МОУ ДПО «Институт повышения квалификации»
г. Новокузнецк

А.А. Киселева

Основы программирования и моделирования в VISUAL BASIC

Учебно-методическое пособие

Новокузнецк, 2004

Печатается по решению редакционно-издательского совета Института повышения квалификации

Киселева А.А. Основы программирования и моделирования в среде Visual Basic. *– Новокузнецк: ИПК, 2004. – 116 с.*

В учебно-методическом пособии рассмотрены основы программирования с использованием среды разработки Windows-приложений Visual Basic. Особое место уделено возможностям Visual Basic, как инструмента компьютерного математического моделирования.

Доступность изложенного теоретического материала и большое количество подробно рассмотренных примеров позволят самостоятельно овладеть основными навыками работы с визуальной средой программирования Visual Basic.

Учебно-методическое пособие предназначено для учащихся профильных классов средних учебных заведений, студентов, учителей информатики. Пособие будет полезно учителям физики, желающим внедрить в учебный процесс такое средство обучения, как компьютерное математическое моделирование.

От автора

Математическое моделирование является быстро развивающейся областью науки и техники. Для ее успешного развития важны отвечающие современным требованиям средства моделирования (методы, программы, накопленные знания о решении классов задач). Среда визуального программирования Visual Basic является идеальным средством для построения и изучения моделей, из-за простой реализации основных конструкций, доступного интерфейса. Именно изучению данной среды посвящено это учебно-методическое пособие.

Учебно-методическое пособие состоит из двух частей.

В первой части «Программирование в среде Visual Basic» рассматриваются: интерфейс программы Visual Basic, основные конструкции языка, базовые элементы управления и контроля и основные методы работы с ними. Большое количество подробно рассмотренных примеров позволят овладеть основами программирования в среде Visual Basic как человеку уже имеющему опыт программирования в различных средах, так и новичку.

Во второй главе «Моделирование в среде Visual Basic» читатель познакомиться с теоретическими основами моделирования, а также с возможностями Visual Basic, как инструмента компьютерного математического моделирования. В связи тем, что рассмотренные во второй части примеры взяты из раздела общего курса физики, пособие будет интересно учителям физики профильной школы, желающим углубить и расширить содержание предмет за счет введения такого метода изучения физики, как компьютерное моделирование.

Все примеры и решенные задания читатель может найти на прилагаемом к пособию диске. Папка «программирование» содержит задания первой части пособия. Задания второй части читатель найдет в папке «моделирование». Диск содержит демоверсию программы Visual Basic 5.0 (папка «дистрибутив VisualBasic5.0»). Файлы «задания_моделирование.doc» и «задания_программирование.doc» облегчат работу учителя при создании заданий для учащихся. В папке «Graphics» находятся графические файлы, для создания более ярких приложений.

ЧАСТЬ I.
Программирование в среде VISUAL BASIC

Тема 1. Знакомство с Visual Basic

Введение

Важнейшие качества языка BASIC (Beginner's All-purpose Symbolic Instruction Code - символический командный код общего назначения для начинающих) - простота и компактность - оказались решающими в период перехода на микрокомпьютеры. Билл Гейтс и Пол Аллен, основатели корпорации MICROSOFT (MS), создали версию BASIC, способную работать в 4 Килобайтах оперативной памяти (!), со временем превратив эту версию в один из самых популярных программных продуктов для персональных компьютеров (ПК).

Впоследствии потребность в более быстром, компактном, простом и структурированном процедурном языке привела к появлению MS QUICK BASIC, поднятого на уровень технологии программирования 80-х годов.

Но большие перемены в компьютерном мире, связанные с принятием стандарта на графический интерфейс пользователя (Graphical User Interface, GUI), требовали средств разработки приложений под Windows. И такое средство появилось в 1991 году - MS Visual Basic, позволяющее, «отстранившись» от сложнейшей внутренней структуры Windows, творить в свое удовольствие.

Нельзя также забывать, что первая версия Visual Basic была создана еще под DOS и позволяла не в графической среде оперировать такими объектами, как окна, кнопки и т.д., приводя в неописуемый восторг начинающих программистов.

Что такое VB

Visual Basic (VB) - это *среда разработки программ,* которая позволяет быстро и легко создавать приложения (прикладные программы) для Windows. В нее включено все, что необходимо для создания, модификации, тестирования, корректирования и компиляции Ваших программ.'Visual Basic - это *полноценный язык программирования* высокого уровня.

Слово *Visual* - «визуальный», «наглядный» - означает способ разработки пользовательского интерфейса программы. Еще на этапе создания программы Вам видно, как будет выглядеть программа в действии. Вы «рисуете» окна, кнопки, текстовые панели, линейки прокрутки и другие компоненты пользовательского интерфейса подобно тому, как рисуют объекты в графическом редакторе для Windows.

Слово *Basic* - «основной» - описывает тип программного кода,

который Вы создаете. Программа Visual Basic представляет собой вариант хорошо известного языка программирования.

Получили распространение такие подмножества языка Visual Basic, как VB For Applications (встроенный язык написания макросов для MS Office приложений) и VB Script (язык для работы в среде WEB-браузера с объектами и элементами HTML).

Что может VB

Visual Basic позволяет создавать прикладные программы, которые могут:

- Использовать стандартные элементы пользовательского интерфейса Windows, такие, как окна, меню, кнопки, линейки прокрутки и т.п.
- Создавать, читать и записывать текстовые файлы и файлы баз данных.
- Выводить данные на печать при помощи стандартных драйверов принтеров Windows.
- Читать графические файлы в форматах .bmp, .jpg, .tif, .ico, .wmf и др.
- Обращаться к базам данных таких форматов, как SQL, dBase, Microsoft Access и другие.
- Взаимодействовать с другими прикладными программами через буфер обмена, DDE (Динамический Обмен Данными) и OLE (Связывание и Встраивание Объектов).
- Использовать элементы управления OCX ActiveX и API-функции.
- Взаимодействовать с Интернет

Запуск программы. Основные окна

Запуск программы

Запустить программу можно с помощью команды меню (см. рис.1):

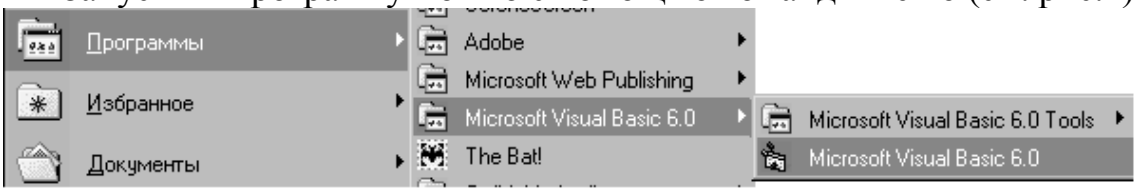


Рисунок 1. Команда запуска Visual Basic в меню

После запуска на экране появится диалоговое окно, в котором можно выбрать тип создаваемого приложения. Из этого окна можно загрузить уже существующий проект. Для создания обычного Windows-приложения нужно выбрать элемент **Standart EXE**.

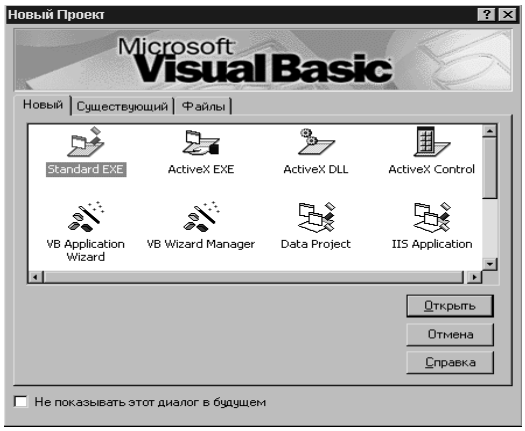


Рисунок 2. Диалоговое окно запуска Visual Basic

Главное окно

Назначение этого окна – доступ ко всем командам через главное меню и панель инструментов.

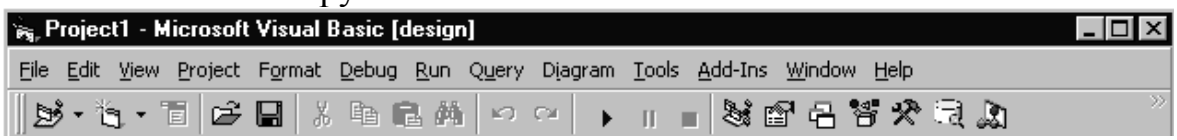


Рисунок 3. Главное окно

Окно формы

В окне формы создается интерфейс будущего Windows-приложения – средства общения пользователя с программой.

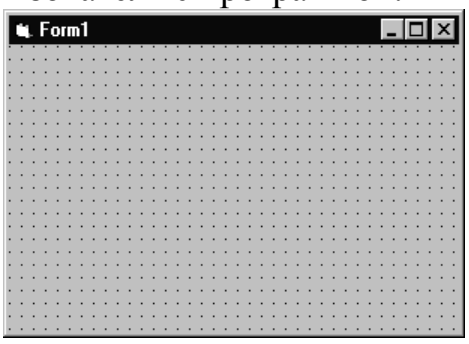


Рисунок 4. Окно формы

Окно инструментария

Toolbox Window (Окно инструментов) предназначено для выбора элементов управления для вашей программы. Часто, при работе с этим окном, элементы управления называют **tools (инструменты)**. Таким образом, существуют три термина, описывающие элементы управления: объекты, инструменты, и, наиболее употребляемые, сами элементы управления (controls).

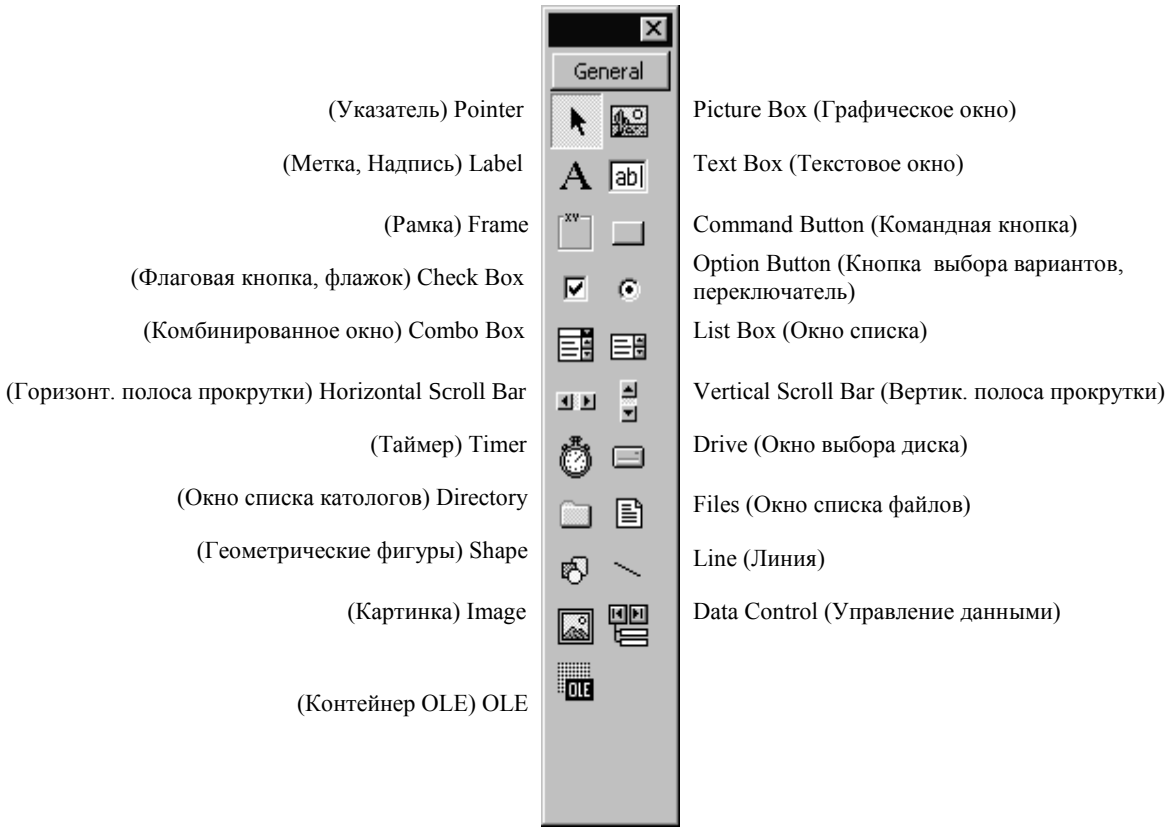


Рисунок 5. Окно инструментов

Окно свойств

Properties Window (Окно свойств) предназначено для установки исходных значений некоторых параметров элементов управления. Раскрывающийся список в верхней части окошка содержит все элементы управления, расположенные на форме, с которой вы работаете. Под этим полем располагается перечень свойств выбранного объекта.

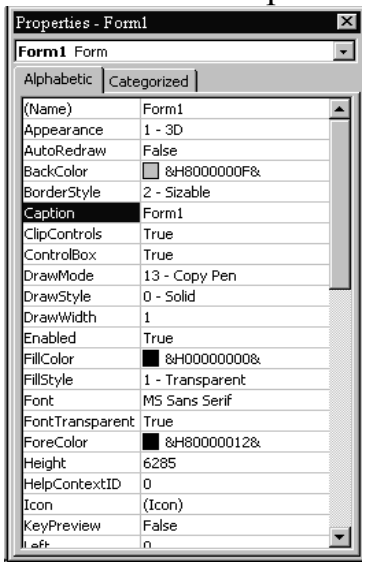


Рисунок 6. Окно свойств

Окно проводника проекта

Project Explorer (Окно проводника проекта) появляется, когда создаётся форма для вашего проекта. Когда вы станете опытным программистом, вы научитесь создавать проекты, состоящие из нескольких форм. В этом случае, все формы вашего проекта будут перечислены в этом окне. Из этого окна, также, вы сможете перейти к просмотру окна **Form** или окна **Code** (это окно, в котором записывается непосредственно код программы на языке BASIC); для этого необходимо щелкнуть по одной из кнопок окна проекта.

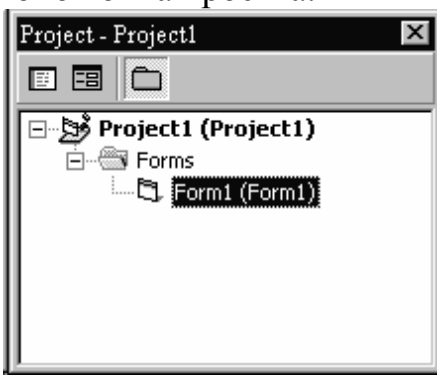


Рисунок 7. Окно проводника проекта

Окно кода

Code Window (Окно кода). В Visual Basic управление происходит с помощью событий, т. е. когда в проекте встречается событие, то выполняется **event procedure (процедура события)**. Процедуры событий сообщают компьютеру, что делать в ответ на событие. Процедуры событий можно назвать непосредственно компьютерным программированием (в данном случае, использующим язык BASIC). Мы рассматриваем процедуры событий в **Code Window (Окне кода)**. Есть несколько способов вызвать на экран окно кода. Например, можно в окне проекта нажать соответствующую кнопку. А можно щелкнуть **View** в главном меню, а затем **Code**. Или просто нажать **F7** на клавиатуре.

Окно просмотра характеристик классов объекта

Браузер Объектов (Object Browser). Вид – Браузер Объектов. Окно позволяет просматривать характеристики класса (свойства, методы, события).

Окно расположения формы Form Layout

В окне расположения формы имеется возможность установить положение экранной формы на экране монитора во время выполнения программы.

Основные термины

Объект управления и контроля (Control) - общий термин, который определяет как экранные формы, так и графические элементы внутри этих форм, в том числе текстовые окна, окна-списки, командные кнопки, графические окна, линейки прокрутки, пиктограммы и т.д. Над объектами управления можно осуществлять только операции, определенные в рамках метода доступа. В системе VB термины «объект управления и контроля» (Control) и «объект» (Object) равнозначны.

Событие *(Event)* - действие, которое распознается объектом управления VB.

Экранная Форма (Form) - окно, созданное пользователем в соответствии с требованиями прикладной программы.

Метод доступа (Method) - ключевое слово VB, аналогичное по смыслу понятиям «функция» или «оператор», но которое воздействует все же на конкретный объект управления. Для каждого объекта система VB определяет несколько стандартных методов, которыми может оперировать пользователь.

Процедура (Procedure) - этим термином обозначаются как подпрограммы, так и функции. Это просто последовательность операторов VB, к которым система обращается, как к связанной группе во время выполнения. Существует два типа процедур: событийная процедура и общая процедура. ***Событийная процедура*** используется только с экранными формами и объектами управления, ***общие*** же ***процедуры*** используются в любом месте программы и могут вызываться событийными процедурами

Проект (Project) - совокупность всех файлов, составляющих прикладную программу.

Параметр или свойство (Property) - характеристика или атрибут объекта управления. Для каждого типа объекта управления VB определяет набор параметров, относящихся только к данному объекту.

Значение (Setting) - значение параметра. Можно изменить значение большинства параметров при разработке программы. Программа может изменить значения параметров и во время работы.

Этапы разработки программ на VB:

- Постановка задачи (словесное описание того, как будет работать приложение, как будет выглядеть…).
- Разработка интерфейса - создание экранной формы, со всеми находящимися на ней объектами и свойствами этих объектов.

- Программирование - определение того, какие события будут происходить в процессе работы приложения, составление алгоритмов процедур для этих событий, написание программных кодов.
- Отладка программы - устранение логических ошибок в процедурах.
- Сохранение проекта, при желании - компиляция (превращение проекта в исполнимое приложение).

Задание 1

Создать форму, содержащую объекты: метка, кнопку "Привет", "Очистка", "Выход". При нажатии на кнопку "Привет", в окне Надписи (Метки) появляется сообщение "Здравствуйте, друзья", при нажатии на кнопку "Очистка" окно Надписи очищается, при нажатии на кнопку "Выход", программа завершает свою работу.

1. Разработка интерфейса приложения:

- Создайте форму и переименуйте ее в соответствии с заданием, свойству Caption (название) присвойте значение "Первая программа на Visual Basic".
- Нанесите на форму Надпись, изменив ее свойство Caption на пустую строку.
- Нанесите на форму командные кнопки 1,2,3, изменив их свойство Caption на "Привет", "Очистка", "Выход" соответственно.

2. Написание программного кода:

- Второе событие, после загрузки формы - это щелчок на кнопку "Привет". В процедуре обработки данного события свойству надписи Caption необходимо присвоить значение символьной строки "Здравствуйте, друзья".

```
Private Sub Command1_Click()
Label1.Caption = "Здравствуйте, друзья"
End Sub
```

- Третье событие - щелчок на кнопку "Очистка", напишите программный код обработки данного события самостоятельно.
- Последнее событие - щелчок на кнопку "Выход". В процедуре обработки этого события необходимо завершить работу программы.

```
Private Sub Command3_Click()
End
End Sub
```

3. Сохраните форму и проект (1.1.).

4. Измените цвет фона формы на голубой, используя свойство формы BackColor.

5. Используя свойство Надписи Allignment (выравнивание), установите текст Надписи по центру; используя свойство Font (шрифт), измените значения шрифта.
6. Измените программу таким образом, чтобы при нажатии на кнопку "Очистка" название формы изменялось бы на "Очистка Надписи", а при нажатии на кнопку "Привет" - на "Заполнение Надписи".

Задание 2. Подсчитать квадрат введенного числа.

Программа после нажатия на кнопку "Вычислить" находит квадрат числа, введенного в текстовое поле.

Рисунок 8

1. Разработка интерфейса приложения:
- Измените заголовок приложения, используя свойство формы Caption.
- Нанесите на форму текстовое поле, изменив его свойство Text на пустую строку.
- На форме сознайте две кнопки, измените их название в соответствии с рисунком.

2. Написание программного кода:
- Напишите самостоятельно процедуру обработки нажатия на кнопку "выход".
- Напишите процедуру обработки нажатия на кнопку "вычислить"

```
Private Sub Command1_Click()
Print Val(Text1.Text) ^ 2
End Sub
```

3. Измените программу таким образом, чтобы результат выводился в текстовое поле.

Задание 3. Определить число рулонов, необходимых для оклейки помещения.

В программе будут вводиться ширина, длина, высота комнаты, ширина и длина рулона, при нажатии на кнопку "Вычислить" должен появиться результат, показывающий число рулонов, необходимых на оклейку помещения.

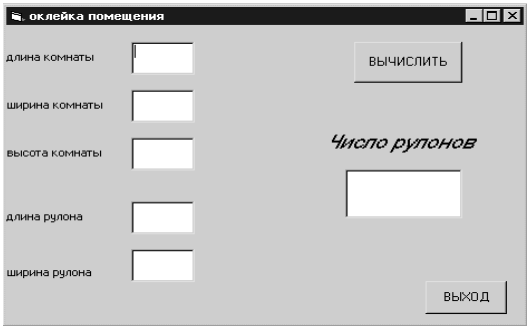


Рисунок 9

1. Разработка интерфейса приложения:
Создайте приложение, используя объекты: Надпись, Текстовое поле, Кнопка.

2. Написание программного кода:
Напишите программный код обработки двух событий: нажатие на кнопку "Вычислить" и кнопку "Выход".

Тема 2. Программирование: константы, переменные, выражения, функции, основные алгоритмические конструкции

Переменная и ее значение

Переменная - часть программы, которая имеет имя и значение.

Имя переменной - строка символов, которая отличает переменную от других объектов программы, т.е. строка, идентифицирующая переменную в программе.

Значение переменной - данные, которые хранятся и обрабатываются компьютером.

Тип данных определяет способ хранения и совокупность функций, которые можно применять к этому типу данных (числовой, строковый, массив, запись, файл)

Таблица 1. Типы данных

Тип данных	Размер	Диапазон значений	Пример использования
Byte	1	0 – 255	Dim I as Byte I=23
Integer	2	-32768 – 32767	Dim Temp as Integer Temp=-55 Dim Temp% Temp%=-55
Long	4	-2147483648 - 2147483648	Dim L& L&=350000
Single	4	-3,402823E38 - 3,402328E38	Dim Price! Price!=69.99
Double	8	-1,797…D308 - 1,797…D308	Dim Pi# Pi#=3.1415926535
Currency	8	-922 trillions - 922 trillions	Dim D@ D@=67.45
String	1байт на символ	0 - 65535 символов	Dim Dog$ Dog$="boxer" Или Dim s As String * 12
Boolean	2	True, False	Dim Flag as Booleal Flag=True
Date	8	January 1,100 - December 31,9999	Dim Birthday as Date Birthday=#6-6-99#
Variant	16 байт (для чисел), 22 байта + 1 байт на символ (для строк)	Для всех типов данных	Dim Total Total=456.13

Объявление переменных

Объявление переменных осуществляется в разделе описания переменных.

Если переменная глобальная - в рамках данной формы, то она описывается в разделе General с помощью ключевого слова **Dim**:

Dim ИмяПеременной [**As** Тип]

Описание локальных переменных осуществляется в разделе описания переменных процедуры.

Объявление типа с помощью суффикса. Чтобы по имени переменной можно было судить об ее типе, часто (но не всегда) к имени переменой приписываю суффикс (%, &…). Суффикс можно описать только один раз в разделе описания переменной.

Режим обязательного объявления переменных. Инструменты – Опции – Require Variable Declaration (установить флажок).

Помимо объявления переменных уровня процедуры (Dim), существует еще один способ объявления переменных и массивов - с помощью ключевого слова **Static**. Это означает, что переменная будет сохранять последнее присвоенное ей значение даже после завершения процедуры. Статическими переменными удобно пользоваться для текущего значения накапливаемой суммы. Если пропустить ключевое слово Static (используя Dim), то при каждом запуске процедуры переменная будет обнуляться вместе с другими числовыми переменными.

Пример 1. Подсчитайте число кликов на кнопке в процессе выполнения приложения.

```
Private Sub Command1_Click()
Static a As Single
a = a + 1
Print a
End Sub
```

Присвоение переменной значения

Для *присвоения переменной* значения - используется *оператор присваивания*

[**Let**] ИмяПеременной = ЗначениеПеременной

Присвоить значение переменной в процессе выполнения программы можно:

- используя текстовое поле a = Text1.Text
- используя диалоговое окно сообщений (см. ниже).

Задание 1. Постройте приложение, в котором бы после нажатия на кнопку "Пуск", содержимое двух текстовых полей менялось бы местами.

Константы

Константы – величины, значения которых не могут меняться.

Const Имя Константы [**As** Тип] = Значение Константы

Кроме объявляемых констант, в программе могут быть использованы *системные, встроенные* константы: vbRed, VbOk.

Выражения и функции

Выражения: арифметические, строковые, логические. В построении выражений используются операторы и функции.

Арифметические выражения

Арифметические выражения - последовательность чисел, констант, переменных, функций, которые соединены между собой знаками арифметических операций.

Арифметические операции: +, -, *, /, \ (целая часть от деления), mod (остаток от деления), ^ (возведение в степень).

Функции должны возвращать определенные числовые значения. Понятие функции в VB такое же, как и в математике.

Встроенные *математические функции*:

cos(n)	косинус числа n
sin(n)	синус числа n
tan(n)	тангенс числа n
abs(n)	модуль числа n
round(n)	целое число, ближайшее к числу n
exp(n)	экспонента числа n
rnd	генерирует случайное вещественное число от 0 до 1
sqr(n)	квадратный корень числа n,
fix(n)	целое число, равное n без дробной части
int(n)	наибольшее целое, не превышающее n

В VB также существуют *финансовые функции, системные функции. Определяемые функции* - собственные функции.

Строковые выражения

Строка - упорядоченная последовательность символов. Для обозначения строки, используют кавычки ("").

Операция & и + - слияние двух строк (конкатенация)

Строковые функции:

len(s)	функция возвращает длину строки s
mid(s,pos[,length])	выделение подстроки длиной length из строки s с позиции pos

left(s,length)	выделение левой подстроки длинной length
right(s,length) -	выделение правой подстроки длинной length
instr([start],s,pods)	в строке s ищется подстрока pods, возвращается номер позиции вхождения подстроки
val(s)	функция преобразует строку s в число
str(n)	функция преобразует число n в строку
asc(s)	преобразует строку в код ASCII первого символа этой строки asc(«1998») возвращает 49 (код цифры 1)
chr(kod)	преобразует код в строку из одного символа chr(49) – возвращает «1»
UCase(s)	возвращает строку, все буквы которой преобразованы в пропосные UCase(“Visual Basic”) – взвращает “VISUAL BASIC
LCase(s)	возвращает исходную строку, все буквы которой преобразованы в строчные

Логические выражения (условные).

Для построения логических выражений используют операции сравнения: >, <, =, <>, >=, <=; логические операции: and, or, xor, not

Организация нелинейных алгоритмов

Условный оператор

однострочная форма синтаксиса

if условное выражение **then** оператор1 [**else** оператор2]

многострочная форма

if условное выражение **then**
  последовательность операторов1
[**else**
  последовательность операторов2]
end if

Существует встроенная функция **Iif** для программирования безусловного перехода: **Iif** (УсловноеВыражение, значение1, значение2)

Задание 2. Создайте программу определения модуля числа (не используя функцию abs(n)).

Задание 3. Используя функцию **Iif**, проверьте введенное число на четность (Составить программу, определяющую результат гадания на ромашке — «любит—не любит», взяв за исходное данное количество лепестков *n.*).

Задание 4. Вычислить корни квадратного уравнения $ax^2+bx+c=0$ с заданными коэффициентами *a, b* и *c*.

Оператор выбора

```
select case переменная
case значение1
  оператор1
case значение2
  оператор2
…
end select
```

VB позволяет использовать операции сравнения, чтобы иметь возможность проверки по нескольким критериям: оператор Is - дает указание машине сравнить значение проверяемой переменной со значением выражения, следующего после Is. To - задает диапазон значений.

Задание 5. Создать программу, которая в зависимости от введенного возраста выдает следующую информацию: "ты ребенок", "ты подросток", "хорошо выглядишь", "прекрасный возраст".

