Министерство науки и высшего образования Российской Федерации

Курский государственный университет

Кафедра ПОиАИС

Основы программирования графики

Отчёт о выполнении лабораторной работы №1

По дисциплине “Компьютерная графика”

Выполнил Студент группы 313 Мусонда.С

Проверил доц. Е.А. Сухотерин

Курск – 2019

# Цель работы

Ознакомление с основными возможностями выбранного языка программирования по созданию произвольных изображений и практическое освоение графических процедур и функций.

# Задание

На выбранном самостоятельно языке программирования создать программу, обеспечивающую построение графика функции, заданной индивидуальным вариантом.

Требования к программе:

* 1. Выводить на экран название работы, номер варианта задания и фамилию автора, название графика.
  2. Позволять в интерактивном режиме изменять значения параметров, включая шаг изменения независимой переменной, с перерисовкой графика функции.
  3. Позволять независимо выводить и отключать оси, масштабные линейки и координатную сетку. Вид линеек и координатной сетки должен соответствовать системе координат, в которой задана функция.
  4. Позволять изменять масштаб вывода графика.
  5. Обеспечивать восстановление (перерисовку) изображения после того, как часть окна программы была невидимой (например, закрыта другими окнами или находилась за пределами экрана).

|  |  |  |
| --- | --- | --- |
| № | Название графика | Формула функции |
| 14 | Циклоида |  |

Краткие теоретические сведения, описание и блок-схемы основных алгоритмов.

При моделировании, расчёте и выводе изображений приходится иметь дело как минимум с двумя системами координат. Первая, мировая, система координат, в которой описывается исходный объект, который надо изобразить. Мировая система координат может иметь произвольный тип, расположение и ориентацию, в зависимости от решаемой задачи. В большинстве случаев это — вещественная правая декартова прямоугольная или полярная система координат. Вторая, экранная, система координат, в которой описывается изображение на экране.

С точки зрения определения координат точки на экране (или задания точки по координатам) дисплей ЭВМ представляет собой прямоугольную сетку (растр) пикселов одинаковой формы, которые представляют собой минимальный адресуемый элемент. При этом, как в обычной таблице, номер строки увеличивается сверху вниз, а номер столбца — слева направо. Таким образом, если принять номер строки, в которой расположен пиксел, за его экранную ординату xЭ, а номер столбца — за экранную абсциссу yЭ, то экранная система координат представляет собой левую декартову прямоугольную систему координат, начало которой расположено в верхнем левом углу экрана. Кроме того, поскольку экранные координаты представляют собой номера (строк и столбцов), экранная система координат является целочисленной.

Расчёт изображения, например, графика функции, удобнее выполнять в мировых координатах, а все графические процедуры и функции языков программирования, как правило, используют экранные координаты. Таким образом, перед выводом на экран необходимо перейти от мировых координат к экранным. Это преобразование имеет вид:

Здесь:

,, – Экранные и мировые координаты точки.

- масштабные коэффициенты при переходе от мировой системы координат к экранной, где и - пределы изменения абсциссы и, соответственно, ординаты в мировой системе координат в окне вывода; , - ширина и высота окна вывода в пикселах.

угловые скобки означают округление.

– экранные координаты начала мировых координат. В случае его расположения в центре экрана , .

Блок-схема алгоритма перевода мировой координаты x точки в декартовы представлена на рисунке 1.

Входные данные: x-мировая абсцисса точки, - пределы изменения абсциссы, – ширина области вывода.

Выходные данные:x1-экранная абсцисса точки.



Округлить x1

*x*,,

Рисунок 1 – блок - схема алгоритма перевода мировой абсциссы точки в экранную

Блок-схема алгоритма перевода мировой координаты y точки в декартовы представлена на рисунке 2.

Входные данные: y-мировая ордината точки, - пределы изменения ординаты, – высота области вывода.

Выходные данные:y1-экранная ордината точки.



Округлить y1

*y*,,

Рисунок 1 – блок - схема алгоритма перевода мировой ординаты точки в экранную

Для построения гладкой (в общем случае) кривой графика функции необходимо выполнить её дискретизацию с некоторым заданным шагом. Тогда кривая превращается в набор точек. Способ дискретизации зависит от вида заданной функции:

При параметрическом задании функции (x = fx(t), y = fy(t)) дискретизация осуществляется по параметру и набор точек имеет вид (при постоянном шаге

дискретизации)

(xi = fx(ti), yi = fy(ti)), ti = tmin + (i − 1)dt, i = 1,Nt, dt =tmax – tmin/Nt – 1.

