МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ПРАКТИКУМ ПО КОМПЬЮТЕРНОМУ МАТЕМАТИЧЕСКОМУ МОДЕЛИРОВАНИЮ ЧАСТЬ I:

ОСНОВЫ РАБОТЫ С ГРАФИКОЙ В СИСТЕМЕ DELPHI

### Казанский университет 2015

УДК 004.94

ББК 32.973.26–018

Рекомендовано к печати решением кафедры высшей математики и математического моделирования отделения педагогического образования Института математики и механики им. Н. И. Лобачевского ФГАОУВПО

«Казанский (Приволжский) федеральный университет»

### Научный редактор –

кандидат физ.-мат. наук Ф. Ш. Зарипов

### Рецензенты:

доктор пед. наук Л. Р. Шакирова канд. физ.-мат. наук Е. П. Шустова

**Практикум по компьютерному математическому моделированию. Часть I: Основы работы с графикой в системе Delphi:** учебно- методическое пособие / О. А. Широкова – Казань: КФУ, 2015. – 50с.

Предлагаемое учебно-методическое пособие предназначено для обеспечения самостоятельных занятий изучающим компьютерное моделирование студентам отделения педагогического образования Института математики и механики им. Н. И. Лобачевского КФУ. Пособие содержит теоретический материал, примеры и задания по следующим темам: «Построение графиков в Delphi», «Динамический рисунок с использованием графических примитивов», «Построение графика функции с заданием размеров, связанных с размерами формы» «Пример компьютерной научной графики». Учебно-методическое пособие предоставляет необходимый теоретический материал по соответствующим темам, методическую разработку лабораторных работ с демонстрациями решения типовых задач и задачи для самостоятельного выполнения.

УДК 004.94

ББК 32.973.26–018

**© Казанский университет, 2015**

**© Широкова О.А., 2015**

## ОГЛАВЛЕНИЕ

[ПОСТРОЕНИЕ ГРАФИКОВ В DELPHI 4](#_bookmark0)

[Общие сведения 4](#_bookmark1)

[Построение графика. 8](#_bookmark2)

[Точечный метод построения графика. 9](#_bookmark3)

[Пример точечного метода построения графика 11](#_bookmark4)

[Лабораторная работа № 1 14](#_bookmark5)

[Кусочно-линейный метод построение графика. 15](#_bookmark6)

[Пример кусочно-линейного метода построения графика 16](#_bookmark7)

[Лабораторная работа №2 22](#_bookmark8)

[Построение алгебраических кривых по их параметрическому](#_bookmark9) [представлению 23](#_bookmark9)

[Пример построения алгебраической кривой 23](#_bookmark10)

[Лабораторная работа №3 26](#_bookmark11)

[ДИНАМИЧЕСКИЙ РИСУНОК С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ](#_bookmark12) [ПРИМИТИВОВ 29](#_bookmark12)

[Пример динамического рисунка с использованием графики 29](#_bookmark13)

[Лабораторная работа №4 32](#_bookmark14)

[ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ С ЗАДАНИЕМ РАЗМЕРОВ,](#_bookmark15) [СВЯЗАННЫХ С РАЗМЕРАМИ ФОРМЫ 33](#_bookmark15)

[Лабораторная работа №5 37](#_bookmark16)

[Построение графика функции с помощью компонента Chart, а также с](#_bookmark17) [помощью точечного и кусочно-линейного методов на одной форме Form1.](#_bookmark17)

[. 38](#_bookmark17)

[Программа с использованием трех методов построения графика:](#_bookmark18) [точечного, линейного и с помощью компонента Chart 38](#_bookmark18)

[Лабораторная работа №6 43](#_bookmark19)

[ПРИМЕР КОМПЬЮТЕРНОЙ НАУЧНОЙ ГРАФИКИ 44](#_bookmark20)

[Литература. 50](#_bookmark21)

[Интернет-источники 50](#_bookmark22)

## ПОСТРОЕНИЕ ГРАФИКОВ В DELPHI

**Общие сведения**

Работа с графикой в Delphi предполагает обращение к канве - свойству Canvas компонентов Delphi. Canvas – это холст, который позволяет программисту иметь доступ к каждой его точке (пикселю), и, словно художнику, отображать то, что требуется.

Канва **Canvas** (холст, полотно) представляет собой область компонента, на которой можно рисовать или отображать готовые изображения. Она содержит свойства и методы, существенно упрощающие графику Delphi. Каждая точка канвы имеет координаты **X** и **Y**. Система координат канвы имеет началом левый верхний угол канвы. Координата **X** возрастает при перемещении слева направо, а координата **Y** — при перемещении сверху вниз. Направление осей показано на рисунке.

х

у

Рис1. Направление осей

Координаты измеряются в пикселях. Пиксель — это наименьший элемент поверхности рисунка, с которым можно манипулировать. Важнейшее свойство пикселя — его цвет.

В работе с графикой в Delphi в распоряжении программиста находятся канва (свойство **Canvas** компонентов), карандаш (свойство **Pen**), кисть (свойство **Brush**) того компонента или объекта, на котором предполагается рисовать.

|  |  |
| --- | --- |
| Имя | Описание |
| **Pen** | Используется для рисования простых линий. Обычно применяется для функции LineTo или при рисовании рамки для определённой  фигуры (например для функции Rectangle). |
| **Brush** | Кисть используется для заполнения области определённым  цветом. Применяется в функциях Rectangle, FillRect или FloodFill. |
| **Font** | Используется для задания шрифта, которым будет нарисован  текст. Можно указать имя шрифта, размер и т.д. |

У карандаша **Pen** и кисти **Brush** можно менять цвет (свойство **Color**) и стиль (свойство **Style**). Доступ к шрифтам предоставляет свойство канвы – **Font**. Эти инструменты позволяют отображать как текст, так и достаточно сложные графики математического и инженерного содержания, а также рисунки. Кроме этого, работа с графикой позволяет использовать в Delphi такие ресурсы Windows как графические и видеофайлы.

Конечно, не все компоненты в Delphi имеют эти свойства. Свойство Canvas имеет форма – компонент TForm. На вкладке Additional расположен специализированный компонент TImage, специально предназначенный для рисования. Также свойство Canvas имеют, например, такие компоненты как TListBox, TComboBox, TStringGrid, а главное – сама форма Form1: TForm, на которой размещены компоненты.

Основное свойство канвы – **Pixels[i, j]** типа TColor. Это двумерный массив точек (пикселей), задаваемых своим цветом. Например, **Canvas.Pixels [10,20]** соответствует цвету пикселя, 10-го слева и 20-го сверху. С массивом пикселей можно обращаться как с любым свойством: изменять цвет, задавая пикселю новое значение, или определять его цвет по хранящемуся в нем значению.

Рисование на канве происходит в момент присвоения какой-либо точке канвы заданного цвета. Каждому пикселю может быть присвоен цвет. Для стандартных цветов в Delphi определён набор текстовых констант. Увидеть его можно, открыв в Инспекторе Объектов компонента (например, формы) свойство Color. Например, выполнение оператора:

### Form1.Canvas.Pixels[100, 100]:=clRed;

приведёт к рисованию красной точки с координатами [100, 100]. Узнать цвет пиксела можно обратным присвоением:

### Color:= Form1.Canvas.Pixels[100,100];

Тип TColor определён как длинное целое (LongInt). Его четыре байта содержат информацию о долях синего (B), зелёного (G), и красного (R)

цветов. В 16-ричной системе это выглядит так: $00BBGGRR. Доля каждого цвета может меняться от 0 до 255.

Например, **Form1.Canvas.Pixels[10,20] := 0**

или **Form1.Canvas.Pixels[10,20] := clBlack** — это задание пикселю черного цвета.

### Рисование линий

Самое главное, что надо знать при рисовании линий и фигур, это различие между пером (Pen) и кистью (Brush). Всё очень просто: перо (Pen) используется при рисовании линий или рамок, а кисть (Brush) для заполнения фигуры. Ниже приведены две функции, которые используются для рисования линий и обе принадлежат TCanvas:

|  |  |  |
| --- | --- | --- |
| Имя | Описание | Пример |
| **MoveTo** | Перемещает точку начала рисования  линии в указанные координаты x и y | Canvas.MoveTo(50, 100); |
| **LineTo** | Рисует линию начиная с текущей  позиции (см. MoveTo) до указанных координат x и y. | Canvas.LineTo(50, 100); |

Эффект перемещения точки начала рисования линии также достигается при помощи установки свойства **PenPos**. Например, "Canvas.PenPos.x := 20;", "Canvas.PenPos.y := 50", или "Canvas.PenPos := Point(20,50);".