Безусловный оператор и безусловный переход

```
GoTo метка
```

Метка - символьная цепочка, которая может быть поставлена перед каким-либо оператором программного кода.

Пример 2. Возможно ли построить треугольник с заданными сторонами?

```
a = Val(Text1.Text)
b = Val(Text2.Text)
c = Val(Text3.Text)
If a + b <= c Then GoTo lb1
If a + c <= b Then GoTo lb1
If c + b <= a Then GoTo lb1 Else GoTo lb2
lb1: MsgBox ("треугольник с такими сторонами постоить нельзя")
GoTo lb3
lb2: MsgBox ("можно построить треугольник с такими сторонами")
GoTo lb3
lb3: End
```

Программирование циклов

цикл со счетчиком

```
for имя = значение1 to значение2 [step значение3]
   повторяющиеся операторы
next [имя]
```

Задание 6. Вычислить факториал числа.

цикл с условием

do условие
Повторяющиеся операторы
loop

do
Повторяющиеся операторы
loop условие

Условие бывает двух типов:

✓ с ключевым словом While, в этом случае оно называется условием продолжения цикла;

✓ с ключевым словом Until, в этом случае оно называется условием завершения цикла.

Таким образом, возможны 4 варианта организации цикла с условием:

do while …	**do until …**	**do**	**do**
…	**…**	**…**	**…**
loop	**loop**	**loop while …**	**loop until …**

Для того, чтобы программа не зависла в случае ошибки, а продолжала работать, удобно использовать оператор перехвата ошибки:

On Error GoTo метка и метка устанавливается в конце обрабатываемого события

Для выхода из зависания можно нажать клавиши **Ctrl + Break**.

Для того чтобы прервать выполнение цикла и перейти к выполнению оператора, следующего за циклом, можно воспользоваться оператором прерывания цикла: **Exit Do** – для цикла с условием; **Exit For** – для цикла со счетчиком.

Задание 7. Найти сумму всех n-значных чисел.

Задание 8. Бегущая строка.

1. Создать форму, как на рисунке, переименовать ее.

2. Нанести на форму Метку и две Командные кнопки. Изменить их свойства в соответствии с рисунком.

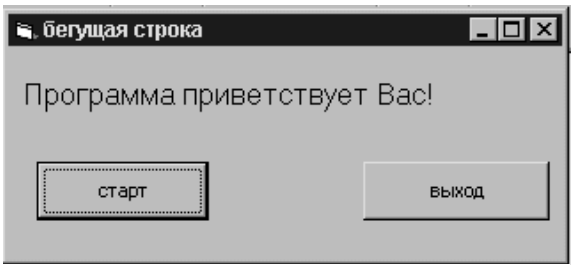


Рисунок 10

3. Написать программу, выполняющую следующие функции:

• при нажатии на Кнопку «Старт» в окне Метки появляется «бегущее» сообщение «Программа приветствует Вас!»;

• при нажатии на Кнопку «Выход» программа завершает работу.

Указания:

1. Для реализации бегущей строки используйте бесконечный цикл Do ... Loop.

2. Для обновления состояния Метки используйте метод Refresh (метод ***Refresh*** - осуществляет полную перерисовку либо формы, либо

соответствующего объекта управления и контроля).

3. Задержку соответствующей продолжительности оформите с помощью пустого цикла For ... Next.

Для того чтобы во время бесконечного цикла произошла обработка события Щелчок на кнопку «Выход», необходимо использовать функцию DoEvents ***(DoEvents*** - придерживает выполнение таким образом, чтобы операционная система могла обрабатывать другие события; возвращаемое значение функции - количество открытых окон).

Тема 3. Идеология объектно-ориентированного программирования и реализация его в VB

Б. Л. Ворф сказал: «Язык формирует наш способ мышления и определяет, о чем мы можем мыслить». К этому можно добавить, что визуальный язык программирования формирует наше воображение и определяет, что мы можем себе представить.

В традиционной, процедурной модели программирования выполнение программы начинается с первой строки и следует стандартным путем с вызовом, по мере необходимости, тех или иных процедур, В объектно-ориентированных языках реализуется событийно-управляемая модель программирования, т. е. программа выполняется, не следуя строго предопределенным путем, а в зависимости от того, какие наступают события. Эти события могут быть вызваны, например, действиями самого пользователя (нажатие на клавишу, щелчок мышью на экранной кнопке и др.) или сообщениями от системы (внутренних компонентов компьютера или подключенных периферийных устройств). Таким образом, последовательность выполнения запущенной программы определяется последовательностью событий. При следующем запуске программы эта последовательность может быть иной.

Итак, как следует из названия, объектно-ориентированное программирование (object-oriented programming, оно же событийно-управляемое программирование — event-driven programming) — это программирование, направленное на объекты.

Что же такое объект (object)? А все, что нас окружает, и с чем мы можем взаимодействовать (потрогать, увидеть, услышать, измерить, провести эксперимент и т. д.). В одном случае это могут быть автомобили или шнурки для ботинок (материальные объекты), а в другом — вой сирены или таблица в книге (информационные). Объекты, в свою очередь, могут состоять из более простых объектов (подобъектов). В автомобиле есть двигатель, руль, сиденья, педали. Объект «Текст» состоит из абзацев, слов, отдельных символов. Если объект сложный, то он получает статус системы. Каждая система характеризуется своей структурой, т. е. входящими в нее элементами (подобъектами) и правилами их взаимодействия, обеспечивающими целостное функционирование системы.

Не бывает двух абсолютно одинаковых объектов. Сорвите с березы несколько листьев и сравните. Но в то же время нельзя сказать, что они совершенно разные. Достаточно одного листочка, чтобы определить породу дерева. Следовательно, есть в этом листочке что-то, что отличает березу от дуба, и поэтому для изучения пород деревьев нет необходимости

запасаться мешками с листьями. Достаточно одного, который назовем моделью «Лист березы». Здесь важно понять, что реальный (исходный) листик и построенная на его основе модель - далеко не одно и то же, так как созданная нами модель - это абстракция, обобщенный образ объекта реального мира и содержит лишь наиболее общие (существенные) свойства или качества, присущие всем березовым листьям (один во многом, многое в одном). Как и объекты, модели могут быть материальными и информационными. Для листа такими моделями могут служить, соответственно, засушенный лист и рисунок листа. То есть объект один, а способы его моделирования разные. Все наше образование, по существу, сводится к изучению всевозможных моделей, а также их созданию и использованию (моделирование).

То же относится и к программированию. Ясно, что здесь мы имеем дело с созданием моделей информационного типа, отражающих (описывающих) свойства и поведение некоторых реальных объектов (например, полет ракеты). В программировании в качестве объектов могут выступать и объекты самого языка. Когда вы щелкаете мышью на какой-нибудь экранной кнопке, например ОК, то щелкаете на объекте. То же самое относится ко всевозможным окнам, рисункам, надписям, меню, квадратикам и кружочкам. Да что там кружочки! Все приложение целиком (например, текстовый редактор Microsoft Word) тоже является объектом.

В каждый момент времени объект характеризуется присущим именно ему набором свойств (properties) и методов (methods) — операций, совершаемых над другими объектами или данным объектом, а также реагирует на события (events).

Возьмем, например, объект «Собака». Он может иметь такие свойства: порода — пудель, имя — Джек, цвет шерсти — белый, цвет глаз — коричневый, внешний вид — грязный, вес — 15 кг, наличие ошейника — да. Слева от знака тире указаны наименования свойств, справа — значения свойств. Если значение свойства нельзя изменить, то такое свойство называется постоянным, в противном случае оно называется переменным. Среди всех свойств особое место занимают так называемые логические свойства, т. е. такие свойства, которые принимают только два значения: True (истина, да) и False (ложь, нет). В нашем примере постоянными являются свойства «порода» и «цвет глаз», остальные свойства переменные: имя для собаки можно придумать другое, шерсть перекрасить, собаку можно помыть, накормить, снять с нее ошейник. Свойство «наличие ошейника» является логическим.

Методы (действия) у собаки могут быть такими: лаять, кусаться, грызть кость, сторожить, вилять хвостом. Как и всякий объект, собака может реагировать, а может и не реагировать на события. Вообще говоря,

событие — это действие, распознаваемое объектом, после чего следует реакция (отклик) со стороны объекта.

Если положить перед собакой аппетитную косточку, то для нее возникнет событие «Кость». Поскольку у всех собак имеется «встроенный метод» «грызть кость», то возникает соответствующий отклик: собака начинает ее грызть. Если же попытаться организовать событие «Ко мне!», то отклика может не последовать, поскольку собаки от рождения не обладают этим методом. Для того чтобы собака могла распознавать данное событие, придется писать специальный «программный код», попросту говоря, дрессировать собаку. В сущности, в этом и заключается процесс создания приложений.

На самом деле значение свойств «Порода» и «Цвет глаз» можно изменить. Да, пока мы имели дело с конкретным пудельком Джеком (а может, уже и не Джеком), то сделать этого, естественно, не могли. Но если мы будем рассматривать собаку абстрактно, как модель, тогда все в порядке. В программировании такие модели называются классами объектов (Classes of Objects). Это понятие, как вы помните, используется в различных науках, например в биологии: класс млекопитающих, класс земноводных и т. д. Одним словом, класс представляет собой шаблон, на основе которого конструируются («рождаются») реальные объекты, называемые также экземплярами класса.

Ясно, что в классе «Собака» вы не найдете таких свойств и методов, как мяукать или лазать по деревьям, но в классе «Кошка» они имеются.

Рассмотрим пример. Вы хотите приобрести компьютер. Для этого вы берете бумагу, пишете на ней свойства класса «Компьютер» и устанавливаете их значения. Например: корпус — MidiTower, процессор — Pentium III, RAM — 64 Мб, HDD — 10 Гб и т. д. Затем с этими записями вы идете в магазин, и вам по ним собирают реальный объект «Компьютер» — один из экземпляров класса.

Если взять набор объектов, как правило, одного и того же типа, хотя и не обязательно, то мы получим семейство или коллекцию (collection), которая, в свою очередь, тоже является объектом. Чтобы можно было обращаться к элементам коллекции, каждому объекту присваивается уникальное имя или номер. Типичным примером коллекции является объект «Группа студентов».

Объединение в объекте его свойств и методов называют ИНКАПСУЛЯЦИЕЙ. Казалось бы, зачем вводить новый термин? На первый взгляд достаточно описать свойства и методы объекта. На самом деле инкапсуляция означает, что объект инкапсулирует (содержит) в себе свойства и методы, но описывать мы ничего не должны. Более того, мы вообще не обязаны знать, как устроен объект (иными словами, мы можем

не задумываться над тем, какой у объекта программный код, его реализующий). В примере с покупкой компьютера можно было ничего не описывать, а просто купить готовый компьютер (как и делает большинство людей, поскольку все эти RAM, HDD, FDD им ничего не говорят). Под термином «инкапсуляция» подразумевается то, что мы работаем (взаимодействуем) с объектом совершенно не зная об его устройстве (т. е. объект — это как бы вещь в себе). Синонимом инкапсуляции может служить более понятный термин «сокрытие информации» о конструкции объекта.

Итак, окончательно, объект можно определить как продукт инкапсуляции данных вместе с кодом, предназначенным для их обработки.

Несмотря на то, что в разных источниках делается акцент на те или иные особенности внедрения и применения ООП, три основных понятия остаются неизменными: инкапсуляция, наследование, полиморфизм. Первое из них мы уже рассмотрели.

НАСЛЕДОВАНИЕ - процесс, посредством которого один объект может наследовать свойства другого объекта и добавлять к ним черты, характерные только для него.

ПОЛИМОРФИЗМ - свойство, которое позволяет одно и то же имя использовать для решения нескольких технических задач. Полиморфизм подразумевает такое определение методов в иерархии типов, при котором, метод с одним именем может применяться к различным родственным объектам. В общем смысле концепцией полиморфизма является "один интерфейс - множество методов". Преимуществом полиморфизма является то, что он помогает снижать сложность программ, разрешая использование одного интерфейса для единого класса действий.

Последнее, что следует отметить – какова разница в оформлении свойства и метода. Свойство объекта имеет определенное значение. У метода нет значений; правда, могут быть определенные параметры, которые указываются за названием метода.

```
frmHello . Caption = “ПРИВЕТ!”      ' пример использования свойства
frmHello . Cls                      ' пример использования метода
frmHello . Print A
```

Тема 4. Основные объекты управления: форма, кнопка, надпись, таймер, текстовое поле, полосы прокрутки

Объект Form (форма)

Форма представляет собой стандартное Windows-окно, которое служит основой для создания интерфейса прикладной программы.

Основные свойства

BackColor - цвет фона (задается 16-ричной константой)
BorderStyle - тип границы: 0 - нет, 1 - одинарная, 2 - изменяемая, 3 - двойная фиксированная
Caption - название формы в заголовке
DrawStyle - стиль рисования
Enabled - доступность формы
FillColor - цвет заполнения
FillStyle - стиль заполнения
Font - параметры шрифта
ForeColor - цвет переднего плана
Height - высота формы
Icon - рисунок в поле заголовка
Left - расстояние до формы от левой границы экрана
ScaleMode - единица измерения в координатной системе объекта
Top - расстояние до формы от верхней границы экрана
Visible – видимость
Name - имя формы в программе
MinButton - кнопка минимизации формы
MaxButton - кнопка максимизации формы
MousePointer - указатель мыши
WindowsState - состояние окна

События для формы

Load - загрузка формы
Unload - выгрузка формы
Click - щелчок мыши
DoubleClick - двойной щелчок

Процедуры и методы

Cls - очистка формы
Print - вывод текста на форму
Show - показать форму
LoadPicture - функция загрузки картинки, записанной в виде файла.

Объект управления и контроля Command button (командная кнопка)

Щелчок командной кнопки обычно активизирует выполнение какой-то команды.

Основные свойства

Caption, ***Enabled***, ***Font***, ***Height***, ***Width***, ***Left***, ***Top***, ***MousePointer***, ***Visible***, ***Name*** - свойства, аналогичные свойствам формы

Cancel - отмена (если true, то щелчок на командную кнопку аналогичен лавише Esc)

Default - по умолчанию (если true, то нажатие Enter аналогично шелчку по этой кнопке)

Основные события

Click - щелчок мыши.

Объект управления и контроля Label (надпись)

Поле, заполняемое программистом текстовой информацией и недоступное пользователю для редактирования.

Основные свойства

Aligment - выравнивание

AutoSize - автоподбор размера

BorderStyle - тип границ: 1 - метка очерчивается одинарными линиями, 0 - контур отсутствует

Enabled - доступность

Основные события

Click, DblClick.

Объект управления и контроля Text Box (текстовое поле)

Позволяет пользователю вводить и редактировать текст.

Основные свойства

Font, ***BoderStyle***, ***Enabled***, ***Left***, ***Name***, ***Top*** ,***Visible***, ***Width***, ***Height*** - такие же, как и остальных элементов управления.

Text - содержимое текстового окна.

MaxLength - максимальная длина, если 0 - то не ограничена

MultiLine - многострочность: False - не более одной строки

Aligment - выравнивание, работает, если MulttiLine = True

PasswordChar - символ пароля

ScrollBar - линейки прокрутки

Основные события

Change - изменение

LostFocus - уход из фокуса.

Задание 1. Перевоз из радианов в градусы.

Написать программу, содержащую два текстовых поля, надписи: "Градусы", "Радианы", кнопки: "Перевод", "Выход".

Программа выполняет следующие функции.

✓ При щелчке на текстовые окна они очищаются;

✓ При нажатии на кнопку "Перевод", если в первом текстовом окне записано число, то во втором появляется соответствующее число в радианах и наоборот, если во втором текстовом окне записано число, то в первом появляется соответствующее число в градусах;

✓ При нажатии на кнопку "Выход" программа завершает свою работу.

Задание 2. Перевод чисел из одной системы счисления в другую.

1. Создать форму для программы, выполняющей перевод из 10-й системы счисления в 2-ю систему и наоборот.

2. Нанести на форму соответствующие задаче объекты управления и контроля. Изменить их свойства.

3. Написать программу, выполняющую следующие функции:

✓ Очистку текстовых окон;

✓ При нажатии на соответствующие кнопки должен осуществляться перевод из одной системы счисления в другую или программа должна завершать свою работу.

Объект управления и контроля Timer

Объект, который способен инициировать события через регулярные промежутки времени.

Основные свойства

Left, Top, Name - свойства, аналогичные свойствам других объектов.

Основное свойство таймера — ***Enabled***, показывает, включен или нет таймер на данный момент. Свойство ***Interval*** определяет, по истечении какого промежутка времени генерируется ***событие Timer*** (число в миллисекундах).

Пример 1. По истечении двух секунд кнопка на форме становится невидимой.

Изначально устанавливается свойства таймера: Enabled = True, Interval=2000.

```
Private Sub Timer1_Timer()
  Command1.Visible = False
End Sub
```

Если в событии Timer задать отключение таймера, то данное событие произойдет только один раз (см. пример 1). Если же в событии Timer для таймера задать условие, по которому таймер остается включенным или

отключается, то событие таймер будет выполняться условленное количество раз.

В приведенном ниже примере при нажатии кнопки включается таймер. В качестве условия действия таймера используется значение переменной k. Таймер действует до тех пор, пока значение k не станет равным 0, т. е. событие Timer выполнится 6 раз, и в надписи метки будут появляться цифры от 6 до 1, после чего появится восклицательный знак, и действие таймера прекратится.

Пример 2.

```
Dim k As Integer

Private Sub Command1_Click()
 Timer1.Enabled = True
 k = 6
End Sub

Private Sub Timer1_Timer()
 If k > 0 Then
   Label1.Caption = k
 Else
   Label1.Caption = "!"
   Timer1.Enabled = False
 End If
 k = k - 1
End Sub
```

Задание 3. Электронные часы.

1. Создать форму, как на рисунке, переименовать ее.

2. Нанести на форму текстовое окно и три командные кнопки. Изменить их свойства в соответствии с рисунком.

3. Поместить на форму таймер, задать свойство Interval.

4. Написать программу, выполняющую следующие функции:

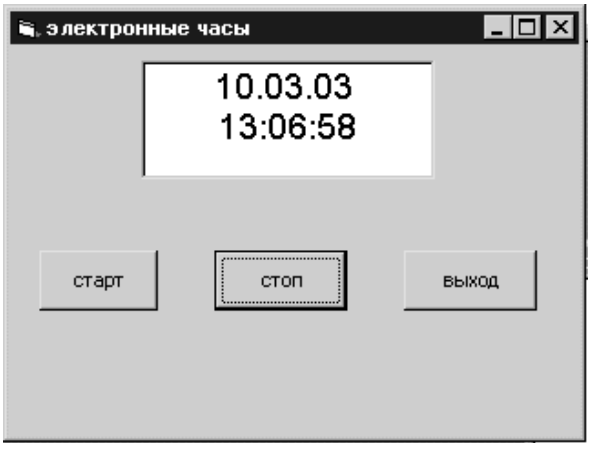


Рисунок 11

✓ При нажатии на кнопку "старт" в текстовом поле появляется текущее время, окно становится недоступным, часы "идут", т.е. каждую секунду меняется время;

✓ При нажатии на кнопку "стоп" в текстовом окне останавливается текущее время, а при повторном нажатии на кнопку "старт" часы продолжают идти;

✓ При нажатии на кнопку "выход" программа завершает свою работу.

ВЫПОЛНЕНИЕ ЗАДАНИЯ

Первая часть - визуальное программирование

1. Необходимо создать форму и переименовать ее в соответствии с рисунком.

2. Нанести на форму текстовое окно.

3. Нанести на форму Командные кнопки 1, 2 и 3, изменив их размеры и свойство Font, а также свойство Caption на «Старт», «Стоп» и «Выход», соответственно.

4. Нанести на форму объект Timer, присвоить свойству Interval значение 1000 (соответствует 1 секунде).

Вторая часть - написание кода программы.

1. В данном приложении не будут использоваться переменные, поэтому не требуется делать описания в процедуре Declarations объекта General.

2. Первое событие - загрузка формы. Необходима инициализация объекта Timer, свойству Enabled присвоить значение False, чтобы таймер не начал работу сразу после запуска программы.

```
Private Sub Form_Load ()
Timerl.Enabled = False
End Sub
```

3. Второе событие после загрузки формы - щелчок на кнопку «Старт». В процедуре обработки данного события свойству Enabled объекта Timer присвоить значение True, чтобы «включить» таймер.

```
Private Sub Command1_Click()
Timerl.Enabled = True
End Sub
```

4. Следующее событие Timer применимо к объекту Timer. Событие наступает, когда исчерпывается интервал времени в 1сек. В процедуре обработки данного события необходимо отразить в Текстовом окне текущее время и дату. Для этого можно использовать функцию Now (встроенная функция, возвращающая текущую дату и время).

```
Private Sub Timer1_Timer()
Text1.Text = Now
End Sub
```

5. Следующее событие - щелчок на кнопку «Стоп». В процедуре обработки данного события свойству Enabled объекта Timer присвоить значение False, чтобы «выключить» таймер, а значит, и электронные часы.

```
Private Sub Command2_Click()
Timer1.Enabled = False
End Sub
```

6. Последнее событие - щелчок на кнопку «Выход». В процедуре обработки этого события необходимо завершить работу программы.

```
Private Sub Command3_Click()
```

```
END
End Sub
```

Запустить программу на выполнение, протестировать ее и завершить.

Задание 4. Шутка.

Рисунок 12

1. Создать форму, как на рисунке, переименовать ее.

2. Нанести на форму Командную кнопку и Таймер. Изменить заголовок кнопки на «Начало».

3. Написать программу, выполняющую следующие функции:

✓ при нажатии на Кнопку «Начало» заголовок кнопки меняет свое название на «Выход» и запускается Таймер. По срабатыванию Таймера кнопка начинает перемещаться по форме произвольным образом;

✓ при нажатии на Кнопку «Выход» программа завершает работу.

Указание:

✓ предусмотреть, чтобы кнопка перемещалась в пределах формы, а Таймер начинал свою работу только по нажатию на кнопку «Начало»;

Объекты управления и контроля Horizontal Scroll Bar и Vertical Scroll Bar (Горизонтальная и вертикальная линейки прокрутки)

Очень распространенные управляющие элементы в Windows-приложениях. Дают возможность пользователю выбирать некоторое числовое значение, передвинув движок (ползунок) на полосе. Эта шкала может использоваться самостоятельно и служить устройством ввода или индикатором скорости, количества и т. п.

Основные свойства:

Свойства, аналогичные свойствам других объектов - ***Enabled, Height, Left, Name, Top, Visible, Width.***

Value (текущая позиция) - это свойство содержит число, которое отражает текущую позицию движка на линейке прокрутки

Min (минимум) - значение этого свойства в пределах целого типа. Определяет крайнюю левую или самую верхнюю позицию движка

Max (максимум) - значение этого свойства в пределах целого типа; определяет крайнюю правую или самую нижнюю позицию движка

LargeChange (постраничное изменение) - это свойство определяет величину, которая добавляется или вычитается из значения свойства Value при щелчке внутри линейки прокрутки

SmallChange (построчное изменение) - это свойство определяет величину, которая добавляется или вычитается из значения свойства Value при щелчке одной из стрелок, расположенных на концах линейки.

Основные события для линеек прокрутки:

Scroll (прокрутка) - непрерывно генерируется при перемещении (мышью) движка по линейке

Change (изменение) - это событие возникает после изменения позиции движка

Задание 5. Метроном.

1. Создать форму, как на рисунке, переименовать ее.

2. Нанести на форму две Командные Кнопки. Изменить их свойства в соответствии с рисунком.

3. Нанести на форму Горизонтальную линейку прокрутки, изменив ее свойства.

4. Поместить на форму Таймер, задать свойство Interval.

5. Написать программу, выполняющую следующие функции:

✓при нажатии на Кнопку «Старт» движок Линейки прокрутки начинает менять свое положение и перемещаться каждую секунду из крайнего левого до крайнего правого положения;

✓при нажатии на Кнопку «Стоп» движок линейки прекращает свое движение;

✓при нажатии на кнопку "Выход" программа завершает свою работу .

ВЫПОЛНЕНИЕ ЗАДАНИЯ

Первая часть - визуальное программирование.

1. Необходимо создать форму и переименовать ее в соответствии с рисунком, свойству Caption (название), присвоив значение «Метроном».

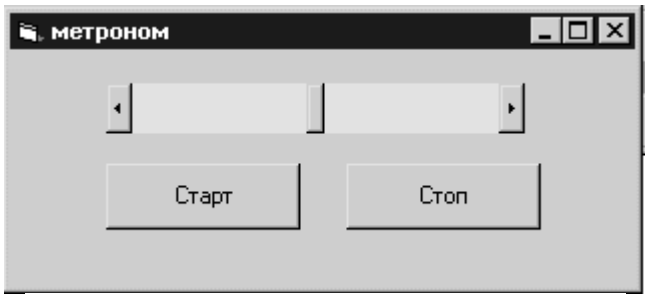


Рисунок 13

2. Нанести на форму Горизонтальную Линейку прокрутки, изменив свойства движка Min и Max на -1 и 1 соответственно, свойство Value на 0.

3. Нанести на форму Командные кнопки 1 и 2, изменив их размеры и свойство Font, а также свойство Caption на «Старт» и «Стоп», соответственно.

Нанести на форму объект Timer.

Вторая часть - написание кода программы.

1. В данном приложении будет использоваться переменная-счетчик k целого типа, поэтому требуется сделать ее описание в процедуре Declarations объекта General.

```
Dim k
```

2. Второе событие после загрузки формы - щелчок на кнопку "Старт". В процедуре обработки данного события свойству Interval объекта Timer присвоить значение 1000 (1 сек), чтобы включить таймер.

```
Private  Sub Command1_Click()
Timer1.Interval = 1000
End Sub
```

3. Следующее событие Timer применимо к объекту Timer. Событие наступает, когда исчерпывается интервал времени в 1сек. В процедуре обработки данного события необходимо увеличить счетчик на 1 и проверить его на четность. При четном значении переместить движок Линейки прокрутки влево, при нечетном - вправо, сопровождая перемещение движка звуковым сигналом (BEEP).

```
Private  Sub Timer1_Timer()
  k = k +  1
  ost = k Mod 2
  If  ost =  1  Then
    HScroll1.Value = HScroll1.Max
  Else
    HScroll1.Value  = HScroll1.Min
  End  If
  Beep
End  Sub
```

4. Следующее событие - щелчок на кнопку «Стоп». В процедуре обработки данного события необходимо завершить программу оператором END.

```
Private  Sub Command2__Click ()
  End
End Sub
```

5. Запустить программу на выполнение, протестировать ее и завершить.

Задание 6. Скорость.

1. Создать форму, как на рисунке, переименовать ее.

2. Нанести на форму Командную Кнопку. Изменить ее свойства в соответствии с рисунком.

3. Нанести на форму Горизонтальную линейку прокрутки, изменив ее свойства в

Рисунок 14

соответствии с рисунком.

4. Поместить на форму Текстовое окно, задав такие свойства, чтобы начальное значение скорости было 50 км/час.

5. Написать программу, выполняющую следующие функции:

✓ при изменении движка Линейки прокрутки начинает меняться значение скорости в Текстовом окне, в крайнем левом положении достигая нулевого значения, а в крайнем правом - 100 км/час;

✓ при нажатии на Кнопку «Выход» программа завершает свою работу.

Задание 7. Термометр.

1. Создать форму, как на рисунке, переименовать ее.

2. Нанести на форму три или более Меток и Вертикальную линейку прокрутки (для изображения термометра с градусами), Текстовое окно (для ввода температуры) и две Командные кнопки. Изменить их свойства в соответствии с рисунком.

