
165	170	171	158	160

2.Прямоугольные (несколько строк)

Все элементы таблицы имеют одно имя и отличаются только номером.

A=

3	**5**	**0**	**-1**
-2	**1**	**-8**	**6**
2	**4**	**-3**	**0**

A(1,1)=3

A(2,3)=-8

A(3,4)= 0

В линейных таблицах – **один** номер(индекс)

В прямоугольных – **два** (номер строки, номер столбца)

Для организации таблиц в программах используют массивы.

Массив - упорядоченная последовательность величин, обозначенных одним именем.

Элементы массива - индексированные переменные.

Индексы - выражения, имеющие целые положительные значения, разделяются запятой, если их несколько. Отрицательный индекс - ошибка

Индексы могут быть заданы:

Числом: A(2), C(3,5)

Переменной: A(i), B(i,j)

Выражением: A(к+1), B(5, к/2+3)

Для описания в программах таблиц (массивов) используется команда **DIM** (DIMENSION - размерность), которая пишется в программе до использования массива.

DIM R(1 To 5) As Integer DIMA(1 To 3, 1 To 4) As Single

Одной командой DIM можно описывать несколько массивов:

DIM R(1 **To** 5) As Integer, A(1 **To** 3, 1 **To** 4) As Single

Команда DIM определяет:

1. **Имя массива** (таблицы)
2. **Тип элементов** массива (числ., символьный)
3. **Размерность** массива (определяется количеством индексов):
   - **одномерный** (линейный, вектор) - один индекс
   - **двумерный** (матрица) – два индекса
4. **Количество элементов** (указан номер последнего элемента, номер первого по умолчанию=0)
5. **Занимает место в оперативной памяти** для элементов массива

Заполнение массива и поиск в массивах

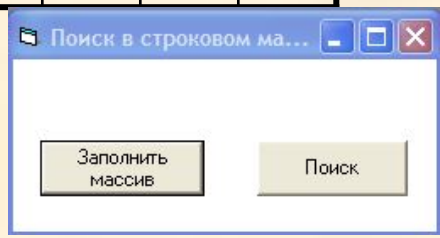
1. Заполнение с клавиатуры (InputBox)

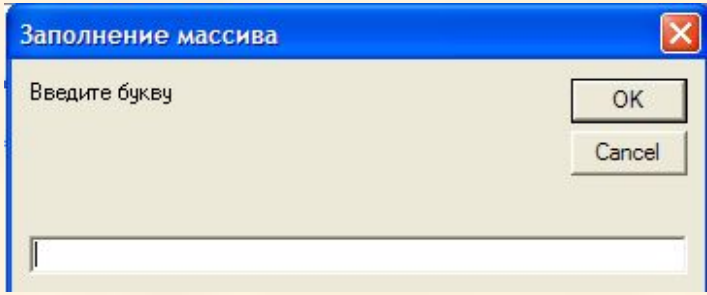
№1.1). Создать массив, элементами которого являются буквы русского алфавита, вывести их на форму.

2). Найти номер заданной буквы

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A(I)	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т

I	21	22	23	24	25	26	27	28	29	30	31	32	33
A(I)	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я



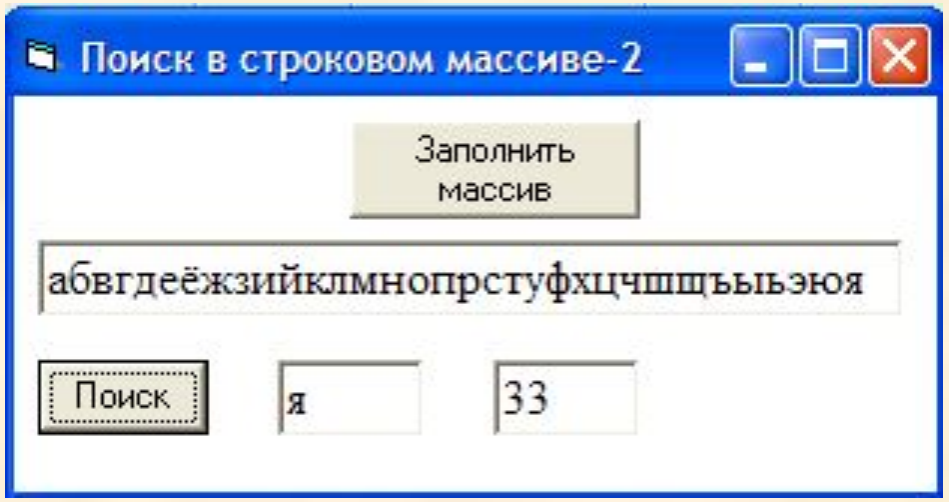



```
Dim A(1 To 33) As String, I, N As Byte
'Заполнение массива с клавиатуры
Sub cmd1_Click()
For I = 1 To 33
A(I) = InputBox("Введите букву", "Заполнение массива")
frm1.Print "A("; I; ")="; A(I)
Next I
End Sub
```

```
'Поиск элемента
Sub cmd2_Click()
B = InputBox("Введите искомую букву", "Поиск")
For I = 1 To 33
If B = A(I) Then N = I
Next I
frm1.Print "Буква и ее номер: "; B; "-"; N
End Sub
```

Задание 4.28

```
Dim A(1 To 33) As String, I, N As Byte
'Заполнение массива с клавиатуры
Sub cmd1_Click()
For I = 1 To 33
A(I) = Mid$(txt1.Text, I, 1)
Next I
End Sub
'Поиск элемента
Sub cmd2_Click()
B = txt2.Text
For I = 1 To 33
If B = A(I) Then N = I
Next I
txt3.Text = N
End Sub
```



2. Заполнение с помощью оператора присваивания

№2. 1).Заполнить массив **A(I) целыми случайными числами в интервале (1;10).**

2) Найти **Min** элемент массива и его номер.