По умолчанию, точка начала рисования установлена в (0,0), то есть, если сразу вызвать "Canvas.LineTo(100,100);" то будет нарисована линия из точки (0,0) в точку (100, 100). Точка начала рисования автоматически переместится в (100, 100), то есть, если выполнить команду "Canvas.LineTo(200, 100);", то следующая линия будет нарисована из точки (100, 100) в (200, 100). Поэтому, если мы хотим рисовать линии несоединённые друг с другом, то придётся воспользоваться методом MoveTo.

Линия, нарисованная при помощи LineTo использует текущее перо канваса (типа TPen). Основные свойства пера, это ширина - "Canvas.Pen.Width := 4;" (при помощи которого можно задавать различную ширину линий), и цвет "Canvas.Pen.Color := clLime;".

### Рисование фигур

Для рисования фигур, в TCanvas предусмотрены следующие функции:

|  |  |  |
| --- | --- | --- |
| Имя | Описание | Пример |
| **Ellipse** | Рисует элипс, вписанный в невидимый квадрат с координатами верхнего левого угла и правого нижнего. Если координаты х и y у углов будут совпадать, то получится  круг. | Canvas.Ellipse(0,0,50,50); |
| **FillRect** | Заполняет прямоугольник цветом текущей кисти (brush), но никак не за  пределами него. | Canvas.FillRect( Bounds(0,0,100,100)); |
| **FloodFill** | Заполняет данную область цветом текущей кисти, до тех пор пока не  будет достигнут край. | Canvas.FloodFill(10, 10, clBlack, fsBorder); |
| **Rectangle** | Рисует прямоугольник (или квадрат), заполненный цветом текущей кисти и обрамлённый цветом текущего  пера | Canvas.Rectangle( Bounds(20, 20, 50, 50)); |
| **RoundRect** | Тоже, что и Rectangle, но с  загруглёнными углами. | Canvas.RoundRect( 20, 20,  50, 50, 3, 3); |

Ещё есть очень нужная функция TextOut, которая позволяет рисовать текст, используя шрифт, заданный в Canvas:

|  |  |  |
| --- | --- | --- |
| Имя | Описание | Пример |
| **TextOut** | Рисует данную строку на канве начиная с координат (x,y) - фон текста заполняется текущим цветом  кисти. | Canvas.TextOut(10, 10, 'Some text'); |

Функция TextOut позволяет рисовать текст, не заполняя его фон. Если Вам необходимо изменить шрифт, используемый в TextOut, то необходимо изменить свойство Font (это свойство имеет тип TFont) - например "Canvas.Font.Name := 'Verdana';", "Canvas.Font.Size := 24;" или "Canvas.Font.Color := clRed;".

**Построение графика.**

Требуется составить программу построения на экране дисплея графика функции *y = F(x).*

Решение этой задачи удобно проводить в следующем порядке:

1. Определим границы значений аргумента в декартовых координатах, в пределах которых будет строиться график X [Xmin, Xmax].
2. Для данной области значений аргумента определим предельные значения функции: Y[Ymin, Ymax]. Эти значения необязательно должны быть точными. Они могут быть оценочными снизу и сверху соответственно.
3. Зададим границы графического окна в графических координатах, в пределах которого будет рисоваться график: [Xgmin, Xgmax] – по горизонтали, [Ygmin, Ygmax] – по вертикали.
4. Учесть, что Ygmin>Ygmax, поскольку в графических координатах вертикальная ось направлена вниз.

Таким образом, имеем две системы координат [x, y] – математическая или декартова система координат и [xg, yg] – экранная система координат.

Получим формулы связи между этими системами.

Нетрудно получить формулу, связывающую экранные и математические координаты:

* 1.  *X*   *X g* max  *X g* min ( *X*  *X* 

*g g* min

*X*



 max

* *X* min

min );



(\*)

* 1.  *Y*  *Yg* max  *Yg* min (*Y*  

*g g* min



 *Y*max

* *Y*min

*Y*min ).



Здесь квадратные скобки означают округление до целого значения (функция Round).

Построение графика функции может производиться либо точечным методом, либо кусочно-линейным. При первом способе график строится как последовательность точек, расположенных максимально близко.

Производится «попикселевый» перебор значений аргумента Xg в интервале

*[Xgmln, Xgmax]* с выставлением точек с соответствующими координатами *Yg.*

При кусочно-линейном методе задается шаг ΔX и рассчитывается последовательность значений *(Xi , Yi):*

*X*  *X*  *i* *X* , *Y*  *F* ( *X* ), *i*  0,1,..., *n*, *n*  *X* max  *X* min .

*i* min *i i* *X*

Данный расчет производится в декартовой системе координат.

График строится в виде отрезков прямых, проведенных через точки *(Хi, Yi), (Xi+1,* Уi+1).

**Точечный метод построения графика.**

Создадим визуальный проект и оставим программу построения графика функции y = sin(2x + 1) для 𝑋[−2𝜋; 0] используя точечный метод.

Из условия задачи следует, что *Xmin=* -2π*, Хтax= 0*.

В этих пределах функция y = sin(2x + 1) меняется от -1до 1. Поэтому *Ymin= -1, Ymax = 1.*

Xmin= -2π Ymin= -1

Хmax= 0 Ymax = 1

Выберем следующие границы графического окна: Xgmin=10 Ygmin= 300

Xgmax=400 Ygmax=40

График строится в виде последовательности точек с математическими координатами

*Xi = Xmin+ i ∙ h; Yi =sin(2Xi+1)*; *i* = *0, …,390. (400-10=390)*

Шаг h выбирается минимально возможным, соответствующим шагу графической сетки:

𝑋𝑚𝑎𝑥−𝑋𝑚𝑖𝑛

ℎ =

𝑋𝑔 𝑚𝑎𝑥−𝑋𝑔 𝑚𝑖𝑛

0 − (−2𝜋)

= =

400 − 10

2𝜋

=

390

𝜋

195

Приведенные выше формулы перевода декартовых координат в экранные примут вид:

𝑋 = 10 + [400−10 (𝑥 + 2𝜋)] = 10 [390 (𝑥 + 2𝜋)] = 10 + [390𝑥 + 390∗2𝜋] = 400 + [195𝑥]

𝑔 0+2𝜋

40 − 300

2𝜋

2𝜋

2𝜋

𝜋

(1)

𝑦𝑔 = 300 + [

(𝑦 + 1)] = 300 + [−130(𝑦 + 1)] = 300 + [−130𝑦 − 130] = 170 − [130𝑦]

1 + 1

Вместе с графиком функции строятся оси координат. Строить их будем с помощью команды рисования линии LineTo и команды MoveTo. Найдем графические координаты точки пересечения осей, то есть выразим декартовое начало координат в графическом виде (Xg нк, Yg нк), для этого подставим значение Xнк = 0, и Yнк = 0 в формулы Xg, yg (1),

195 ∗ 0

𝑋𝑔 нк = 400 + [

] = 400

𝜋

𝑌𝑔 нк = 170 − [130 ∗ 0] = 170,

таким образом, графическое начало координат находится в точке [400;170]. Графические оси координат пройдут через эту точку.

Далее на оси *OX* строятся засечки, обозначающие точки -2, - 3, -, - 

2 2

Для нахождения их графических координат используются формулы (1).

Например: найдем графические координаты засечки -2. Используем формулу для 𝑋𝑔, при 𝑋 =-2

195 ∗ (−2)

𝑋𝑔(−2) = 400 + [  ] = 400 − 390 = 10

Вторая координата определяется осью *OX:* yg=170. Высота засечки выбирается произвольно: [170-5, 170+5] и выводится надпись:

Form1.Canvas.MoveTo(10,165); Form1.Canvas.LineTo(10,175); Form1.Canvas.TextOut(10,180,'-2п');

Аналогично строим остальные засечки.

В программе также можно предусмотреть цикл, обозначающий точки на оси *OХ,* в которых производная равна нулю.