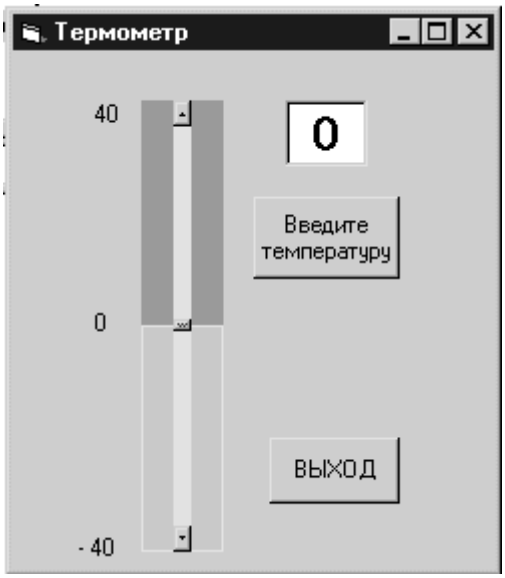


Рисунок 15

3. Написать программу, выполняющую следующие функции:

✓ при нажатии на Кнопку «Введите температуру» бегунок на линейке прокрутки перемещается до метки, соответствующей введенной в Текстовом окне температуре;

✓ при нажатии на Кнопку «Выход» программа завершает работу.

Указания:

А) Предусмотреть видимое перемещение бегунка по температурной шкале (используйте цикл и задержку в виде пустого оператора цикла For… Next).

В) Для обновления состояния вертикальной линейки прокрутки, используйте свойство формы AutoRedraw - автоматическая перерисовка.

Задание 8. Светофор.

1. Создать форму как на рисунке, переименовать ее.

2. Нанести на форму три командные кнопки «Включить», «Выключить», «Выход». Изменить их свойства в соответствии с рисунком.

Рисунок 16

3. Написать программу, выполняющую следующие функции:

✓ при загрузке формы появляется светофор в

выключенном состоянии (три не закрашенных круга);

- ✓ при нажатии на кнопку «Включить» светофор загорается (верхний круг закрашивается красным цветом, средний – желтым, нижний – зеленым;);
- ✓ при нажатии на кнопку «Выключить» светофор снова выключается;
- ✓ при нажатии на кнопку «Выход» программа завершает работу.

Указания:

А) Для рисования кругов используйте элемент управления Геометрическая фигура (Shape), для изменения формы — свойство Shape, для изменения цвета — свойство BackStyle (Oraque), FillStyle (0 Solid) и свойство FillColor (цвет заливки).

Измените программу таким образом, чтобы огни светофора загорались поочередно.

Тема 5. Процедуры. Модульный принцип построения проекта и программного кода. Создание проекта, состоящего из нескольких форм, создание исполнимого файла

Понятия процедуры и функции

Эти понятия аналогичны тем же понятиям в других языках программирования.

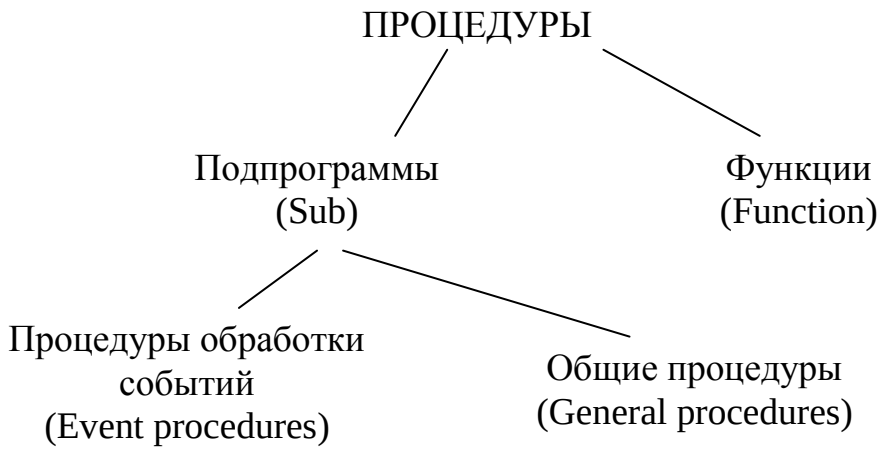


Рисунок 17. Классификация процедур, определяемых программистом

Будем говорить об ***общих процедурах и функциях***. Очень часто в процессе выполнения программы требуется многократное выполнение какой-либо ее части. Логичнее всего эту часть программы выделить, обозначить ее имя, задать ей необходимые параметры и вызвать эту часть по ее имени. Для этого используются процедуры и функции. Каждой вызываемой процедуры присваивается уникальное *имя*, устанавливается перечень *входных* и *выходных* параметров.

Входной параметр – это переменная, значение которой должно быть установлено до начала работы процедуры и которая участвует в работе процедуры.

Выходной параметр – это переменная, которая получает свое значение в результате работы процедуры.

Пример. Процедура рисования треугольника заданного цвета по заданным координатам его вершин. Треугольник – имя процедуры; x1, x2, x3, y1, y2, y3, цвет – входные параметры; выходных параметров нет.

Пример. Определение площади треугольника по формуле Герона. Площадь треугольника – название процедуры; сторона1, сторона2, сторона3 – входные параметры; площадь – выходной параметр.

Синтаксис оператора вызова процедуры следующий:

Call ИмяПроцедуры (СписокПараметровВызова)

или

ИмяПроцедуры (СписокПараметровВызова)

Среди параметров вызова могут быть входные и выходные параметры.

Объявление процедуры

[ОбластьВидимости] [**Static**] **Sub** ИмяПроцедуры_
(СписокПараметровВызова)
Выполняемые операторы
End Sub

Область видимости - **Public** или **Private**. Процедура локальная (private), если она доступна внутри данного модуля.

Наличие или отсутствие ключевого слова **Static** говорит о статусе локальных переменных. При наличии этого слова локальные переменные будут сохранять свои значения между последовательными вызовами этой процедуры.

Список параметров – это переменные (с их типами), которые играют роль входных и выходных параметров процедуры.

Объявление функции

[ОбластьВидимости] [**Static**] **Function** ИмяФункции_
(СписокПараметровВызова) **as** ТипЗначения
Выполняемые операторы
End Function

В VB ПараметрыВызова (аргументы) могут передаваться двумя способами: либо как ссылки (**ByRef**), либо как значение (**ByVal**).

Различие между двумя видами передачи аргументов состоит в том, что при передаче аргумента как ссылки можно изменять значение этого аргумента. **ByRef** можно опускать. Процедура может возвращать несколько значений. При вызове процедуры ей передаются аргументы, значение которых она может изменить. Если процедура не должна изменять аргументы, их следует передавать как значения. Для передачи аргументов в качестве значений перед именем аргумента ставится ключевое слово **ByVal** (должно быть обязательно). В этом случае процедуре передается копия этого значения.

Пример 1. Определить наибольшее из трех введенных чисел

```
Dim a As Integer
```

```
Dim b As Integer
Dim c As Integer

Function max(x, y) As Integer
If x > y Then m = x Else m = y
max = m
End Function

Private Sub Form_Load()
a = Val(InputBox("введите первое число"))
b = Val(InputBox("введите второе число"))
c = Val(InputBox("введите третье число"))
Text1.Text = max(max(a, b), max(a, c))
End Sub
```

Куда и как помещается программный код общей процедуры

Проект - совокупность экранных форм и программных кодов. Если программа сложная, то программный код удобно составлять не одним сплошным куском, а частями. Эти части называют модулями. Модуль - программный код - часть программного кода, который храниться в отдельном файле. Существует модуль формы - файл FRM. И стандартный (универсальный) - в нем хранится программный код, который можно использовать в нескольких формах, - BAS. Для создания модуля: Проект – Добавить модуль - Новый. Возможно использование уже существующих модулей: Проект – Добавить модуль – Существующий.

Пример 2. Найти все числа-палиндромы не превосходящие заданного n (см. диск). *Примечание*: запускаемый файл – файл проекта.

Пример 2.1. Вывести все четные трехзначные числа-палиндромы (см. диск). *Примечание:* для выполнения этого задания используется уже существующий модуль, созданный при выполнении предыдущего примера.

Задание 1. Вывести все двухзначные четные числа-палиндромы, используя уже существующий модуль с функцией palindrom(x).

Задание 2. Составить программу для вычисления суммы факториалов всех четных чисел от t до n.

Задание 3. Дано простое число. Написать программу, которая будет находить следующее за ним простое число (используйте логическую функцию, определяющую является ли число простым).

Задание 4. Выполните задание 2, используя специальный модуль.

Создание проекта, состоящего из нескольких форм

Типичное приложение имеет более одной формы. Для добавления в проект новой формы: Проект – Добавить Форму. Появится диалоговое окно, содержащее набор различных типов форм, вот некоторые из них:
Form – новая стандартная пустая форма;
About Dialog – диалоговое окно «О программе»;
Log In Dialog – диалоговое окно «Вход в систему»;
Splash Screen – экран-заставка.
Tip Of The Day – диалоговое окно «Совет дня».

Когда приложение начинает работу, загружается главная форма. После загрузки форма захватывает требуемые ресурсы, поэтому неиспользуемые формы из памяти можно выгружать. Но если формы содержат, например, большие растровые изображения. Которые загружаются и выгружаются довольно долго, то имеет смысл такие формы держать в памяти и отображать по мере надобности. Для загрузки и выгрузки формы используются операторы ***Load*** и ***Unload***. Синтаксис операторов:
Load FormName.
Unload FormName

Для отображения и скрытия формы используются методы ***Show*** и ***Hide***. Синтаксис операторов:
FormName.**Show**
FormName.**Hide**

Для указания главной формы: Проект – ИмяПроекта - Параметры - Загружаемый объект.

Элементы управления проекта являются доступными из разных форм. Обращение к ним осуществляется следующим образом: ИмяФормы.ИмяОбъектаНаФорме.Свойство = Значение

Задание 5. Цвета.
Создайте две формы. На первую нанести пять кнопок: «Красный», «Синий», «Желтый», «Зеленый», «Выход». На вторую форму нанести кнопку «Закрыть окно» и объект Фигура (Shape). Написать программу, выполняющую следующие функции:
✓ в зависимости от того, какая кнопка нажата, появляется вторая форма и объект Фигура закрашивается соответствующим цветом, при этом первая форма скрывается.
✓ при нажатии на кнопку «Закрыть окно» скрывается вторая форма и появляется первая форма;
✓ программа завершает свою работу при нажатии на кнопку «Выход».

Создание выполняемого файла (.exe)

Чтобы превратить готовый проект в выполняемый файл выберите команду Делать ProjectName.exe… (Make PiojectName.exe…) из меню Файл (File) программы, где ProjectName —имя вашего проекта. Программа VB создаст по указанному вами пути файл с заданным именем и расширением exe, который вы сможете загружать на компьютере непосредственно из программной оболочки типа Проводник или DN не открывая программу VB.

Задание 6. Сделать проект «Цвета» - исполнимым файлом.

Тема 6. Диалоговые окна

Функция и окно INPUTBOX

Окно вывода. Для ввода небольших фрагментов текста можно использовать окно ввода, точнее, функцию окна ввода.

Окно ввода имеет:

- строку заголовка,
- приглашение к вводу,
- поле ввода со значением,
- предлагаемым по умолчанию, кнопки: OK и Cancel.

Функция вызова InputBox имеет следующий синтаксис с соответствующими именованными аргументами:

Возвращаемое значение = **InputBox** (сообщение [, заголовок] [, значение по умолчанию] [, координата X верхнего левого угла] [, координата Y][,параметры, позволяющие открывать опред файлы справочной системы]

Пример 1.

```
Private Sub Command1_Click()
Dim s As String
s = InputBox("введите ваше имя")
MsgBox "привет " & s & " "
End Sub
```

Окно сообщений MSGBOX

Окно сообщений имеет:

- текст сообщений,
- заголовок,
- пиктограмму,
- набор кнопок.

Может использоваться как оператор и как функция.

Синтаксис процедуры:

MsgBox сообщение[, атрибуты] [, заголовок][,параметры, позволяющие открывать опред файлы справочной системы]

Атрибуты определяют особенности окна: пиктограмму на окне, кнопки.

Сообщение может вызываться при помощи функции - возвращаемое значение зависит от кнопки, нажатой пользователем. Синтаксис функции:

Возвращаемое значение = **MsgBox** сообщение[, атрибуты] [, заголовок][,параметры, позволяющие открывать определенные файлы справочной системы]

Значения, возвращаемые функцией MsgBox:

Таблица 2. Значения, возвращаемые функцией MsgBox

Константа	Значение	Нажатая кнопка
vbOk	1	ОК
vbCancel	2	Отмена
vbAbort	3	Стоп
vbRetry	4	Повторить
vbIgnore	5	Пропустить
vbYes	6	Да
vbNo	7	Нет

Функция inputbox возвращает строку, а msgbox - значение целого типа.

Пример 2.

```
Private Sub Form_Load()
 If MsgBox("Начать работу программы?", vbYesNo) = vbNo Then End
 End If
End Sub
```

Задание 1. Написать программу, которая запрашивает Фамилию, Имя, Отчество через окно сообщений и выводит их на форму.

Задание 2. Составить программу, которая запрашивает пароль (например, четырехзначное число) до тех пор, пока он не будет правильно введен. После введения правильного пароля появляется окно сообщения «Пароль введен правильно и загружается форма».

Задание 3. Составить программу, которая запрашивает пароль через текстовое поле (вводимый пароль отображается в виде звездочек, использовать свойство текстового поля PasswordChar). Необходимо ограничить время ввода и число попыток. По истечении определенного времени и числа попыток выводится окно сообщений «Программа завершает свою работу».

Тема 7. Тип данных: массив. Массив элементов управления. Элементы управления переключатель, флажок и рамка

Тип данных: массив

Массивы - тип данных, представляющий собой совокупность переменных одного типа. Возможно обращение к любому элементу массива, с помощью его индекса. В VB возможна реализация статических и динамических массивов

Описание статического массива в разделе описания переменных:

[**Static | Public | Dim**] Имя Переменной (ВерхняяГраница) [as ТипЭлементовМассива]

Dim s(2) as single - массив, состоящий их 3-х элементов.

Dim temp (1 to 30) as single – массив, состоящий из 30-ти элементов.

Dim b(13,35) as integer – двумерный массив размера 14х35.

Пример 1. Сформировать двумерный массив (NxN), найти сумму элементов массива находящихся на главной диагонали.

```
Const n = 3
Dim a(1 To n, 1 To n) As Integer
Dim sum

Private Sub Command2_Click()
End
End Sub

Private Sub Form_Load()
For j = 1 To n
For i = 1 To n
a(i, j) = InputBox("введите число")
Next i
Next j
End Sub

Private Sub Command1_Click()
sum = 0
For i = 1 To n
For j = 1 To n
Print a(i, j),
If i = j Then sum = sum + a(i, j)
Next j
Print
Next i
Print "Сумма элементов, находящихся на главной диагонали = "; sum
End Sub
```

Задание 1. Дан одномерный массив, представляющий собой набор температур первой недели марта. Определить: среднюю максимальную и минимальную температуры.

Задание 2. Дан двухмерный массив. При выводе на форму, поменяйте местами строки и столбцы массива.

В VB возможна работа с динамическими массивами. Описание динамического массива: первоначальное объявление делается с помощью слова Dim, при этом после имени массива в скобках ничего не указывается: **Dim** a() **as double**. В процессе работы можно многократно заново объявлять этот массив и каждый раз устанавливать для него верхнюю границу значений индекса: **ReDim** a(n). При переобъявлении массива, чтобы старые значения массива не стирались, после слова **ReDim** следует поставить ключевое слово **Preserve.** Можно воспользоваться встроенной функцией **UBound**, аргументом которой является имя массива, а значением - верхняя граница значений массива.

Пример 2. По ходу выполнения программы создать массив, состоящий из N элементов, вывести все четные элементы массива.

```
Dim a()
Dim n As Integer

Private Sub Form_Load()
n = InputBox("число элементов массива")
ReDim a(1 To n)
For i = 1 To n
a(i) = InputBox("введите значение ")
Next i
End Sub

Private Sub Command1_Click()
For i = 1 To n
 If a(i) Mod 2 = 0 Then Print a(i)
Next i
End Sub
```

Элемент управления Option Button (переключатель)

Кнопка-переключатель, иногда называется также радио-кнопкой. С ее помощью отображается устройство, индуцирующее включение или выключение чего-либо. Обычно такие кнопки используют как часть некоторой группы или множества устройств, из которых пользователь может выбрать только одно. Когда пользователь выбирает кнопку-переключатель, другие кнопки из той же группы автоматически

отключаются.

Основные свойства: ***Alignmemt***, ***BackColor***, ***Caption***, ***Font*** – такие же как у других элементов управления и контроля.

Value (значение) - при True кнопка активна, при False кнопка пассивна.

***Основные события для Кнопки опций*:**

Click (DblCIick) - событие наступает, когда пользователь делает щелчок (или двойной щелчок) на объект кнопкой мыши,

Элемент управления Check Box (флажок)

Окно контроля, отображающее информацию о включении или выключении каких-то операций в дальнейших действиях пользователя. В отличие от Кнопок опций, контрольных окон может быть выбрано любое число.

Основные свойства и события у Флажка такие же, как у Кнопки опций.

Элемент управления Frame (Рамка)

Объект Frame (Рамка) предназначен для помещения в него других объектов, или, как говорят, служит контейнером для других элементов управления. Изменение значения свойств объекта-контейнера будет влиять на соответствующие свойства всех составляющих его объектов. Чтобы поместить объекты в рамку (группу), необходимо проследить, чтобы перед выбором других объектов рамка находилась в активном состоянии. Нельзя поместить в группу уже существующие на форме элементы управления.

Основные свойства: Caption, BorderStyle, Enabled, FontBold, ***FontName, FonlSize, Height, Left, Top, Width, Name*** - свойства, аналогичные свойствам других объектов.

Visible, установленное в False, приведет к тому, что сама рамка и входящие в нее объекты исчезнут с экрана.

Массив элементов управления

Кроме обычных массивов, хранящих данные различного типа, в VB разрешается определять массивы объектов или элементов управления (control arrays), что весьма удобно, если в программе имеются группы объектов, действующих примерно одинаково. Такие массивы позволяют привязывать разные элементы управления к одной процедуре обработки события. Например. Если в программе создан массив из нескольких командных кнопок, щелчок на любой из них вызывает одну и ту же процедуру обработки события Click. В то же время VB дает возможность

различать конкретные объекты в массиве – это достигается передачей в процедуру индекса нужного элемента.

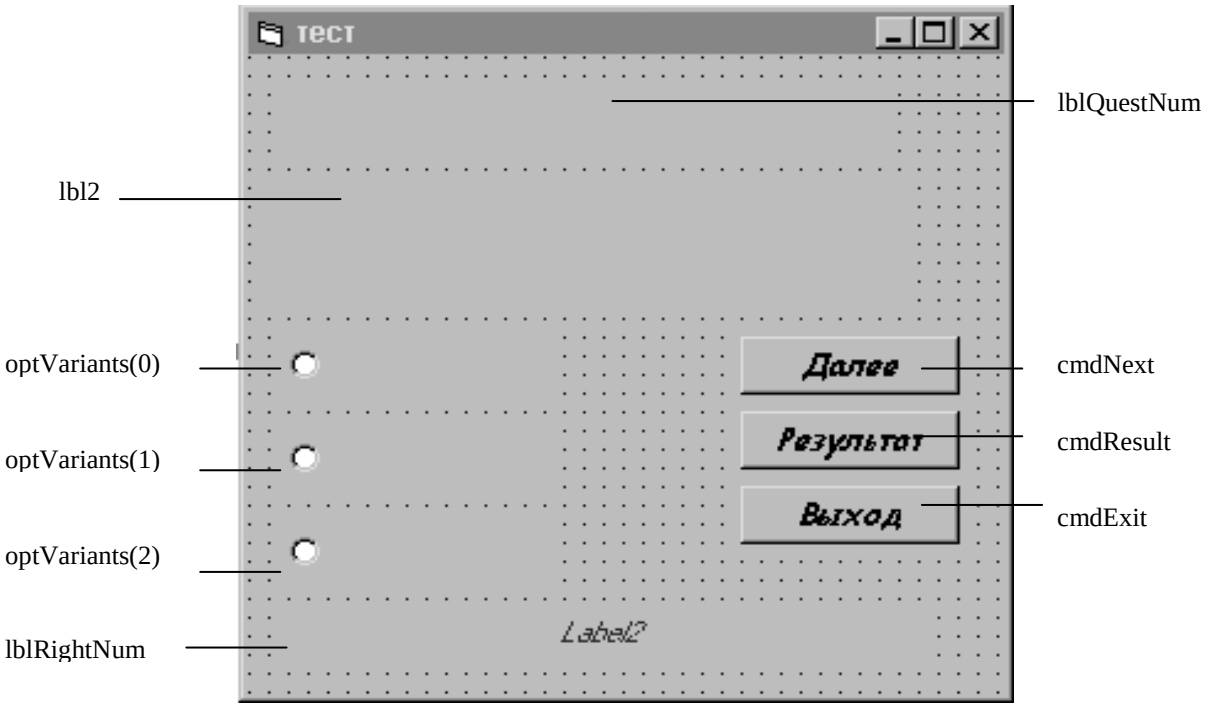
Как создать массив объектов?

1. нанести объект на форму;
2. скопировать его в буфер памяти (Правка - Копировать);
3. произвести вставку объекта из памяти (Правка - Вставить);
4. при появлении диалогового окна с вопросом «У Вас уже есть объект с таким именем. Вы желаете создать массив объектов?» ответить Да.

В созданном массиве VB присвоит каждому объекту индекс по порядку. Это отразится в значениях свойства Index у каждого объекта в окне Properties. Нумерация начинается с нуля. Чтобы обратиться к элементу массива объектов, необходимо указать имя массива, а за ним индекс в круглых скобках: cmdOk(0).Caption="Кнопка 1".

Задание 3. Создайте проект, состоящий из пяти вопросов с тремя вариантами ответов, который после завершения выдавал бы информацию о количестве правильных ответов (Используйте массивы).

Внесите следующие изменения в программу: после прохождения теста программа выставляет оценку, если не выбран не один вариант ответа, и нажата кнопка «Далее», появляется окно сообщений «Выберите вариант ответа».



```
Dim nq As Integer  'номера текущего вопроса
Dim na As Integer  'номер выбранного ответа
Dim i  As Integer  'счетчик правильных ответов
Dim q(1 To 3) As String 'массив вопросов
Dim a(1 To 3, 1 To 3) As String 'массив_
вариантов ответов
Dim ar(1 To 3) As Integer 'массив номеров_
правильных ответов

Sub Quest()              'процедура, при_
выполнении которой появляется номер_
текущего вопроса, вопрос и варианты ответа
lblQuestNum.Caption = "номер вопроса " & nq
lbl2.Caption = q(nq)
optVariants(0).Value = False
optVariants(1).Value = False
optVariants(2).Value = False
optVariants(0).Caption = a(nq, 1)
optVariants(1).Caption = a(nq, 2)
optVariants(2).Caption = a(nq, 3)
End Sub

Private Sub Form_Load()
lblRightNum.Visible = False
q(1) = "2x2=" 'формирование массива_ вопросов
q(2) = "5+6="
q(3) = "12/3="
a(1, 1) = "4" 'формирование массива ответов
a(1, 2) = "3"
a(1, 3) = "-2"
a(2, 1) = "7"
a(2, 2) = "11"
a(2, 3) = "1"
a(3, 1) = "36"
a(3, 2) = "5"
a(3, 3) = "4"
ar(1) = 1     'формирование массива номеров_
правильных ответов

ar(2) = 2
ar(3) = 3
nq = 1 'при загрузке формы номер вопроса=1
i = 0    'число правильных ответов=0
Quest
End Sub

Private Sub optVariants_Click(Index As Integer)
'процедура выбора ответа
na = Index + 1   'путем выбора переключателя
End Sub

Private Sub cmdNext_Click() 'процедура_ щелчка
на кнопке "Далее"'
If ar(nq) = na Then
i = i + 1
End If
If nq = 3 Then
cmdNext.Enabled = False
cmdResult.Enabled = True
Else
nq = nq + 1
Quest
End If
End Sub

Private Sub cmdResult_Click()
lblRightNum.Visible = True
lblRightNum.Caption = "Количество_ правильных
ответов:" & i & " из 3"
End Sub

Private Sub cmdExit_Click()
End
End Sub
```

Тема 8. Файлы: запись и чтение данных

Работа с файлом начинается с его открытия. Оператор открытия файла:

Open *Filename* **for** *Mode* **as** *Fileno*

Filename - имя открываемого файла (путь к нему).
Mode - режим доступа: (Append - дополнение, Input - ввода, Output - вывод) – последовательный доступ (текстовые файлы); Random – произвольный (для файлов, с постоянной структурой, типа БД); Binary – двоичный (возможен доступ к определенному байту файла).
Fileno - файловый номер - целое число, по которому идентифицируется файл: #1, #2…

Для считывания данных из файла, открытого для последовательного доступа, существует несколько возможностей:

- ✓ **Line Input #** - считывает одну строку;
- ✓ **Input #** - считывает последовательность символов, обычно записанных с помощью оператора **Write #;**
- ✓ **Input $** считывает определенное количество символов.

Для записи информации в файл используются операторы:

- ✓ **Print #** - для форматирования записываемой информации следует по-разному разделять данные в операторе **Print**. Если в операторе данные разделять запятыми (,), то в файле они будут разделены символами табуляции. Если же в операторе для разделения данных использовать точку с запятой (;), то данные в файл записываются без разделителей.
- ✓ **Write #**. Отличие оператора **Write #** от **Print #** состоит в следующем: если **Print #** сохраняет данные в виде обычного текста, то **Write #** заключает текстовые строки в кавычки, а цифры выводятся без кавычек.

```
Print #1, text1.text, «фрагмент1»; «фрагмент2»
```

После завершения работы с файлом его следует закрыть оператором **Close** с одним параметром - файловым номером:

Close *Fileno*

Пример 1. Чтение из файла.

```
Private Sub Command1_Click()
Do Until EOF(1)
   Line Input #1, strString
   strText = strText & strString & vbLf
Loop
Print strText
End Sub

Private Sub Command2_Click()
```

```
strText = Input$(LOF(1), 1)
Print strText
End Sub

Private Sub Command3_Click()
Close #1
End
End Sub

Private Sub Form_Load()
Open "c:\text.txt" For Input As #1
End Sub
```

Задание 1.

Создайте форму, как на рисунке, переименуйте ее.

Нанесите на форму Метку, Текстовое окно, две Кнопки Опций и две Командные Кнопки. Измените их свойства в соответствии с рисунком.

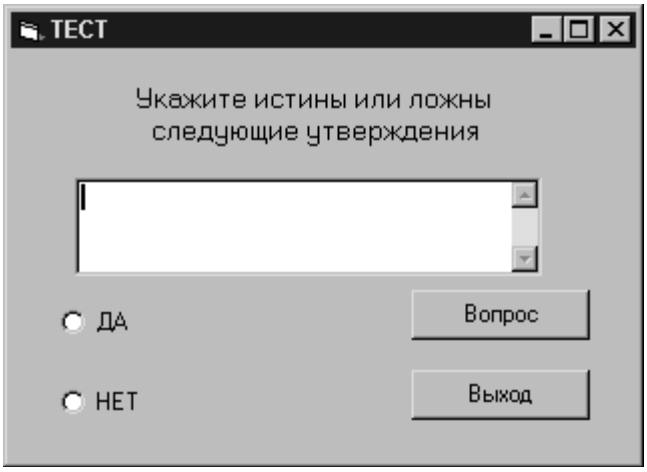


Рисунок 18

1. Напишите программу, выполняющую следующие функции:

- При нажатии на Кнопку «Вопрос» анализируется ответ ДА или НЕТ на предыдущий вопрос, считается количество правильных ответов, а в Текстовом окне появляется очередной вопрос теста. Когда все вопросы закончены, в Текстовом окне появляется сообщение о количестве правильных ответов;
- При нажатии на Кнопку «Выход» программа завершает свою работу.

ВЫПОЛНЕНИЕ ЗАДАНИЯ

Первая часть - визуальное программирование.

1. Необходимо создать форму и переименовать ее в соответствии с рисунком, свойству Caption (название) присвоив значение «ТЕСТ».

2. Нанести на форму метку, изменив свойство Caption на строку «Укажите, истинны или ложны следующие утверждения», свойство Font в соответствии с требованиями к размеру, цвету и начертанию букв, свойству Alignment присвоить значение 2 - Center.

