МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ РАДИОТЕХНИКИ, ЭЛЕКТРОНИКИ И АВТОМАТИКИ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

ИНФОРМАТИКА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

Для студентов специальностей

072000, 190400, 190700,
200100, 200300

МОСКВА 2002

Составители: Индришенок В.И.,
Парамонов Н.Б.,
Певцов Е.Ф.,
Русанов К.Е.

Редактор Е.Ф. Певцов

Учебно-методические указания для выполнения курсовой работы по предмету «Информатика», обучение которому проводится по специальностям 072000, 190400, 190700, 200100, 200300 факультета «Электроника» в I и II семестрах. Рассмотрены основные алгоритмические структуры программирования, графические средства языка, а также вычислительные методы определения корней уравнений, численного дифференцирования и интегрирования и разобрана примерная последовательность действий при разработке и отладке курсовой работы по программированию в среде Visual Basic.

Печатаются по решению редакционно-издательского совета Московского Государственного института радиотехники, электроники и автоматики (технического университета)

Рецензенты: д. т. н., проф. В.С. Солдатов
к. т. н., доц. В.К. Григорьев

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА VISUAL BASIC
ПРОГРАММА ИССЛЕДОВАНИЯ ФУНКЦИЙ

Цель курсовой работы. Приобретение навыков проектирования в интегрированной среде Visual Basic (VB). При работе над проектом программы исследования функций студенты обучаются на практике применять основные алгоритмические структуры, графические средства языка, а также вычислительные методы определения корней уравнений, численного дифференцирования и интегрирования.

ОБЩИЕ ЗАМЕЧАНИЯ

Начальные сведения по технике работы в среде VB приведены в методических указаниях [1]. При выполнении упражнений и курсовой работы следует особенно обратить внимание на: 1) формулировку алгоритма решения, обязательно позволяющего полностью выполнить задание; 2) правильное определение областей видимости переменных, процедур и функций, определяемых в программах (см. [1, 2 и 3]); 3) применение методов отладки и тестирования программ [1 и 4]; 4) четкое понимание математических основ численных методов, используемых в расчетах [5].

Переменная - это такая часть программы, которая имеет имя и значение. Строку символов, которая отличает переменную от других объектов программы, *идентифицирует* ее называют *именем* переменной. Переменные создаются программистом по определенным правилам при написании программного кода. ***Значение переменной*** - это данные, которые хранятся и обрабатываются компьютером. Способ хранения определяется ***типами*** переменных, которые следует объявлять в программе.

Основные типы данных: ***Byte, Integer, Long, Single, Double, String, Currency, Boolean, Date, Object, Variant.***

Оператор объявления типа записывается в начале программного кода или процедуры и имеет синтаксис:

Dim *ИмяПеременной* **As** *ТипПеременной*

После слова **Dim** через запятую можно записывать несколько таких конструкций:

Dim *sngX* **As Single,** *intЧислоАрбузов* **As Integer,** *dblPI* **As Double.**

В этом примере при задании имен переменных использована рекомендуемая разработчиками языка венгерская нотация, согласно которой слева от имени переменной следует приписывать

соответствующие приставки: *byt, int, lng, sng, dbl, str, cur, bln, dtm, obj, vnt*.

Существует несколько способов, уточняющих *область видимости* объявляемой переменной, процедуры или функции. В том случае, когда переменная объявляется с ключевым словом Dim, областью ее использования будет только та процедура, в которой она была описана. Такие переменные называются *локальными (закрытыми)*. После выполнения процедуры их значения будут потеряны. В разных процедурах для локальных переменных можно использовать одинаковые имена (например, в счетчиках циклов). Если используется ключевое слово Static, то переменная тоже будет локальной, однако ее последнее значение сохраняется. Если требуется, чтобы переменные, функции и процедуры были доступны в нескольких разделах одного модуля, следует использовать оператор Private и объявление переменных следует осуществлять в разделе Declarations текущего модуля. Когда нужно, чтобы переменная была доступна всем модулям, ее следует объявить в этом разделе оператором Public. Такие переменные называются *глобальными* (*открытыми*).

Оператор присваивания:

ИмяПеременной=ЗначениеПеременной

Функциональный оператор - название и встроенная в язык функция или процедура, подразумевающая выполнение вполне определенной последовательности действий. Примеры:

– **Математические функции: sin(), exp(), log(), log10(), randomize** и т.п.

– **Функции обработки строк:**

Val(Строка$) – возвращает число, соответствующее строке символов, представляющих запись числа;

Str(Число#) – преобразует число в строку символов;

LCase(Строка$) - возвращает исходную строку, в которой все буквы преобразованы в строчные.

Asc(Строка$) - возвращает код ASCII первого символа строки.

Chr(Код&) – возвращает строку из одного символа, код которого задан.

UCase(Строка$) - возвращает исходную строку, в которой все буквы преобразованы в прописные.

В примерах функций обработки строк для обозначения типов переменных использована устаревшая конструкция с суф-

фиксами после имени. Допустимые суффиксы, служащие для автоматического назначения типов данных:

Суффикс	**%**	**&**	**!**	#	**$**	**@**
Тип	Integer	Long	Single	Double	String	Currency

– **Функции ввода данных и выдачи сообщений:**

InputBox(приглашение[,заголовок][,начЗначение]), где приглашение - любой текст в окне ввода, который д.б. туда помещен программистом, заголовок - заголовок окна, начЗначение - то, что будет введено автоматически (по умолчанию).

MsgBox(Текст[,Опция][,Заголовок]**).**

Эта функция возвращает значение, которое затем как-то используется. Другой вариант:

MsgBoxТекст[,Опция][,Заголовок]

В этом варианте функция действует как оператор, т.е. не возвращает никакого значения и просто выдает информацию в окне сообщения. Текст – это строка сообщения, может содержать до 1024 символов. Разбиение на строки и страницы реализуется вставкой в текст сообщения символов переноса, возврата и конца страницы (Например, Строка1$ & Chr(13) & Chr(10) &Строка2$). Значение аргумента Опция является целое число: Op1+Op2**.** При этом Op1 может принимать значения 16, 32, 48 или 64 и определяет вид сообщения и пиктограмму в окне сообщения: критическое сообщение, вопрос, предупреждение или информация. Аналогично Op2 принимает значения 0, 1, 2, 3, 4 или 5 и определяет, соответственно, набор кнопок в окне: *OK*, *OK* и *отмена*, *стоп-повтор-пропустить, да-нет-отмена, да-нет, повтор-отмена.* Возвращаемое значение - целое число от 1 до 7. Оно определяется однозначно нажатой кнопкой. (Например, *нет* соответствует 7).

Пример: ввод фамилии, имени и отчества.

```
Private Sub Start_Click()
    Dim Фамилия As String, Имя As String,_
    Отчество As String, soob As String
      Фамилия = InputBox("Введите, пожалуйста, _ Вашу
    фамилию:", "Ввод фамилии")
      Имя = InputBox("Введите, пожалуйста, _
    Ваше имя:", "Ввод имени")
```

```
    Отчество = InputBox("Введите, пожалуйста, _
  Ваше отчество:", "Ввод отчества")
  Text1(0).Text = Фамилия: _
  Text1(1).Text = Имя: _
  Text1(2).Text = Отчество
  soob = "Студент(ка), " + Фамилия, + _
  "Вас вызывают в деканат"
  MsgBox soob, 50, "Назначение на практику"
End Sub
```

В итоге последовательно появляются три окна ввода. После данных все текстовые поля формы будут заполнены, и появится сообщение о вызове в деканат для назначения на практику.

