Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра прикладной математики и программирования

ОТЧЕТ

о выполнении лабораторной работы № 5 по дисциплине
«Математические основы компьютерной графики»

Автор работы,
студент группы ЕТ-211
_____ Савонин М.В.
«____» _____ 2022 г.

Руководитель работы,
старший преподаватель
_____ Шелудько А.С.
«____» _____ 2022 г.

Челябинск 2022

1 ЗАДАНИЕ

1. Написать программу для построения гладкой кривой по четырем опорным точкам. При выборе опорных точек текущие координаты указателя мыши должны отображаться в графическом окне. Интерфейс программы должен содержать следующие элементы управления:

- выбор опорных точек;
- построение кубической кривой Безье;
- построение кривой по алгоритму Чайкина;
- сохранение результата в файл;
- выход из программы.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

КРИВАЯ БЕЗЬЕ

Пусть x_i, y_i одна из n точек кривой. Находим треугольник Паскаля и берём строку номер n в массив pasc.

Проходимя от t=0 до 1 с шагом step.

Проходим от i=0 до i<n с шагом 1.

рисуем линию от предыдущей точки до точки:

$$x = pasc_i * (1-t)^{n-i-1} * t^i * x_i.$$

$$y = pasc_i * (1-t)^{n-i-1} * t^i * y_i.$$

КРИВАЯ ЧАЙКИНА

Берём первые 2 точки, находим на отрезке между ними 1/4 и 3/4 и забываем предыдущие.

Потом берём 2 и 3 точку и выполняем те же действия до последней точки.

Выполняем эти действия 5 раз для получившихся точек.

рисуем линии от точки к точки по получившимся точкам.