3. Нанести на форму Текстовое окно, изменив свойство Text на пустую строку, свойству Multiline присвоив значение True, а свойству ScrollBars присвоив значение 2 - Vertical.

4. Нанести на форму Кнопки опций 1 и 2, изменив их размеры и свойство Font, а также свойство Caption на «ДА» и «НЕТ», соответственно.
5. Нанести на форму Командные кнопки 1 и 2, изменив их размеры и свойство Font, а также свойство Caption на «Вопрос» и «Выход» соответственно.
Вторая часть - написание кода программы.
6. В данном приложении будут использоваться переменная-счетчик k целого типа, нумерующая вопросы; переменная kp, считающая количество правильных ответов, и переменная n, содержащая номер правильного ответа, поэтому требуется сделать их описание в общем разделе объявлений General Declarations.

```
Dim k, kp, n
```

7. Первое событие - загрузка формы. В процедуре обработки данного события необходимо подготовиться к работе с файлом, в котором содержится тест с вопросами и номерами правильных ответов, то есть открыть его. Предполагается, что файл текстовый, последовательного доступа, организован следующим образом: первая строка - вопрос; вторая строка - номер правильного ответа: 1, если правильный ответ ДА, или 2, если правильный ответ НЕТ.

```
Private Sub Form_Load()
Open "test.txt" For Input As #1
End Sub
```

8. Второе событие после загрузки формы - щелчок на кнопку «Вопрос». В процедуре обработки данного события необходимо предусмотреть следующие действия:

- проверить номер правильного ответа и нажатие соответствующей Кнопки опций, в случае верного ответа увеличить счетчик правильных ответов на 1 (первая и вторая строки);
- увеличить счетчик вопросов на 1 (третья строка);
- кнопки опций сделать свободными для следующего нажатия (строки 4-я и 5-я);
- используя функцию проверки конца файла EOF (строка 6), прочитать из файла строку вопроса и строку с номером ответа (строки 7 и 8) с помощью оператора LINE INPUT;
- занести в текстовое окно вопрос (строка 9);
- при обнаружении конца файла в текстовое окно вывести сообщение о количестве правильных ответов (строка 11) и сделать кнопку «Вопрос» недоступной (строка 12).

```
Private Sub CmdQuestion_Click()
If n = 1 And OptYes.Value = True Then kp = kp + 1
If n = 2 And OptNo.Value = True Then kp = kp + 1
```

```
k = k + 1
OptYes.Value = False
OptNo.Value = False
If Not (EOF(1)) Then
   Line Input #1, a$
   Line Input #1, n
   TxtQuestion.Text = a$
Else
 TxtQuestion.Text = "Из " + Str$(k - 1) + " вопросов правильно отвечено на "_
+ Str$(kp)
 CmdQuestion.Enabled = False
End If
End Sub
```

9. Следующее событие - щелчок на кнопку «Выход». В процедуре обработки данного события необходимо завершить программу оператором END, предварительно закрыв файл с помощью оператора CLOSE.

```
Private Sub CmdExit_Click()
Close #1
End
End Sub
```

10. Запустить программу на выполнение, протестировать ее и завершить.

Задание 2. Внесите в программу первого задания следующие изменения: перед запуском программы запрашивается фамилия, имя, отчество ученика которые поле завершения теста вместе с количеством правильных ответов записываются в файл.

Тема 9. События клавиатуры и мыши, проектирование меню

Клавиатура

Фокус

Если говорят, что элемент управления имеет фокус, это означает, что ввод информации с клавиатуры относится к этому элементу. Элемент, имеющий фокус, можно распознать по различным визуальным признакам. Чаще всего это мигающий курсор или пунктирная рамка.

Клавиша табуляции

Перемещать фокус можно с помощью клавиши Tab. Все элементы управления, на которые можно установить фокус обладают свойством ***TabIndex***. Элемент управления, у которого свойство TabIndex равно 0 получает фокус сразу после загрузки формы. После каждого нажатия клавиши Tab фокус переходит к элементу со следующим значением TabIndex.

Горячие клавиши

В Windows почти во всех надписях на элементах управления есть подчеркнутые символы. С помощью клавиши Alt и этого символа можно переместить фокус на этот элемент управления или выполнять соответствующее действие. В VB при помощи свойства Caption можно задать горячую клавишу путем установки перед нужной буквой символа амперсанд, воспользовавшись окном свойств: command1.caption = ”&Exit”

События клавиатуры

Ввод с клавиатуры может обрабатываться не только Windows, но и непосредственно элементами управления. Для этого необходимо обрабатывать события: KeyDown, KeyPressed и KeyUp.

В качестве параметра процедуре обработки события ***KeyPressed*** передается переменная KeyASCII, содержащая ANSI-код нажатой клавиши (различия между ASCII- и ANSI-кодом клавиши важно только для символов с кодом, больше 127).

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
   MsgBox KeyAscii
End Sub
```

Событие ***KeyPressed*** вызывается только при нажатии клавиш, имеющих ANSI-код. А нажатие, например, клавиш управления курсором

или функциональных это событие не вызывает. Для обработки этих событий следует использовать события : ***KeyDown***, и ***KeyUp***.

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
   MsgBox KeyCode
End Sub
```

Параметр *KeyCode* содержит клавиатурный код нажатой клавиши, а *Shift* позволяет определить состояние клавиш Shift, Ctrl, Alt. Значения кодов отдельных клавиш можно добавить в программу как константы, воспользовавшись каталогом объектов (см. класс KeyCodeConstants). В качестве констант для аргумента Shift используются следующие значения:

Таблица 3

Константа	Значение	Описание
vbShiftMask	1	Нажата клавиша Shift
vbCtrlMask	2	Нажата клавиша Ctrl
vbAltMask	4	Нажата клавиша Alt

События мыши

Наиболее распространенные событиями мыши - **Click** (одинарный щелчок) и ***DblClick*** (двойной щелчок). Процедуры данных событий активируются, когда пользователь один или два раза (соответственно) щелкнет мышкой на объекте.

MouseDown - событие генерируется, если нажать кнопку мыши и не отпускать ее.

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

X, Y- параметры, определяющие положение указателя.

Button - значением данного параметра является:

1 - если была нажата левая кнопка мыши;

2 - если была нажата правая кнопка мыши;

4 - средняя кнопка

Shift - описывает состояние клавиш Shift, Ctrl, Alt на момент нажатия кнопки.

1- клавиша Shift;

2- клавиша Ctrl;

4 - клавиша Alt.

Сумма этих величин определяет различные сочетания клавиш. Например, значение 3 означает, что были нажаты Shift и Ctrl, а 7 - что все три управляющие клавиши были нажаты одновременно.

```
Private Sub Form_MouseDown(…)
  If (Button=2) And (Shift=4) Then MsgBox "Surprise-surprise!"
End Sub
```

MouseMove - генерируется при перемещении указателя мыши над элементом, в его процедуре можно изменить состояние объекта, находящегося под указателем, например, сделать его невидимым или передать ему фокус.

Private Sub target_MouseMove(Button As Integer, Shift As Integer, X As Single, _ YAs Single)

Параметры этого события идентичны параметрам события MouseDown. В следующем примере заголовок кнопки Exit меняется на Quit, когда над ней проходит указатель мыши:

```
Private Sub cmdExit_MouseMove(... )
  CmdExit. Caption = "Quit"
End Sub
```

MouseUp - событие, которое генерируется, если отпустить нажатую кнопку мыши над элементом.

Private Sub target_MouseUp(Button As Integer, Shift As Integer, X As Single, _ YAs Single)

Параметры этого события совпадают с параметрами двух предыдущих, Это событие включает в себя событие Click (в этом случае Button=l).

Кроме этого, вы также можете изменить вид указателя мыши на один из 12 видов, определенных для свойства MousePointer (Указатель мыши), или же можно загрузить свой собственный вид указателя с помощью свойства MouseIcon (Значок мыши) Установленные виды указателей мыши позволяют информировать пользователя о предстоящем использовании мыши. Если установить свойство MousePointer для какого-либо объекта формы, то при помещении указателя на этот объект он будет принимать заданный вид. Если установить свойство MP для формы целиком, то указатель будет иметь заданный вид в любой точке формы, независимо от того, над каким объектом он находится. Полный список значений свойства можно получить в окне Properties. Если свойству MP задать значение 99, то программа будет использовать вид указателя, заданный для свойства MouseIcon В приведенном ниже примере при помещении указателя мыши на кнопку cmdMove указатель мыши превращается в четырехстороннюю стрелку, показывая, что данную кнопку можно перемещать.

```
Private Sub cmdMove_MouseMove(... )
  CmdMove. MousePointer = 5
End Sub
```

События перетаскивания объекта

Рассмотрим следующие события - перетаскивание объектов прижиманием левой клавиши мыши. Чтобы добавить в вашу программу

поддержку операции перетащить-и-оставить, нужно выполнить следующие действия:

1. Позволить перетаскивание объекта, для этого требуется свойству DragMode объекта задать значение 1 или в процедуру загрузки формы ввести строку Object. Drag 1, где Object- имя перетаскиваемого объекта.

2. Определить вид значка, в который превращается указатель мыши при перетаскивании объекта - свойство объекта DragIcon. Значение этого свойства (файл с расширением. ico) либо задается в окне свойств объекта, либо изменяется в ходе программы.

По умолчанию VB использует прямоугольник, определяющий границы объекта.

3. Написать процедуру обработки событий DragDrop или DragOver для объекта назначения, т. е. того объекта, на который будут перетаскиваться объекты, которым в п.1 разрешено перетаскивание.

DragDrop (Перетащить-и-оставить) - событие генерируется при сбрасывании объекта, перетаскиваемого мышью

Private Sub target_DragDrop (Source As Control, X As Single, Y As Single)

Source - параметр, содержащий имя элемента, сброшенного на объект *target.*

X, Y- параметры, определяющие координаты указателя мыши на момент генерации события, пользуясь ими можно задать расположение элемента на принимающем объекте.

```
Private Sub fraPut__DragDrop (Source As Control, X As Single, Y As Single)
  Source. Left = X - Source. Width / 2
  Source. Top = Y - Source. Height / 2
End Sub
```

В этом примере объекты (здесь все они объединены ключевым словом Source) перетаскиваются на рамку fraPut и центр объекта помещается в том месте, где отпустили левую клавишу мыши.

DragOver - генерируется при протаскивании объекта над элементом Процедура события имеет следующий синтаксис:

Private Sub target_DragOver(Sourse As Control, X As Single, Y As Single, State _As Integer)

Source - имя протаскиваемого объекта

X, Y — координаты указателя мыши

State - определяет состояние перетаскиваемого объекта по отношению к принимающему объекту события, значения этого параметра.

0 - объект входит в границы приемника,

1 - объект покидает границы приемника,

2 - объект перетаскивается внутри границ приемника.

Пример 1. Цвет фона надписи lblColor изменяется в зависимости от положения объекта над ней - при вхождении объекта в границу надпись становится темно-серым, при нахождении объекта внутри границ надпись становится черной, а при выходе объекта за границу - белой.

```
Private Sub lblColor_DragOver(Sourse As Control, X As Single, Y As Single,_
State_ As Integer)
  Select Case State
    Case 0       'вхождение внутрь надписи
      LblColor. BackColor = &H80000003
    Case 1       'выход за границу надписи
      LblColor. BackColor = &H80000005
    Case 2       'перетаскивание объекта внутри надписи
      LblColor. BackColor = &H80000007
  End Select
End Sub
```

Задание 1. Игра «Поместите в нужный раздел».

Разработайте игру с пользовательским интерфейсом, приведенном на рисунке, и следующими правилами:

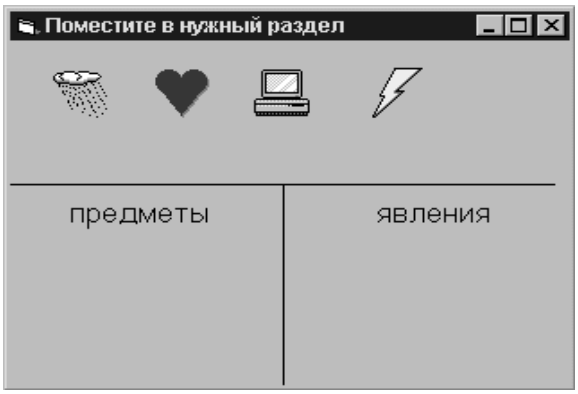


Рисунок 19

• из предложенного набора рисунков выбрать те, на которых изображены предметы, и переместить их в левый раздел с соответствующим названием.

• из предложенного набора рисунков выбрать те, на которых изображены явления природы, и переместить их в правый раздел с соответствующим названием.

• если изображение на рисунке не подходит ни под какой из разделов - сделать перемещение недопустимым.

ВЫПОЛНЕНИЕ ЗАДАНИЯ

Первая часть - визуальное программирование.

1. Необходимо создать форму и переименовать ее в соответствии с рисунком.

2. Нанести на форму Метку 1, изменив ее свойство Caption на «Предметы».

3. Нанести на форму Метку 2, изменив ее свойство Caption на «Явления».

4. Нанести на форму массив из 4 Окон рисунка (элемент управления Image), присвоив свойству Name значение pic, а в свойство Picture занести различные картинки (смотрите на рисунке: тучка, сердце, компьютер,

молния). Свойству DragMode для Окон рисунков придать значение 1.
Вторая часть - написание кода программы.
1. В процедуре обработки события DragDrop для Метки 1, являющейся целью, параметр Source является объектом-источником, а параметры X и Y - координатами курсора мыши относительно Метки 1. Для того чтобы разрешить «приземление» источнику, нужно проверить, является ли он предметом (предмет же в данном случае - только компьютер, изображенный в Picture Box с индексом 2).

```
Private Sub Label1_DragDrop(Source As Control, X As Single, Y As Single)
If Source.Index = 2 Then Source.Move X + Label1.Left, Y + Label1.Top
End Sub
```

2. В процедуре обработки события DragDrop для Метки 2 необходимо проверить, относится ли перемещаемое изображение к явлениям природы (таковыми являются изображения тучки - индекс 0 и изображение молнии - индекс 3).

```
Private Sub Label2_DragDrop(Source As Control, X As Single, Y As Single)
If Source.Index = 0 Or Source.Index = 3 Then
   Source.Move X + Label2.Left, Y + Label2.Top
End If
End Sub
```

3. Изображение сердца, имеющее индекс 1, не будет помещаться ни в один из разделов, так как к нему не применяется метод Move.

Задание 2. Корзина.

1. Создайте форму, как на рисунке, переименуйте ее в соответствии с заданием.

2. Нанесите на форму Окна изображения, используя иконки: корзина пустая, корзина заполненная, любые четыре предмета; кнопку "очистить корзину"; надпись, отображающую количество предметов в корзине.

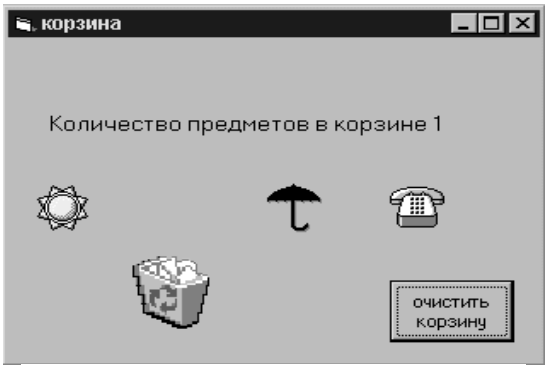


Рисунок 20

3. Напишите программу, выполняющую следующие функции:
✓При загрузке формы отображаются: пустая корзина, надпись и четыре предмета.
✓При перемещении любого из предметов в корзину, сам предмет исчезает, надпись показывает количество предметов в корзине, а корзина "становится заполненной".
✓В корзину нельзя поместить более двух предметов, при попытке поместить больше, появляется сообщение "Корзина переполнена".

✓При нажатии на кнопку "очистить корзину", корзина очищается, появляется сообщение: "Корзина пуста".

Проектирование меню

Создание меню в среде Visual Basic является очень легкой задачей. Но именно меню делает собственные программные разработки наиболее похожими на профессиональные WINDOWS-приложения. Создается меню с помощью Проектировщика меню *Menu Design* в более ранних версиях VB или Редактора меню *Menu Editor* (рисунок 1) в современных версиях VB. Для этого необходимо войти в пункт горизонтального меню *Инструменты (Tools)* и активизировать окно *Редактор меню (Menu Editor).*

Рисунок 21. Редактор меню

Заголовок (Caption). Содержит заголовок пункта меню, определяющий или название команды, или название выпадающего меню. Все, что вводится в этом окне, автоматически появляется в нижней части окна. Знак «&» около буквы дает подчеркивание этой буквы в пункте меню. Символ «-» создает разделительную черту в меню, например, перед пунктом Выход (рисунок).

Имя (Name). В текстовом окне для каждого из элементов, входящих в меню, вводится название или специальный идентификатор, который впоследствии используется при написании текста событийной процедуры, вызываемой при выборе этого пункта или команды

Индекс (Index). Текстовое окно *Index* позволяет ввести значение этого параметра для некоторого элемента, входящего в состав меню. После того, как введено значение параметра, управляющая структура автоматически становится частью массива подобных объектов.

Shortcut. В этом окне высвечивается полный список всех клавиш и комбинаций клавиш сокращенного доступа (горячие клавиши), которые можно назначать командам меню.

Windowlist. По умолчанию, присвоено значение *false.* Активизируется, если пункт меню содержит некоторые подпункты, образуя ниспадающее меню.

Checked. Рядом с названием будет поставлена галочка (при активизации). При выборе этого режима создается двоичный переключатель для обозначения того, выбрана ли данная команда в меню.

Enabled. Проверочное окно отмечено галочкой (по умолчанию), значение параметра *true.* В случае, когда команда в меню доступна, ее выбор влечет за собой выполнение назначенной ей событийной процедуры; если команда недоступна, то ее параметр устанавливается в *false* (выводится с пониженной яркостью и не реагирует на выбор).

Visible. Проверочное окно отмечено галочкой (по умолчанию). Для того чтобы сделать команду невидимой в меню, уберите отметку, после чего она станет недоступной и уберется из меню.

Кнопки для создания пунктов меню

• *Следующий (Next)* - кнопка используется при составлении меню, означает переход на следующую строку в созданном меню.

• *Вставить (Insert)* - с помощью данной кнопки можно вставить в меню новую строку.

• *Удалить (Delete)* - с помощью этой кнопки можно удалить строку в готовом меню.

• *Кнопки с изображениями стрелок* используются при создании подпунктов основного или ниспадающего меню, либо для изменения их порядка.

Существует *всплывающее меню (popup menu),* создаваемое с помощью метода PopupMenu. Статус всплывающего может быть присвоен любому пункту меню.

Задание 3.

На форме программы «Корзина» (задание 2) создайте главное меню, как на рисунке. При нажатии на любом из пунктов меню активизируется соответствующая команда.

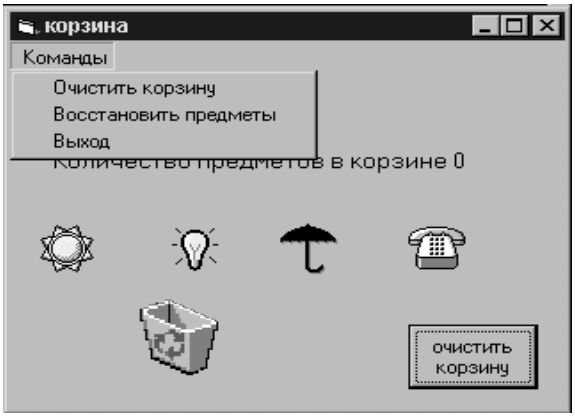


Рисунок 22

Тема 10. Графика и анимация в VB

Способы задания цвета объекта

Существует несколько способов задания цвета графического объекта:

1. использование системных цветов (их список вы можете найти в разделе System Color Constants в браузере объектов);

2. использование стандартных цветовых констант программы (функция VBColor, раздел ColorConstants браузера объектов);

Таблица 4. Значения некоторых цветовых констант

vbBlack	&HO	Black
vbRed	&HFF	Red
vbGreen	&HFFOO	Green
vbYellow	&HFFFF	Yellow
vbBIue	&HFFOOOO	Blue
vbMagenta	&HFFOOFF	Magenta
vbCyan	&HFFFFOO	Cyan
vbWhite	&HFFFFFF	White

3. использование палитры цветов и функции RGB, которая возвращает целое число, представляющее сочетание красного, зеленого и синего цветов: RGB(red, green, blue). Red — целое число от 0 до 255, обозначающее содержание красной компоненты цвета; Green - целое число от 0 до 255, обозначающее содержание зеленой компоненты цвета; Blue - целое число от 0 до 255, обозначающее содержание синей компоненты цвета. Если значение аргумента превышает 255, то оно автоматически считается равным 255. В следующей таблице представлены некоторые стандартные цвета и их красная, зеленая и синяя составляющие:

Таблица 5. Значение функции RGB для некоторых цветов

Color	Red Value	Green Value	Blue Value
Black	0	0	0
Blue	0	0	255
Green	0	255	0
Cyan	0	255	255
Red	255	0	0
Magenta	255	0	255
Yellow	255	255	0
White	255	255	255

Графические методы и свойства элементов управления и контроля Форма (Form) и Окно рисунка (Picture Box)

Как и в любой другой программе в VB имеются средства для создания графических объектов. А именно - вы можете рисовать линии и фигуры, задавать цвет и толщину рисунка, изменять и создавать пользовательские систему координат и единицы измерения, В VB существует только два объекта, поддерживающие графику - это форма и графическое поле (Picture Box), поэтому все нижеследующие относится только к ним. Рассмотрим свойства и методы, работающие с графикой.

Свойства

Back Color – цвет фона

ForeColor – свойство устанавливает цвет, используемый для отображения текста или графики в элементе управления

FillColor – свойство устанавливает цвет так называемых Shapes (рисованных объектов)

FillStyle - свойство устанавливает стиль заливки (7 значений)

DrawWidth - задает или возвращает в пикселях толщину линии рисунка. По умолчанию = 1

ScaleMode - задает или возвращает единицы измерения системы координат объекта (твипы, пиксели (3), миллиметры(6), сантиметры(7), либо создавать собственную систему координат (0)) По умолчанию единица измерения - твипы, значение = 1:

ScaleWidth и ***ScaleHeight*** - задают ширину и высоту внутренней части объекта в системе координат. При изменении этих свойств, свойство ScaleMode автоматически приравнивается к 0. Задавая положительные значения свойств, координаты увеличиваются слева направо и сверху вниз соответственно. При задании отрицательных значений свойств координаты будут расти справа налево и снизу вверх.

ScaleLeft, ***ScaleTop*** - задают или возвращают горизонтальные (ScaleLeft) и вертикальные (ScaleTop) координаты для левого и верхнего краев объекта

CurrentX, ***CurrentY*** - задают или возвращают горизонтальные (CurrentX) или вертикальные (CurrentY) координаты следующего графического метода или выводимого текста (с помощью метода Print) Используются только в программном коде, не изменяются в окне свойств. Координаты измеряются от верхнего левого угла объекта. Значение свойства Current X равно 0 на левом краю объекта, а Current Y - на верхнем.

Координаты выражаются в твипах, или текущих единицах измерения, определенных свойствами ScaleHeight, ScaleWidth, ScaleLeft, ScaleTop, и ScaleMode. Используя все четыре Scale свойства можно устанавливать полную систему координат с положительными и

отрицательными координатами

Основные графические методы

Cls - очищает графику и текст, образованные во время выполнения программы на форме Form или в графическом окне Picture Box:

object.**Cls**

Scale – позволяет переходить к собственной системе координат, это особенно удобно при построение графиков функций:

object. **Scale** (x1, y1) - (x2, y2)

где x1, y1, x2, y2 - координаты верхнего левого и нижнего левого угла соответственно

Point - возвращает как целое число типа Long красно-зелено-синий (RGB) цвет определенной точки на объекте:

object. **Point** (x, y)

где x, y - координаты точки на объекте, если координаты выходят за границы объекта, то метод Point возвращает 1

Pset - рисует точку на объекте определенного цвета:

object. **PSet** (x, y), [color]

color – необязательный параметр - число типа long, выражающее RGB цвет, определенный для точки, если аргумент опущен, то используется текущее значение свойства ForeCoIor, можно использовать функцию RGB или VBColor для определения цвета.

Пример 1. При щелчке правой клавишей мыши на форме в позиции курсора ставится точка толщиной 4 пикселя произвольного цвета.

```
Sub Form_MouseDown (...)
        If Button=2 Then
                DrawWidth = 4
                Pset (X, Y), RGB (255*Rnd, 255*Rnd, 255*Rnd)
        End If
End Sub
```

Line - рисует линии, и прямоугольник на объекте:

object. **Line**(xl, y1) - (x2, y2), [color], [B][F]

(xl, yl) – необязательные параметры, отдельное значение определяет координату (горизонтальную и вертикальную) начальной точки линии или верхнего левого угла прямоугольника, если значение пропущено, линия или прямоугольник начинаются с позиции, определенной свойств CurrentX и CurrentY;

(x2, y2) – обязательные значения определяют горизонтальную и вертикальную координаты конечной точки линии, либо нижнего правого угла прямоугольника;

Color - необязательный аргумент, задает цвет рисуемого объекта;

B - необязательная опция, если включена в свойство, то обозначает

рисование прямоугольника с заданными верхним левым и нижним правым углами. F необязательная опция. Если используется опция B, данная опция обозначает что прямоугольник закрасится в цвет границы. Нельзя использовать F без B. Чтобы нарисовать связанную линию, начинайте следующую линию с конечной точки предыдущей линии.

Пример 2. Рисование параболы на форме с заданием пользовательской системы координат.

```
Const xmax = 5, xmin = -5, ymax = 26, ymin = -4

Private Sub Form_Load()
Picture1.Scale (xmin, ymax)-(xmax, ymin) 'масштабирование графического
                                          ‘поля
End Sub

Private Sub Command1_Click() ‘при нажатии на кнопку рисуется парабола с
                             ‘сеткой координат
For i = xmin To xmax Step 0.5   ‘нанесение вертикальных линий
Picture1.Line (i, ymin)-(i, ymax)
Picture1.CurrentX = i
Picture1.CurrentY = 0
Picture1.Print i
Next i

For i = ymin To ymax Step 2    ‘нанесение горизонтальных линий
Picture1.Line (xmin, i)-(xmax, i)
Picture1.CurrentX = 0
Picture1.CurrentY = i
If (i < -0.1) Or (i > 0.1) Then Picture1.Print i
Next i

For i = xmin To xmax Step 0.01 ‘построение параболы точечным методом
   Picture1.PSet (i, i ^ 2)
Next i

End Sub
```

Circle - рисует окружность, эллипс или их дугу на объекте:

object. **Circle** (x, y), radius, [color, start, end, aspect]

(x, y) обязательные аргументы, обозначают координаты центра; radius обязательный аргумент, задает радиус окружности, эллипса или дуги;
color - необязательный аргумент, задает цвет окружности;

start, end – необязательны, при рисовании дуги окружности или эллипса задает начальное и конечное положение дуги. По умолчанию значение start - 0 радиан, end - 2*pi радиан; aspect необязательный агрумент, задает отношение осей эллипса (по умолчанию равно 1, что обозначает правильную окружность).

Пример 3. Рисование концентрических эллипсов.

```
radius = 50
Do While radius <> 400
   Circle (200, 200), radius, vbBlue, , , 2
   radius = radius+50
Loop
```

Объект управления и контроля Окно изображения (Image)

Image (Окно изображений), как и Picture Box (Окно рисунка), позволяет размещать графическую информацию в определенных участках формы. Окна рисунков - более гибкие объекты, но требуют большего объема памяти и времени на обработку. Они лучше подходят для динамической среды, когда при выполнении программы изображения рисуют непосредственно на экране или создают анимационные эффекты, перемещая значок по экрану. Графические объекты Окна изображений удобнее использовать в статической среде, то есть в тех случаях, когда созданную или скопированную растровую картинку не предполагается изменять.