Иногда для выдачи сообщений проще воспользоваться выводом в текстовое поле или изменением свойства *Caption* в элементе управления *Label*.

– **Функция определения условия:**

IIf(*УсловноеВыражение,Значение1,Значение2*)

эта функция возвращает *Значение1*, если условное выражение истинно и *Значение2*, если оно ложно.

Программирование условий

Два выражения, между которыми помещен знак сравнения, называются ***простыми условиями*.** Последовательность простых условий, соединенных знаками логических операций AND, OR, NOT называется ***сложным условием***. Алгоритм, последовательность выполнения операций которого зависит от выполнения некоторых условий, проверяемых исполнителем, называют нелинейным. Реализация нелинейных алгоритмов в VB основывается на ***операторах условных переходов***.

Однострочная форма оператора условного перехода:

If *УсловноеВыражение* **Then** *Оператор1* [**Else** *Оператор2*]

Многострочная форма:

```
If УсловноеВыражение Then
    Последовательностьоператоров1
[Else
    Последовательностьоператоров2]
End If
```

Оператор условного перехода может быть с вложенными условиями. Тогда можно просто вкладывать эти операторы друг в друга:

If *УсловноеВыражение* **Then**

```
Else
       If         Then
       Else
         If           Then

         End If
       End If
End If
```

Применение конструкции **ElseIf** позволяет сократить запись (см. пример из упражнения 1).

Более простой способ записи - применение оператора **Select Case.** Синтаксис:

```
Select Case Переменная
       Case Значение1
         Последовательность операторов1
         ...
       Case Значение(N-1)
         Последовательность операторов(N-1)
       [Case Else
         Последовательность операторовN]
End Select
```

Пример: проверка значения переменной *x*

```
Select Case x
    Case Is =3
     'Выполнить некоторые действия
    Case Is>17
     'Выполнить другие действия
    Case Else
        'Действия для случая, когда не выполнено ни
        одно из предыдущих условий
End Select
```

УПРАЖНЕНИЕ 1. ПРОГРАММИРОВАНИЕ УСЛОВИЙ

Задание. Составьте алгоритм и программу вычисления данных зависимостей. Назначьте значения ***x*** отладочных тестов по всем ветвям (обратите внимание на особые точки). Выведите на экран соответствующие зависимости параметра *a* и значения *Y*. Варианты заданий по упражнению 1 приведены в табл. 1.

Таблица 1. Варианты заданий к упражнению 1.

№	Y=	a=
1	$Y=\frac{2a^2+7a-2}{x-2.5}+X^3$	$a=\begin{cases} x+1.5x^3+e^{x+1} & (-2\le x\le 5)\\ 4.5x+1.5 & (5<x\le 7.5)\\ x+1 & (x>7.5)\\ \text{не определено} & (x<-2)\end{cases}$
2	$Y=a+\frac{a}{a+1}+2.5x$	$a=\begin{cases} 3.5x & (x\ge -2)\\ x+1.5 & (x<-2)\end{cases}$
3	$Y=\frac{2}{x}+a^3$	$a=\begin{cases} 2+2x & (x\le 4)\\ 3 & (4<x\le 5)\\ x+1 & (x>5)\end{cases}$
4	$Y=\frac{2.2a+3.2a^2-2}{x-1.5}$	$a=\begin{cases} 2\sin x+4\cos^2 x^2 & (-1\le x\le 1)\\ x+3.5 & (1<x\le 5)\\ -4.6 & (x>5)\\ \text{не определено} & (x<-1)\end{cases}$
5	$Y=\frac{x+(2.2a+xa)^2}{x+1}$	$a=\begin{cases} 5+2x & (x\le 3)\\ 4 & (3<x\le 7)\\ x-1 & (x>7)\end{cases}$
6	$Y=3x+a^2-\frac{x}{2ax}$	$a=\begin{cases} \sin x+2\cos x & (-\frac{\pi}{2}\le x\le \frac{\pi}{2})\\ \frac{x-2.5}{2+x} & \text{для остальных x}\end{cases}$
7	$Y=a+2x+\frac{3x^2+a^2}{a+x}$	$a=\begin{cases} 3.2x+1 & x<3.14\\ x\sin x & (x=3.14)\\ x & (x>3.14)\end{cases}$
8	$Y=7.3-\frac{a}{(1+x)}+x^3a$	$a=\begin{cases} 2x+1 & (x\le 2)\\ \sqrt{x+3} & (x>2)\end{cases}$

9	$Y = \frac{2}{x} + a^3 + e^x$	$a = \begin{cases} x^2 + \frac{4}{\sin x + 2} & (-\frac{\pi}{2} \le x \le \frac{\pi}{2}) \\ 5 & (\frac{\pi}{2} < \text{x} \le 2.5) \\ x + 1 & (x > 2.5) \\ \text{не определено} & (x < -\frac{\pi}{2}) \end{cases}$
10	$Y = a + 2.8x + \frac{x + a}{3 - x}$	$a = \begin{cases} 4x + 2.5 & (x \le 2.5) \\ 1.5x + 8 & \text{для остальных x} \end{cases}$
11	$Y = 2.5a^2 + \sqrt{a + x}$	$a = \begin{cases} 5 & (x < 3) \\ 1.5x + 8 & (x \ge 3) \end{cases}$
12	$Y = a - \frac{x^2}{a}$	$a = \begin{cases} 3.5x & (x < 4) \\ x - 5 & (x \ge 4) \end{cases}$
13	$Y = \frac{2 + x^3}{x + 2} + a^4$	$a = \begin{cases} 3.7x & (-6 < x \le -3) \\ x^2 + 3x - 1 & (-12 < x \le -6) \\ \text{не определено для остальных x} \end{cases}$
14	$Y = \frac{2 + x}{x} + a^3 - 3x$	$a = \begin{cases} 3.7x & (-6 < x \le -3) \\ x^2 + 3x - 3 & (-3 < x \le 8) \\ \text{не определено для остальных x} \end{cases}$

Пример программного кода для варианта 1:

```
Private Sub Командная_кн_Click()
Dim Y As Double
Dim x As Double, a As Double
Dim strA As String, str1 As String, var1 As String, _
var2 As String, var3 As String
strA = "A=?"
var1 = "x+1.5x^3+e^(x+1)"
var2 = "4.5x+1.5"
var3 = "x+1"
x = Val(TxtX)
If x >= -2 And x <= 5 Then
```

```
        a = x + 1.5 * x ^ 3 + exp (x + 1)
        strA = var1
    ElseIf x > 5 And x <= 7.5 Then
        a = 4.5 * x + 1.5
        strA = var2
    ElseIf x > 7.5 Then
        a = x + 1
        strA = var3
    Else: strA = "Not defined"
End If
If strA = "Not defined" Then
        str1 = "No"
        Else: Y=(2*a^2+7*a-1)/2.5+x^3
        str1 = Str(Y)
End If
TxtA.Text = strA
TxtY.Text = str1
End Sub
```