### Пример точечного метода построения графика:

Unit unit\_new; interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls;

type

{ TForm1 }

TForm1 = class(TForm) Button1: TButton; Button2: TButton; Label1: TLabel;

procedure Button1Click(Sender: TObject); procedure Button2Click(Sender: TObject); private

{ private declarations } public

{ public declarations } end;

var

Form1: TForm1; implementation

{$R \*.lfm}

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject); var x:real;

xg,yg,i:integer; begin

Form1.Canvas.MoveTo(400,20); Form1.Canvas.LineTo(400,320); //*ось Y*

Form1.Canvas.MoveTo(5,170); Form1.Canvas.LineTo(420,170); //*ось X*

Form1.Canvas.MoveTo(10,165); Form1.Canvas.LineTo(10,175); //*засечка -2* Form1.Canvas.TextOut(10,180,'-2п');

Form1.Canvas.MoveTo(205,165); Form1.Canvas.LineTo(205,175); Form1.Canvas.TextOut(205,180,'-п');

Form1.Canvas.MoveTo(108,165); Form1.Canvas.LineTo(108,175); Form1.Canvas.TextOut(108,180,'-3/2п');

Form1.Canvas.MoveTo(303,165); Form1.Canvas.LineTo(303,175); Form1.Canvas.TextOut(303,180,'-п/2');

Form1.Canvas.MoveTo(10,165); Form1.Canvas.LineTo(10,175); Form1.Canvas.TextOut(10,180,'-2п');

Form1.Canvas.MoveTo(395,300); Form1.Canvas.LineTo(405,300); Form1.Canvas.TextOut(415,295,'-1');

Form1.Canvas.MoveTo(395,40); Form1.Canvas.LineTo(405,40); Form1.Canvas.TextOut(415,30,'1');

Form1.Canvas.MoveTo(395,105); Form1.Canvas.LineTo(405,105); Form1.Canvas.TextOut(410,100,'0,5');

Form1.Canvas.TextOut(405,175,'0');

Form1.Canvas.MoveTo(395,235); Form1.Canvas.LineTo(405,235); Form1.Canvas.TextOut(410,235,'-0,5');

Form1.Canvas.MoveTo(400,20); Form1.Canvas.LineTo(405,25); Form1.Canvas.MoveTo(400,20); Form1.Canvas.LineTo(395,25); Form1.Canvas.TextOut(380,20,'Y');

Form1.Canvas.MoveTo(420,170); Form1.Canvas.LineTo(415,165); Form1.Canvas.MoveTo(420,170); Form1.Canvas.LineTo(415,175); Form1.Canvas.TextOut(435,170,'X'); Form1.Canvas.TextOut (20, 20,'y=sin (2x+1)');

x:=-2\*pi;

for i:=0 to 390 do begin

xg:=400+round((195/(pi))\*x); yg:=170-round(130\*sin(2\*x+1)); Form1.Canvas.Pixels[xg,yg]:=clblack; x:=x+pi/195;

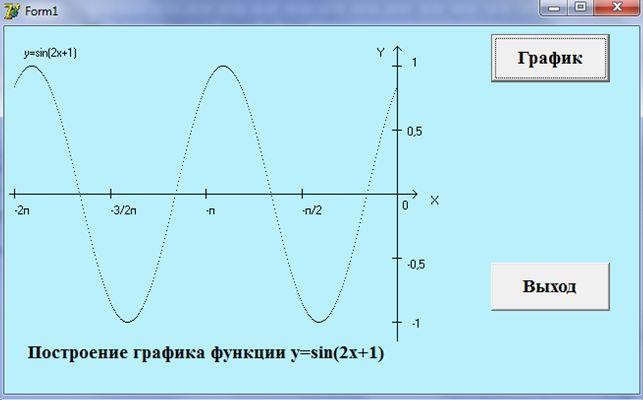
end; end;

procedure TForm1.Button2Click(Sender: TObject); begin

Form1.Close; end;

end.

Результат работы программы:



**Лабораторная работа № 1.**

Создать визуальный проект и построить график функции, используя точечный метод. Выбрать произвольно цвет фона, цвет изображения.

Построить разметки осей координат

1. *y=1/3cos(2x)* на отрезке [-2π, 2π].
2. *y=2sin(2x)* на отрезке [0, 3π].
3. *y=sin(3x+1)* на отрезке [-π, π].
4. *y=3sin(x)* на отрезке [-2π, 0].
5. *y=sin(1/2x)* на отрезке [-2π, 2π].
6. *y=cos(2x+1)* на отрезке [-2π, 0].
7. *y=1/2cos(x)* на отрезке [-π, π].
8. *y=1/2cos(2x+1)* на отрезке [-2π, 2π].
9. *y=3cos(x-1)* на отрезке [-2π, 2π].
10. *y=1+cos(x+1/2)* на отрезке [-2π, 2π].
11. *y=1-cos(x)* на отрезке [-π, 0].
12. *y=1+2cos(2x)* на отрезке [0, 3π].
13. *y=1-3sin(x)* на отрезке [-2π, 2π].
14. *y=2+sin(x)* на отрезке [-π, π].
15. *y=1-sin(1/2x)* на отрезке [0, 3π].
16. *y=1+-sin(x-1)* на отрезке [0, 3π].
17. *y=1-sin(x+1)* на отрезке [-2π, 2π].
18. *y=1-sin(2x)* на отрезке [0, 4π].
19. *y=1-2sin(1/2x)* на отрезке [-π, 4π].
20. *y=1-3sin(1/2x)* на отрезке [-π, 5π].

**Кусочно-линейный метод построение графика.**

Составим программу построения графика функции

*y*  *x*2 cos(

1

*x*  2

), *x* [*a*, *b*], используя кусочно-линейный метод.

Алгоритм действия следующий:

1. Из условия задачи следует, что *Xmin= a, Хтax= b.*
2. Предельные значения функции для данной области значений аргумента *х* определим, используя подпрограмму поиска максимума.

Тогда

*m*  max( *f* (*xi* ) ) в массиве чисел

*i*

Программный код поиска m имеет вид:

*m:=Abs(f(a)); for i:=1 to n do*

*if m<Abs(f(a+i\*h)) then m:=Abs(f(a+i\*h));*

*yi = │f(xi)│, где хi = a + i∙h,*

*h*  (*b*  *a*) *;*

*n*

*i*  0, *n,*

1. Зададим границы графического окна. В программе будем строить график данной функции 2 раза – в пределах следующих графических окон:

*а) Xgmin=100; Xgmax=500; Ygmin= 150; Ygmax=450 б) Xgmin=550; Xgmax=620; Ygmin= 10; Ygmax=100.*

1. Проведем преобразования декартовых координат в графические(\*):
2.  *X*   *X g* max  *X g* min ( *X*  

*g g* min 



*b*  *a*

*a*);



1.  *Yg* max  *Yg* min     *Yg* max  *Yg* min 

*g*  2





 *Y*

 

2*m* .

*X*  100   400 ( *X*  *a*);

*g*

А) *b*  *a* 

*Y*  300  *Y*  150 .

*g*  *m* 

*X*  550   70 ( *X*  *a*);

Б) *g* *b*  *a* 

*Y*  55  *Y*  45.

*g*  *m* 

Тогда построение графика функции на экране происходит установкой точек (*Xg,, Yg*) соответствующих координатам математическим (*xi, f(xi)*) и эти точки соединяются отрезками ломаной с применением процедуры lineto.

Оси координат на графическом экране строятся с учетом формул для экранного начала координат:

*xv*   *X*  *a*  *X g* max  *X g* min ;

 *g* min





*b*  *a* 

*yv*  *Yg* max  *Yg* min 



#  2

*xv*  

.



# 400 

100  *a*  *b*  *a* ;

А) 

*yv*  300.

*xv*  

 *a* 



# 70 ;

550



*Б)* 

*yv*  55

*b*  *a* 

### Пример кусочно-линейного метода построения графика

unit Unit1; interface uses

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

type

TForm1 = class(TForm) lbl1: TLabel;

lbl2: TLabel; lbl3: TLabel; lbl4: TLabel; lbl5: TLabel; edt1: TEdit; edt2: TEdit; edt3: TEdit; edt4: TEdit; edt5: TEdit; btn1: TButton; btn2: TButton; lbl6: TLabel;

procedure btn1Click(Sender: TObject); procedure btn2Click(Sender: TObject); private

{ Private declarations } public

{ Public declarations } end;

var

Form1: TForm1; implementation

{$R \*.dfm}

function fun(x:real):real; begin

if x<>2 then fun:=x\*x\*cos(1/(x-2)); end;

procedure grafun(xgmin,xgmax,ygmin,ygmax,n:Integer;a,b:Real); var h,m,x, t1,t2:Real;

i,xg,yg,xv,yv:Integer; begin

h:=(b-a)/n;

m:=abs(fun(a)); for i:=1 to n do

if m<abs(fun(a+i\*h)) then m:= abs(fun(a+i\*h));

t1:= (xgmax-xgmin)/(b-a); t2:=(ygmax-ygmin)/(2\*m);

Form1.Canvas.Brush.color:=clWhite; Form1.Canvas.Pen.color:=clred;

Form1.Canvas.Rectangle((xgmin-5),(ygmin-5),(xgmax+5),(ygmax+5)); xv:=Round(xgmin-a\*t1);

yv:=Round((ygmin+ygmax)/2); Form1.Canvas.Brush.color:=clblack; Form1.Canvas.MoveTo(xv,ygmin,);Canvas.LineTo(xv,ygmax); Form1.Canvas.MoveTo(xgmin,yv,);Canvas.LineTo(xgmax,yv); Form1.Canvas.MoveTo(xgmin, yv-round(fun(a)\*t2));

for i:=1 to n do begin

end; end;

x:=a+i\*h; xg:=xgmin+round((x-a)\*t1); yg:=yv-round(fun(x)\*t2); Form1.Canvas.LineTo(xg,yg);

procedure TForm1.btn1Click(Sender: TObject); var n:Integer;

a,b,a1,b1:Real; begin

a:=StrToFloat(edt1.Text); b:=StrToFloat(edt2.Text); n:=StrToInt(edt3.Text); a1:=StrToFloat(edt4.Text); b1:=StrToFloat(edt5.Text);

edt1.Visible:=False; edt2.Visible:=False; edt3.Visible:=False; edt4.Visible:=False; edt5.Visible:=False; btn1.Visible:=False; lbl2.Visible:=False; lbl3.Visible:=False; lbl4.Visible:=False; lbl5.Visible:=False; lbl1.Visible:=False;

lbl6.Caption:='Grafic funcsii y=x\*x\*cos(1/(x-2)); na otrezke['+floattostr(a)+';'+floattostr(b)+']i na otrezke['+floattostr(a1)+';'+floattostr(b1)+']'; grafun(550,620,10,100,2000, a1,b1); grafun(100,500,150,450,n,a,b);

end;

procedure TForm1.btn2Click(Sender: TObject); begin

close; end;

end.