Основные свойства:

Наряду со стандартными свойствами ***BackColor***, ***BorderStyle***, ***Enabled***, ***Left***, ***Name***, ***Top***, ***Visible***, ***Width*** и ***Height***, Окно рисунка и Окно изображения обладают еще одним важным свойством ***Picture*** (Картинка), которое позволяет выводить на экран либо растровые, либо векторные картинки. Для его настройки при проектировании программы служит окно Properties, а в период выполнения - функция LoadPicture(). Необходимо помнить, что координаты объекта в случае свойств Height, Left, Top и Width относительны, а не абсолютны, то есть изменяются как смещение от начала координат формы, а не экрана.

У Окна изображений имеют одно дополнительное свойство, отсутствующее у Окон рисунков, ***Stretch*** (масштабирование), по умолчанию, оно равно False**;** это приводит к автоматической подгонке размеров графического объекта - элемента управления под содержащуюся в нем картинку. При изменении значения False на True картинка будет масштабироваться в соответствии с размерами Окна изображения.

Основные события для Окон рисунков и изображений:

Click (DblClick) - событие наступает, когда пользователь делает щелчок (или двойной щелчок) на объект кнопкой мыши.

MouseDown, MouseUp - событие наступает, когда пользователь нажимает или отпускает левую кнопку мыши.

Процедуры и методы:

Функция ***LoadPicture*** (имя файла-картинки) устанавливает свойство Picture Окна рисунка или изображения.

Например: Picture1.Picture = LoadPicture («cloud.ico») загружает картинку с облаком в Окно рисунка.

Задание 1. Цветовая палитра.

1. Создайте форму, как на рисунке, переименуйте ее.

2. Нанесите на форму Окно рисунка и три Кнопки опций. Измените их свойства в соответствии с рисунком.

3. Напишите программу, выполняющую следующие функции:

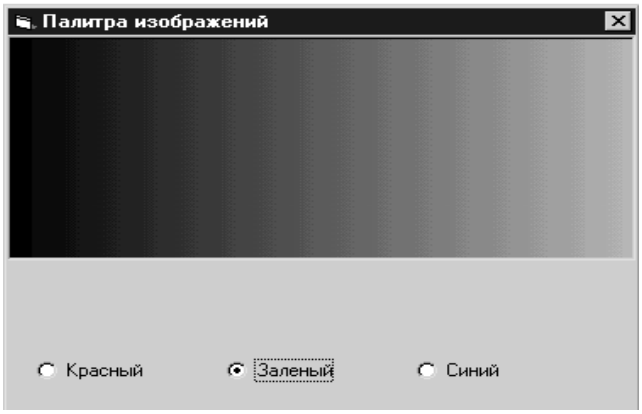


Рисунок 23

- При нажатии на одну из Кнопок опций в Окне рисунка появляется изображение палитры соответствующего цвета. Программа завершает работу при нажатии на кнопку «Закрыть» системного меню.

ВЫПОЛНЕНИЕ ЗАДАНИЯ.

Первая часть - визуальное программирование.

1. Необходимо создать форму и переименовать ее в соответствии с рисунком, свойству Caption присвоив значение «Цветовая палитра».

2. Нанести на форму Окно рисунка.

3. Нанести на форму Кнопки опций 1, 2 и 3, изменив их свойство Caption на «Красный», «Зеленый» и «Синий», соответственно.

Вторая часть - написание кода программы.

4. В данном приложении не будут использоваться никакие глобальные переменные, следовательно, нет описаний в общем разделе объявлений.

5. При загрузке формы не происходит никаких событий.

6. Событие после загрузки формы - щелчок на одну из Кнопок опций. В процедуре обработки данного события необходимо предусмотреть следующее действие: нарисовать палитру соответствующего цвета. Каким образом это можно осуществить?

✓ Создать общую процедуру DrawPalette, которая, в зависимости от входного параметра - цвета Base, заполняет окно рисунка линиями, используя метод Line и постепенно увеличивая номера цветов.

```
Sub DrawPalette(Base)
For i = 0 To 255
Picture1.ForeColor = i * Base
Picture1.Line (i * 15, 0)-(i * 15, Picture1.Height)
Next
End Sub
```

✓ При нажатии на Кнопку опций вызвать процедуру DrawPalette с соответствующим номером цвета: красным (&H1), зеленым (&H100) или синим (&H10000).

```
Private Sub Option1_Click()
Call DrawPalette(&H1)
End Sub

Private Sub Option2_Click()
Call DrawPalette(&H100)
End Sub

Private Sub Option3_Click()
Call DrawPalette(&H10000)
End Sub
```

8. Запустить программу на выполнение, протестировать ее и завершить.

Задание 2. Абстрактный рисунок.

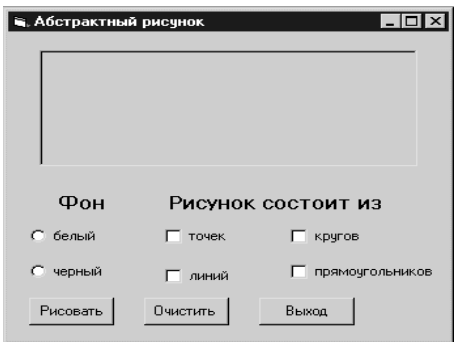


Рисунок 24

1. Создать форму как на рисунке, переименовать ее.

2. Нанести на форму Окно рисунка, выровнять его поверху формы (свойству Align придать значение 1)

3. Нанести на форму две метки «Фон» и «Рисунок состоит из».

4. Нанести на форму две кнопки опций с названиями «Белый» и «Черный»

5. Нанести на форму четыре Флажка с названиями «Точек», «Линий», «Кругов», «Прямоугольников».

6. Нанести на форму три Командные кнопки «Рисовать», «Очистить», «Выход».

7. Написать программу, выполняющую следующие функции:

✓ При загрузке формы появляется Окно рисунка серого цвета. Все переключатели и флажки выключены.

✓ При выборе одной из Кнопок опций в Окне рисунка устанавливается соответствующий цвет.

✓ Флажками помечаются те детали, из которых будет состоять абстрактный рисунок.
✓ При нажатии на Кнопку «Рисовать» в Окне рисунка появляются случайным образом разбросанные разноцветные фигуры из отмеченных флажками.
✓ При нажатии на Кнопку «Очистить» Окно рисунка очищается.
✓ При нажатии на Кнопку «Выход» программа завершает работу.

Задание 3. **Просмотрщик изображений.**

1. Создайте форму, как на рисунке, переименуйте ее.

2. Нанесите на форму Окно изображения и три Кнопки опций. Измените их свойства в соответствии с рисунком.

3. Напишите программу, выполняющую следующие функции:

При нажатии на одну из Кнопок опций в Окне рисунка появляется рисунок в соответствующем графическом формате. Причем, если это иконка формата ICO, размер ее увеличивается до размеров Окна изображения, если это фотография формата JPG или рисунок BMP, размер их уменьшается до размеров Окна. Программа завершает работу при нажатии на кнопку «Закрыть» системного меню.

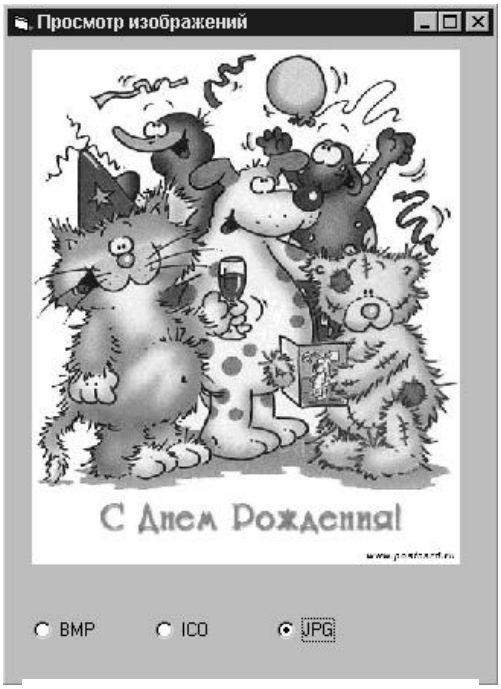


Рисунок 25

ВЫПОЛНЕНИЕ ЗАДАНИЯ

Первая часть - визуальное программирование.

7. Необходимо создать форму и переименовать ее в соответствии с рисунком.

8. Нанести на форму Окно изображения, свойству Stretch Окна изображения присвоить значение True для возможности масштабирования рисунков.

9. Нанести на форму Кнопки опций 1, 2 и 3, изменив их свойство Caption на «BMP», «ICO» и «JPG», соответственно.

Вторая часть - написание кода программы.

10. При загрузке формы можно сделать активной любую из Кнопок опций, например, третью.

```
Private Sub Form_Load()
Option3.Value = True
End Sub
```

11. Событие после загрузки формы - щелчок на одну из Кнопок опций. В процедуре обработки данного события необходимо предусмотреть следующее действие: загрузить рисунок из файла соответствующего формата с помощью функции LoadPicture().

```
Private Sub Option1_Click()
Image1.Picture = LoadPicture(path + "options.bmp")
End Sub

Private Sub Option2_Click()
Image1.Picture = LoadPicture(path + "Face03.ico")
End Sub

Private Sub Option3_Click()
Image1.Picture = LoadPicture(path + "mirinda.jpg")
End Sub
```

Path – путь к папке, в которой лежат картинки, описывается в разделе описания констант следующим образом: Const path = ".."

Задание 4. С Новым годом!

1. Создать форму как на рисунке, переименовать ее.

2. Нанести на форму Окно рисунка, Метку, Текстовое окно и две Командные кнопки. Изменить их свойства в соответствии с рисунком.

3. Написать программу, выполняющую следующие функции:

✓При загрузке формы появляется Окно рисунка с изображением елочки (рисунок необходимо заранее создать и сохранить в виде файла, а на этапе визуального программирования занести его в свойство Picture).

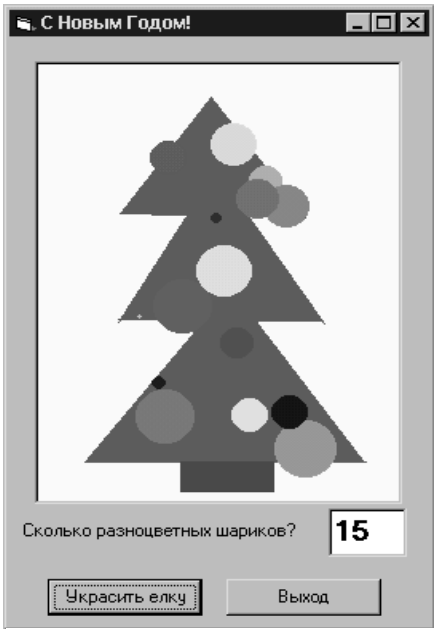


Рисунок 26

✓В изначально пустое Текстовое окно вводится количество шариков для украшения елочки.

✓При нажатии на Кнопку «Украсить елку» в Окне рисунка именно на зеленых ветках елочки, а не на стволе и не на пустом месте, появляются случайным образом разбросанные разноцветные шарики в том количестве, которое занесено в Текстовое окно. При желании процесс украшения повторяется.

✓При нажатии на Кнопку «Выход» программа завершает работу.

Указание:

Для того чтобы узнать, попадает или не попадает шарик на елочку, используйте метод Point, предназначенный для определения цвета точки с указанными координатами. Для упрощения такого определения лучше рисовать елочку чисто зеленым цветом, и тогда возможна, например, проверка типа
<= RGB(0, 255, 0).

Задание 5. Рисовальщик.

1. Создать форму как на рисунке, переименовать ее/

2. Нанести на форму два Окна рисунка, три Вертикальные линейки прокрутки, Метки с названиями «R», «G», «B2, «Ширина линии» и еще три метки для чисел, показывающих цвета RGB-палитры, одно Текстовое окно и две Командные кнопки «Сохранить» и «Выход». Изменить свойства всех элементов управления в соответствии с рисунком.

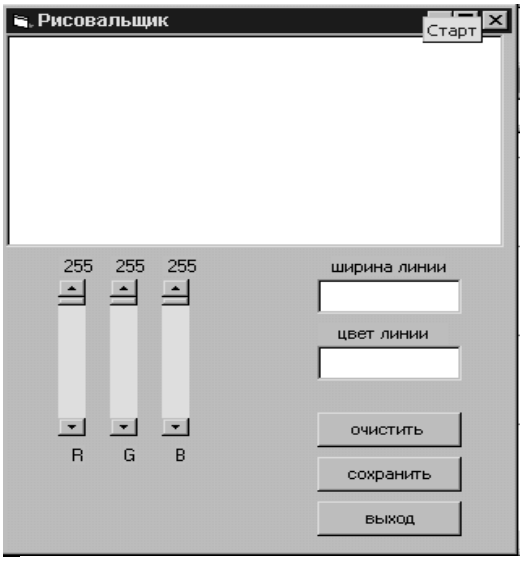


Рисунок 27

3. Написать программу, выполняющую следующие функции:

✓ В Текстовом окне можно вводить число для указания толщины линии, которой можно будет рисовать.

✓ С помощью движков на Линейках прокрутки устанавливается соответствующий уровень цветов RGB-палитры (R - красный, G - зеленый, B - синий), а в маленьком Окне рисунка показывается соответствующий комбинированный цвет.

✓ После установки параметров толщины линии и цвета с помощью мышки в основном Окне рисунка можно рисовать в режиме «карандаша» при нажатой левой клавише.

✓ При нажатии на Кнопку «Сохранить» рисунок сохраняется в файле в BMP-формате с произвольно взятым именем в текущем каталоге.

✓ При нажатии на Кнопку «Выход» программа завершает работу.

Указания:

✓Для того чтобы осуществить рисование с помощью мыши, используйте событийную процедуру Picture_MouseDown, параметры которой X, Y дают первоначальную координату точки в Окне рисунка при нажатии клавиши мыши.

✓Для непосредственного рисования используйте событийную процедуру Picture_MouseMove, параметры которой X, Y дают текущую координату точки в Окне рисунка при перемещении.
✓Для сохранения рисунка в файле используйте оператор SavePicture, который сохраняет графическое изображение Окна рисунка, используя его свойство Image.
Например, SavePicture Picture1.Image, «TEST.BMP».

Анимация

Анимация - создание эффекта движения за счет быстрого отображения объекта в разных точках экрана. Перетаскивание объектов представляет собой простейший вид анимации, поскольку при этом объект перемещается из одного места к другому на форме. Настоящая анимация включает в себя программное управление движением объекта, а зачастую и изменение размера изображения при его перемещении.

Каждый объект перемещается в системе относительных экранных координат формы, каждая форма имеет свою систему координат, представляющую собой сетку из строк и столбцов, наложенную на форму. Начало системы координат находится в верхнем левом углу формы, ось X направлена вправо, ось Y - вниз. Значение X находится в свойстве Left объект, а Y - в свойстве Top. Метод ***Move*** дает возможность перемещать объект на форме.

Object1.**Move** Left, Top

Object - имя перемещаемого объекта;
Left - расстояние между левой границей формы и объектом;
Top - расстояние между верхней границей формы и объектом.
Picture1.Move 1400, 600
Picture1.Move Picture1.Left-100, Picture1.Top+300

Суть анимации заключается в том, что один или несколько методов Move помещается в процедуру обработки события таймер, чтобы объект или объекты перемещались по экрану с интервалом времени, задаваемым таймером. Выбор значения свойства Interval таймера определяет скорость передвижения объекта по экрану (степень анимации). Свойства Top и Left перемещаемого объекта используются для остановки анимации - достигая определенного положения (например, одной из границ формы) объект останавливается, а таймер отключается.

Пример 4. При загрузке программы включается таймер и по истечении значения свойства Interval кнопка Command1 сдвигается на 50 единиц вверх. Так как после первого выполнения условного оператора таймер не отключается, то по истечении времени, заданного в свойстве Interval снова

генерируется событие Timer и так далее, пока объект не достигнет верхней границы, т.е. значение его свойства Top не станет равным 0.

```
Private Sub Form_Load()
Timer1.Enabled = True
Command1.Top = 3000
End Sub
Private Sub Timer1_Timer()
If Command1.Top > 0 Then
   Command1.Move 3000, Command1.Top - 50
Else
   Timer1.Enabled = False
End If
End Sub
```

Задание 6. Полет бабочки.

1. Создать приложение, в котором при нажатии на кнопку «Старт» по форме начинает перемещаться бабочка.

ВЫПОЛНЕНИЕ ЗАДАНИЯ

Первая часть - визуальное программирование

1. Необходимо создать форму и переименовать ее в соответствии с заданием - "Полет бабочки". Изменить цвет фона формы на белый.

2. Нанести на форму два окна рисунка. Изменить их свойства следующим образом: name первого - Picfly1, второго - Picfly2; свойство AutoSize на True; Appearance – на 0 (Flat); BorderStyle – на 0 (None); изменить свойства Picture, задав путь к файлу (…бабочка1.bmp, бабочка2.bmp - соответственно).

3. Нанести на форму Командную кнопку, изменить свойство Caption на «старт», Name - cmdrun.

4. Нанести на форму объект Timer, присвоить свойству Interval значение 100, свойству name – tmrfly, Enadled – False.

Вторая часть - написание кода программы.

5. Опишем переменную k, определяющую номер кадра, в процедуре Declarations объекта General.

```
Dim k As Integer
```

6. Первое событие - загрузка формы. При загрузке формы переместим окна рисунков в нижний левый угол экрана.

```
Private Sub Form_Load()
Picfly1.Top = Form1.Height - Picfly1.Height
Picfly2.Top = Form1.Height - Picfly2.Height
Picfly1.Left = 0
Picfly2.Left = 0
End Sub
```

7. Второе событие после загрузки формы - щелчок на кнопку «Старт». В процедуре обработки данного события свойству Enabled объекта Timer присвоить значение True, чтобы «включить» таймер.

```
Private Sub cmdrum_Click()
tmrfly.Enabled = True
End Sub
```

8. Следующее событие Timer применимо к объекту Timer. Событие наступает, когда исчерпывается интервал времени в 0,1 сек. В процедуре обработки данного события необходимо отображать рисунки поочередно, и перемещать их по экрану.

```
Private Sub tmrfly_Timer()
If (Picfly1.Top > 0) And (k Mod 2 <> 0) Then
Picfly1.Move Picfly1.Left + 100, Picfly1.Top - 100
Picfly1.Visible = True
Picfly2.Visible = False
ElseIf (Picfly2.Top > 0) And (k Mod 2 = 0) Then
Picfly2.Move Picfly2.Left + 100, Picfly2.Top - 100
Picfly2.Visible = True
Picfly1.Visible = False
Else
tmrfly.Enabled = False
Picfly1.Visible = False
Picfly2.Visible = False
End If
k = k + 1
End Sub
```

9. Запустить программу на выполнение, протестировать ее и завершить.

Тема 11. Другие элементы управления и контроля, их использование

Элемент управления и контроля ListBox (Список)

Объект ListBox (Список) используется обычно в программах для наглядного представления информации, а также для предоставления пользователю возможности выбора одного (или нескольких) элемента из некоторого перечня.

Ниже перечислены основные элементы списка.

- Список элементов.
- Выбранный элемент (либо подсвечивается, либо помечается флажком в зависимости от типа списка).
- Полоса прокрутки.

Основные свойства:

Enabled, ***Font***, ***Left***, ***Top***, ***Width***, ***Name*** - свойства, аналогичные свойствам других объектов.

Columns (колонки) - количество колонок в списке, по умолчанию, равно нулю, что соответствует одной колонке

List (список) - представляет собой массив, состоящий из элементов списка

ListCount - количество элементов в списке

Listlndex (индекс текущего элемента в списке) - соответствует номеру последнего подсвеченного элемента списка. Индекс первого элемента списка равен нулю

Sorted (сортировка) - если это свойство приравнено True, то элементы в списке располагаются по алфавиту

Style (вид списка) - по умолчанию, равен нулю, при задании значения 1 список приобретает вид с элементами-флажками.

Основные методы:

AddItem - включает элемент в список (синтаксис: Listl.Addltem текст [,индекс])

RemoveItem - удаляет элемент из списка (синтаксис: Listl.Removcltem индекс)

Clear - удаляет из списка все элементы (синтаксис: List1.Clear).

Пример 1. Калькулятор.

В первые два текстовых поля записываются числа, в зависимости от того, какой математический оператор в поле со списком был выбран, в третье текстовом окне появляется результат сложения, вычитания, умножения или деления этих двух чисел.

Рисунок 28

```
Dim a, b, c As Double

Private Sub List1_Click()
a = Val(Text1.Text)
b = Val(Text2.Text)
Select Case List1.ListIndex
Case 0
Text3.Text = a + b
Case 1
Text3.Text = a - b
Case 2
Text3.Text = a * b
Case 3
Text3.Text = a / b
End Select
End Sub
```

Задание 1. Калькулятор 2.
Создать приложений как на рисунке. Написать программу, выполняющую следующие действия: в первое текстовое поле вводится угол в градусах, в зависимости от выбранного элемента списка (SIN, COS, TN, CTN), во втором текстовом поле находится: синус введенного угла, косинус, тангенс, или котангенс.

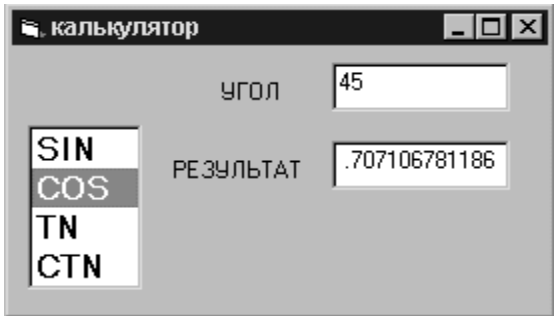


Рисунок 29

Элемент управления Combo Box (Комбинированный Список)

Объект Combo Box (Комбинированный Список) представляет собой объединение Текстового окна и обычного Списка. В отличие от обычного Комбинированный Список не позволяет размещать элементы в несколько колонок, но дает пользователю возможность выбора одного из трех режимов работы. Ниже перечислены основные элементы Комбинированного Списка (см. рисунок).

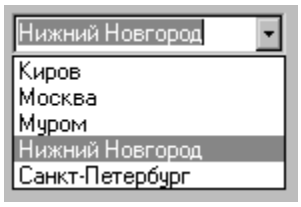


Рисунок 30

- Поле ввода (или текстовое окно).
- Список элементов.
- Выбранный элемент (подсвечивается и отображается в поле ввода).
- Полоса прокрутки появляется в случае длинного списка.

Основные свойства:

Кроме свойств, аналогичных свойствам Списка, Комбинированный Список обладает следующими:

Style (стиль списка) - определяет стиль (или тип) Комбинированного Списка и его режим работы, принимает значения 0, 1 и 2. По умолчанию свойство Style равно 0, список является раскрывающимся (drop-down combo box). Изначально представляет собой одну строку, но при щелчке на стрелку на правой стороне объекта разворачивается в список. Может быть выбран элемент из списка или введено значение в текстовом поле. При задании значения Style=l (simple combo box) список не раскрывается, а остается открытым на экране. Пользователь может вводить текст в поле или выбирать элемент из списка. Для Style=2 (drop-down list box) возможен только выбор элемента из списка.

List (список) - содержит список элементов по одному в каждой строке; вводить очередной элемент можно программно или во время разработки приложения, при этом для перехода на новую строку необходимо нажать клавиши <Ctrl+ Enter>

Установка начального значения поля ввода в Комбинированном Списке осуществляется для стилей 0 и 1 с помощью свойства Text или программно с помощью свойства Listlndex.

Text (текст) - содержит либо текст, набранный в поле ввода, либо текст элемента, выделенного в окне списка

Основные методы:

Методы ***Addltem***, ***Removeltem***, ***Clear*** с Комбинированным Списком работают так же, как и с обычным.

Пример 2. Добавление элементов к комбинированному списку: при нажатии на кнопку "Добавить".

```
Private Sub Command1_Click()
Combo1.AddItem Text1.Text
End Sub
```

Элемент управления DriveListBox (список устройств)

Элемент управления DriveListBox относится к группе элементов управления, предназначенных для отображения и работы с дисками, каталогами и файлами. DriveListBox служит для отображения списка всех доступных дисков и устройств системы и обеспечивает возможность их выбора.

Основные события

Событие ***Change*** вызывается при смене носителя данных.

Свойства

Элемент ***DriveListBox*** обладает почти всеми обычного поля со списком, но чаще используется только свойство ***Drive***, возвращающее выбранный диск или устройство (например C:\).

Элемент управления DirectoryListBox (список каталогов)

DirListBox – это второй элемент управления, предназначенный для выбора файлов. Он отображает структуру выбранного диска и позволяет осуществлять выбор и смену каталога.

Основное событие

Событие ***Change*** вызывается в результате двойного щелчка мыши по имени каталога в окне просмотра.

Свойства

Главным свойством является свойство ***Path***, возвращающее полный путь к выбранному каталогу, включая имя диска, например: C:\Windows\Word.

После добавления в форму элементов управления DriveListBox и DirListBox они еще не работают совместно. То есть в один и тот же момент в DriveListBox может отображаться имя диска C, а в DirListBox – структура каталогов диска D. Поэтому прежде чем использовать эти элементы управления, их необходимо синхронизировать. Это происходит при обработке события Change в DriveListBox:

```
Private Sub Drive1_Change()
   Dir1.Path = Drive1.Drive
End Sub
```

Элемент управления и контроля FileListBox (список файлов)

Последний элемент управления, который можно использовать для выбора файлов. Он отображает файлы текущего каталога, откуда их можно выбирать.

Основные события

Основное событие ***Click***, которое вызывается при выборе пользователем имени файла в списке.

PathChange происходит после изменения пути (свойство Path)

PatternChange – после изменения маски выбора файлов (свойство Pattern)

Основные свойства

FileName – содержит имя выбранного файла

Pattern – позволяет определить тип тех файлов, которые должны отображаться в списке: File1.Pattern=”*.ico”;”*.bmp”

Список файлов также должен синхронизироваться с выбранными устройствами и каталогом. Это происходит при обработке события Change для DirListBox, при этом используется свойство Path элемента FileListBox:

```
Private Sub Dir1_Change()
   File1.Path = Dir1.Path
End Sub
```

Задание 2. Определение размера файла.

1. Создайте форму, как на рисунке, переименуйте ее.
2. Нанесите объекты управления, измените их свойство Name в соответствии с рисунком.
3. Напишите программу, выполняющую следующие функции:
✓ существует возможность выбора файла
✓ при нажатии на кнопку ОК, появляется окно сообщений, содержащее информацию о местоположении, имени и размере выбранного файла
✓ при нажатии на кнопку Cancel приложение завершает работу.
ВЫПОЛНЕНИЕ ЗАДАНИЯ

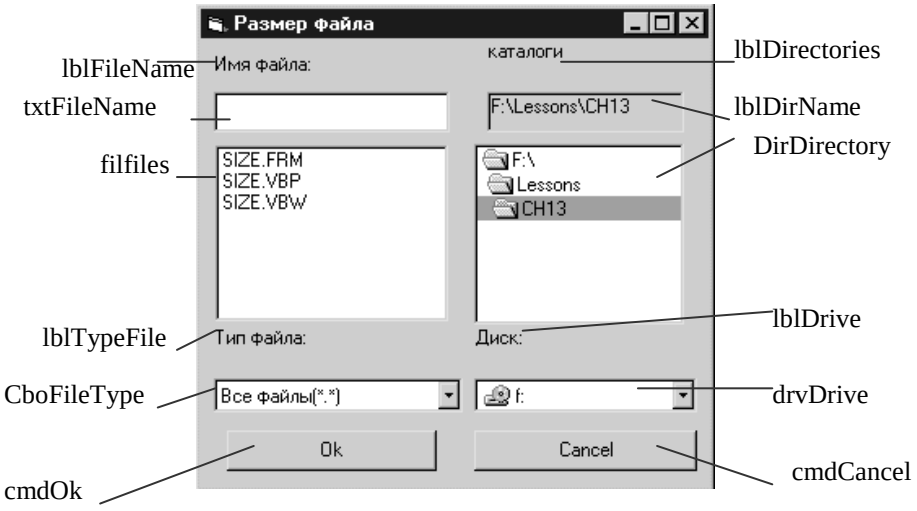


Рисунок 31

Первая часть - визуальное программирование
1. Создайте форму и переименуйте ее в соответствии с рисунком.
2. Нанесите на форму следующие элементы управления и измените их свойства Name следующим образом.