Блок-схема алгоритма построения графика функции представлена на рисунке 3.

Входные данные: - пределы изменения параметра, – количество значений параметра,a – постоянное значение для функции.

Выходные данные:отсутствуют.



Провести отрезок между точками с экранными координатами, соответствующими точкам с мировыми координатами (x1,y1) и (x,y)

,,a

Рисунок 3 – блок – схема алгоритма построения графика функции

Для вывода оси OX(OY) необходимо провести отрезок между точками с экранными координатами, соответствующими точкам с мировыми координатами (xmin,0) и (xmax,0) ((0,ymin) и (0,ymax)).

Для вывода координатной сетки можно поступить следующим образом: ввести количество вертикальных Nx и горизонтальных Ny линий. Тогда расстояние между линиями равны:

,

Блок – схема алгоритма рисования вертикальных представлена на рисунке 4.

Входные данные: – число вертикальных линий - пределы изменения абсциссы.

Выходные данные:отсутствуют.



Провести отрезок между точками с экранными координатами, соответствующими точкам с мировыми координатами (xmin+i\*DX,ymin) и (xmin+i\*DX,ymax)

,

Рисунок 4 – блок – схема алгоритма построения вертикальных линий сетки



,

Провести отрезок между точками с экранными координатами, соответствующими точкам с мировыми координатами (xmin,ymin+i\*DY) и (xmax,ymin+i\*DY)

Рисунок 4 – блок – схема алгоритма построения горизонтальных линий сетки

Формирование горизонтальной и вертикальной масштабных линеек сводится к выполнению тех же двух циклов, что и при выводе координатной сетки

только вместо рисования прямой линии нужно вывести само значение текущей абсциссы или ординаты.

Листинг программы

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

int xmin,xmax,ymin,ymax,cx,cy;

double tmin,tmax,m;

double DX,DY,a;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

int SCX(double x,TImage\*Image1)

{double x1=((double)(Image1->Width)/(xmax-xmin))\*x+Image1->Width/2 ;

int x2=round(x1);

return x2;

}

int SCY(double x,TImage\*Image1)

{double x1=((double)(Image1->Height)/(ymax-ymin))\*x\*(-1)+Image1->Height/2 ;

int x2=round(x1);

return x2;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

Image1->Canvas->Pen->Color=clWhite;

Image1->Canvas->Rectangle(0,0,Image1->Width,Image1->Height);

xmin=StrToInt(Edit3->Text);

a=StrToFloat(Edit1->Text);

xmax=StrToInt(Edit4->Text);

ymin=StrToInt(Edit5->Text);

ymax=StrToInt(Edit6->Text);

cy=StrToInt(Edit7->Text);

m=StrToFloat(Edit11->Text);

xmin\*=(double)1/m;

xmax\*=(double)1/m;

ymin\*=(double)1/m;

ymax\*=(double)1/m;

cx=StrToInt(Edit8->Text);

tmin=StrToFloat(Edit9->Text);

tmax=StrToFloat(Edit10->Text);

int nt=StrToInt(Edit2->Text);

if (CheckBox4->Checked==true){

Image1->Canvas->Pen->Color=clBlack;

Image1->Canvas->Pen->Style=psDot;

DX=(double)(xmax-xmin)/(cx+1);

DY=(double)(ymax-ymin)/(cy+1);

Image1->Canvas->Pen->Style=psDot;

for(int i=1;i<=cy;i++)

{

Image1->Canvas->MoveTo(SCX(xmin,Image1),SCY(ymin+i\*DY,Image1));

Image1->Canvas->LineTo(SCX(xmax,Image1),SCY(ymin+i\*DY,Image1));}

for(int i=1;i<=cx;i++)

{Image1->Canvas->MoveTo(SCX(xmin+i\*DX,Image1),SCY(ymin,Image1));

Image1->Canvas->LineTo(SCX(xmin+i\*DX,Image1),SCY(ymax,Image1));}

}

if (CheckBox3->Checked==true)

{ DX=(double)(xmax-xmin)/(cx+1);

DY=(double)(ymax-ymin)/(cy+1);

for(int i=1;i<=cx;i++)

{ double x=round((xmin+i\*DX)\*100)/100;