Данная программа строит 2 графика. Первый график строится в пределах графического окна (550, 620, 10, 100), а второй график строится в

пределах графического окна (50, 450, 140, 440). В программе предполагается обращение к процедуре grafun дважды:

* первое обращение к процедуре grafun предполагает задание только диапазона изменения X в пределах [a1;b1] с помощью редакторов Edit на окне формы. Количество точек n=2000 задается в явном виде,
* второй раз необходимо предусмотреть задание параметров n, a, b с помощью редакторов Edit на окне формы Form1.

Поскольку в заданной функции

*y*  *x*2 cos(

1 )

*x*  2

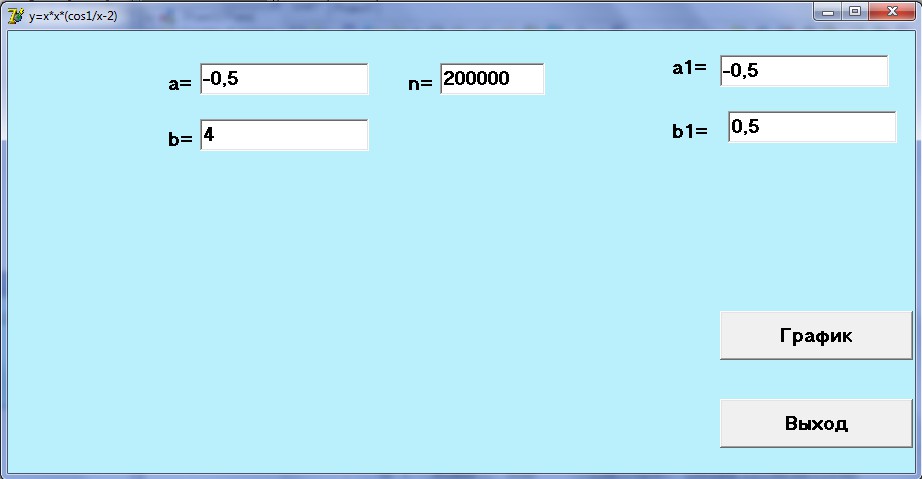
есть особенность в точке

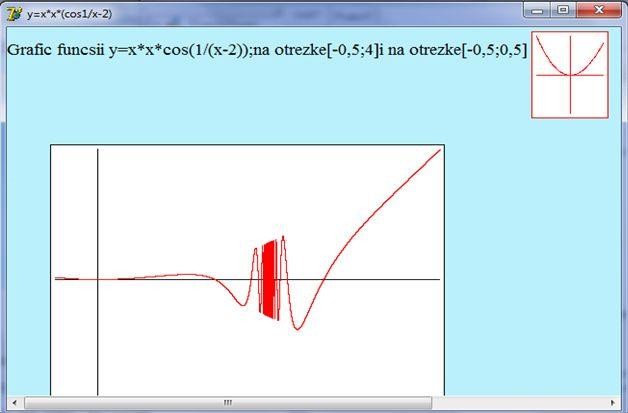
x=2, то интересно рассмотреть графики при задании разных диапазонов значения аргумента X.

Например:

В первом случае указать отрезок, включающий особую точку [-0,5;4] и не включающий [-0,5;0,5]. Во втором случае указать два отрезка, включающих особую точку, например, [-1;7] и [1,5;3].

При нажатии на кнопке График часть интерфейсных компонентов становятся невидимыми и на компоненте lbl6 и появляется надпись о функции, график которой строится.





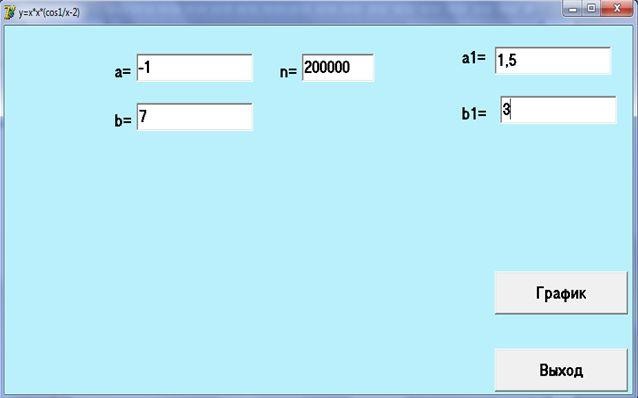
Графики функции

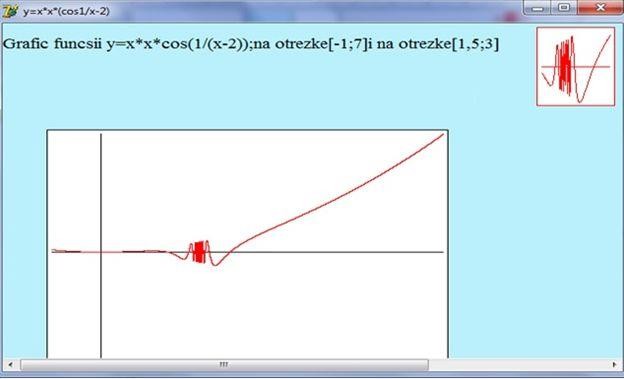
*y*  *x*2 cos(

1 )

*x*  2

на отрезке [-0,5;4] и на отрезке [-0,5;0,5]





Графики функции

*y*  *x*2 cos(

1 )

*x*  2

на отрезке [-1;7] и на отрезке [1,5;3]

**Лабораторная работа №2.**

Создать визуальный проект и построить график заданной функции

*y=f(x).* Выбрать произвольно цвет фона, цвет изображения.

|  |  |
| --- | --- |
| Номер варианта | Функция |
| 1 | *y*  *x* sin( 1 )  *x* 1 |
| 2 | *y*  *x* cos( 1 )  *x* 1 |
| 3 | *y*  *x*2 sin( 1 )  2*x* |
| 4 | *y*  *x*2 cos( 1 )  2*x* |
| 5 | *y*  *x*2 sin( 1 )  *x*  2 |
| 6 | *y*  *x*2 sin( 1 )  *x*  2 |
| 7 | *y*  *x*2 cos( 1 )  *x*  2 |
| 8 | *y*  *x*2 cos( 1 )  *x*  4 |
| 9 | *y*  2*x* sin( 1 )  *x* |
| 10 | *y*  2*x* cos( 1 )  *x* 1 |
| 11 | *y*  *x* sin( 3 )  *x* 1 |
| 12 | *y*  *x* sin( 2)  *x* |
| 13 | *y*  sin( 3 )  *x* 1 |
| 14 | *y*  2 cos( 1 )  *x* 1 |
| 15 | *y*  3sin( 1 )  *x* 1 |
| 16 | *y*  2sin( 1 )  *x* 1 |
| 17 | *y*  5sin( 1 )  *x*  2 |
| 18 | *y*  2*x* cos( 1 )  *x* 1 |
| 19 | *y*  3*x* sin( 2 )  *x*  3 |
| 20 | *y*  2*x* cos( 2 )  *x* 1 |

**Построение алгебраических кривых по их параметрическому представлению.**

Кривая *L* называется алгебраической кривой порядка *n*, если имеются, декартова система координат и многочлен *F(x,y)* степени *n*, такой что уравнение кривой *L* в этой системе координат имеет вид *F(x,y)=0.*

Для аналитического представления линии *L* часто выражают переменные *x* и *y* координат ее точек при помощи третьей, вспомогательной переменной, или параметра *t*: *x=φ(t), y=ψ(t),* где функции *φ(t)* и *ψ(t)* являются непрерывными по параметру *t* (в некоторой области изменения этого параметра).