Элемент управления	Соответствующее свойство Name
Label	lblFileName
TextBox	txtFileName
FileListBox	Filfiles
Label	lblTypeFile
ComboBox	CboFileType
CommandButton	cmdOk
Label	lblDirectories
Label	lblDirName
DirListBox	DirDirectory
Label	lblDrive
DriveListBox	drvDrive
CommandButton	cmdCancel

Вторая часть – написание кода программы

1. Первое событие – загрузка формы. Происходит заполнение комбинированного списка с расширениями файлов; и в надписи lblDirName отображается путь к текущей директории:

```
Private Sub Form_Load()
   CboFileType.AddItem "Все файлы(*.*)
   CboFileType.AddItem "Текстовые файлы(*.txt) "
   CboFileType.AddItem "Документы Word(*.doc)"
   CboFileType.ListIndex = 0
   lblDirName.Caption = DirDirectory.path
End Sub
```

2. Второе событие смены диска, при котором происходит синхронизация объектов DirDirectory и drvDrive:

```
Private Sub drvDrive_Change()
   DirDirectory.path = drvDrive.Drive
End Sub
```

3. Третье событие - событие смены директории (папки), при котором синхронизируются объекты filfiles и DirDirectory; в надписи отображается путь к текущей директории:

```
Private Sub DirDirectory_Change()
   filfiles.path = DirDirectory.path
   lblDirName.Caption = DirDirectory.path
End Sub
```

4. Четвертое событие – выбор типа файлов, отображаемых в списке файлов с помощью комбинированного списка CboFileType. В зависимости от выбранного расширения, свойство Pattern (определяет тип файлов, которые должны отображаться в списке) принимает разное значение:

```
Private Sub CboFileType_Click()
   Select Case CboFileType.ListIndex
   Case 0
   filfiles.Pattern = "*.*"
   Case 1
   filfiles.Pattern = "*.txt"
   Case 2
   filfiles.Pattern = "*.Doc*"
   End Select
End Sub
```

5. Следующее событие - 'выбор файла, при котором имя файла отображается в поле txtFileName:

```
Private Sub filfiles_Click()
   txtFileName.Text = filfiles.FileName
End Sub
```

6. При нажатии на кнопку ОК, определяется размер файла

```
Private Sub cmdOk_Click()
   Dim PathAndName As String 'переменная хранит путь и имя файла
   Dim FileSize As String    'размер файла
   Dim path As String        'путь к файлу
```

```
    If txtFileName.Text = "" Then
       MsgBox "Файл не выбран!"
       Exit Sub
    End If
    If Right(filfiles.path, 1) <> "\" Then
       path = filfiles.path + "\"
    Else
       path = filfiles.path
    End If
    PathAndName = path + filfiles.FileName
    FileSize = Str(FileLen(PathAndName))
    MsgBox "Размер файла " + PathAndName + ": " + FileSize + " bites"
    Exit Sub
End Sub
```

7. Последнее событие – нажатие на кнопку Cancel – выход из программы.

```
Private Sub cmdCancel_Click()
   End
End Sub
```

Дополнительные элементы управления и контроля

При создании Visual Basic были предусмотрены специальные средства для расширения набора элементов управления и контроля, доступных программе. Для этого Вы можете выбрать пункт меню Components (контекстное меню Окна инструментария Toolbox, либо в пункте Project главного меню). В появившемся списке вкладки Controls поставьте галочку против нужного элемента управления. Дополнительные стандартные элементы управления и контроля расположены в следующих группах: *Microsoft Windows Common Controls 6.0* (содержит, например, такие объекты, как ImageList - Список изображений, StatusBar - Строка состояния, TreeView - Дерево просмотра, ProgressBar - Индикатор процесса и другие); *Microsoft Windows Common Controls-2 6.0* (содержит такие объекты, как UpDown - Счетчик., Month View - Календарь); *Microsoft Windows Common Controls-3 6.0* (содержит объект CoolBar - Панель инструментов). Кроме стандартных компонент, можно подключать к своим разработкам и другие интересные компоненты.

Для добавления такого элемента:

1. Добавить инструмент в Toolbox (панель инструментов). Проект – компоненты, в открывшемся окне в списке Control указать группу элементов управления. Выбранный инструмент или группа окажутся в наборе Toolbox.

2. Разместить объект на экранной форме.

Элемент управления и контроля Common Dialog (общий диалог)

Рассмотрим пример использования дополнительного элемента управления и контроля CommonDialog (общий диалог), который позволяет выполнять одну из шести следующих функций:

- ✓ Открывать файл
- ✓ Сохранять файл
- ✓ Выбирать шрифт
- ✓ Управлять печатью
- ✓ Выбирать цвет
- ✓ Получать справочную информацию.

Задание 3

На форме рисуется линия, цвет линии задается с помощью диалогового окна выбора цвета.

1. Создать форму «Выбор цвета»

2. Нанести на форму линию (Line), изменить свойство BorderWidth на 3.

3. Нанести кнопку «Выбрать цвет», при нажатии на которую появляется диалоговое окно выбора цвета.

4. Вынести объект CommonDialog на панель элементов управления. Проект – Компоненты, выделить флажком Microsoft Common Dialog Control 6.0.

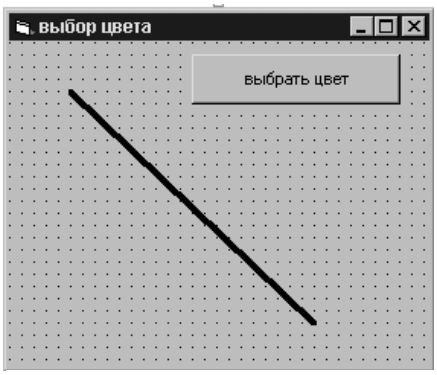


Рисунок 32

5. Разместить данный объект на форме.

6. Написать программный код для события – нажатие на кнопку «выбрать цвет»:

```
Private Sub Command1_Click()
  CommonDialog1.ShowColor
  Line1.BorderColor = CommonDialog1.Color
End Sub
```

ЧАСТЬ II. Моделирование в среде VISUAL BASIC

Тема 1. Теоретические основы моделирования

Введение

Тема «Компьютерное моделирование в Visual Basic» посвящена той из компьютерных технологий обработки информации, ради которой когда-то создали первую ЭВМ и ради которой сегодня в значительной мере создают супер-ЭВМ - решению прикладных научно-технических задач, среди которых задачи математического моделирования составляют видную долю.

Что же такое модель? Что общего между игрушечным корабликом и рисунком на экране компьютера, изображающим сложную математическую абстракцию? И все же общее есть: и в том, и в другом случае мы имеем образ реального объекта или явления, «заместителя» некоторого «оригинала», воспроизводящего его с той или иной достоверностью и подробностью. Или то же самое другими словами: модель является представлением объекта в некоторой форме, отличной от формы его реального существования.

Практически во всех науках о природе, живой и неживой, об обществе, построение и использование моделей является мощным орудием познания. Реальные объекты и процессы бывают столь многогранны и сложны, что лучшим способом их изучения часто является построение модели, отображающей лишь какую-то грань реальности и потому многократно более простой, чем эта реальность, и исследование вначале этой модели. Многовековой опыт развития науки доказал на практике плодотворность такого подхода.

В моделировании есть два заметно разных пути. Модель может быть похожей копией объекта, выполненной из другого материала, в другом масштабе, с отсутствием ряда деталей – материальная модель. Модель может отображать реальность более абстрактно – словесным описанием в свободной форме, описанием, формализованным по каким либо правила, математическими соотношениями и т.д.

В данной теме предлагаются к рассмотрению математические компьютерные модели (разновидность абстрактных моделей). Данные вид моделей выражает существенные черты объекта или процесса языком уравнений и других математических средств, инструментом моделирования выступает компьютер, если более точно, система программирования Visual Basic. Компьютерные математические модели дают широкие возможности представить изучаемые процессы наглядно с различных сторон, а также способствует углубленному их пониманию. Но самым замечательным является то, что кроме создания простых моделей,

адекватных некоторым сложным процессам, можно еще и управлять моделирующими процессами (интерактивные модели).

Этапы и цели компьютерного моделирования

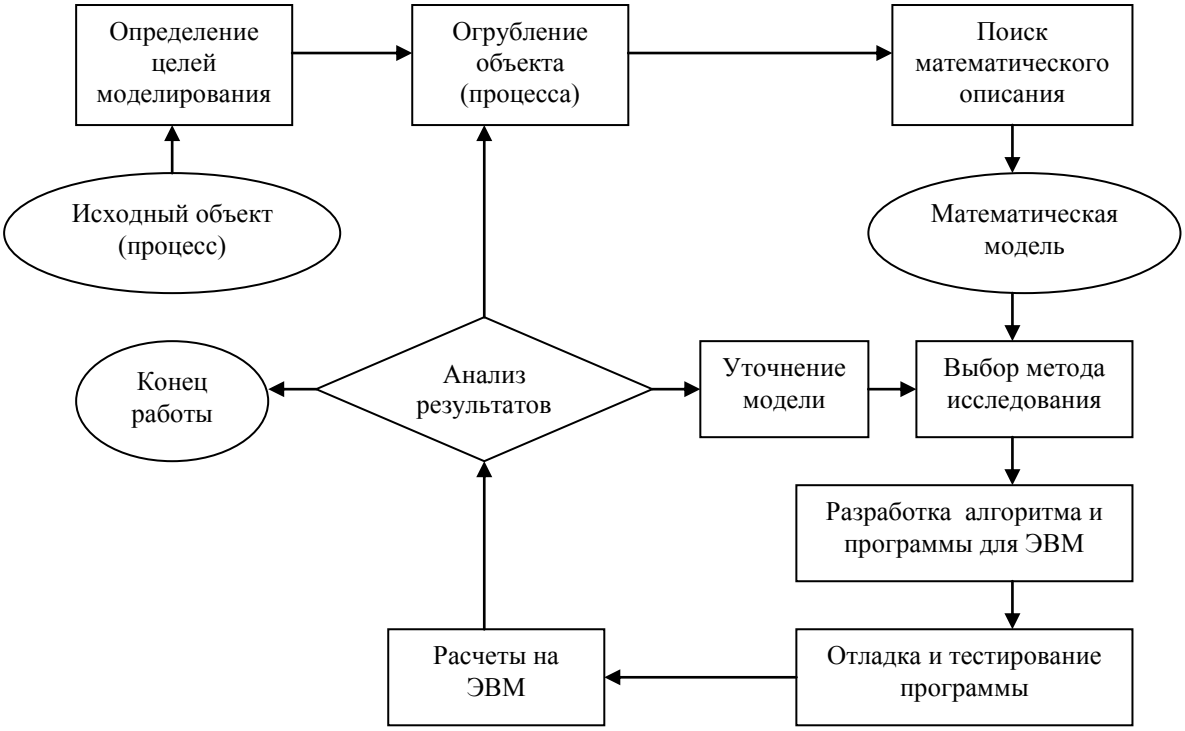


Рисунок 33. Этапы компьютерного моделирования

Здесь мы рассмотрим процесс компьютерного математического моделирования, включающий численный эксперимент с моделью (см. рис.).

Первый этап - *определение целей моделирования*, основные из них таковы:

1) модель нужна для того, чтобы понять, как устроен конкретный объект, какова его структура, основные свойства, законы развития и взаимодействия с окружающим миром (понимание);

2) модель нужна для того, чтобы научиться управлять объектом (или процессом) и определить наилучшие способы управления при заданных целях и критериях (управление);

3) модель нужна для того, чтобы прогнозировать прямые и косвенные последствия реализации заданных способов и форм воздействия на объект (прогнозирование).

Поясним это на примерах. Пусть объект исследования - взаимодействие потока жидкости или газа с телом, являющимся для этого потока препятствием. Опыт показывает, что сила сопротивления потоку со стороны тела растет с ростом скорости потока, но при некоторой

достаточно высокой скорости эта сила скачком уменьшается с тем, чтобы с дальнейшим увеличением скорости снова возрасти. Что же произошло, обусловив уменьшение силы сопротивления? Математическое моделирование позволяет получить четкий ответ: в момент скачкообразного уменьшения сопротивления вихри, образующиеся в потоке жидкости или газа позади обтекаемого тела, начинают отрываться от него и уноситься потоком.

Выработка концепции управления объектом - другая возможная цель моделирования. Какой режим полета самолета выбрать для того, чтобы полет был вполне безопасным и экономически наиболее выгодным? Множество таких проблем систематически возникает перед экономистами, конструкторами, учеными.

Наконец, прогнозирование последствий тех или иных воздействий на объект может быть как относительно простым делом в несложных физических системах, так и чрезвычайно сложным — на грани выполнимости - в системах биолого-экономических, социальных. Если относительно легко ответить на вопрос об изменении режима распространения тепла в тонком стержне при изменениях в составляющем его сплаве, то несравненно труднее проследить (предсказать) экологические и климатические последствия строительства крупной ГЭС. Возможно, и здесь методы математического моделирования будут оказывать в будущем более значительную помощь.

Составим список величин, от которых зависит поведение объекта или ход процесса, а также тех величин, которые желательно получить в результате моделирования. Обозначим первые (входные) величины через $x_1, x_2, x_3 \ldots, x_n$; вторые (выходные) через $y_1, y_2, y_3 \ldots, y_k$. Символически поведение объекта или процесса можно представить в виде:

$$y_j = F_j(x_1, x_2, ..., x_n), (j = 1,2,..,k),$$

где F_J - те действия, которые следует произвести над входными параметрами, чтобы получить результаты. Хотя запись $F(x_1, x_2, \ldots, x_n)$ напоминает о функции, мы здесь используем ее в более широком смысле. Лишь в простейших ситуациях $F(x)$ есть функция в том смысле, который вкладывается в это понятие в учебниках математики, чтобы это подчеркнуть, лучше использовать по отношению к $F(x)$ термин «оператор».

Входные параметры x_i могут быть известны «точно», т.е. поддаваться (по крайней мере, в принципе) измерению однозначно и с любой степенью точности - тогда они являются детерминированными величинами. Так, в классической механике, сколь сложной ни была бы моделируемая система, входные параметры детерминированы - соответственно, детерминирован, однозначно развивается во времени

процесс эволюции такой системы. Однако в природе и обществе гораздо чаще встречаются процессы иного рода, когда значения входных параметров известны лишь с определенной степенью вероятности, т.е. эти параметры являются вероятностными (стохастическими), и, соответственно, таким же является процесс эволюция системы - случайным.

«Случайный» - не значит «непредсказуемый»; просто характер исследования задаваемых вопросов резко меняется (они приобретают вид «С какой вероятностью...», «С каким математическим ожиданием...» и т.п.). Примеров случайных процессов не счесть как в науке, так и в обыденной жизни (силы, действующие на летящий самолет в ветреную погоду, переход улицы при большом потоке транспорта и т.д.).

Для стохастической модели выходные параметры могут быть как величинам вероятностными, так и однозначно определяемыми.

Важнейшим этапом моделирования является разделение входных параметров по степени важности влияния их изменений на выходные. Такой процесс называете *ранжированием* (разделением по рангам). Чаще всего невозможно (да и не нужно учитывать все факторы, которые могут повлиять на значения интересующих нас величин *y*). От того, насколько умело выделены важнейшие факторы, зависит успех моделирования, быстрота и эффективность достижения цели. Выделить боле важные (или, как говорят, значимые) факторы и отсеять менее важные может лишь специалист в той предметной области, к которой относится модель. Так, опытный учитель знает, что на успех контрольной работы влияет степень знания предмета и психологический настрой класса; однако, влияют и другие факторы - например, каким уроком по счету идет контрольная, какова в этот момент погода и т.д. - фактически проведено ранжирование.

Отбрасывание (по крайней мере, при первом подходе) менее значимых факторов огрубляет объект моделирования и способствует пониманию его главных свойств и закономерностей. Умело ранжированная модель должна быть адекватна исходному объекту или процессу в отношении целей моделирования. Обычно определить, адекватна ли модель можно только в процессе экспериментов с ней, анализа результатов.

Следующий этап - *поиск математического описания*. На этом этапе необходимо перейти от абстрактной формулировки модели к формулировке, имеющей конкретное математическое наполнение. В этот момент модель предстает перед нами в виде уравнения, системы уравнений, системы неравенств, дифференциального уравнения или системы таких уравнений и т.д.

Когда математическая модель сформулирована, *выбираем метод ее исследования*. Как правило, для решения одной и той же задачи есть несколько конкретных методов, различающихся эффективностью, устойчивостью и т.д. От верного выбора метода часто зависит успех всего процесса.

Разработка алгоритма и составление программы для ЭВМ - это творческий и трудно формализуемый процесс. В настоящее время при компьютерном математическом моделировании наиболее распространенными являются приемы процедурно-ориентированного (структурного) программирования.

После составления программы решаем с ее помощью простейшую *тестовую задачу* (желательно, с заранее известным ответом) с целью устранения грубых ошибок. Это лишь начало процедуры тестирования, которую трудно описать формально исчерпывающим образом. По существу, тестирование может продолжаться долго и закончиться тогда, когда пользователь по своим профессиональным признакам сочтет программу верной. Программистский фольклор полон историй об ошибках на этом пути.

Затем следует собственно *численный эксперимент*, и выясняется, соответствует ли модель реальному объекту (процессу). Модель адекватна реальному процессу, если некоторые характеристики процесса, полученные на ЭВМ, совпадают с экспериментальными с заданной степенью точности. В случае несоответствия модели реальному процессу возвращаемся к одному из предыдущих этапов.

Тема 2. Некоторые приемы программирования для визуализации результатов моделирования

Геометрические модели

Пространственные соотношения между реальными объектами (положение и ориентация объектов в пространстве и их размеры) изучаются с помощью геометрических моделей. Для визуализации геометрических моделей используются идеализированные геометрические объекты (точка, линия, плоскость…). С помощью таких моделей можно решать задачи вида: изучение хода луча в линзе, с заданными характеристиками; изучение условий образования теней и полутеней и др.

Задание 1. Визуализировать процесс преломления света при переходе из одной среды в другую.
В самой постановке задачи сформулирована цель моделирования: визуализация процесса преломления света.

Теоретические основы преломления света
Закон преломления света:
1. Луч падающий, луч преломленный и перпендикуляр, восстановленный в точку падения луча, лежат в одной плоскости.

2. Действует следующее математическое отношение: $\frac{\sin\alpha}{\sin\beta} = n_{21} = \frac{n_2}{n_1}$,

где α, β – угол падения и преломления соответственно, n_1, n_2 – абсолютные показатели преломления первой и второй сред, n_{21} – относительный показатель преломления.

Построение модели
Визуализацию будем проводить средствами среды визуального программирования Visual Basic 6.0.

Первая часть - визуальное программирование

1. Разместить на форме два текстовых поля (Text1, Text2) для ввода угла падения и относительного показателя преломления, а также соответствующие надписи (Label1, Label2).

2. Поместить на форме два графических поля для отображения хода луча в двух средах (Picture1,Picture2).

3. С помощью редактора меню создать меню с соответствующим набором

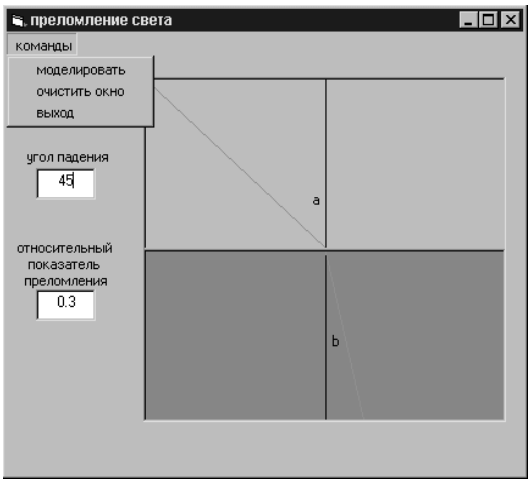


Рисунок 34

команд (см.рис.).

4. Нанести еще две надписи, отображающие название углов (Label3, Label4)

Вторая часть - написание кода программы.

1. Описать константу Пи и переменную *k*, хранящую число вызовов команды «моделировать» после запуска программы и очистки окна.

```
Const pi = 3.14159265358979
Dim k
```

2. Программный код обработки события вызова команды «очистить».

```
Private Sub Clear_Click()
Picture1.Cls
Picture2.Cls
Label3.Visible = False
Label4.Visible = False
k = 1
Picture1.ForeColor = vbBlack
Picture2.ForeColor = vbBlack
End Sub
```

3. Программный код обработки события вызова команды «выход»

```
Private Sub exit_Click()
End
End Sub
```

4. Программный код обработки события загрузки формы

```
Private Sub Form_Load()
Picture1.Scale (-1, 1)-(1, 0)
Picture2.Scale (-1, 0)-(1, -1)
k = 1
End Sub
```

5. Программный код обработки события вызова команды «моделировать»

```
Private Sub model_Click()
a = Val(Text1.Text)
If k = 1 Then
Picture1.Line (0, 1)-(0, 0)
Picture2.Line (0, 0)-(0, -1)
End If
col = RGB(Rnd() * 255, Rnd() * 255, Rnd() * 255)
Picture1.ForeColor = col
Picture2.ForeColor = col
Picture1.Line (0, 0)-(-Tan(a * pi / 180), 1)
n21 = Val(Text2.Text)
x = Sin(a * pi / 180) * n21 / Sqr(1 - (Sin(a * pi / 180) * n21) ^ 2)
Picture2.Line (0, 0)-(x, -1)
Label3.Visible = True
Label4.Visible = True
k = k + 1
End Sub
```

6. Протестировать программу

Задание 2. Исследовать ход луча в плоскопараллельной пластинке (или призме с заданными характеристиками).
В программе предусмотреть возможность изменения угла падения и относительного показателя преломления.

Траектории движения тел, графики

В ряде задач компьютерного моделирования уместно иллюстрировать процесс моделирования изображениями движущихся объектов и их траекториями. Мы сознательно ограничивались случаями плоских (двумерных) движений, которые легко отобразить на плоском экране компьютера.

Задание 3. Изучить движение тела, брошенного под углом к горизонту при отсутствии сопротивления воздуха.
Цели моделирования сформулируем следующими вопросами: Какой вид имеет траектория тела, брошенного под углом к горизонту? Как изменяется скорость во время полета?

Теоретическое описание исследуемой модели. Траектория движения тела будет описываться следующими формулами:

$$\begin{cases} S_x = v_0 \cdot \cos\alpha \cdot t \\ S_y = v_0 \cdot \sin\alpha \cdot t - \dfrac{gt^2}{2} \end{cases},$$

где S_x – перемещение вдоль оси x, S_y - перемещение вдоль оси y в момент времени t, α – угол бросания относительно оси x. Зная координаты тела в любой момент времени, будем точкой отображать положение тела через равные промежутки времени. Т.о. будет понятен характер изменения скорости движения, и вид траектории.

Построение модели

Первая часть - визуальное программирование

1. Разместить на форме три текстовых поля (Text1, Text2, Text3) для ввода угла бросания, начальной скорости и интервала времени (в миллисекундах), через который будет определяться новое положение тела (строиться точка), а также соответствующие надписи (Label1, Label2, Label3).

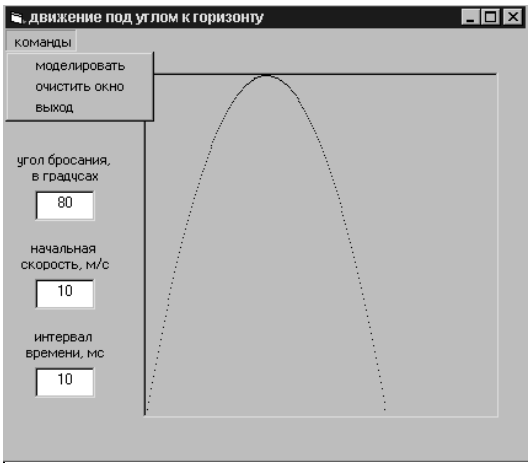


Рисунок 35

2. Поместить на форме графическое

поле для отображения положения тела через определенные промежутки времени (Picture1).

3. Нанести на форму таймер, который будет регулировать построение траектории.

4. С помощью редактора меню создать меню с соответствующим набором команд (см. рис.).

Вторая часть - написание кода программы.

1. Описать константу ПИ и g (ускорение свободного падения), переменные, в которых будут храниться текущие значения начальной скорости движения, угла бросания, и время:

```
Const pi = 3.14159265358979
Const g = 9.8
Dim v0, a, T
```

2. Программный код обработки событий вызова команды «выход» и «очистить» будут иметь следующий вид:

```
Private Sub Clear_Click()
Picture1.Cls
End Sub

Private Sub exit_Click()
End
End Sub
```

3. При вызове команды «моделировать» главного меню, происходит присвоение значений переменным a, v0, устанавливается значение свойства Interval для таймера. Происходит масштабирование графического поля, исходя из коэффициента пропорциональности, активизируется таймер.

```
Private Sub model_Click()
T = 0
Picture1.Cls
a = Val(Text1.Text) * pi / 180
v0 = Val(Text2.Text)
Timer1.Interval = Val(Text3.Text)
tm = v0 * Sin(a) / g                    'время достижения максимальной высоты
symax = v0 * Sin(a) * tm - g * tm ^ 2 / 2 'максимальная высота полета
sxmax = (v0 * Cos(a) * tm) * 2          'максимальная дальность полета
k0 = Picture1.Width / Picture1.Height 'коэффициент пропорциональности
If symax > sxmax Then
 Picture1.Scale (0, symax)-(k0 * symax, 0)
Else
 Picture1.Scale (0, sxmax / k0)-(sxmax, 0)
End If
Timer1.Enabled = True
End Sub
```

4. Через интервал времени, равный значению свойства Interval таймера строится точка. При достижении нижней границы графического поля таймер отключается

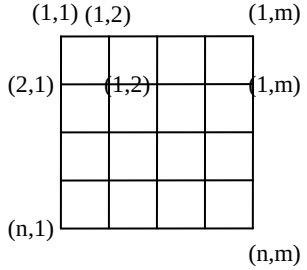
```
Private Sub Timer1_Timer()
T = T + Timer1.Interval / 1000
y = v0 * Sin(a) * T - g * T ^ 2 / 2
'MsgBox (y)
If y < 0 Then
 Timer1.Enabled = False
Else
 x = v0 * Cos(a) * T
 Picture1.PSet (x, y)
End If
End Sub
```

Задание 4. С какой скоростью, и под каким углом нужно бросить мяч, чтобы он попал в площадку, размером 1м, лежащую на расстоянии десяти метров от точки бросания?

Изолинии

В задачах моделирования достаточно стандартная проблема - построение линий (поверхностей), вдоль которых некоторая функция имеет одинаковое значение, называемых изолиниями (изоповерхностями).

Опишем типичную процедуру построения изолиний на экране компьютера. На старте мы имеем двумерную таблицу значений некоторой величины A, полученную в ходе математического моделирования; числа в этой таблице соответствуют значениям этой величины в узлах пространственной сетки (см. рис.).



A_{11}	A_{12}	A_{13}	…	A_{1m}
A_{21}	A_{22}	A_{23}	…	A_{2m}
…	…	…	…	…
A_{n1}	A_{n2}	A_{n3}	…	A_{nm}

Рисунок 36. Пространственная сетка и соответствующая ей таблица значений величины A

Зададим некоторый, совершенно условный, пространственный шаг *h* между соседними узлами по горизонтали и вспомогательную систему координат, в которой узел *(1, 1)* имеет координату *(0, 0)*, узел *(1, 2)* - координату *(h, 0)*, узел *(1, 3)*- координату *(2h, 0)* и т.д. Если шаг по вертикали *h*, то узел *(i, j)* в этой системе имеет координату *((j-1)*h, (i-1)*h)*.