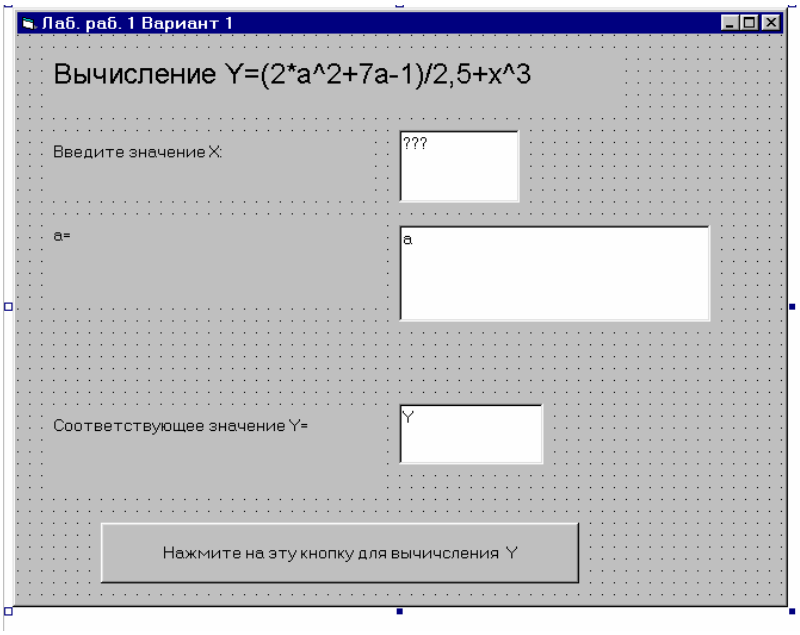

Рис.1. Пример пользовательского интерфейса к упражнению 1.

ПРОГРАММИРОВАНИЕ ПОВТОРЕНИЙ

Повторение - многократное выполнение одного или нескольких предписаний алгоритма.

Цикл - оператор, с помощью которого повторение будет выполнено компьютером. Принято различать два типа циклов: *со счетчиком* и *с условием.*

Цикл со счетчиком. Синтаксис:

For *Имя=Значение1* **To** *Значение2* [**Step** *Значение3*]
ПовторяющиесяОператорыЦикла
Next[*Имя*]

Имя - имя переменной, называемой *счетчиком* или *индексом* цикла, *Значение1* - начальное значение счетчика, *Значение2* - конечное значение счетчика, *Значение3* - величина, на которую изменяется значение счетчика за одно повторение, *ПовторяющиесяОператорыЦикла* - часть программы, которая должна повториться.

В процессе выполнения этого оператора цикла в начале проводится проверка на непротиворечивость, если есть противоречие, то работа цикла прекращается и у счетчика остается его начальное значение. Если противоречия нет, то повторяющиеся операторы будут выполнены при начальном значении счетчика, после чего его значение будет увеличено на шаг и будет снова проведена проверка на превышение конечного значения.

Циклы могут быть вложенными. Число повторений не всегда известно заранее тогда можно конечно, сделать переменной конечное значение цикла. Однако, следует следить, чтобы это значение было всегда определено до входа в цикл. По существу именно так организуется цикл с условием.

Циклы с условием. Синтаксис цикла имеет две формы в зависимости от местоположения условий.

Форма 1. **Do** *Условие*
ПовторяющиесяОператоры
Loop

Если условие стоит в первой строчке, то может случиться так, что *ПовторяющиесяОператоры* не будут выполнены ни разу. Иногда это полезно.

Форма 2. **Do**
ПовторяющиесяОператоры
Loop *Условие*

В этой форме *ПовторяющиесяОператоры* всегда будут выполнены хотя бы один раз.

Условие тоже бывает двух типов:

– С ключевым словом **While** (условие продолжения цикла). В этом случае *ПовторяющиесяОператоры* выполняются, если

значение *УсловногоВыражения* есть Истина (True), иначе цикл завершается.

– С ключевым словом **Until** (условие завершения цикла). В этом случае ПовторяющиесяОператоры выполняются, если значение УсловногоВыражения есть Ложь (False), иначе цикл завершается.

После ключевого слова должно следовать *УсловноеВыражение*. Итак, при организации циклов с условием возможны 4 варианта:

1	2	3	4
Do While... **...** **Loop**	**Do Until ...** **...** **Loop**	**Do** **...** **Loop While...**	**Do** **...** **Loop Until ...**

УПРАЖНЕНИЕ 2. ПРОГРАММИРОВАНИЕ ПОВТОРЕНИЙ

Задание. Вычислить сумму S, прекращая суммирование, когда очередной член станет меньше $eps=10^{-4}$, при изменении аргумента ***x*** в указанном диапазоне ***[a,b]*** с шагом ***h***. Для проверки в каждой точке вычислить также и функцию ***y=f(x),*** представляющую собой аналитическое выражение ряда. Варианты заданий приведены в табл. 2.

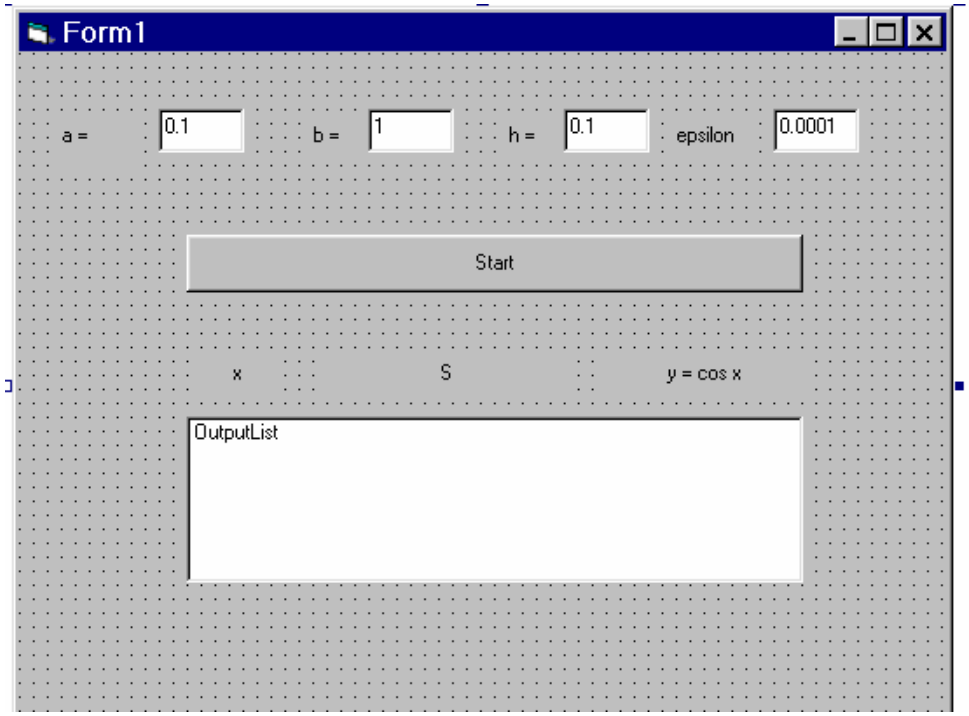

Рис. 2. Пример экранной формы в упражнении 2.

Таблица 2. Задания для упражнения 2.