### Пример построения алгебраической кривой

Создать визуальный проект и построить кардиоиду в центре окна по заданному параметрическому представлению:

*x=a*  cos*(t)(*1*+*cos*(t)),*

 *y=a*  sin *(t)(*1*+*cos*(t)),*



*a>*0*, t*

### Решение:

*Дано:*

0*,* 2*π*

*а* – параметр уравнений кардиоиды;

*t* – генерирующий параметр (измеряется в радианах);

*Δt* – шаг изменения значения параметра, частота вывода точек на экран.

*Построить:* кардиоиду как последовательность точек.

*Граничные условия*: *а>0, 0≤t≤2π, Δt>0. Алгоритм решения:*

1. Ввести значение *a* с клавиатуры.
2. Определить начальное значение параметра *t* и значение шага *Δt*.
3. Вычислить *x* и *y* по формулам:

*x=a∙cos(t)(1+cos(t)),*

*y=a∙sin(t)(1+cos(t)).*

1. Вывести точку с координатами (*x, y),* учитывая особенности системы координат для графического режима и координаты центра. Например, координаты центра *(320, 240).* Тогда координаты очередной точки кардиоиды равны *(320+х, 240-y)*, причем *x* и *y* предварительно необходимо округлить до значений целого типа;
2. Повторить шаги 3 и 4 для каждого значения *t*.

### Код программы:

unit Unit1; interface uses

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

type

TForm1 = class(TForm) lbl1: TLabel;

edt1: TEdit; btn1: TButton; btn2: TButton;

procedure btn1Click(Sender: TObject); procedure btn2Click(Sender: TObject); private

{ Private declarations } public

{ Public declarations } end;

var

Form1: TForm1; implementation

{$R \*.dfm}

procedure TForm1.btn1Click(Sender: TObject); var

a,x,y,t,dt:Real; begin

InvalidateRect(0, nil, true); a:=StrToFloat(edt1.Text); if a>0 then

begin

t:=0; dt:=0.01;

while t<2\*Pi do

begin x:=a\*cos(t)\*(1+cos(t));

y:=a\*sin(t)\*(1+cos(t));

Form1.Canvas.Pixels[320+Round(x), 240- Round(y)]:=clBlack; t:=t+dt;

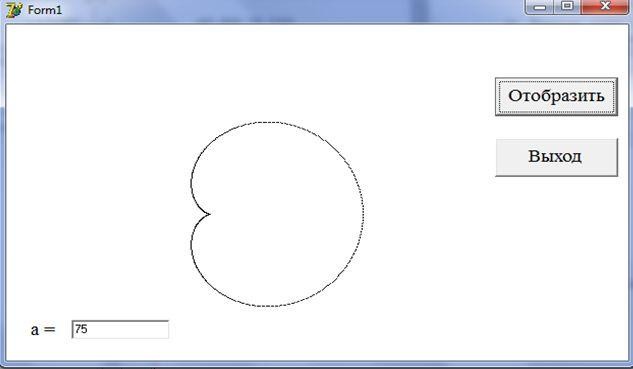
Sleep(5); end;

end; end;

procedure TForm1.btn2Click(Sender: TObject); begin

Form1.Close end;

end.



**Лабораторная работа №3.**

Создать визуальный проект и построить в центре окна кривую по заданному параметрическому представлению:

1. **Улитку Паскаля**: при *a>0, b>0*, *x=a∙cos2(t)+b∙cos(t), y=a∙cos(t)∙sin(t)+ b∙sin(t),*

*t**[0,2π].*

*(*рассмотреть случаи, когда *b≥2a, a<b<2a, a>b).*



*a>b (вариант 1)*

1. **Конхоиду Никомеда**: *x=a+l∙cos(t), y=a∙tg(t)+l∙sin(t),*

*a<b<2a (вариант 2)*

*b≥2a (вариант 3)*

*a>0,l>0, t**(-π/2,π/2)- правая ветвь, t**(π/2,3π/2)- левая ветвь (рассмотреть случаи, когда l>a,l<a, l=a)*

*l>a (вариант 4)*

### Эпициклоиду:

*x=(a+b)∙cos(t)-a∙cos((a+b)∙t/a),*

*l<a*

*(вариант 5)*

*l=a (вариант 6)*

*y==(a+b)∙sin(t)-a∙sin((a+b)∙t/a),*

*a>0,b>0 (рассмотреть случаи, когда b/a есть целое положительное число, t**[0,2π], и b/a=p/q, где p и q – положительные целые взаимно простые числа, t**[0,2qπ].*

b/a=3

*(вариант 7)*

b/a=3/2

*(вариант 8)*

|  |  |
| --- | --- |
| d. **Астроиду**:  *x=b∙cos3(t), y= b∙sin3(t), t**[0,2π].* | *(вариант 9)* |
| e. **Циссоиду**:  *x= a∙t2/(1+t2),*  *y= a∙t3/(1+t2), a>0, t**[-20,20].* | *(вариант 10)* |
| f. **Строфоиду:** *x= a∙(t2-1)/(t2+1), y= a∙t∙(t2-1)/ (t2+1),*  *a>0, t**[-2,2].* | *(вариант 11)* |
| g. **Циклоиду:**  *x= a∙( t - sin(t)),*  *y= a∙(1 - cos(t)),*  *a>0 – радиус катящейся, окружности, t**[-5,5].* | *(вариант 12)* |
| h. **Удлиненную и укороченную циклоиды:**  *x= a∙( t - λ∙sin(t)),*  *y= a∙(1 - λ∙cos(t)),*  *a>0 – радиус, окружности; рассмотреть случаи:*  *λ>1 – удлиненная циклоида λ<1 – укороченная циклоида.* | *t**[-400,400];*  λ>1*(вариант 13)* |
|  |
|  |
| *t**[-20,20];*  λ<1 *(вариант 14)* |
| 1. **Удлиненную гипоциклоиду:**   *x=(b-a)∙cos(t)+ka∙cos((b-a)∙t/a),*  *y=(b-a)∙sin(t)-ka∙sin((b-a)∙t/a), b>a>0, t**[-5,5];k>1*   1. **Укороченную гипоциклоиду:**   *x=(b-a)∙cos(t)+ka∙cos((b-a)∙t/a),*  *y=(b-a)∙sin(t)-ka∙sin((b-a)∙t/a), b>a>0, t**[-3,3];k<1* | *(вариант 15)*  *(вариант 16)* |

|  |  |
| --- | --- |
| k. **Архимедову спираль**  *y=r\*t\*cos(t) x=r\*t\*sin(t) 0<r<10, 1< t< 10π.* |  |
|  | *(вариант 17)* |
| m. **Логарифмическую спираль**  x=r1\*sin(t), y=r1\*cos(t), r1=a\*ebt ; a=2; b=0.1; t[0,10 π]; |  |
|  | *(вариант 18)* |
| n. **Эвольвенту окружности**  *x=a∙cos(t)+ at∙sin(t), y=a∙sin(t)- at∙ cos(t), a>0,*  *две ветви: t**[-10, 0]; t**[0, 10];* |  |
|  | *(вариант 19)* |
| k**. Лемнискату Бернулли**  2  2*с*2 cos 2*t* **,**  *y=ρ\*cos(t ), x=ρ\*sin(t);*  *t**(-π/4, π/4)- правая ветвь, t**(π/2,3π/2)- левая ветвь, c>100* | *(вариант 20)* |

## ДИНАМИЧЕСКИЙ РИСУНОК С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ ПРИМИТИВОВ

Создать видимость движения на экране можно таким способом: запрограммировать многократное выполнение программой набора действий: нарисовать — пауза — стереть рисунок (нарисовать его в том же месте цветом фона)— изменить координаты изображения.

Перед началом составления программы надо продумать описание

«движущегося» объекта; характер изменения координат, определяющих текущее положение объекта; диапазон изменения и шаг.

**Пример.** Динамическая модель движения Земли вокруг Солнца и движения Луны вокруг Земли (без учета физического аспекта движения планет вокруг Солнца).