Предварительно найдем в таблице наибольшее и наименьшее значения величин $a_{i,j}$ - допустим, это a_{min} и a_{max}. Пусть b - некоторое промежуточное значение: $a_{min} < b < a_{max}$ - Обсудим в общих чертах, как построить изолинию $A = b$. Будем для этого (в цикле) просматривать вначале все пары ближайших чисел в первой строке таблицы в поисках такой пары, для которой b находится «внутри». Допустим, число b находится между a_{lk} и a_{lk+1} т.е. либо $a_{lk} < b < a_{lk+1}$, либо $a_{tk} > b > a_{l,k+1}$.

С помощью линейной интерполяции найдем соответствующую горизонтальную координату точки, в которой $A = b$.

$$x = (k-1)\cdot h + \frac{b - a_{l,k}}{a_{l,k+1} - a_{l,k}} h$$

(координата y определяется номером горизонтальной линии; в данном случае $y = 0$). Найденные координаты запомним и просмотрим первую строку в таблице до конца, затем просмотрим вторую строку и т.д. Покончив с просмотром строк, мы получим часть точек, соответствующих изолинии $A = b$.

После этого займемся просмотром столбцов. Допустим, во втором столбце нашлась пара чисел, для которой число b находится между a_{p2} и $a_{p+1,2}$. Она дает следующую точку для изолинии. Закончив просмотр всех столбцов, мы получим максимально возможный набор координат точек, принадлежащих данной изолинии. Выводя их на экран в нужном масштабе, получим точечное изображение изолинии $A = b$, после чего можем, взяв другое значение b, построить следующую изолинию.

Задание 5. Построить эквипотенциальные линии электростатического поля, созданного двумя зарядами

Теоретическое описание исследуемой модели. Для построения эквипотенциальных линий поля, созданного системой зарядов, можно воспользоваться принципом суперпозиции: потенциалы полей, созданных разными зарядам, алгебраически складываются. Поскольку потенциал поля, созданного зарядом q на расстоянии r от него равен $-\frac{1}{4\pi\varepsilon_0}\frac{q}{r}$, легко определить результирующий потенциал в любой точке.

Разработка алгоритма. Поле создается ситстемой точечных электрических зарядов Q_1, ..., Q_p с координаты которых равны соответственно (x_1, y_1), ..., (x_p, y_p). Типичная процедура построения состоит в следующем. Выберем по осям x, y шаги h_x, h_y и покроем плоскость сеткой, образованной прямыми, параллельными осям x и y и отстоящими друг от друга на расстояниях hx, hy соответственно. Точки пересечения этих прямых – узлы сетки. Значения потенциала, создаваемого системой зарядов $Q1$, ..., Qp в узле (k, i), согласно принципу суперпозиций, таково:

$$\Phi_{ik} = -\sum_{l=1}^{p} \frac{1}{4\pi\varepsilon_0} \frac{q}{r} \frac{Q_l}{\sqrt{(x_i - kh_x)^2 - (y_i - kh_y)^2}}$$

Ограничимся прямоугольной областью в плоскости *xy*: [$-mh_x$, mh_x] по оси *x* и [$-nh_y$, nh_y] по оси *y*. В этой области *(2m+1) (2n+1)* узлов. Вычислим значение потенциала в каждом из них по указанным формулам. В результате получим матрицу значений потенциала.

Зафиксируем некоторое значение потенциала $\overline{\Phi}$ и построим изолинию, соответствующую этому значению. Для этого проходим, к примеру, по *i*-ой горизонтальной линии сетки и ищем среди ее узлов такие соединения значений потенциала, в которых "захватывают" $\overline{\Phi}$ между собой; признаком этого может служить выполнение неравенства $(\Phi_{i,k} - \overline{\Phi})(\Phi_{i,k+1} - \overline{\Phi}) < 0$. Если такая пара узлов найдена, то координату точки, в которой $\Phi = \overline{\Phi}$ найдем следующем образом: $x = kh_x + \frac{\overline{\Phi} - \Phi_{ik}}{\Phi_{i,k+1} - \Phi_{i,k}} h_x$, $y = ih_y$.

Найдя в данной горизонтали все такие точки, перейдем к следующей горизонтали, пока не исчерпаем их все. Для этого нужно совершить двойной циклический проход: во внешнем цикле перебирать *i* от *–n* до *+n*, во внутреннем перебирать *k* от *–m* до *+m*. После этого следует аналогично заняться поиском нужных точек на вертикальных линиях сетки. Формулы аналогичны: $y = ih_y + \frac{\overline{\Phi} - \Phi_{ik}}{\Phi_{i+1,k} - \Phi_{i,k}} h_y$ $x = kh_x$.

После прохождения всех горизонтальных и вертикальных линий сетки находятся все те точки на этих линиях, в которых потенциал равен $\overline{\Phi}$. Выводим на экран все эти точки, получая первую изолинию.

Построение модели

Первая часть - визуальное программирование

Линии равного потенциала будем выводить на форму в событии ее активизации.

Вторая часть - написание кода программы.

1. Описание переменных и констант, используемых в программе:

```
Const n = 100 'константа, определяющая размер сетки
Const e0 = 0.00000000000885 'электрическая постоянная
Const k = 2 'количество зарядов
Const pi = 3.14159265358979
Dim f(0 To n, 0 To n) 'массив, хранящий значение потенциала в узлах
Dim q(1 To k) 'массив величин зарядов
Dim x(1 To k) 'массив координат X зарядов (число от 0 до 1)
Dim y(1 To k) ' массив координат Y зарядов (число от 0 до 1)
Dim g(1 To 11) 'массив значений потенциала, для построения изолиний
```

2. Программный код написать для события активизации формы:

```
Private Sub Form_Activate()
```

```
End Sub
```

3. Первое действие – присвоение значений входным параметрам:

```
q(1) = -1 * 0.00000000001: q(2) = -2 * 0.00000000001
x(1) = 0.5: y(1) = 0.4
x(2) = 0.4: y(2) = 0.7
g(1) = -2          'заполнение массива значений потенциала для построения
For i = 2 To 11 ‘эквипотенциальных линий
 g(i) = g(i - 1) + 0.6
Next i
```

4. Второе действие – очистка и масштабирование формы:

```
Form1.Cls                    'очистка формы
Form1.Scale (0, 1)-(1, 0) 'масштабирование формы
```

5. Третье – рисование зарядов

```
Form1.ForeColor = vbWhite
For i = 1 To k
 Form1.Circle (x(i), y(i)), 0.01
 Form1.Print (q(i))
Next i
```

6. Определить значения потенциала в узлах сетки

```
For i = 0 To n
 For j = 1 To n
  For m = 1 To k
   r = Sqr((i / n - x(m)) ^ 2 + (j / n - y(m)) ^ 2)
   If r <> 0 Then f(i, j) = f(i, j) + (-1) * q(m) / (r * 4 * pi * e0)
  Next m
 Next j
Next i
```

7. Используя описанный выше алгоритм, построить линии равного потенциала:

```
For m = 1 To 11
b = g(m)
col = RGB(Rnd() * 255, Rnd() * 255, Rnd() * 255)
Form1.ForeColor = col
 For i = 0 To n
  For j = 0 To n - 1
   If (f(i, j) - b) * (f(i, j + 1) - b) < 0 Then
    a = (j + (b - f(i, j)) / (f(i, j + 1) - f(i, j))) / n
    Form1.PSet (i / n, a)
   End If
  Next j
 Next i
 For j = 0 To n
  For i = 0 To n - 1
   If (f(i, j) - b) * (f(i + 1, j) - b) < 0 Then
    a = (i + (b - f(i, j)) / (f(i + 1, j) - f(i, j))) / n
    Form1.PSet (a, j / n)
```

```
        End If
      Next i
    Next j
  Next m
```

8. Протестировать программу. Для этого, в программном коде можно поменять значение величины заряда, их координаты, значения потенциала для построения линий равного потенциала (начальное значение, шаг). Характер расположения линий равного потенциала зависит от величины и расположения зарядов.

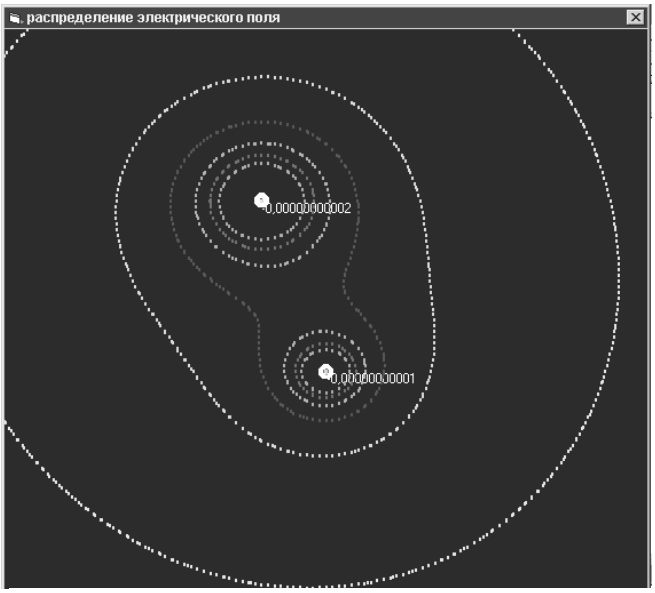


Рисунок 37

Задание 6. На основе предыдущего проекта создать интерактивную модель отображения электростатического поля средствами построения линий равного потенциала.
Интерактивность заключается в возможности задания любого числа зарядов и линий равного потенциала с определенными характеристиками.
Примечание: данную программу реализуйте средствами динамических массивов.

Условные цвета, условное контрастирование

Еще один интересный прием современной научной графики - условная раскраска. Она находит широчайшее применение в самых разных приложениях науки и представляет собой набор приемов по максимально удобной, хотя и очень условной, визуализации результатов компьютерного моделирования.

В различных исследованиях температурных полей встает проблема наглядного представления результатов. Самый простой (и, с точки зрения специалиста, весьма неэффективный) - привести карту (чертеж, план), в некоторых точках которой обозначены значения температуры.

Другой способ - набор изотерм - гораздо эффективнее; к нему прибегают некоторые газеты, давая состояние и прогноз погоды. Но можно добиться еще большей наглядности, учитывая, что большинству людей свойственно, сравнивая разные цвета, воспринимать красный как «горячий», голубой как «холодный», а все остальные - между ними. Допустим, что на некоторой территории температура в данный момент

имеет в разных местах значения от -25°С до + 15°С. Разделим этот диапазон на участки с шагом, равным, например, 5°

[-25,-20], [-20,-15],...,[+10,+ 15],

и закрасим первый из них в ярко-голубой, последний - в ярко-красный, а все остальные - в промежуточные оттенки голубого и красного цветов. Получится наглядная картина температурного поля.

То же самое можно делать при иллюстрации температурного поля и на поверхности обрабатываемой на станке детали, и на поверхности далекой планеты.

Т.о. изображения в условных цветах и контрастах - мощнейший прием научной графики. Он позволяет понять строение не только плоских, но и объемных (трехмерных) объектов, дает в руки исследователя один из замечательных методов познания.

Задание 7. Визуализировать неравномерно нагретый стержень, используя условную раскраску.
В постановке задачи определена цель моделирования.

Разработка алгоритма и создание вспомогательной программы
Чтобы решить следующую задачу рассмотрим:
А) задачу о переходе из красного цвета в синий за 256 шагов
Б) задачу о переходе из красного цвета в синий за 100 шагов.
В обоих случаях используем возможности функции RGB.
Первая часть - визуальное программирование
Переход цвета будем отображать в графических полях, поэтому вынесем на форму два графических поля.
Вторая часть - написание кода программы.

1. Программный код напишем для события активизации формы:

```
Private Sub Form_Activate()

End Sub
```

2. При активизации формы происходит заливка графических полей в соответствующие цвета.
А) Первая задача решается довольно-таки просто: в функции RGB каждая составляющая цвета может принимать 256 значений – от 0 до 255. Соответственно, чтобы реализовать переход от красного к синему цвету, необходимо в 256 раз нарисовать вертикальную линию, цвет которой будет определяться значением функции RGB (значение RGB первой линии будет равно (255,0,0), а последней – (0,0,255)). В каждой последующей линии содержание красного цвета будет уменьшаться на 1, а содержание синего – увеличиваться на 1.

```
Picture1.Scale (0, 1)-(255, 0) 'масштабирование графического поля
For i = 0 To 255
```

```
 Picture1.ForeColor = RGB(255 - i, 0, i)
 Picture1.Line (i, 0)-(i, 1)
Next i
```

Б) Во втором случае необходимо реализовать переход цвета за 100 шагов. Для этого, введем коэффициент *k=256/100*, определяющий степень изменения цвета в каждый последующий шаг.

```
Picture2.Scale (0, 1)-(100, 0)
k = 256 / 100
For i = 0 To 99
 Picture2.ForeColor = RGB(255 - i * k, 0, i * k)
 Picture2.Line (i, 0)-(i, 1)
Next i
```

А теперь вернемся к заданию 7. Пусть заданы температуры пяти узлов, и переход температуры осуществляется по линейному закону. При построении линий, находящихся между узлами будем определять температуру в точке и, в соответствии с температурой, устанавливать цвет линии.

Построение модели

Первая часть - визуальное программирование

На форму нанести графическое поле (Picture1), которое будет имитировать неравномерно нагретый стержень, именного его мы будем закрашивать, используя возможности функции RGB.

Рисунок 38

Вторая часть - написание кода программы.

1. Описать используемые константы, переменные и массивы в разделе описания переменных:

```
Const n = 5 'количество узлов
Const w = 100 'количество линий между узлами
Dim mast(1 To n) As Double ‘массив, хранящий значения температур
Dim maxt, mint ‘максимальная и минимальные тем-ры (из массива тем-р)
```

2. Программный код написать для события активизации формы:

```
Private Sub Form_Activate()

End Sub
```

3. Ввести температуры узлов:

```
mast(1) = 6: mast(2) = 3.667: mast(3) = 4.333: mast(4) = 5: mast(5) = 3
```

4. Подсчитать количесво всех строимых линий:

```
a = w * (n - 1)
```

5. Определить максимальную и минимальную температуры (в дальнейшем узлы, сответствующие этим значениям температуры будут окрашены в ярко-красный и ярко-синий цвета соответственно):

```
maxt = mast(1)
mint = mast(1)
For i = 2 To n
 If mast(i) > maxt Then maxt = mast(i)
 If mast(i) < mint Then mint = mast(i)
Next i
```

6. Масштабировать графическое поле в соответствии с ранее найденными значениями переменных *a* и *maxt*:

```
Picture1.Scale (0, maxt)-(a, 0)
```

7. В цикле определить значение температуры точки между узлами, в соответствии с этим значением устанавить цвет, и построить линию. Для большей наглядности в узлах построить линии желтого цвета, высота которой будет определяться значением температуры узла:

```
kc = 256 / (maxt - mint) 'коэффициент цвета
For i = 1 To n - 1
 k = (mast(i + 1) - mast(i)) / w 'линейный коэф-т измен тем-ры м/у узлами
 b = mast(i) - mint
 For j = 0 To w - 1
  t = k * j + b 'значение температуры в точке между узлами
  Picture1.ForeColor = RGB(t * kc, 0, 256 - t * kc)      ’цвет линии
  Picture1.Line ((i - 1) * w + j, 0)-((i - 1) * w + j, maxt) ’построение линии
 Next j
 Picture1.ForeColor = vbYellow                          ‘построение линии желтого
 Picture1.Line ((i - 1) * w, 0)-((i - 1) * w, mast(i))    ‘в узлах, высота определяет
Next i                                                  ‘значение температуры
Picture1.Line ((i - 1) * w - 1, 0)-((i - 1) * w - 1, mast(i))’ линия последнего узла
End Sub
```

8. Протестировать программу, для этого, можно поменять значения температур в узлах, тогда вид стержня существенно изменяется.

Задание 8. Изменить программу таким образом, чтобы неравномерно нагретый стержень задавался десятью узлами.

Тема 3. Детерминированные модели

Модель остывания чашки кофе

Задание 1. Изучить процесс остывания чашки кофе.
Цель моделирования заключается в визуализации процесса. Данную цель будем достигать построением графика зависимости температуры чашки кофе от времени.

Теоретическое описание модели.
Природа переноса тепла от кофе к окружающему пространству сложна и в общем случае включает в себя механизм конвекции, излучения, испарения и теплопроводности. В том случае, когда разность температур между объектом и окружающей средой не очень велика, скорость изменения температуры T объекта можно считать пропорциональной этой разности температур. Это утверждение формулируется на языке дифференциальных уравнений так:

$$dT/dt=-r(T-Ts) \quad (1)$$

где T – температура кофе, Ts – температура окружающей среды, r – коэффициент остывания. Соотношение данное называется законом теплопроводности Ньютона. Данное уравнение является примером дифференциального уравнения первого порядка. Ввиду того, что множество процессов, происходящих в природе, описывается дифференциальными уравнениями, важно уметь решать эти уравнения. Рассмотрим уравнение первого порядка вида:

$$dy/dx=g(x) \quad (2)$$

В общем случае аналитического решения уравнения (2) не существует. Даже в том случае, если оно есть, необходимо уметь представлять решение в графическом виде, чтобы понять его характер. Эти причины побуждают нас искать не точные, а приближенные численные решения дифференциальных уравнений и познакомиться с простыми методами графического представления решения.
Алгоритм Эйлера. Типичный метод решения дифференциального уравнения включает в себя преобразование его в конечно-разностное уравнение. Проанализируем уравнение (2). Положим, что при $x=x_0$ функция y принимает значение y_0. Поскольку уравнение (2) описывает изменения функции y в точке x_0, то можно найти приближенное значение функции y в близлежащей точке $x_1=x_0+\Delta x$, если приращение аргумента Δx мало. В первом приближении предполагается, что функция g(x) (или скорость изменения y) постоянна на отрезке от x_0 до x_1. В этом случае

приближенное значение функции в точке $x_1 = x_0 + \Delta x$ определяется выражением:

$$y_1 = y(x_0) + \Delta y = y(x_0) + g(x_0)\Delta x \quad (3)$$

Мы можем повторить эту процедуру еще раз и найти значение y в точке $x_2 = x_1 + \Delta x$:

$$y_2 = y(x_1 + \Delta x) = y(x_1) + g(x_1)\Delta x$$

Обобщая это, можно вычислить приближенное значение функции в любой точке $x_n = x_0 + n\,\Delta x$ по итерационной формуле:

$$y_n = y_{n-1} + g(x_{n-1})\Delta x \quad (n=1,2,3\ldots) \quad (4)$$

Данный метод называется методом касательных или методом Эйлера.

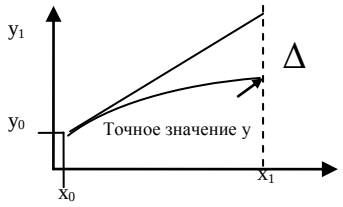


Рисунок 39. Геометрическая интерпретация метода Эйлера

Приближению Эйлера и истинной функции соответствует прямая и кривая рисунка.

Разработка алгоритма. Основной целью моделивания процесса остывания чашки кофе является построение графика зависимости температуры чашки кофе T от времени t. Воспользуемся алгоритмом Эйлера для нахождения приближенного значения температуры в определенный момент времени: $T_n = T_{n-1} + (-r)(T_{n-1} + Ts)\Delta t$. Число шагов *n*, будет определяться интервалом времени Δt и временем наблюдения *tmax*: $n = tmax/\Delta t$.

Построение модели

Первая часть - визуальное программирование.

1. Разместить на форме пять текстовых полей (Text1, Text2, Text3, Text4, Text5) для ввода начальной температуры тела, температуры окружающей среды, коэффициента остывания, времени наблюдения (tmax) и интервала времени, через который будет определяться новое значение температуры тела, а также соответствующие надписи (Label1, Label2, Label3, Label4, Label5).

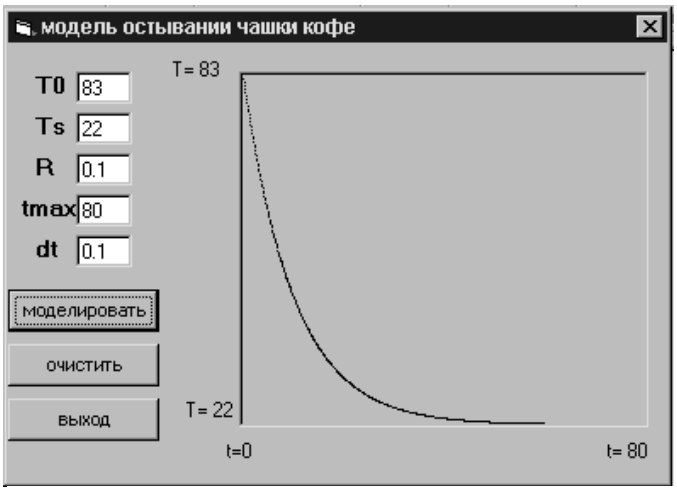


Рисунок 40

2. Поместить на форме графическое поле для графика (Picture1).
3. Установить на форме еще четыре надписи (Label5, Label6, Label7, Label8) для вывода минимальных и максимальных значений времени и температуры.
4. Нанести на форму три кнопки: «моделировать», «очистить», «выход», назначение которых – вызов соответствующих команд.
Вторая часть - написание кода программы.
1. При нажатии на кнопку «очистить» графическое поле должно очищаться, а также перестают отображаться надписи графика (Label5… Label8). Программный код обработки данного события будет иметь следующий вид:

```
Private Sub Command2_Click()
Picture1.Cls
Label7.Caption = ""
Label6.Caption = ""
Label8.Caption = ""
Label9.Caption = ""
End Sub
```

2. Программный код обработки события нажатия на кнопку «выход»:

```
Private Sub Command3_Click() 'выход из программы
End
End Sub
```

3. При нажатии на кнопку моделировать должен строится график зависимости T от t. График будет представлять совокупности точек, определяющих значение температуры (найденной с помощью алгоритма Эйлера) от времени. Программный код события будет иметь следующий вид:

```
Private Sub Command1_Click()
T0 = Val(Text1.Text)     'начальная температура кофе
Ts = Val(Text2.Text)     'конечная темп-ра кофе (т-ра окружающей среды)
R = Val(Text3.Text)      'коэффициент остывания
tmax = Val(Text4.Text) 'время наблюдения
dt = Val(Text5.Text)     'промежуток времени измерения
tv = 0                   'начальный момент времени
T = T0                   'температура кофе
n = tmax / dt            'число шагов
Picture1.Scale (0, T0)-(tmax, Ts) 'масштабирование графического поля
Label7.Caption = "T=" + Str(Ts)
Label6.Caption = "T=" + Str(T0)
Label8.Caption = "t=0"
Label9.Caption = "t=" + Str(tmax)
For i = 1 To n                'определение текущей температуры
 T = T + (-R) * (T - Ts) * dt 'в момент времени tv,
 tv = tv + dt
 Picture1.PSet (tv, T)
```

```
Next i
End Sub
```

Задание 2. Изучить процесс нагревания чашки с водой с заданными параметрами (T_0=Ts=20, $T_к$=100, r=0,15).
Цель моделирования заключается в визуализации данного процесса.

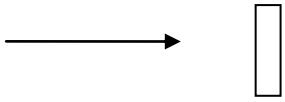
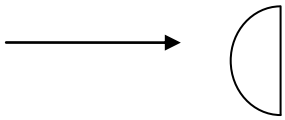
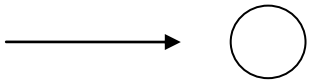
Моделирование свободного падения тела с учетом сопротивления среды

Теоретическое описание модели. Эмпирически установлено, что сила сопротивления определяется следующим образом:

$$F_c=k_1V-k_2V^2,$$

k_1 – коэффициент, который определяется свойствами среды и формой (размером) тела $k_1=6\pi\mu r$, где μ - динамическая вязкость среды ($\mu_{воздуха}$=0,0182 Н·с/м^2, $\mu_{воды}$=1,002 Н·с/м^2, $\mu_{глицерина}$=1480 Н·с/м^2), r – радиус тела;
$k_2=0,5cS\rho_{среды}$, где c – коэффициент лобового сопротивления, S – площадь поперечного сечения по отношению к потоку.

Таблица 6

	Диск	c=1,11
	Полусфера	c=0,55
	Шар	c=0,045

Второй закон Ньютона будет иметь следующий вид: $\frac{d\vec{V}}{dt}=\frac{\vec{F}_{тяж}+\vec{F}_{сопр}}{m}$, в проекции на ось y: $\frac{dV}{dt}=\frac{mg-k_1V-k_2V^2}{m}$ (1)

При изучении свободного падения тела возникает вопрос: *Каков характер изменения скорости со временем, если все характеристики, входящие в уравнение (1) заданы?*

Можно определить, до какой величины скорость будет увеличиваться:

$\frac{d\vec{V}}{dt} = 0$, $mg - k_1 V - k_2 V^2 = 0$ ⇨ $\tilde{V} = \sqrt{\frac{k_1^2}{4k_2^2} + \frac{mg}{k_2}} - \frac{k_1}{2k_2}$, т.е. можно определиться с тем, что скорость возрастает от V_0 до $\tilde{V}$, после чего скорость остается постоянной, каков характер изменения скорости можно узнать, решив дифференциальное уравнение (1). Если необходимо решить задачу о перемещении, то будем решать систему уравнений: $\begin{cases} \frac{dh}{dt} = V \\ \frac{dV}{dt} = \frac{mg - k_1 V - k_2 V^2}{m} \end{cases}$

Задание 3: Герой фильма «Небесный тихоход» майор Булочкин, упав с высоты 6000м в реку без парашюта, остался жив. Смоделируйте данную ситуацию и ответьте на вопрос: возможно ли это на самом деле?

Примечания:

- произведя ранжирование входных данных в рамках математической модели: линейной составляющей скорости можно пренебречь, оставив лишь квадратичную составляющую;
- массу тела взять равную 80 кг;
- c = 1,22;
- S = 0,7 м^2;
- $\rho_{среды}$ = 1,29 кг/м^3;
- для ответа на поставленный вопрос будет интересен следующий факт: один из американских каскадеров совершив прыжок в воду с высоты 75 м (Бруклинский мост), остался жив, а скорость приземления была 33 м/с.
- для визуализации результатов моделирования, построить график функции $V = f(t)$

Задание 4. Изучить свободное падение шарика с заданными характеристиками (m = 100г, r = 10см) в различных вязких средах. Определите влияние вязкости на характер движения ($\mu_{воздуха}$ = 0,0182Н·с/м^2, $\mu_{воды}$ = 1,002 Н·с/м^2, $\mu_{глицерина}$ = 1480 Н·с/м^2, $\rho_{воздуха}$ = 1,29 кг/м^3, $\rho_{воды}$ = 10^3 кг/м^3, $\rho_{глицерина}$ =1,2·10^3 кг/м^3).

Моделирование движения точки, совершающей колебания по оси X и Y (построение фигур Лиссажу)

Задание 5. Изучить траекторию точки, совершающей колебания по осям *x* и *y*. Цель моделирования заключается в построении фигур Лиссажу.

Теоретическое описание модели. Пусть материальная точка совершает колебания как вдоль оси *x*, так и вдоль перпендикулярной ей оси *y*. Если возбудить оба колебания, точка будет двигаться по некоторой

криволинейной траектории, форма которой зависит от разности начальных фаз, соотношения амплитуд и частот складываемых колебаний. Такое движение определяется системой уравнений в параметрической форме:

$$\begin{cases} x(t) = A_1 \sin(w1t + f1) \\ y(t) = A_2 \sin(w2t + f2) \end{cases}$$

$$t=0 \div 2\pi$$

$$f_1=0 \div 2\pi,\ f_2=0 \div 2\pi$$

Такие траектории точки, одновременно совершающей гармонические колебания в двух взаимно перпендикулярных направлениях, называются фигурами Лиссажу.

Фигуры Лиссажу вписываются в прямоугольник, центр которого совпадает с началом координат, а стороны параллельны осям *OX* и *OY* и расположены по обе стороны от них на расстояниях соответственно равных A_1 и A_2. Отношение частот складываемых колебаний w_1 и w_2 равно отношению касаний соответствующей им фигуры Лиссажу со стороной прямоугольника, параллельной оси *OX* и оси *OY*.