№ варианта	S=	y=	a=	b=	h=
1	$\sum_{i=1}^{\infty} i^2 \cdot x^{i-1}$	$\frac{1+x}{(1-x)^3}$	0	0.9	0.1
2	$\sum_{i=1}^{\infty} (2i-1)^2 \cdot x^{i-1}$	$\frac{1+6x+x^2}{(1-x)^3}$	0	0.9	0.1
3	$\sum_{i=1}^{\infty} (-1)^{i+1} \cdot \frac{x^{2i-1}}{(2i-1)!}$	$\sin x$	0.1	0.9	0.1
4	$\sum_{i=1}^{\infty} (-1)^{i+1} \cdot \frac{x^{2(i-1)}}{2(\mathrm{i}-1)!}$	$\cos x$	0.1	1	0.1
5	$\sum_{i=0}^{\infty} \frac{(2x)^i}{\mathrm{i}!}$	e^{2x}	0.1	1	0.05
6	$\sum_{i=0}^{\infty} \frac{x^{2i}}{(2\mathrm{i})!}$	$chx = \frac{e^x + e^{-x}}{2}$	0.1	1	0.05
7	$\sum_{i=0}^{\infty} \frac{x^{(2i+1)}}{(2\mathrm{i}+1)!}$	$shx = \frac{e^x - e^{-x}}{2}$	0.1	1	0.05
8	$\sum_{i=0}^{\infty} (-1)^i \cdot \frac{x^{(2i+1)}}{(2\mathrm{i}+1)}$	$arctgx$	0.1	0.5	0.05

Продолжение таблицы 2.

9	$\sum_{i=0}^{\infty}(2i+1)\cdot x^i$	$\frac{1+x}{(1+x)^2}$	0	0.9	0.05
10	$\sum_{i=1}^{\infty}(-1)^i\frac{\cos(ix)}{i^2}$	$\frac{x^2-\frac{\pi^2}{3}}{4}$	-0.5	0.5	0.1
11	$\sum_{i=1}^{\infty}(-1)^{i+1}\frac{(x)^i}{i}$	$\ln(1+x)$	-0.5	0.5	0.1
12	$\sum_{i=1}^{\infty}(-1)^{i+1}\cdot\frac{x^{2i}}{2i(2i-1)}$	$x\cdot arctg(x)-$ $-\ln\sqrt{1+x^2}$	0.1	0.8	0.05
13	$\sum_{i=0}^{\infty}\frac{(2i+1)x^{2i}}{\mathrm{i!}}$	$(1+2x^2)\cdot e^{x^2}$	0.1	1	0.1
14	$\sum_{i=0}^{\infty}\frac{\left(\frac{x-1}{x+1}\right)^{2i+1}}{(2i+1)}$	$\frac{\ln x}{2}$	0.2	1	0.08

Указания: При выполнении этого упражнения для вывода значений аргумента, суммы ряда и функции удобно применить элемент управления, называемый *ListBox (Список)*. Для вывода следует объединить эти значения в несколько одну строку. Для добавления строк в список применяется метод *AddItem*. Примеры программного кода и интерфейса (см. рис. 2) варианта 4 приведены ниже:
Листинг программы:

```
Private Sub StartCommand_Click()
Dim a As Double, b As Double, _
h As Double, st As Double, _
epsilon As Double, member As Double, y As Double,
Dim mi as Long, fc as Long
Dim i As Integer, j As Integer
Dim tmps As String
a = Val(LeftSideText.Text)
b = Val(RightSideText.Text)
h = Val(StepText.Text)
epsilon = Val(PrecisionText.Text)
st = a
Do While st <= b
   i = 0
   mi = 1
   s = 0
    Do
      member = mi * (st ^ (2 * i))
      fc = 1
         For j = 1 To 2 * i
         Fc = fc * j
         Next j
      i = i + 1
      member = member / fc
      s = s + member
      mi = mi * (-1)
    Loop Until Abs(member) < epsilon
       y = Cos(st)
   tmps = Str(st)
   tmps = tmps + "          " + Str(s)
   tmps = tmps + "          " + Str(y)
   OutputList.AddItem (tmps)
   st = st + h
Loop
End Sub
```

ОСНОВНЫЕ СВЕДЕНИЯ О ГРАФИЧЕСКИХ СРЕДСТВАХ VB

В данном разделе изучаются объекты: *Form (Экранная форма)* и *PictureBox (Графическое окно)*. При работе с графикой для этих элементов управления могут быть применены следующие методы:

– **Scale** (масштабирование) - для задания определяемой программой декартовой системы координат;

– **Pset** (Point Set)- для изображения точки в заданной системе координат;

– **Line** - для изображения на данном объекте отрезка с заданными декартовыми координатами его концов или прямоугольник с заданными декартовыми координатами его углов;

– **Circle** - для построения кругов и эллипсов, дуг и секторов;

– **Cls** - для очистки данного объекта от всего, что было на нем созданос момощью графических методов и метода Print;

– **Print** - для печати на объекте символов.

Для того, чтобы выяснить, какими методами владеет класс объектов, можно обратиться к *Окну просмотра характеристик*, открыв его щелчком кнопки **Object Browser** на *Главной панели* проекта или выбрав соответствующую команду из меню **View**. После этого в списке *Classes* следует указать необходимый класс объектов и просмотреть появившийся справа список с его характеристиками.

Метод **Scale**

Объекты *Form* и *PuctureBox* задаются по умолчанию абсолютными координатами левого верхнего угла от левого верхнего угла на экране и размерами. Единицей измерения служит *твип* (1 твип=1/1440 от *логического дюйма*, т.е. от такого расстояния на экране, которое при печати на принтере будет равно 1 дюйму). Текущие координаты курсора мыши в твипах можно увидеть на панели инструментов. Для этого применяется Метод **Scale**:

```
[ИмяОбъекта.]Scale (X1,Y1)-(X2,Y2)
```

Здесь в условных единицах, вычисляемых относительно размеров объекта, записываются новые координаты левого верхнего и правого нижнего углов *Объекта*.

Пример: запись оператора

```
Picture1.Scale (-7,5)-(7,-5)
```

помещает начало координат в центр объекта *Графическое окно* с именем *Picture1*, которое уже имеется на экранной форме и которое разделено нами на 14 условных единиц по ширине и 10 условных единиц по высоте.

Того же результата можно добиться, если установить соответствующие свойства объекта *Picture1*:

– значение свойства *ScaleMode = 0 (User);*
– *ScaleWidth = 14* (разность координат правого и левого краев);
– *ScaleHeidth = -10* (разность координат нижнего и верхнего краев)
– *ScaleLeft = -7*
– *ScaleTop = 5*

Метод **Pset**

```
[ИмяОбъекта.] PSet (X,Y)[, Цвет]
```

Задаются координаты точки и целое число типа `Long` для кодировки цвета. Палитра цветов имеет до 16 кбит значений. Некоторые цвета обозначены через встроенные постоянные (константы) языка VB:

Цвет	Константа	Числовое значение
Черный	**vbBlack**	0
Красный	**vbRed**	255
Зеленый	**vbGreen**	65280
Желтый	**vbYellow**	65535
Синий	**vbBlue**	16711680
Сиреневый	**vbMagenya**	16711935
Голубой	**vbGyan**	167776960
Белый	**vbWhite**	16772215

Этот метод следует применять совместно с установкой свойства *DrawWidth* (размер точки, целое число 1,2...) объекта *Picture*:

Picture1.DrawWidth=3

Метод **Line**

```
[ИмяОбъекта.]Line (X1,Y1)-(X2,Y2)[,Цвет[,Флаг]]
```

По умолчанию Цвет совпадает со значением свойства объекта *ForeColor*. Флаг= символ **B**, тогда рисуется незаполненный прямоугольник, если значение свойства *FilleStile*=1 *(Transparent)*, или заполненный цветом *ForeColor* прямоугольник, если указанное свойство имеет другое значение; **BF**, тогда прямоугольник будет закрашен указанным цветом. Толщина отрезка или контура определяется свойством *DrawWidth*.