Организуем, движение точки (Земли) по окружности, в центре которой размещается круг (Солнце). Установку точки на орбите осуществим по параметрическим формулам окружности:

x0:=320 + rl \* sin(Аl); y0:=240 + rl \* cos(Аl);

где rl – радиус орбиты Земли, Al – параметрический угол, меняющийся от 0 до 360 градусов. Чтобы организовать движение, достаточно в цикле устанавливать точку с координатами (х0,у0) для всех углов Аl, принимающих значение от 0 до 360 с шагом h. Аналогичная процедура справедлива и для второй точки (Луны), которая изображается по подобным формулам, в которых центр орбиты (Земля) является подвижным:

### Пример динамического рисунка с использованием графики

unit Unit1;

interface uses

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

type

TForm1 = class(TForm) btn1: TButton;

procedure btn1Click(Sender: TObject);

procedure FormKeyPress(Sender: TObject; var Key: char); private

{ Private declarations } public

{ Public declarations } end;

const pi=3.1415; var

Form1: TForm1; flag:boolean; implementation

{$R \*.dfm}

procedure TForm1.FormKeyPress(Sender: TObject; var Key: char); begin

flag:=false; end;

procedure TForm1.btn1Click(Sender: TObject); var x,y,r,rl,x0,y0:integer;

fi,h,fil,hl:real; begin

flag:=true;

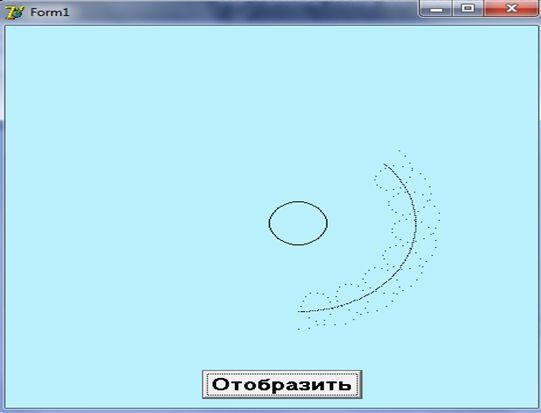
h:=20; hl:=1; fi:=0; fil:=0; r:=20; rl:=100;

Form1.Canvas.Pen.Width:=1; Form1.Canvas.Ellipse(225,200,275,250); repeat

x0:=Round(rl\*sin(fil))+250; y0:=Round(rl\*cos(fil))+225; x:=x0+round(r\*sin(fi)); y:=y0+round(r\*cos(fi)); fi:=fi+2\*pi\*h/360; fil:=fil+2\*pi\*hl/360; Form1.Canvas.Pixels[x0,y0]:=clred; Form1.Canvas.Pixels[x,y]:=clgreen; sleep(200); Form1.Canvas.Pixels[x,y]:=clblue; Form1.Canvas.Pixels[x0,y0]:=clblack;

until not flag; end;

end.



Движение Земли и Луны по окружностям (начальный вариант)



Движение Земли и Луны по окружностям (конечный вариант)

**Лабораторная работа №4**

Создать визуальный проект и смоделировать динамику движения Земли вокруг Солнца, а Луны вокруг Земли (без учета физического аспекта движения планет вокруг Солнца).

.

Номер варианта Параметры

1 h=15, h1=1;

2 h=27, h1=1;

3 h=18, h1=2;

4 h=30, h1=2;

5 h=27, h1=2;

6 h=35, h1=2;

7 h=25, h1=2;

8 h=21, h1=2;

9 h=22, h1=1;

10 h=25, h1=1;

11 h=31, h1=1;

12 h=32, h1=1;

13 h=17, h1=1;

14 h=20, h1=2;

15 h=16, h1=1;

16 h=38, h1=1;

17 h=35, h1=1;

18 h=14, h1=1;

19 h=36, h1=2;

20 h=21, h1=1;

Задания. Увеличить масштаб отображения (радиус земли, размеры планет и т.д.). Изменить все цвета всех планет, а также отображаемый след. Луна должна обойти Землю 4 раза.

## ПОСТРОЕНИЕ ГРАФИКА ФУНКЦИИ С ЗАДАНИЕМ РАЗМЕРОВ, СВЯЗАННЫХ С РАЗМЕРАМИ ФОРМЫ

*x*

Составим программу построения графика функции *y*  2sin(*x*)*e*5

используя точечный метод.

* 1. Зададим границы графического окна, в пределах которого будет рисоваться график. Определим высоту окна изображения графика, связанную с высотой формы.

*h:=Form1.ClientHeight-40*

Рабочую область высоты формы для высоты изображения уменьшаем на 40 пикселей.

Аналогично определим ширину окна изображения графика.

*w:=Form1.Width-40*

Рабочую область ширины формы для изображения уменьшаем на 40 пикселей.

* 1. Пусть *x1*:=0 нижняя граница диапазона аргумента, *x2*:=25 верхняя граница диапазона аргумента
  2. Найдем максимальное *y2* и минимальное *y1* значение функции на отрезке *[х1, х2]*.
  3. Вычислим масштаб:

*mx* 

*my* 

*h* ,

*w* .

*y*2  *y*1

*x*2  *x*1

* 1. Перейдем от декартовой системы к экранной:

*xg=x0 + x∙[mx],*

*yg=y0 - y∙[my]*, где *х0* – координата левого верхнего угла.

*(х0, у0) –* точка пересечения координатных осей экранной системы координат. Здесь квадратные скобки означают округление до целого зна- чения (функция Round).

* 1. Вместе с графиком функции строятся оси координат. Строить их будем с помощью команды рисования линии lineto.
  2. Процедура FormPaint обеспечивает вычерчивание графика после появления формы на экране в результате запуска программы.
  3. Процедура FormResize обеспечивает вычерчивание графика после изменения размера формы.

### Код программы построения графика

unit; interface uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type

TForm1 = class(TForm) procedureFormPaint(Sender: TObject); procedureFormResize(Sender: TObject); private

{ Private declarations } public

{ Public declarations } end;

var

Form1: TForm1; implementation

{$R \*.DFM}

// *Функция, график которой надо построить*

Function f(x:real):real; begin f:=2\*Sin(x)\*exp(x/5); end;

*// Процедура построения графика функции*

procedureGrOfFunc; var

x1,x2:real; *// границы изменения аргумента функции* y1,y2:real; *// границы изменения значения функции* x:real; *// аргумент функции*

y:real; *// значение функции в точке x*

dx:real; // *приращение аргумента*

l,b:integer; // *левый нижний угол области вывода графика* w,h:integer; *// ширина и высота области вывода графика* mx,my:real; // *масштаб по осям X и Y*

x0,y0:integer; // *точка начала координат*

begin // *область вывода графика*

l:=10; // X - *координата левого верхнего угла* b:=Form1.ClientHeight-20; // Y - *координата левого верхнего угла* h:=Form1.ClientHeight-40; // *высота*

w:=Form1.Width-40; // *ширина*

x1:=0; // *нижняя граница диапазона аргумента* x2:=25; // *верхняя граница диапазона аргумента* dx:=0.01; // *шаг аргумента*

// *найдем максимальное и минимальное значения функции на отрезке [x1,x2]*

y1:=f(x1); // *минимум* y2:=f(x1); // *максимум* x:=x1;

repeat

y := f(x);

if y < y1 then y1:=y; if y > y2 then y2:=y; x:=x+dx;

until (x>=x2); // *вычислим масштаб* my:=h/abs(y2-y1); // *масштаб по осиY* mx:=w/abs(x2-x1); // *масштаб по оси X* x0:=l;

y0:=b-Abs(Round(y1\*my)); with form1.Canvas do

begin // оси MoveTo(l,b);LineTo(l,b-h); MoveTo(x0,y0);LineTo(x0+w,y0);

TextOut(l+5,b-h,FloatToStrF(y2,ffGeneral,6,3)); TextOut(l+5,b,FloatToStrF(y1,ffGeneral,6,3));

// *построение графика*

x:=x1; repeat y:=f(x);

Pixels[x0+Round(x\*mx),y0-Round(y\*my)]:=clgreen;

x:=x+dx; until (x>=x2); end;

end; //конец GrOfFunc

// OnPaint *– событие при отрисовке объекта на экране (например, формы).*

procedure TForm1.FormPaint(Sender: TObject); begin

GrOfFunc; end;

// *изменился размер окна приложения, OnResize -событие при изменении размеров формы*

procedure TForm1.FormResize(Sender: TObject); begin

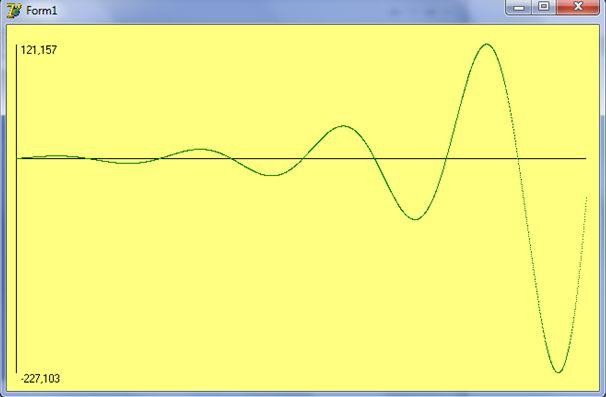
// *очистить форму*

form1.Canvas.FillRect (Rect(0,0,ClientWidth,ClientHeight));

// *построить график*

GrOfFunc; end;

end.



**Лабораторная работа №5**

Создать визуальный проект и построить график функции. Выбрать произвольно цвет фона и цвет изображения.