Разработка алгоритма. Решение задачи состоит в построении графика, заданного в параметрической форме. Будем определять координаты *x* и *y* точки в определеный момтент времени *t*, и выводить ее в графическое поле, в результате получим траекторию движения.

Построение модели

Первая часть - визуальное программирование.

1. Нанести на форму две надписи (Label1, Label2) с формулировкой задачи (см. рис.).

2. Разместить на форме шесть текстовых полей (Text1, … Text6) для ввода амплитуд колебаний, частот и начальных фаз колебаний, а также соответствующие текстовым полям надписи с обозначениями вводимых величин (Label3, … Label8).

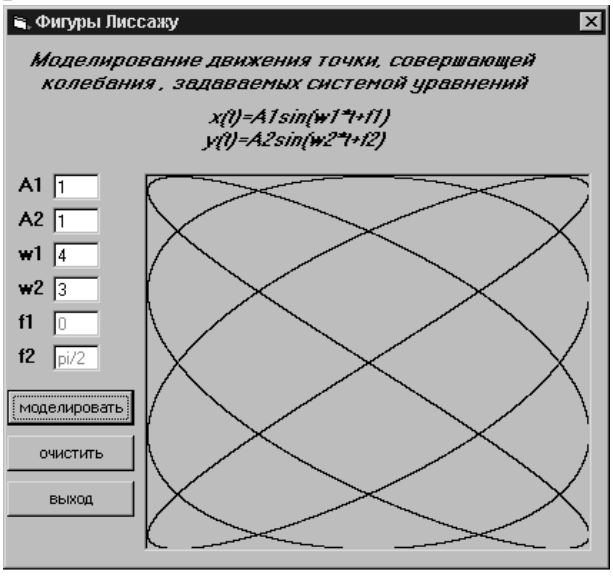


Рисунок 41

В целях упрощения модели установить значения *f1* = 0 и *f2* = pi/2, с невозможностью изменения этих значений. Для этого нужно поменять свойства Enabled (доступность) текстовых полей Text5 и Text6 на False, и свойству Text присвоить соответственно: «0», «pi/2».

3. Поместить на форме графическое поле для отображения траектории движения (Picture1).

4. Нанести на форму три кнопки: «моделировать», «очистить», «выход», назначение которых – вызов соответствующих команд.

Вторая часть - написание кода программы.

1. Описать константу ПИ, которая будет использоваться в программе:

```
Const pi = 3.14159265358979
```

2. При нажатии на кнопку «очистить» графическое поле очищается. Программный код этого события написать самостоятельно.

3. При нажатии на кнопку «выход» программа завершает свою работу. Программный код этого события написать самостоятельно.

4. При нажатии на кнопку «моделировать» определяются входные параметры модели в процессе считывания информации из соответствующих текстовых полей. Осуществляется масштабирование графического поля в соответствии с амплитудами колебания. Рассчитывается положение точки в определенный момент времени (изменяется в интервале от 0 до 2π, т.е. учитывается один период колебания, далее движение точки по такой же траектории). Программный код имеет следующий вид:

```
Private Sub Command1_Click()
A1 = Val(Text1.Text)
A2 = Val(Text2.Text)
w1 = Val(Text3.Text)
w2 = Val(Text4.Text)
f1 = Val(Text5.Text)
f2 = pi / 2
ax = A1
ay = A2
If A1 = 0 Then ax = 1
If A2 = 0 Then ay = 1
Picture1.Scale (-ax, ay)-(ax, -ay)
For t = 0 To 2 * pi Step 0.0001
x = A1 * Sin(w1 * t + f1)
y = A2 * Sin(w2 * t + f2)
Picture1.PSet (x, y)
Next t
End Sub
```

Задание 6. Программу изменить следующим образом: используя анимацию отобразить движение материальной точки, совершающей колебания по оси X и У (законы изменения координаты взять из предыдущей задачи).

В программе осуществить возможность изменения начальных фаз колебания по оси X и У с помощью объекта комбинированный список.

Тема 4. Стохастическое моделирование

Техника стохастического моделирования

Понятие «случайный» одно из самых фундаментальных, как и математике, так и в повседневной жизни. Моделирование случайных процессов - мощнейшее направление в современном математическом моделировании.

Событие называется случайным, если оно достоверно непредсказуемо. Случайность окружает наш мир и чаще всего играет отрицательную роль в нашей жизни. Однако есть обстоятельства, в которых случайность может оказаться полезной.

В сложных вычислениях, когда искомый результат зависит от результатов многих факторов, моделей и измерений, можно сократить объем вычислений за счет случайных значений значащих цифр. Из теории эволюции следует, что случайность проявляет себя как конструктивный, позитивный фактор. В частности, естественный отбор реализует как бы метод проб и ошибок, отбирая в процессе развития особи с наиболее целесообразными свойствами организма. Далее случайность проявляется в множественности ее результатов, обеспечивая гибкость реакции популяции на изменения внешней среды.

В силу сказанного имеет смысл положить случайность в основу методов получения решения посредством проб и ошибок, путем случайного поиска.

Итак, пусть в функционале модели значения некоторых входных параметром определены лишь в вероятностном смысле. В этом случае значительно меняется сам стиль работы с моделью.

При серьезном рассмотрении в обиходе появляются слова «распределение вероятности», «достоверность», «статистическая выборка», «случайный процесс» и т.д.

При компьютерном математическом моделировании случайных процессов нельзя обойтись без наборов, так называемых, случайных чисел, удовлетворяют заданному закону распределения. На самом деле эти числа генерирует компьютер по определенному алгоритму, т.е. они не являются вполне случайными хотя бы потому, что при повторном запуске программы с теми же параметрами последовательность повторится; такие числа называют «псевдослучайными».

Генерация чисел равновероятно распределенных на некотором отрезке

Рассмотрим вначале генерацию чисел равновероятно распределенных на некотором отрезке. Большинство программ - генераторов случайных чисел – выдают последовательность, в которой

предыдущее число используется для нахождения следующего. Первое из них - начальное значение. Все генераторы случайных чисел дают последовательности, повторяющиеся после некоторого количества членов, называемого периодом, что связано с конечной длиной машинного слова. Самый простой и наиболее распространенный метод - метод вычетов, или конгруэнтный метод, в котором очередное случайное число x_n определяется «отображением».

$$x_n = (a\, x_{n-1} + c) \bmod m,$$

где a, c, m - натуральные числа, mod -так называемая, функция деления по модулю. Наибольший возможный период датчика равен m, однако, он зависит от а и с. Ясно, что чем больше период, тем лучше; однако реально наибольшее m ограничено разрядной сеткой ЭВМ, в любом случае используемая в конкретной задаче выборка случайных чисел должна быть короче периода, иначе задача будет решена неверно. Заметим, что обычно генераторы выдают отношение x_n/m, которое всегда меньше 1, т.е.генерируют последовательность псевдослучайных чисел на отрезке [0,1].

Вопрос о случайности конечной последовательности чисел гораздо сложнее, чем выглядит на первый взгляд. Существует несколько статистических критериев случайности, но все они не дают исчерпывающего ответа. Так, последовательно генерируемые псевдослучайные числа могут появляться не идеально равномерно, и проявлять тенденцию к образованию групп (т.е. коррелировать). Один из тестов на равномерность состоит в делении отрезка [0, 1] на M равных частей - «корзин», и помещения каждого нового случайного числа в соответствующую «корзину». В итоге получается гистограмма, в которой высота каждого столбика пропорциональна количеству попавших в «корзину» случайных чисел (см. рис.).

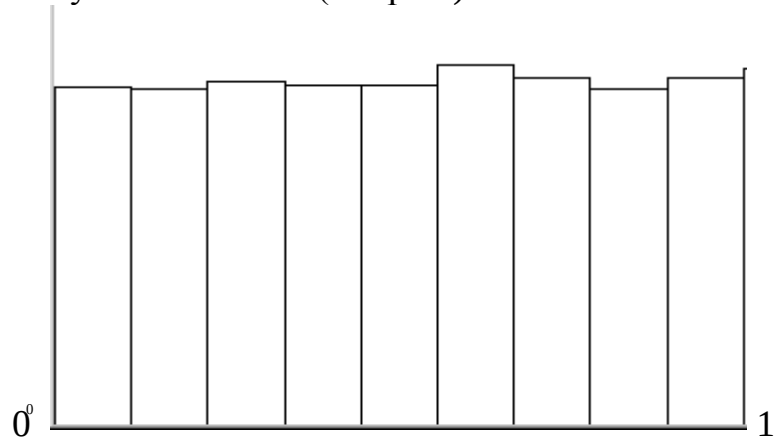


Рисунок 42. Вид гистограммы для равномерно распределенных на отрезке [0, 1] чисел при достаточно большой выборке

Понятно, что при большом числе испытаний высоты столбиков должны быть почти одинаковыми. Однако, этот критерий является необходимым, но не достаточным; например, он «не замечает» даже очень короткой периодичности. Для не слишком требовательного пользователя обычно достаточны возможности датчика (генератора) случайных чисел, встроенного в большинство языков программирования. Так, в Visual Basic есть функция *rnd*(), значения которой - случайные числи в диапазона [*0, 1*]. Ее использованию обычно предшествует использование процедуры *randomize*, служащей для начальной «настройки» датчика, т.е. получения при каждом из обращений к датчику разных последовательностей случайных чисел. Для задач, решение которых требует очень длинных некоррелированных последовательностей, вопрос осложняется и требует нестандартных решений. Равномерно распределенные случайные числа - простейший случай. Располагая датчиком случайных чисел, генерирующим числа *r* принадлежащим [*0, 1*], легко получить числа из произвольного интервала [*a, b*]*:*

$$x = a + (b - a) \bullet r.$$

Задание 1. Протестировать генератор случайных чисел среды программирования Visual Basic, используя алгоритм, описанный выше.

Построение модели

Первая часть - визуальное программирование.

Столбчатую гистограмму будем строить на форме без использования других элементов управления и контроля.

Вторая часть - написание кода программы.

1. Описать переменные, и константы, используемые в программе:

```
Const n = 100 'выборка
Const m = 10   'число столбцов
Dim vis(1 To m) 'массив, определяющий высоту строимых столбцов
```

2. Процесс построения гистограммы будет осуществляться в событии активизации формы. Программный код данного события будет иметь следующий вид:

```
Private Sub Form_Activate()
Randomize
maxi = 0
For j = 1 To n 'заполение массива, определяющего высоту столбцов
 a = Rnd
 i = Fix(a * m)
 vis(i + 1) = vis(i + 1) + 1
 If vis(i + 1) > maxi Then maxi = vis(i + 1) 'определение максимальной
                                              'высоты столбца
Next j
Form1.Scale (0, maxi + 1 / 4 * maxi)-(1, 0) 'масштабирование формы
```

```
h = 1 / m
For i = 1 To m 'построение гистограммы
 Form1.FillColor = RGB(Rnd() * 250, Rnd() * 250, Rnd() * 250)
 Form1.FillStyle = 0
 Form1.Line ((i - 1) * h, vis(i))-((i - 1) * h + h, 0), f, B
Next i
End Sub
```

При увеличении числа испытаний (выборки) высота столбцов будет приближаться к одной высоте.

Генерация случайных чисел с некоторой нормированной функцией распределения

Более сложные распределения часто строятся с помощью равномерного распределения. Упомянем здесь лишь один достаточно универсальный метод Неймана (часто называемый также методом отбора-отказа), в основе которого лежит простое геометрическое соображение. Допустим, что необходимо генерировать случайные числа с некоторой нормированной функцией распределения $f(x)$ на интервале $[a, b]$. Введем положительно определенную функцию сравнения $w(x)$, такую, что $w(x)=const$ и $w(x)>f(x)$ на $[a, b]$ (обычно $w(x)$ равна максимальному значению $f(x)$). Поскольку площадь под кривой $f(x)$ равна для интервала $[x,x+dx]$ вероятности попадания x в этот интервал, можно следовать процедуре проб и ошибок. Генерируем два случайных числа, определяющих равновероятные координаты в прямоугольнике $ABCD$ с помощью датчика равномерно распределенных чисел

$$x = a + (b-a)\cdot r, \quad y = wr$$

и если точка $M(x, y)$ не попадает под кривую $f(x)$, мы ее отбрасываем, а если попадает – оставляем. При этом множество координат x оставленных точек оказывается распределенным в соответствие с плотностью вероятности $f(x)$.

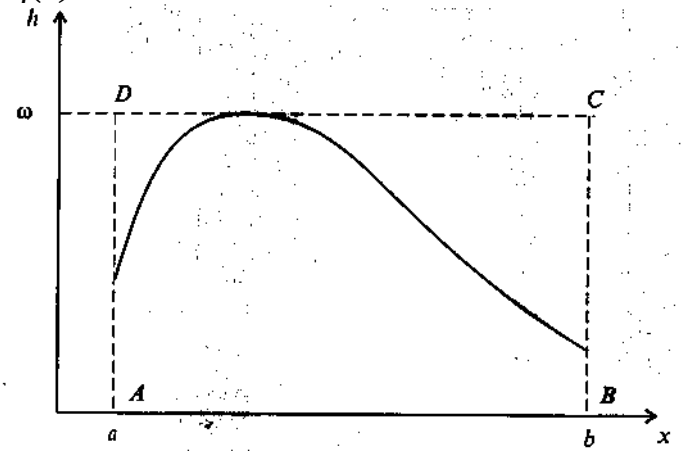


Рисунок 43. Метод отбора-отказа

Этот метод для ряда распределений не самый эффективный, но он универсалом, прост и понятен. Эффективен он тогда, когда функция сравнения *w(x)* близка к *f(x)*. Заметим, что никто не заставляет нас брать *w(x)* = const на всем промежутке [*a, b*]. Если *f(x)* имеет быстро спадающие «крылья», то разумнее взять *w(x)* в виде ступенчатой функции.

Задание 2. Осуществить генерацию чисел, удовлетворяющих распределению Пуассона, использовать метод отбора-отказа. Протестировать сгенерированные числа, с помощью построения гистограммы. Распределение Пуссона имеет следующий вид:

$$p_n(t) = \frac{(\lambda t)^n}{n!} \cdot e^{-\lambda t}$$

где λ - некоторая константа, n – произвольное целое. Функции Пуассона имеют максимум при $\tau = \frac{n}{\lambda}$ и нормированы: $\int_0^\infty p_n(t)dt = 1$.

Построение модели

Первая часть - визуальное программирование.

Столбчатую гистограмму будем строить на форме без использования других элементов управления и контроля.

Вторая часть - написание кода программы.

1. Описать переменные, и константы, используемые в программе:

```
Const n = 100000 'выборка
Const m = 30     'число столбцов
Const a = 3      'произвольная константа лямбда
Const n1 = 1     'произвольное целое
Dim vis(1 To m) 'массив, хранящий число попаданий в диапазоны,
                ‘определяющий высоту столбца
```

2. Т.к. в функции Пуассона используется факториал числа, оформим блок программы, определяющий факториал в виде функции factorial:

```
Function factorial(b As Double) As Double
 f = 1
 For i = 1 To b
  f = f * i
 Next i
 factorial = f
End Function
```

3. Процесс построения гистограммы будет осуществляться в событии активизации формы. Программный код данного события будет иметь следующий вид:

```
Private Sub Form_Activate()
Randomize
x = n1 / a 'максимум функции Пуассона
```

```
w = ((a * x) ^ n1) / factorial(n1) * Exp(-a * x) 'значение функции в максимуме
maxi = 0 ' максимальный элемента массива
For j = 1 To n
 x = Rnd
 y = Rnd() * w
 Y1 = (a * x) ^ n1 / factorial(n1) * Exp(-a * x)
 If y <= Y1 Then
  i = Fix(x * m)
  vis(i + 1) = vis(i + 1) + 1
  If vis(i + 1) > maxi Then maxi = vis(i + 1)
 End If
Next j
h = 1 / m 'построение гистограммы
Form1.Scale (0, maxi + 1 / 4 * maxi)-(1, 0)
For i = 1 To m
 Form1.Line ((i - 1) * h, vis(i))-((i - 1) * h + h, 0), f, B
Next i
                                    End Sub
```

Результат программы:

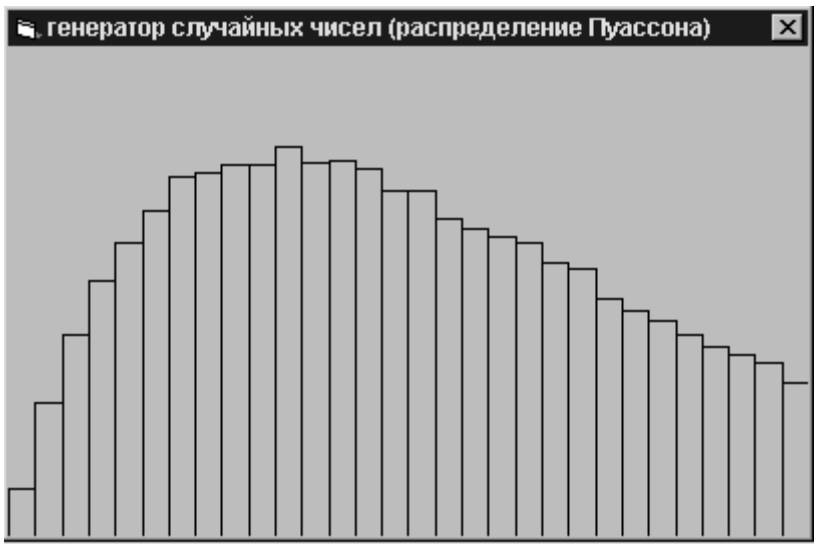


Рисунок 44. Распределение Пуассона

Максвелловское распределение молекул по скоростям

Задание 3. Решить следующую задачу: какая часть молекул водорода при температуре 300^0К обладает скоростями, лежащими в интервале от 1800 до 1910 м/сек. Построить график распределения по модулю скорости с выделением диапазона, указанного в условии задачи.

Теоретическое описание модели.

Молекулы газа участвуют в непрерывном хаотическом движении. В газе скорость одной молекулы при столкновениях будет меняться довольно разнообразно, т.к. молекула будет сталкиваться с другими молекулами газа, движущимися с разными скоростями. Такие столкновения будут случайными и скорость молекулы газа можно рассматривать как случайную величину. Функция распределения скоростей молекул будет иметь следующий аналитический вид:

$$f(v) = 4\pi\sqrt{\left(\frac{m}{2\pi kT}\right)^3} v^2 e^{-\frac{mv^2}{2kT}}$$

где m – масса молекулы (для водорода m=3.34·10^{-24}кг), k=3.38·10^{-23}Дж/К – постоянная Больцмана. Распределение такого вида называется максвелловским, т.к. выведено оно Максвеллом в 1860г. Вид распределения по скоростям см. ниже.

Самое большое число молекул приходится на скорость, соответствующую максимуму кривой *f(v)*. Поэтому скорость, соответствующая максимуму кривой распределения, получила название наивероятнейшей скорости: $v_n = \sqrt{\frac{2kT}{m}}$, значение этой скорости нам понадобится для решения задачи.

Разработка алгоритма. Будем генерировать число, удовлетворяющее распределению Максвелла и проверять принадлежность этого числа к указанному в условии задачи интервалу. Если N определяет число сгененированных значений скорости, а n – число значений скорости, удовлетворяющих заданному интервалу, то отношение N/n – определит искомую величину т.е. часть молекул, обладающих скоростями, лежащими в определенном условием задачи интервале. Точность решения задачи достигается за счет большой выборке значений скорости.

При построении графика для большей наглядности будем выделять интересующий нас диапазон. Для этого, график условно разделим на три части: посторение первой части реализуется с помощью построения точек, второй – построения вертикальных линий, третьей – построения точек.

Построение модели

Первая часть - визуальное программирование.

1. В надписи оформить условие задачи (см. рис.). Для интерактивности модели: разместить на форме три текстовых поля (Text1, Text2, Text3) для ввода двух значений интервала скорости и температуры.

2. Поместить на форме графическое поле для отображения графика функции (Picture1).

3. На графическое поле вынести надпись (lblRez) для вывода искомой величины.

4. Еще две надписи, расположенные на форме, рядом с верхним левым и нижним правом углами графического поля будут обозначать оси графика: *f(v), v*.

5. С помощью редактора меню создать меню с соответствующим набором команд (см.рис.).

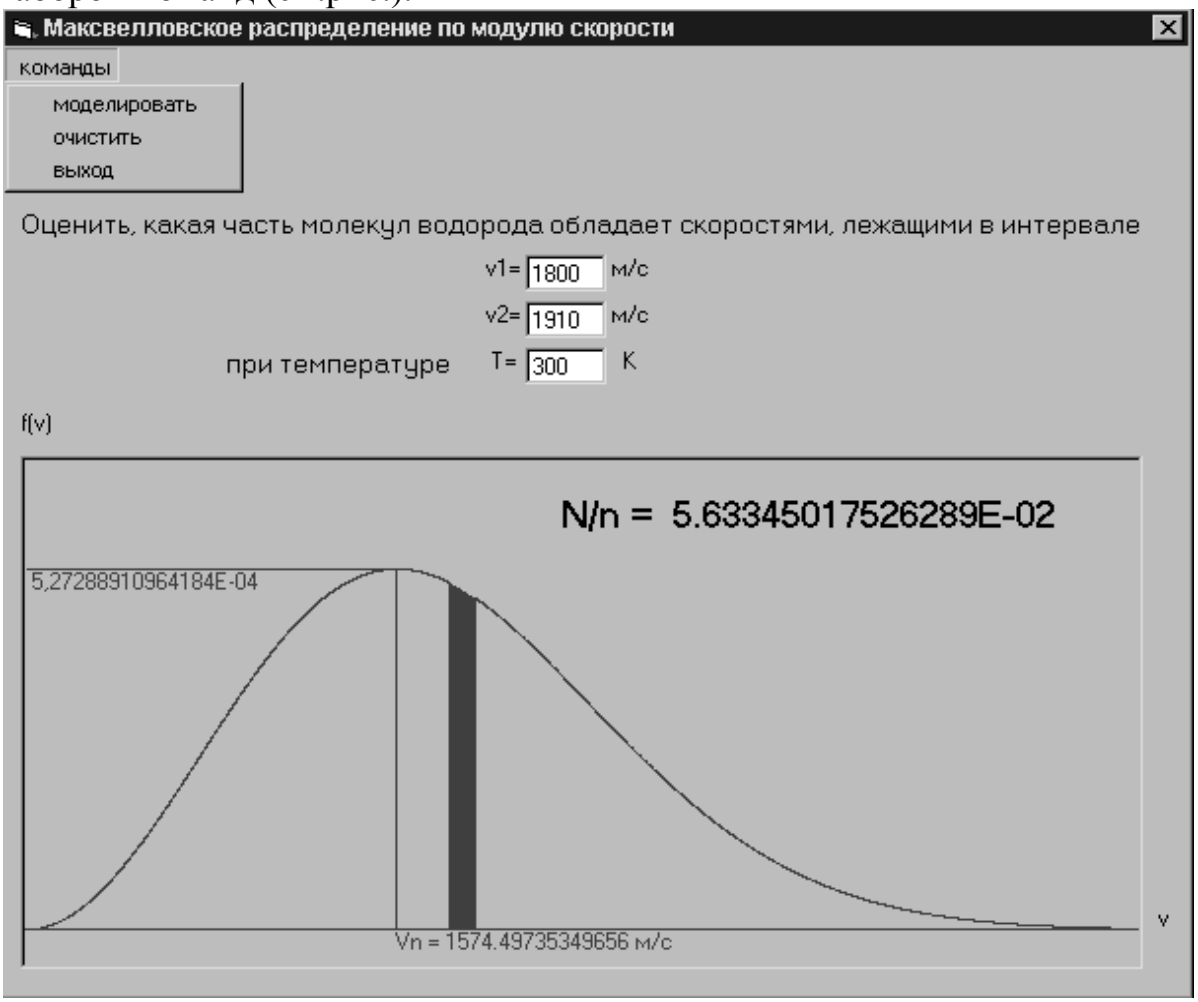


Рисунок 45

Вторая часть - написание кода программы.

1. Описать константы

```
Const n = 10000 'выборка
Const pi = 3.14159265358979
Const m = 3.34E-27 'масса молекулы водорода
Const k = 1.38E-23 'постоянная Больцмана
Const k1 = 10 'число рисуемых линий
```

2. Программный код обработки событий вызова команды «выход» и «очистить» будут иметь следующий вид:

```
Private Sub clear_Click()
Picture1.Cls
End Sub
```

```
Private Sub exit_Click()
End
End Sub
```

3. Используя алгоритм, описанный выше, определение отношения N/n, и построение графика осуществить в событии вызова команды «моделировать». Программный код этого события будет иметь следующий вид:

```
Private Sub model_Click()

'заполнение параметров моделей
v1 = Val(Text1.Text) '
v2 = Val(Text2.Text)
T = Val(Text3.Text)

'настройка генератора случайных чисел
Randomize

'определение наивероятнейшей скорости
vn = (2 * k * T / m) ^ (1 / 2)
'определение значения функции в макимуме
w = 4 * pi * (m / (2 * pi * k * T)) ^ (3 / 2) * vn ^ 2 * Exp((-1) * m * vn ^ 2 / (2 * k * T))

'определение части молекул водорода, обладающих заданными скоростями
'n1 - всего число частиц
'n2 - число частиц, принадлежащих указанному диапазону
For j = 1 To n
 v = Rnd() * 3 * vn
 f1 = Rnd() * w
 f = 4 * pi * (m / (2 * pi * k * T)) ^ (3 / 2) * v ^ 2 * Exp((-1) * m * v ^ 2 / (2 * k * T))
 If (f1 <= f) Then
  n2 = n2 + 1
   If (v >= v1) And (v <= v2) Then n1 = n1 + 1
 End If
Next j

'определение искомой величины и заполнение записи результата
lblRez.Caption = "N/n = " + Str(n1 / n2)

'масштабирование графического поля
xmax = 3 * vn
ymax = w * 1.3
ymin = 0 - w * 0.1
Picture1.Scale (0, ymax)-(xmax, ymin)
Picture1.Line (0, 0)-(xmax, 0)
Picture1.Line (vn, w)-(0, w) 'максимальное значение функции
Picture1.Print w
Picture1.Line (vn, w)-(vn, 0) 'наивероятнейшая скорость
Picture1.Print "Vn =" + Str(vn) + " м/с"
```

```
'построение графика
For v = 0 To xmax Step vn / 1000
 f = 4 * pi * (m / (2 * pi * k * T)) ^ (3 / 2) * v ^ 2 * Exp((-1) * m * v ^ 2 / (2 * k * T))
  Select Case v
  Case Is <= v1
   Picture1.PSet (v, f)
  Case v1 To v2
   Picture1.Line (v, 0)-(v, f)
  Case Is >= v2
   Picture1.PSet (v, f)
  End Select
Next v
End Sub
```

4. Протестировать программу.

Содержание

Литература

1. Волчёнков Н.Г. Программирование на Visual Basic 6: В 3-х ч. Часть 2. – М.: ИНФРА-М, 2002. – 280 с.
2. Информатика: Учеб. пособие для студ. пед. вузов / А.В. Могилев, Н.И. Пак, Е.К. Хеннер; Под ред. Е.К. Хеннера. – 2-е изд., стер. – М.: Изд. Центр «Академия». 2001. – 816 с.
3. Паньгина Н.Н. Первое знакомство с Visual Basic: Заочная школа программирования. Занятие 1: Учебное пособие. – СПб.: Издательство ЦПО «Информатизация образования», 2001. – 26 с.
4. Паньгина Н.Н. Работа с файлами и графикой в Visual Basic: Заочная школа программирования. Занятие 2: Учебное пособие. – СПб.: Издательство ЦПО «Информатизация образования», 2001. – 24 с.
5. Райтингер М., Муч Г. Visual Basic 6.0: для пользователя: пер с нем. – К.: Издательская группа BHV, 1999. -416 с.
6. Угринович Н.Д. Практикум по информатике информационным технологиям. Учебное пособие для общеобразовательных учреждений / Н.Д. Угринович, Л.Л. Босова, Н.И. Михайлова. – М.: Бином. Лаборатория Знаний, 2002. 400 с: ил.