Пример:

```
'Просто линии и прямоугольники:
Private Sub Command1_Click()
'Изображение тонкого красного горизонтального отрезка:
Line(200,200) -(2200,200), 255
'Изображение толстого голубого вертикального отрезка:
DrawWidth=4
Line(200,400) -(200,1400), vbCyan
'Изображение красного наклонного отрезка средней толщины:
DrawWidth=2
Line(200,1800) -(2000,1100), vbRed
'Изображение незакрашенного прямоугольника цвета, установленного по _
умолчанию:
Line(1400,1000) -(400,500),  , B
'Изображение красного закрашенного прямоугольника:
Line(1600,500) -(2400,1000), 255, BF
End Sub
```

Метод **Circle**

[ИмяОбъекта.]**Circle** *(X,Y),Радиус[,Цвет[,Угол1_*
,Угол2[,КоэффициентСжатия]]]

Здесь (X,Y) – координаты центра. Цвет – выражение, определяемое также как для методов, описанных выше. Угол1 и Угол2 – начальный и конечный углы дуги или сектора. Углы задают в радианах в интервале от 0 до 2π. Нулевой угол соответствует горизонтальной оси, направленной вправо. Если перед ненулевым значением угла стоит знак "–" (минус), то это означает, что должна быть нарисована дуга, а не сектор. При рисовании движение «пера» происходит всегда от Угла1 к Углу2, причем независимо от соотношения между ними и против часовой стрелки. Если угли не заданы, то рисуется полный круг или эллипс. КоэффициентСжатия – это положительное число, большее или меньшее единицы. Сжатие происходит вертикали.

УПРАЖНЕНИЕ 3. ПРОГРАММИРОВАНИЕ ГРАФИКИ

Задание 3.1. Напишите программу-анимацию, изображающую на экране подъем флага.

Указания:

- Для рисования флага воспользуйтесь методом *Line*:

```
Picture1.Line (1600,2500)-(4400,3000),vbWhite, BF
Picture1.Line (1600,3000)-(4400,3500), vbBlue, BF
Picture1.Line (1600,3500)-(4400,4000), 255, BF
```

- Для анимации используйте перерисовку линии в цикле. Для того, чтобы перерисовка нижней линии соответствовала фону, следует установить в свойствах *Picture1* значение *FilleStile <>1* и проследить, что *BackColor=ForeColor*. Чтобы перерисовка шла в удобном для восприятия темпе, используйте цикл с пустыми операторами внутри.

Задание 3.2.

3.2.1. Разработайте и протестируйте программу, выполняющую построение графика заданной функции *Y = X ^ 2 - 2 * X - 3*, координатных осей и подписей на осях в заданном интервале изменения значений аргумента [-3;5].

3.2.2. Дополните программу подпрограммой определения максимального и минимального значения функции и измените программу так, чтобы ею можно пользоваться как процедурой при произвольных значениях пределов изменения аргумента и функции.

Указание: Программный код простой программы построения графика и примерный вид соответствующей экранной формы:

```
Private Sub Command1_Click()
Picture1.BackColor = 0
Picture1.Scale (-3, 5)-(5, -5)
        'Построение графика
For X = -2 To 4 Step 0.01
    Y = X ^ 2 - 2 * X - 3
    Picture1.PSet (X, Y), vbGreen
Next X
        'Axe X
Picture1.Line (-3, 0)-(5, 0), vbCyan
For i = -3 To 4
    Picture1.PSet (i, 0), vbCyan
    Picture1.Print i
Next i
        'Axe Y
    Picture1.Line (0, -5)-(0, 5), vbCyan
For i = -4 To 5
    Picture1.PSet (0, i), vbCyan
    Picture1.Print i
Next i
End Sub
```

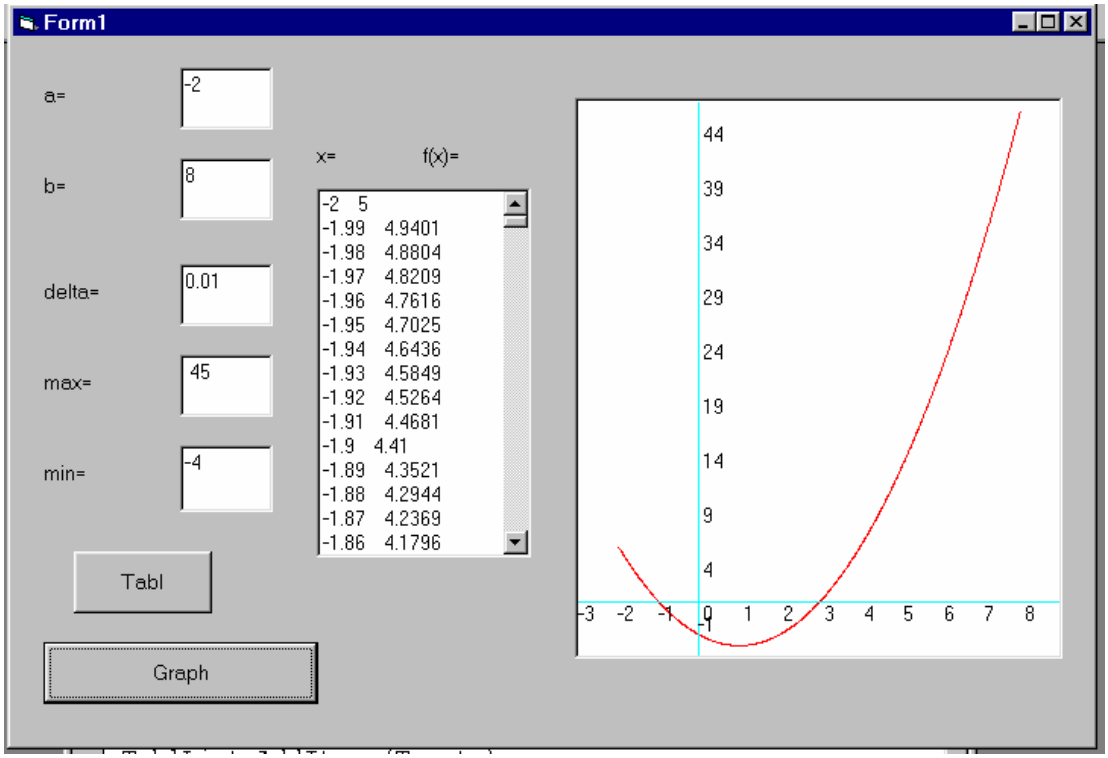

Рис. 3. Пример экранной формы с графиком функции.

ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ

ПРОГРАММА ИССЛЕДОВАНИЯ ФУНКЦИЙ

1.1. Разработайте и протестируйте программу, выполняющую построение графика заданной функции, координатных осей и подписей на осях в заданном интервале изменения значений аргумента.

1.2. Дополните программу п.1.1. так, чтобы на экранной форме выводились точки пересечения заданной функции с осью абсцисс. Решение соответствующего уравнения выполните с точностью до 0,0001 одним из численных методов: дихотомия, метод хорд, касательных и т.п.

1.3. Дополните программу п.1.2., запрограммировав вывод на экран изображения интеграла и производной функции (вариант: изображения графика производной и интеграла выполните на отдельных формах). Численное интегрирование проведите методом трапеций (методом Монте-Карло), задав не менее 100 точек для вычисления интеграла в интервале задания функции.

1.4. С точностью до 0,001 методом трапеций вычислите площадь фигуры, образованной заданной функцией между 2 и 3-им корнем и осью абсцисс.

Варианты заданий по курсовому проекту приведены в табл. 3.

Таблица 3. Варианты заданий по курсовой работе.