Варианты лабораторной работы:

|  |  |  |
| --- | --- | --- |
| Вариант | Функция | Диапазон |
| 1. | y=2sin(2x)ex | [0;24] |
| 2. | y=sin(2x)ex/5 | [0;22] |
| 3. | y=exsin(3x+1) | [0;20] |
| 4. | y=3 exsin(x) | [0;23] |
| 5. | y= ex/5sin(1/2x) | [0;21] |
| 6. | y= ex/3cos(2x) | [0;18] |
| 7. | y=1/2 ex/2cos(x) | [0;24] |
| 8. | y=1/3 ex/3cos(2x) | [0;22] |
| 9. | y=3excos(x) | [0;20] |
| 10. | y=4+ excos(x) | [0;25] |
| 11. | y=1- ex/2cos(x) | [0;19] |
| 12. | y=2+ excos(x) | [0;24] |
| 13. | y=1+ ex/2sin(x) | [0;23] |
| 14. | y= ex/5sin(5x) | [0;25] |
| 15. | y=ex/2sin(x/2) | [0;18] |
| 16. | y=1/3 ex/2sin (x) | [0;21] |
| 17. | y=1/3 ex/3 sin (2x) | [0;22] |
| 18. | y=3 ex sin (x) | [0;24] |
| 19. | y=4+ ex sin (x) | [0;20] |
| 20. | y=1- ex/2 sin (x) | [0;23] |

**Построение графика функции с помощью компонента Chart, а также с помощью точечного и кусочно-линейного методов на одной форме Form1.**

Для построения диаграмм и графиков, которые позволяют наглядно увидеть все преимущества системы и к тому же выглядят очень эффектно, существует компонент **Chart**. У него есть большой выбор методов, свойств, событий, для того что бы предоставить пользователю максимальные удобства при работе с системой.

Компонент Chart можно найти во вкладке Additional. Чтобы добавить график в Chart необходимо зайти во вкладку Additional, выбрать вид графика, затем закрыть окно редактора компонента.

Составим программу построения графика функции

*x*

*y*  2sin(*x*)*e*5

### Программа с использованием трех методов построения графика: точечного, линейного и с помощью компонента Chart:

unit Unit1; interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart;

type

TForm1 = class(TForm) chrt1: TChart;

fstSeries1: TFastLineSeries;

procedure FormPaint(Sender: TObject); procedure FormResize(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations } end;

var

Form1: TForm1; implementation

{$R \*.dfm}

function f(x:Real):Real; begin f:=2\*Sin(x)\*exp(x/5); end;

function f2(x:Real):Real; begin f:=2\*Sin(x)\*exp(x/5); end;

procedure GrOfFunc; var

x1,x2,y1,y2,x,y,dx,mx,my:Real; l,b,w,h,x0,y0:Integer;

begin

l:=10;

b:=Form1.ClientHeight-Round(Form1.ClientHeight/2)-25; h:=Form1.ClientHeight-Round(Form1.ClientHeight/2)-25; w:=Form1.Width-Round(Form1.Width/2);

x1:=0; x2:=25; dx:=0.01;

y1:=f(x1);

y2:=f(x1); x:=x1; repeat

y:=f(x);

if y<y1 then y1:=y; if y>y2 then y2:=y; x:=x+dx;

until (x>=x2); my:=h/abs(y2-y1); mx:=w/abs(x2-x1) ; x0:=1;

y0:=b-Abs(Round(y1\*my)); with Form1.canvas do begin

MoveTo(l,b); LineTo(l,b-h); MoveTo(x0,y0); LineTo(x0+w,y0);

TextOut(l+5,b-h, FloatToStrF(y2,ffGeneral,6,3)); TextOut(l+5,b, FloatToStrF(y1,ffGeneral,6,3));

//*построение графика точечным методом на Canvas и на компоненте Chart*

x:=x1; repeat y:=f(x);

*// построение точечным методом*

Pixels[x0+Round(x\*mx), y0-round(y\*my)]:=clgreen;

*// построение на компоненте Chart* Form1.chrt1.SeriesList[0].addxy(x,y,'',clRed); x:=x+dx;

until (x>=x2);

end;

l:=10;

b:=Form1.ClientHeight-20;

h:=Form1.ClientHeight-Round(Form1.ClientHeight/2)-25; w:=Form1.Width-Round(Form1.Width/2);

x1:=0; x2:=25; dx:=0.01;

y1:=f2(x1);

y2:=f2(x1);

x:=x1; repeat

y:=f2(x);

if y<y1 then y1:=y; if y>y2 then y2:=y; x:=x+dx;

until (x>=x2); my:=h/abs(y2-y1); mx:=w/abs(x2-x1) ; x0:=1;

y0:=b-Abs(Round(y1\*my)); with Form1.canvas do begin

MoveTo(l,b); LineTo(l,b-h); MoveTo(x0,y0); LineTo(x0+w,y0);

TextOut(l+5,b-h, FloatToStrF(y2,ffGeneral,6,3)); TextOut(l+5,b, FloatToStrF(y1,ffGeneral,6,3));

// *построение графика кусочно-линейным методом на Canvas*

x:=x1;

Form1.Canvas.MoveTo(Round(x1), y0-round(f2(x1)\*my)); repeat

y:=f2(x);

LineTo(x0+Round(x\*mx), y0-round(y\*my)); x:=x+dx;

until (x>=x2); end;

end; //конец GrOfFunc

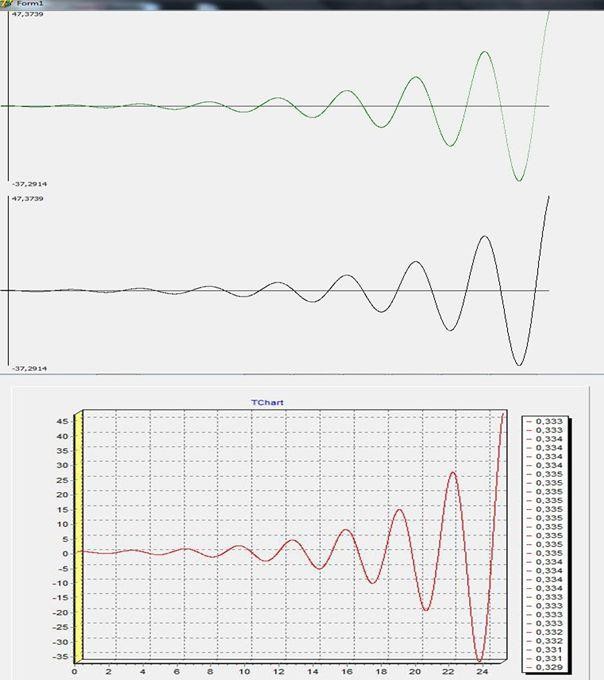
procedure Tform1.FormPaint(Sender:Tobject); begin

GrOfFunc; end;

procedure TForm1.FormResize(Sender: TObject); begin

Form1.Canvas.FillRect(Rect(0,0,ClientWidth,ClientHeight)); GrOfFunc;

end; end.



**Лабораторная работа №6**

Создать визуальный проект и построить график функции тремя способами на одной форме.

Варианты лабораторной работы:

Вариант Функция

1. y=sin(2x)ex/5

2. y= ex/3cos(2x)

3. y=1/3 ex/2sin (x)

4. y=3 ex sin (x)

5. y= 2+ ex/5sin(1/2x)

6. y=1- ex/2 sin (x)

7. y=1- ex/2cos(x)

8. y=3 exsin(x)

9. y=1- ex/2 sin (x)

10. y=4+ ex sin (x)

11. y=exsin(3x+1)

12. y=3 exsin(x)

13. y= ex/5sin(1/2x)

14. y=2 ex/2cos(x)

15. y=1/2 ex/2cos(x)

16. y=1/3 ex/3cos(2x)

17. y=3excos(x)

18. y=4+ excos(x)

19. y=2+ excos(x)

20. y=1+ ex/2sin(x)

## ПРИМЕР КОМПЬЮТЕРНОЙ НАУЧНОЙ ГРАФИКИ

Условные цвета, условное контрастирование. Интересный прием современной научной графики – условная раскраска. Она находит широчайшее применение в самых разных приложениях науки.

Приведем примеры. В различных исследованиях температурных полей встает проблема наглядного представления результатов.

Самый простой способ – привести карту (чертеж, план), в некоторых точках которой обозначены значения температуры. Другой способ – набор изотерм. 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 замечательная наглядная картина температурного поля.

***Пример. Условная раскраска неравномерно нагретого стержня в разные моменты времени*** *(по заранее заготовленным данным).*

Рассмотрим динамику изменения температуры в стержне длиной 4 м с теплоизолированными концами, температура на которых поддерживается постоянной и равна 3°С с заданным начальным условием.

Ограничимся пятью узлами на пространственной сетке. В начальный

момент *(t = 0)* температура в пяти узлах задана: *u* (0) = 3,0000, *u* (0) = 3,667, *и* (0) =

4,333, *и* (0)

3

*=* 5,000, *и* (0)

4

*=* 3,0000.