Image1->Canvas->TextOut(SCX(xmin+i\*DX,Image1)+3,SCY(0,Image1)+3,FloatToStr(x));}

for(int i=1;i<=cy;i++)

{ double y=round((ymin+i\*DY)\*100)/100;

Image1->Canvas->TextOut(SCX(0,Image1)+3,SCY(ymin+i\*DY,Image1)-3,FloatToStr(y));}

}

if (CheckBox1->Checked==true) {

Image1->Canvas->Pen->Style=psSolid;

Image1->Canvas->Pen->Color=clBlack;

Image1->Canvas->MoveTo(SCX(xmin,Image1),SCY(0,Image1));

Image1->Canvas->LineTo(SCX(xmax,Image1),SCY(0,Image1));

Image1->Canvas->MoveTo(SCX(xmax,Image1)-7,SCY(0,Image1)+3);

Image1->Canvas->LineTo(SCX(xmax,Image1),SCY(0,Image1));

Image1->Canvas->MoveTo(SCX(xmax,Image1)-7,SCY(0,Image1)-3);

Image1->Canvas->LineTo(SCX(xmax,Image1),SCY(0,Image1));

Image1->Canvas->TextOut(SCX(xmax,Image1)-7,SCY(0,Image1)-20,"X");

}

if (CheckBox2->Checked==true)

{ Image1->Canvas->Pen->Style=psSolid;

Image1->Canvas->Pen->Color=clBlack;

Image1->Canvas->MoveTo(SCX(0,Image1),SCY(ymin,Image1));

Image1->Canvas->LineTo(SCX(0,Image1),SCY(ymax,Image1));

Image1->Canvas->MoveTo(SCX(0,Image1)+3,SCY(ymax,Image1)+7);

Image1->Canvas->LineTo(SCX(0,Image1),SCY(ymax,Image1));

Image1->Canvas->MoveTo(SCX(0,Image1)-3,SCY(ymax,Image1)+7);

Image1->Canvas->LineTo(SCX(0,Image1),SCY(ymax,Image1));

Image1->Canvas->TextOut(SCX(0,Image1)+7,SCY(ymax,Image1)+10,"Y");}

double DT= (double)(tmax-tmin)/(double)(nt-1);

double t=tmin;

double x=0,y=0;

Image1->Canvas->Pen->Color=clBlack;

for(int i=1;i<=nt;i++)

{ double x1=x;

double y1=y;

x=a\*(cos(t)+t\*sin(t));

y=a\*(sin(t)-t\*cos(t));

Image1->Canvas->Pen->Style=psSolid;

Image1->Canvas->Pen->Color=clBlack;

Image1->Canvas->Pixels[SCX(x,Image1)][SCY(y,Image1)]=clBlack;

if(i!=1)

{Image1->Canvas->MoveTo(SCX(x1,Image1),SCY(y1,Image1));

Image1->Canvas->LineTo(SCX(x,Image1),SCY(y,Image1));}

t=tmin+(i-1)\*DT;}

}

//---------------------------------------------------------------------------

Результаты работы программы

Тестирование задачи 1 представлено на рисунках 5 – 6.