№	Аргумент	Функция (N=1,2…5)
1.	$x := -10, -9.9 .. 10$	$y(x) := N - \frac{((2 - 18 \cdot N \cdot x) + 16 \cdot ((N \cdot x)^2))}{1 + N \cdot (x)^4}$
2.	$x := -3, -2.99 .. 1$	$y(x) := \frac{\sqrt{N}}{2} - \left[\frac{\left[\left[1 + 2 \cdot (x)^4 - 8 \cdot (x)^2 \right]^2 - 4 \cdot N \cdot x^3 + N^2 \cdot x^4 \right]}{(1 - N \cdot x)^4} \right]$
3.	$x := -5, -4.99 .. 2$	$y(x) := 5 + \frac{\left[\left[\left(-5 + 2 \cdot x + 1 \cdot x^2 \right) + 4 \cdot N \cdot x^3 \right] + N \cdot x^4 \right]}{(1 - N \cdot x)^{\frac{4}{5}}}$
4.	$x := -1, -0.99 .. 4$	$y(x) := \left[(((2 \cdot N \cdot x - 1) \cdot (2 \cdot x - 2)) \cdot (x - 3))^2 \right]^{\frac{2}{3}} - N$
5.	$x := -4, -3.99 .. 2$	$y(x) := -2 \cdot \sqrt{N} + \frac{\left[\left(0.5 \cdot x^5 - 8 \cdot x^3 - 8 \cdot x^2 \right) - N \cdot x + 10 \right]^3}{100 \cdot (x - 1)^4}$

6.	$x := 0, 0.01 .. 2$	$y(x) := \sin\left(N \cdot x^{\frac{3}{4}} + 4 \cdot x^2\right) + 2 \cdot x^{\frac{3}{4}} - 2$
7.	$x := -1.9, -1.89 .. 3$	$y(x) := e^{-2x+1} \cdot \sqrt{N} \cdot (\sin(4 \cdot x - 1))^2 - 8 \cdot (x)^2$
8.	$x := 0.2, 0.21 .. 10$	$y(x) := -3 \cdot \sqrt{N} + \left[-\frac{\left[x^5 - (3x)^3 - N^2 \cdot x + 10 \right]}{(x+1)^4} \right]^2$
9.	$x := 0, 0.01 .. 10$	$y(x) := 0.2 \cdot (x+2) + \sqrt{N} \cdot \sin(x) + \frac{4 \cdot \sin(3 \cdot x)}{3}$
10.	$x := 0, 0.01 .. 2$	$y(x) := \frac{\ln\left(8 \cdot \cos\left(4x^2\right)^2 + 1\right) - x^3 + 0.4}{N \cdot (\lvert x \rvert)^{\frac{1}{5}} - 4}$
11.	$x := -4, -3.99 .. 4$	$y(x) := 0.25 \cdot x^2 - 0.8 \cdot \sqrt{N} + (\sin(2 \cdot x - 1))^4 + N \cdot (\cos(x))^4$
12.	$x := -3, -2.99 .. 3$	$y(x) := 0.25 \cdot x^2 + 0.1 \cdot x - 0.8 \cdot \sqrt{N} + N \cdot (\sin((2 \cdot x)))^4 + N \cdot (\cos(x))^4$

13.	$x := 2, 2.01 .. 18$	$y(x) := \left\lvert \left(x^3 - 2\,x^2 + 1\right)^{\frac{(2\cdot N\cdot \sin(x))}{6}} \right\rvert - \frac{x\cdot N}{3}$
14.	$x := -2, -1.99 .. 5$	$y(x) := \left[\left[\cos\left[\frac{\left[x^3 - N\cdot(x)^2 + x - 1\right]}{20}\right]\right]^4 + N\cdot\left(\frac{x-1}{10}\right)^2\right] - 1 + \frac{x}{20}$
15.	$x := 0, 0.01 .. 3$	$y(x) := -6 + \left[10\cdot(N\cdot(e))^{\left[\left(x^5 + 6\cdot x\right)^{\frac{1}{2}} - x^2 - x - 1\right]} - (x)^2\right]$
16.	$x := 0, 0.01 .. 6$	$y(x) := N\cdot(e)^{\frac{x-4}{8}} - N\cdot x + \left[4\cdot\sin\left[\left(\frac{x}{4}\right)^2 + x\right]\right]^2$
17.	$x := -4, -3.99 .. 5$	$y(x) := N\cdot(e)^{\frac{x}{8}} - x^2 + \left[4\cdot\sin\left[\left(\frac{x}{4}\right)^2 + \sqrt{N}\cdot x\right]\right]^2$
18.	$x := -2, -1.99 .. 16$	$y(x) := \left[\left[x^5 - N\cdot(x)^4 - 4\cdot x^3 + (N\cdot x)^2 - 6\cdot N\cdot x\right] - 10\right]\cdot e^{-(x)} - 1$

19.	$x := -10, -9.99 .. 1$	$y(x) := 0.2\cdot(x+1) + N\cdot\sin(x) + \frac{2\cdot\sin(3\cdot x - 1)}{3}$
20.	$x := -4, -3.99 .. 4$	$y(x) := (2\cdot x)^2 - N^2\cdot(\sin((2\cdot x)))^4 + 4\cdot x + \left(\cos\left(\frac{x^2}{4}\right)\right)^4$
21.	$x := -4, -3.99 .. 4$	$y(x) := -(2\cdot x)^2\cdot N\cdot\sin((2\cdot x))^4 + 4\cdot N\cdot x + \left(\cos\left(\frac{x^2}{4}\right)\right)^4$
22.	$x := 0, 0.01 .. 10$	$y(x) := 0.2\cdot(x+2) + \sin(x) + \frac{4\cdot\sqrt{N}\cdot\sin(3\cdot x)}{3}$
23.	$x := -4, -3.99 .. 4$	$y(x) := 0.25\cdot x^3 - 0.8\cdot\sqrt{N} + (\sin(2\cdot x - 1))^4 + N^2\cdot(\cos(x))^4$
24.	$x := 0, 0.01 .. 6$	$y(x) := e^{\frac{x^{x-4}}{8}} - N\cdot x + \left[4\cdot\sin\left[\sqrt{N}\cdot\left(\frac{x}{4}\right)^2 + x\right]\right]^2$

УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

Пример выполнения задания приведен на рис.4

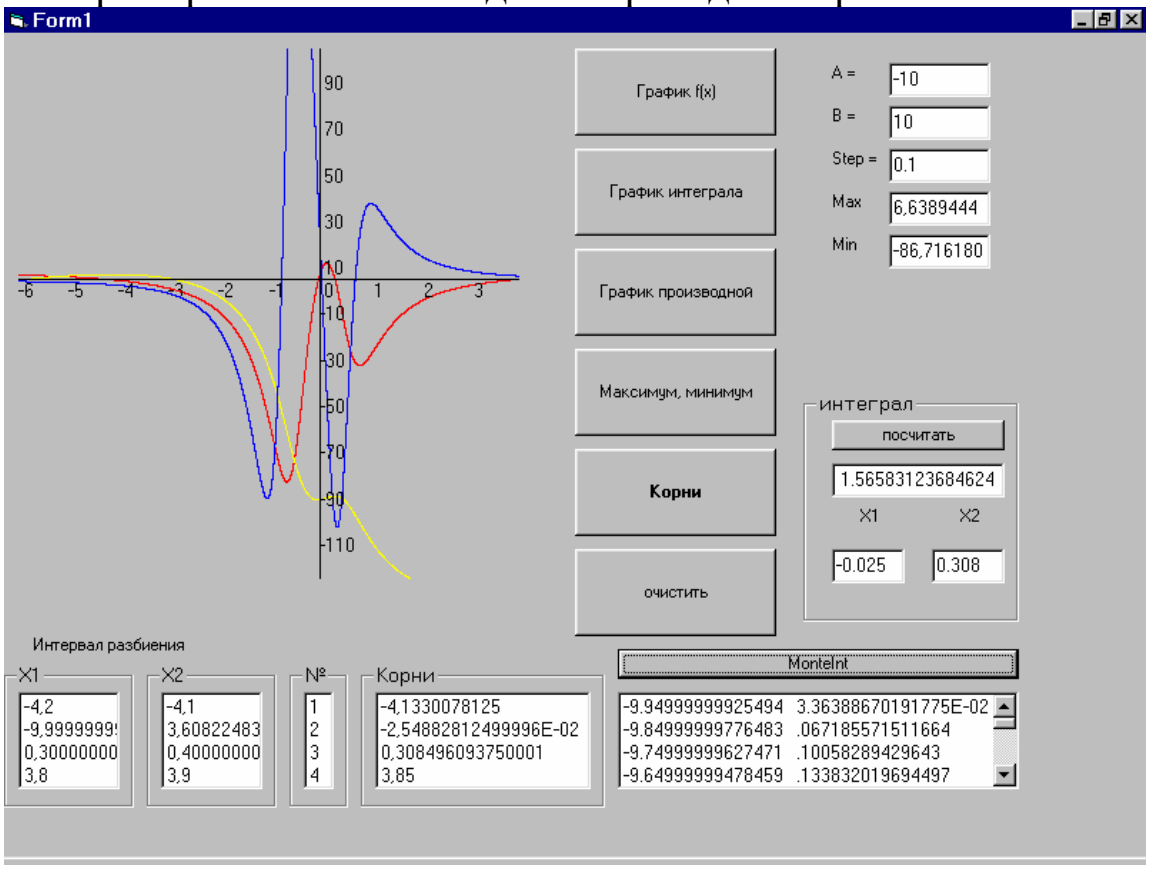

Рис.4. Результат работы программы исследования функции.

Определение корней уравнения

Дихотомия - метод последовательного деления отрезка пополам. Пусть задано уравнение $F(x)=0$, причем функция $F(x)$ непрерывна на некотором отрезке $[a,b]$ и $F(a)\cdot F(b) < 0$. Если функция меняет знак на отрезке $[a,b]$, то на этом отрезке она пересекает ось абсцисс нечетное число раз. Пусть корень только один. Разделим отрезок $[a,b]$ пополам: $x=(a+b)/2$. Далее возможны три случая:

– если $F(x)=0$, то x есть искомый корень уравнения;

– если $F(a)\cdot F(x)>0$, то перемена знака функции имеет место в правой половине отрезка и следует положить, что $a=x$ и продолжить процесс деления пополам;

– если $F(a)\cdot F(x)<0$, то перемена знака функции имеет место в левой половине отрезка и, задав $b=x$ опять продолжить процесс деления отрезка пополам;

Процесс деления пополам можно завершить, когда выполнится одно из условий: либо $F(x)<\varepsilon$, либо b-a<ε, где ε - заданная погрешность вычислений.

Отыскание корня уравнения удобно оформить отдельной функцией.

Для определения интервалов изоляции корней необходимо просмотреть последовательно пары соседних значений функции при заданном шаге разбиения области задания функции. При выполнении условия отрицательности произведения соседних значений функции следует обратиться к функции отыскания корня.

Пример: построение графика функции $Y(x)=x^2-2x-3$ и определение точек пересечения с осью абсцисс.

```
'Определение типов переменных:
Public mx, mn, a, b, d, x, ff As Double
Public num As Integer
Public i As Integer
Public xmin As Integer
Public xmax As Integer
Public ymin As Integer
Public yman As Integer
Public epsilon As Double
Public X1 As Double
Public X2 As Double
Public Maxstr, Minstr, Tmpstr As String
'Динамическое определение массива
Dim f() As Double
'Вывод таблицы значений функции и экстремальных значений:
Private Sub Command_TABL_Click()
a = Val(AText.Text)
b = Val(BText.Text)
d = Val(StepText.Text)
num = Int((b - a) / d)
ReDim f(num + 1)
mx = a ^ 2 - 2 * a - 3
mn = a ^ 2 - 2 * a - 3
TablList.Clear
For i = 0 To num
    x = a + i * d
    Tmpstr = Str(x) & "   "
    f(i) = x ^ 2 - 2 * x - 3
    If mx < f(i) Then mx = f(i)
    If mn > f(i) Then mn = f(i)
    Tmpstr = Tmpstr + Str(f(i))
```

```
    TablList.AddItem (Tmpstr)
Next i
Tmpstr = Str(B) & "   "
f(num + 1) = b ^ 2 - 2 * b - 3
    If mx < f(num + 1) Then mx = f(num + 1)
    If mn > f(num + 1) Then mn = f(num + 1)
Tmpstr = Tmpstr + Str(f(num + 1))
TablList.AddItem (Tmpstr)
MAXText.Text = Str(mx)
MINText.Text = Str(mn)
End Sub
```

'Построение графика:

```
Private Sub Command_GRAPH_Click()
Dim yymax As Integer
Dim yymin As Integer
Dim xxmin As Integer
Dim xxmax As Integer
yymax = Int(Abs(mx)) + 1
yymin = Int(Abs(mn)) + 1
xxmax = Int(Abs(B)) + 1
xxmin = Int(Abs(A)) + 1
```

'Процедура построения графика

```
Call Figureee(xxmin, xxmax, yymin, yymax)
End Sub
```

'Определение корней:

```
Private Sub RootCom_Click()
Dim FA, FB, x1, x2 As Double
Dim xroot, xroot1 As Double
Dim j As Integer
Dim Xstr As String
Dim K As Integer
K = 0
RootList1.Clear
For j = 0 To num
x = A + j * d
FA = xfunc(x)
FB = xfunc(x + d)
If FA = 0 Then
    xroot = x
    x1 = x
    x2 = x
```

```
    j = j + 1
    K = K + 1
    Xstr  =  Str(x1)  +  "       "  +  Str(x2)  +_
"    " + Str(xroot) +  "  " + Str(K)
    RootList1.AddItem (Xstr)
ElseIf FA * FB < 0 Then
    x1 = x
    x2 = x + d
    xroot = rootfunc(x1, x2)
    K = K + 1
    Xstr  =  Str(x1)  +  "       "  +  Str(x2)  +  _
"    " + Str(xroot) + "  " + Str(K)
    RootList1.AddItem (Xstr)
End If
Next j
End Sub
```

'Функция вычисления значения функции в заданной точке

```
Public Function xfunc(ByVal u As Double) As Double
    xfunc = u ^ 2 - 2 * u - 3
End Function
```

'Функция определения корня методом дихотомии:

```
Public Function rootfunc(ByVal xleft, xright As Dou-
ble) As Double
epsilon = 0.00000001
    Dim AA As Double
    Dim BB As Double
    Dim FA As Double
    Dim Fx As Double
    Dim xx As Double
    AA = xleft
    BB = xright
    FA = xfunc(AA)
    Do
        xx = (AA + BB) / 2
        Fx = xfunc(xx)
        If FA * Fx > 0 Then
        AA = xx
        Else
        BB = xx
        End If
```

```
    Loop    Until    Abs(Fx)    <    epsilon    Or    _
BB - AA < epsilon
    rootfunc = xx
End Function
```

Построение графика производной

Для построения графика производной достаточно вычислить значения производной функции в точках, соответствующих центрам интервалов сетки (т.е. между точками разбиения области задания функции). В большинстве случаев при этом можно использовать приближенное соотношение, вытекающее из определения производной. Для построения графика производной в том же графическом окне следует изменить масштаб оси ординат (практически удобнее рядом со старой осью ***0Y*** дорисовать новую ось производной ***0Y'*** в измененном масштабе).