0 1 2

Из краевых условий получаем *и* (1)

0

*= и* (1)

*=* 3,000.

Решая данную задачу, получаем распределение температуры по длине стержня в узлах пространственной сетки в разные моменты времени t (см. решение задачи теплопроводности в части II: «Компьютерное моделирование физических процессов»).

4

Результаты моделирования процесса теплопроводности, представим в виде таблицы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *x*  *t* | 0 | 1 | 2 | 3 | 4 |
| 0 | 3,000 | 3,667 | 4,333 | 5,000 | 3,000 |
| 1 | 3,000 | 3,628 | 4,128 | 3,952 | 3,000 |
| 2 | 3,000 | 3,514 | 3,783 | 3,593 | 3,000 |
| 3 | 3,000 | 3,377 | 3,546 | 3,396 | 3,000 |
| 4 | 3,000 | 3,267 | 3,381 | 3,272 | 3,000 |
| 5 | 3,000 | 3,187 | 3,266 | 3,188 | 3,000 |
| 6 | 3,000 | 3,131 | 3,185 | 3,131 | 3,000 |
| 7 | 3,000 | 3,091 | 3,129 | 3,091 | 3,000 |
| 8 | 3,000 | 3,064 | 3,090 | 3,064 | 3,000 |
| 9 | 3,000 | 3,044 | 3,063 | 3,044 | 3,000 |
| 10 | 3,000 | 3,031 | 3,044 | 3,031 | 3,000 |

Таблица 2. Моделирование процесса теплопроводности в стержне

unitUnit1; interface uses

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

Type

TForm1 = class(TForm) procedureFormPaint(Sender: TObject); private

{ Private declarations } Public

{ Public declarations } end;

var

Form1: TForm1; Implementation

{$R \*.dfm}

procedure Raskraska; varDate,

u:array[0..10,0..4] of Double;

M, I, J, N1, Nt : Integer; MaxF, L, T, HI, Ht : Double; X\_N, Shag, Y\_N, Shir, Dlin, Color, K, Y : Integer;

// U:array [0..4,0..10] of Double;

Flag: Boolean; Ff : String; Col : Array [0..15] Of TColor;

Begin Date[0,0]:= 3 ;

Date[1,0]:= 3;

Date[2,0]:= 3;

Date[3,0]:= 3;

Date[4,0]:= 3;

Date[5,0]:= 3;

Date[6,0]:= 3;

Date[7,0]:= 3;

Date[8,0]:= 3;

Date[9,0]:= 3;

Date[10,0]:= 3;

Date[0,1]:= 3.667;

Date[1,1]:= 3.628;

Date[2,1]:= 3.514;

Date[3,1]:= 3.377;

Date[4,1]:= 3.267 ;

Date[5,1]:= 3.187;

Date[6,1]:= 3.131;

Date[7,1]:= 3.091;

Date[8,1]:= 3.064 ;

Date[9,1]:= 3.044;

Date[10,1]:= 3.031;

Date[0,2]:= 4.333;

Date[1,2]:= 4.128;

Date[2,2]:= 3.783;

Date[3,2]:= 3.546;

Date[4,2]:= 3.381;

Date[5,2]:= 3.266;

Date[6,2]:= 3.185;

Date[7,2]:= 3.129 ;

Date[8,2]:= 3.090;

Date[9,2]:= 3.063;

Date[10,2]:= 3.044;

Date[0,3]:= 5.000;

Date[1,3]:= 3.952;

Date[2,3]:= 3.593;

Date[3,3]:= 3.396;

Date[4,3]:= 3.272 ;

Date[5,3]:= 3.188 ;

Date[6,3]:= 3.131;

Date[7,3]:= 3.091 ;

Date[8,3]:= 3.064 ;

Date[9,3]:= 3.044 ;

Date[10,3]:= 3.031;

Date[0,4]:= 3;

Date[1,4]:= 3;

Date[2,4]:= 3;

Date[3,4]:= 3;

Date[4,4]:= 3 ;

Date[5,4]:= 3 ;

Date[6,4]:= 3;

Date[7,4]:= 3;

Date[8,4]:= 3;

Date[9,4]:= 3;

Date[10,4]:= 3;

L := 4; T := 10; Hi := 1; Ht := 1;

N1 := Trunc(L / HI); Nt := Trunc(T / Ht); MaxF := 5; X\_N :=Form1.Width div 6;

If Nt<= 6 Then M :=Nt Else M := NtDiv 2;

Y\_N := Form1.Height Div M - 20; Shir := Y\_N Div 2;

Dlin := Form1.Height -2 \* X\_N; Shag := Trunc(Dlin / N1); Str(Shag,Ff) ;

*//Палитра цветов*

Col[0] := 0; Col[1] :=RGB(128,128,128); Col[2] := RGB(80,0,128);

Col[3] := RGB(0,0,255); Col[4] := RGB(0,225,225); Col[5] := RGB(0,255,255);

Col[6] := RGB(128,128,128); Col[7] := RGB(0,255,0); Col[8] := RGB(255,255,0);

Col[9] := Rgb(225,0,225); Col[10] := rgb(255,0,255); Col[11] :=RGB(255,0,0); Col [12] := RGB(225,0,0);

For I := 0 To M - 1 Do *// номер временного промежутка Begin*

For J := 0 To N1 - 1 Do *// номер участка стержня*

Begin

Flag := False;

For K := 0 To Shag Do Begin

For y:= 0 To Shir Do

Begin *// определение номера цвета*

Color := 1 + Round((Date[I, J] + (Date[I, J + 1] - Date[I, J]) \* K / Shag - Date[0, 0]) / 3 \* 16);

If Random(64) > 32

Then If Random(64) > 32 Then Color := Color + 1 Else Color := Color - 1;

If Not Flag Then *// вывод текущей температуры*

Begin

Str((Date[I,J]+(Date[I,J+1]-Date[I,J])\*K/Shag) : 5 : 3, Ff); Form1.Canvas.TextOut(K+X\_N+Shag\*J,Y\_N\*(1+1)-19,Ff); Flag := True;

End;

*// рисование точки*

Form1.Canvas.Pixels[K+X\_N+Shag\*J, Y+Y\_N\*(1+I)]:= Col[Color]; End;

End; End;

*End;*

End;

procedure TForm1.FormPaint(Sender: TObject); begin

Raskraska; end;

end.

Результатом выполнения данной программы будет диаграмма условной раскраски неравномерно нагретого стержня с теплоизолированными концами, температура на которых равна 3°С, в разные моменты времени. Диаграмма отображена на форме проекта:



### Литература.

1. *Могилев А. В., Пак Н. И., Хённер Е. К.* Информатика: Учеб. пособие для студ. высш. учеб. заведений - М.: Издательский центр «Академия», 2004.– 816с.
2. *Могилев А. В., Пак Н. И., Хённер Е. К.* Практикум по информатике: Учеб. пособие для студ. высш. учеб. заведений - М.: Издательский центр

«Академия», 2001.–608с.

1. *Культин Н. Б.* Delphi в задачах и примерах / Н. Б. Культин. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2008. — 288 с. : ил. + CD-ROM.

- ISBN 978-5-94157-997-6.

<http://znanium.com/bookread.php?book=350283>

1. *Хомоненко А. Д.* Delphi 7 / А. Д. Хомоненко, В. Э. Гофман, Е. В. Мещеряков. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2010. — 1136 с. — (В подлиннике). - ISBN 978-5-9775-0425-6.

<http://znanium.com/bookread.php?book=350727>

1. *Сулейманов Р. Р.* Компьютерное моделирование математических задач. Элективный курс [Электронный ресурс]: учебное пособие / Р. Р. Сулейманов. - Эл. изд. - М.: БИНОМ. Лаборатория знаний, 2012. - 381 с.: ил. - ISBN 978-5-9963-1484-3.

<http://znanium.com/bookread.php?book=485565>

1. *С. А. Канцедал* Алгоритмизация и программирование : Учебное пособие / С. А. Канцедал. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 352 с.: ил.;

60x90 1/16. - (Профессиональное образование). (переплет) ISBN 978-5- 8199-0355-1

<http://znanium.com/catalog.php?bookinfo=391351>

1. *Мясникова О. К.* Моделирование и формализация в курсе информатики// Информатика и образование. – 2003. – № 9, 10, 11, 12.
2. *Павлова И. М.* Графика на Паскале// Информатика и образование. – 2003. – №7 –с.54-61

### Интернет-источники

1. [www.delphisources.ru](http://www.delphisources.ru/)
2. [www.delphi.int.ru](http://www.delphi.int.ru/)
3. [www.beluch.ru/progr/100comp/index.htm](http://www.beluch.ru/progr/100comp/index.htm)
4. <http://znanium.com/bookread.php?book=319046>
5. <http://znanium.com/bookread.php?book=251095>