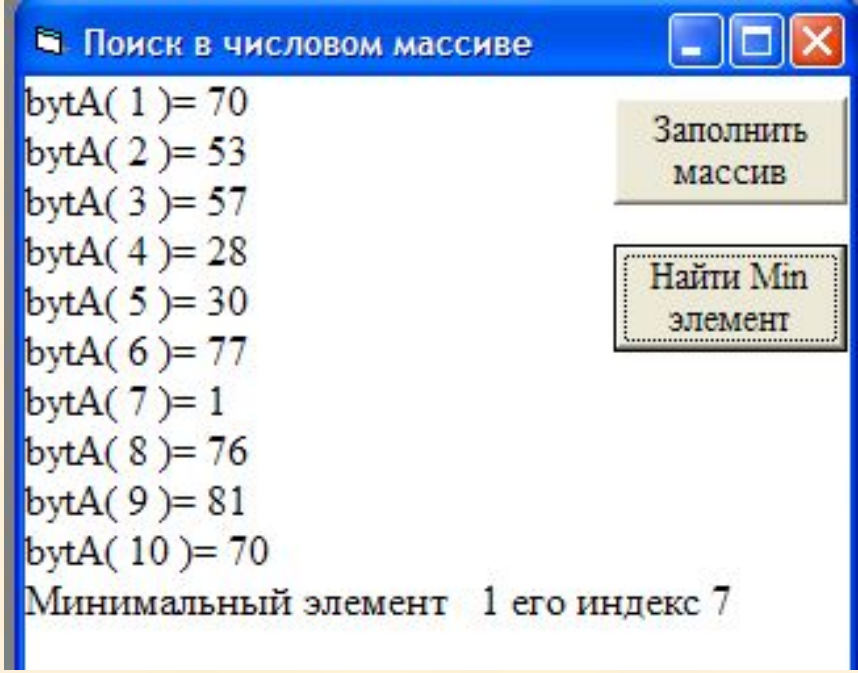
Rnd[(Число)] - Генерирует случайное число от 0 до 1.

Для генерации случайного числа **X ∈ [A,B]** используют формулу:
X=RND*(B-A) +A или
X=RND*(B-A+1) +A (включает крайние знач. интервала [A,B])

Каждый раз при запуске программы, если не переустанавливается база генератора случайных чисел, формируется одна и та же последовательность чисел.
RANDOMIZE (*база*) - переустанавливаем базу генератора случайных чисел.
Пример:

```
Dim V
RANDOMIZE TIMER
V = Int((6 * Rnd) + 1) ' Генерирует случайное число от 1 до 6
```

```
Dim A(1 To 10), I, N, Min As Byte
Sub cmd1_Click()
For I = 1 To 10
A(I) = Int(Rnd * 100)
frm1.Print "A("; I; ")="; A(I)
Next I
End Sub

Sub cmd2_Click()
'Поиск минимального элемента
Min = A(1)
N = 1
For I = 2 To 100
If A(I) < Min Then Min = A(I): N = I
Next I
frm1.Print "Минимальный элемент  "; Min; "его индекс"; N
End Sub
```



№4.27. Разработать проект, в котором массив заполняется значениями текущего времени.


```
Dim Time(1 To 100) As String, I As Byte
Sub tmr1_Timer()
I = I + 1
Time(I) = Time$
Print I; " - "; Time(I)
End Sub
```

Сортировка по возрастанию

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)
21	15	2	-8	34	-5

-8	15	2	21	34	-5

-8	-5	2	21	34	15

-8	-5	2	21	34	15

-8	-5	2	15	34	21

-8	-5	2	15	21	34

P.S. Для выполнения сортировки по убыванию необходимо искать максимальный элемент и ставить его на первое место в рассматриваемой части массива.

Сортировка методом прямого выбора:

Min - минимальный элемент массива

N - его номер

1.Выбирается элемент с наименьшим значением

2.Меняется с первым элементом массива

3.Процесс повторяется с оставшимися N-1 элементами, N-2 элементами и т.д., до тех пор, пока не останется один, самый большой элемент

R = A(I): A(I) = A(N): A(N) = R - обмен значениями элементов A(I), A(K)

```
Dim A(1 To 10), Min, I, J, K, R, N As Byte

'общая процедура поиска минимального элемента
Sub МинЭлемент(I, N As Byte)
Min = A(I)
N = I
For J = I + 1 To 10
If A(J) < Min Then Min = A(J): N = J
Next J
End Sub

'заполнение массива
Private Sub cmd1_Click()
For I = 1 To 10
A(I) = Int(Rnd * 10)
frm1.Print "A("; I; ")="; A(I)
Next I
End Sub
```

```
'событийная процедура сортировки
Private Sub cmd2_Click()
txtSort.Text = ""
For I = 1 To 9
'вызов общей процедуры поиска минимального элемента
Call МинЭлемент(I, N)
'перестановка
R = A(I)
A(I) = A(N)
A(N) = R
'печать массива для каждого цикла перестановки
For K = 1 To 10
txtSort.Text = txtSort.Text + Str(A(K))
Next K
Next I
End Sub
```

Проект «Таблица умножения»

```
Dim tA(1 To 9, 1 To 9), K, N As Byte
Sub cmd1_Click()
For K = 1 To 9
For N = 1 To 9
A(K, N) = K * N
frm1.Print K; "*"; N; "="; A(K, N)
Next N
Next K
End Sub
```