Интегрирование функций

Метод средних прямоугольников. Отрезок интегрирования делится на n равных частей длины $h=(b-a)/h$. Обозначим $x_0=a$, $x_1=x_0+h$, ... $x_i=x_0+i\cdot h$, ... $x_n=b$, $y_i=f(x_i)$. Заменяя на каждом отрезке длиной h подинтегральную функцию $f(x)$ постоянной величиной, равной значению функции в середине интервала разбиения $f(x_i+h/2)$, получаем:

$$\int_a^b f(x)dx \approx h\left(y_{1/2}+y_{3/2}+\ldots+y_{i-1/2}+\ldots y_{n-1/2}\right)=h\left(\sum_{i=1}^{n} y_{i-1/2}\right)$$

Метод трапеций. Отрезок интегрирования делится на n равных частей длины $h=(b-a)/h$. Обозначим $x_0=a$, $x_1=x_0+h$, ... $x_i=x_0+i\cdot h$, ... $x_n=b$, $y_i=f(x_i)$. Заменяя на каждом отрезке длиной h подинтегральную функцию $f(x)$ отрезком прямой, проходящей через точки y_i,y_{i+1}, получаем формулу трапеций:

$$\int_a^b f(x)dx \approx \frac{h}{2}\left(y_0+2y_1+2y_2+\ldots+2y_{n-1}+y_n\right)=\frac{h}{2}\left(y_0+2\sum_{i=1}^{n-1} y_i+y_n\right)$$

Метод Симпсона. Предусматривает интерполяцию подинтегральной функции параболой второго порядка, проводимой через три соседние точки y_{i-1}, y_i, y_{i+1}. Поэтому отрезок интегрирова-

ния следует разделить на четное число частей. В этом случае формула Симпсона имеет вид:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(y_0 + 4y_1 + 2y_2 + ... + 4y_{n-1} + y_n\right) =$$

$$= \frac{h}{3}\left(y_0 + \sum_{i=1}^{n-1}(3 + (-1)^{i+1})y_i + y_n \right)$$

О точности вычислений. Оценка погрешности вычисления определенного интеграла делается по формулам Рунге. Практически требуется выбор надлежащего шага разбиения h. Тогда алгоритм вычисления сводится к следующему. Выбираются шаги h_1 и h_2, причем $h_2<h_1$. Используя эти шаги, по какому-либо методу вычисляют величины интеграла $A_1=A(h_1)$ и $A_2=A(h_2)$, которые суть приближения к истинному значению A. Если A_1 и A_2 оказываются близкими по некоторому критерию точности, то за искомое значение принимается A_2. В противном случае выбирается новый шаг $h_3<h_2$ и снова вычисляется значение $A_3=A(h_3)$, которое сравнивается с A_2. Удобно принимать $h_{i+1}=h_i/2$, (метод двойного пересчета). Значение A считается найденным, если выполняется условие

$$\left|A_{i+1} - A_i\right| \le \varepsilon \text{ при } \left|A_{i+1}\right| \le 1 \text{ или } \left|\frac{A_{i+1} - A_i}{A_{i+1}}\right| \le \varepsilon \text{ при } \left|A_{i+1}\right| > 1$$

Если эти условия не выполняются, то процесс уменьшения шага продолжается. Таким образом, реализация указанного метода сводится к вычислению определенного интеграла при заданном шаге h_1 и при шаге $h_2=h_1/2$, что удобно оформить в виде процедуры или функции. Полученные значения сравниваются и, если условия достижения заданной точности не выполняются, то цикл ***Do-Loop*** продолжается.

Пример: интегрирование методом срединных треугольников.

```
Public Function IntegrFunc(ByVal Left, Rigth As Dou-
ble, N As Long) As Double
    Dim summ, pas As Double
    pas = (Rigth - Left) / N
    XI = Left + pas / 2
    summ = 0
    i = 0
```

```
    For i = 1 To N
    summ = summ + xfunc(XI)
    XI = XI + pas
    Next i
    IntegrFunc = summ * pas
End Function
Private Sub IntegrCom_Click()
Dim eps,Integral1,Integral2,R,Del,Left,Rigth As Double
Dim Nnn As Long
eps = 0.000001
Nnn = 24
Left = Val(LeftText)
Rigth = Val(RigthText)
Integral1 = IntegrFunc(Left, Rigth, Nnn)
Do
Nnn = 2 * Nnn
Integral2 = IntegrFunc(Left, Rigth, Nnn)
If Abs(Integral2) <= 1 Then R = 1 Else R = Integral2
Del = Abs((Integral1 - Integral2) / R)
Integral1 = Integral2
Loop Until Del < eps
IntegText = Str(Integral2)
Nnn = 24
End Sub
```

Методы Монте-Карло. С развитием вычислительной техники все чаще применяются **статистические методы**. Их применение наиболее целесообразно при вычислении кратных интегралов, а также при решении некоторых физических задач. Разновидность одного из статистических методов сводится к тому, что при интегрировании функции на отрезке методом прямоугольников в качестве узла разбиения x_0 выбирается случайное число, распределенное на интервале интегрирования *[a,b]*. Проводится *N* вычислений со случайными узлами x_j, результат усредняется и принимается за приближенное значение интеграла:

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{j=1}^{N} f(x_j)$$

Погрешность вычисления интеграла ε зависит от числа испытаний *N*: $\varepsilon \cong N^{1/2}$.

Пример: Вычисление определенного интеграла методом Монте-Карло:

```
Private Sub Com_Monte_Int_Click()
Dim x1, x2, ff, xx, sf, MonteInt As Double
MonteInt = 0
Int_List.Clear
For i = 0 To num
x1 = A + i * d
x2 = x1 + d
sf = 0
For j = 1 To 1000
Randomize
xx = x1 + Rnd(i) * d
ff = xfunc(xx)
sf = sf + ff
Next j
sf = sf / 1000
sf = sf * d
MonteInt = MonteInt + sf
Int_List.AddItem (Str(x1 + d / 2) + " " +
Str(MonteInt))
Next i
End Sub
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Индришенок В.И., Певцов Е.Ф., Русанов К.Е. Информатика: Практикум по программированию: Основы программирования на VISUAL BASIC 6.0 // Методические указания по выполнению лабораторных работ. Московский государственный институт радиотехники, электроники и автоматики (Технический Университет). М.: 2001, 32 с.
2. Глушаков С.В., Мельников И.В., Сурядный А.С. Программирование в среде Windows: Учебный курс / Харьков: Фолио; Ростов-н/Д: Феникс; Киев: Абрис, 2000. – 487 с.
3. Волченков Н.Г. Программирование на Visual Basic 6: В 3-х ч. М.: ИНФРА-М, 2000.
4. Стивенс Р. Тестирование и отладка программ на Visual Basic: Пер. с англ. – М.: ДМК Пресс, 2001. – 384 с.
5. Ракитин В.И., Первушин В.Е. Практическое руководство по методам вычислений с приложением программ для персональных компьютеров: Учеб. пособие. – М.: Высш. шк., 1998. – 383с.