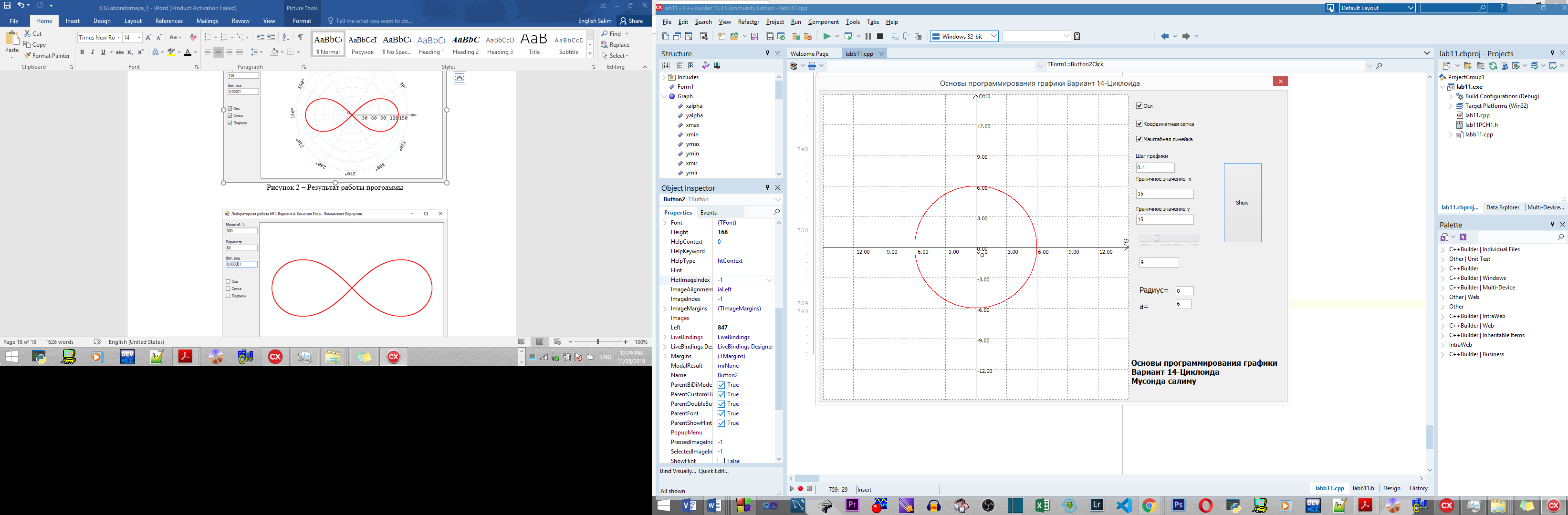


Рисунок 5 – Тест 1 задачи 1

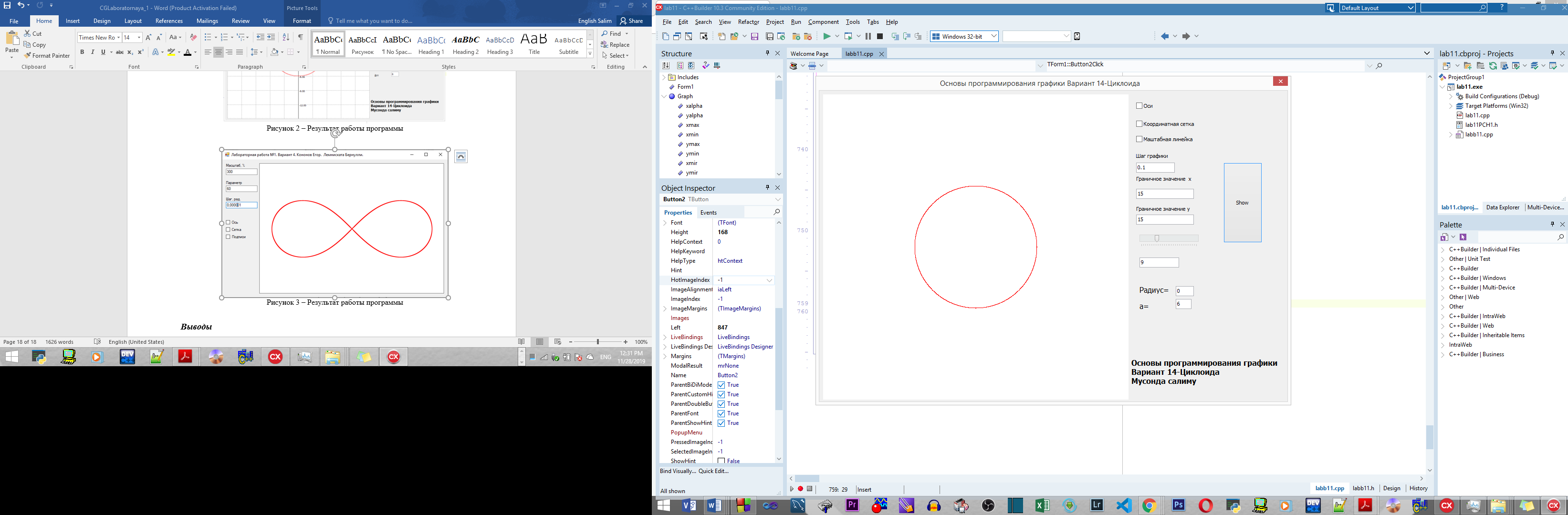


Рисунок 6 – Тест 2 задачи 1

Выводы

В результате выполнения лабораторной работы я ознакомился с основными возможностями языка c++ по созданию произвольных изображений и практически освоил работу с графическими процедурами и функциями.