3 ТЕКСТ ПРОГРАММЫ

Файл main.cpp

```
#define _USE_MATH_DEFINES
#include <iostream>
#include <random>
#include <time.h>
#include <math.h>

#include "graphics.h"
#include "control.h"
#include "draw.h"

using namespace std;

struct Point
{
   int x, y;
   Point *next = NULL;
};

struct Fill
{
   int x, y;
   int color;
};

struct act
{
   int num = 0;
   lineB lin;
   star sta;
   circl circ;
   Fill fil;
   lineBez linBz;
   lineCha lineCh;
};

void pefog(int pef[100], int k, int n)
{
   if(n <= k) return;
   int a[100] = {0};
   for(int i = 0; i < k; i++) a[i] = pef[i];
   for(int i = 1; i < k+1; i++)
   {
      pef[i] = a[i-1]+a[i];
   }
   pefog(pef, k+1, n);
}

int main()
```

```
{
    setlocale(LC_ALL,"Russian");
    setlocale(LC_ALL,"rus");
    srand(time(0));
    initwindow(1200, 900);

    setlinestyle(SOLID_LINE, 0, 1);
    setbkcolor(WHITE);

    srand(time(0));

    IMAGE *imager;
    imager = loadBMP("Red.bmp");

    IMAGE *imageg;
    imageg = loadBMP("Green.bmp");

    IMAGE *imageb;
    imageb = loadBMP("Blue.bmp");

    int col[3][2] = {{255, 0}, {255, 0}, {255, 0}};
    int r = 0, wrt[2] = {0};
    int n = 5, p = 0, t = 0;
    int mode = 0;

    lineB but1 = lineB(20, 25, 50, 45, BLACK, 3);
    star but2 = star(85, 35, 24, 5, 0, BLACK);
    circl but3 = circl(135, 35, 21, BLACK);

    lineB but4[5];
    but4[0].setStart(160, 43);
    but4[0].setEnd(177, 60);
    but4[0].setColor(BLACK);
    but4[0].setWidth(2);
    but4[1].setStart(160, 27);
    but4[1].setEnd(193, 60);
    but4[1].setColor(BLACK);
    but4[1].setWidth(2);
    but4[2].setStart(160, 10);
    but4[2].setEnd(210, 60);
    but4[2].setColor(BLACK);
    but4[2].setWidth(2);
    but4[3].setStart(177, 10);
    but4[3].setEnd(210, 43);
    but4[3].setColor(BLACK);
    but4[3].setWidth(2);
    but4[4].setStart(193, 10);
    but4[4].setEnd(210, 26);
    but4[4].setColor(BLACK);
    but4[4].setWidth(2);

    lineBez but5 = lineBez(BLACK, 2);
```

```
but5.add(220, 20);
but5.add(250, 20);
but5.add(220, 50);
but5.add(250, 50);
but5.setStep(0.02);

lineCha but6 = lineCha(BLACK, 2);
but6.add(270, 20);
but6.add(300, 20);
but6.add(270, 50);
but6.add(300, 50);

lineB butsave[18] = {lineB(1150, 20, 1177, 20, BLACK, 1),
                                lineB(1177, 20, 1180, 23, BLACK, 1)
                                lineB(1180, 23, 1180, 50, BLACK, 1)
                                lineB(1180, 50, 1150, 50, BLACK, 1)
                                lineB(1150, 50, 1150, 20, BLACK, 1)
                                lineB(1155, 20, 1155, 28, BLACK, 1)
                                lineB(1155, 28, 1175, 28, BLACK, 1)
                                lineB(1175, 28, 1175, 20, BLACK, 1)
                                lineB(1170, 22, 1173, 22, BLACK, 1)
                                lineB(1173, 22, 1173, 26, BLACK, 1)
                                lineB(1173, 26, 1170, 26, BLACK, 1)
                                lineB(1170, 26, 1170, 22, BLACK, 1)
                                lineB(1155, 50, 1155, 31, BLACK, 1)
                                lineB(1155, 31, 1175, 31, BLACK, 1)
                                lineB(1175, 31, 1175, 50, BLACK, 1)
                                lineB(1157, 35, 1173, 35, BLACK, 1)
                                lineB(1157, 40, 1173, 40, BLACK, 1)
                                lineB(1157, 45, 1173, 45, BLACK, 1)

lineB butclear[13] = {lineB(1130, 20, 1115, 35, BLACK, 5),
                               lineB(1116, 36, 1099, 49, BLACK, 1),
                               lineB(1114, 34, 1101, 51, BLACK, 1),
                               lineB(1116, 36, 1098, 48, BLACK, 1),
                               lineB(1114, 34, 1102, 52, BLACK, 1),
                               lineB(1117, 37, 1097, 47, BLACK, 1),
                               lineB(1113, 33, 1103, 53, BLACK, 1),
                               lineB(1117, 37, 1096, 46, BLACK, 1),
                               lineB(1113, 33, 1104, 54, BLACK, 1),
                               lineB(1118, 38, 1095, 45, BLACK, 1),
                               lineB(1112, 32, 1105, 55, BLACK, 1),
                               lineB(1118, 38, 1094, 44, BLACK, 1),
                               lineB(1112, 32, 1106, 56, BLACK, 1)}

lineB butback[3] = {lineB(1080, 35, 1050, 35, BLACK, 3),
                               lineB(1050, 35, 1065, 20, BLACK, 3),
                               lineB(1050, 35, 1065, 50, BLACK, 3)}
act acts[1000];
int countDo = 0;
int start[2] = {0}, end[2] = {0};
```

```
while(1)
{
   p = 1-p;
   setactivepage(p);
   clearviewport();

   drawAll(acts, countDo);

   setcolor(BLACK);
   rectangle(10, 10, 60, 60);
   but1.draw();

   setcolor(BLACK);
   rectangle(60, 10, 110, 60);
   but2.draw();

   setcolor(BLACK);
   rectangle(110, 10, 160, 60);
   but3.draw();

   setcolor(BLACK);
   rectangle(160, 10, 210, 60);
   for(int i = 0; i < 5; i++) but4[i].draw();

   setcolor(BLACK);
   rectangle(210, 10, 260, 60);
   but5.draw();

   setcolor(BLACK);
   rectangle(260, 10, 310, 60);
   but6.draw();

   setcolor(BLACK);
   rectangle(1140, 10, 1190, 60);
   for(int i = 0; i < 18; i++)
   {
      butsave[i].draw();
   }

   setcolor(BLACK);
   rectangle(1090, 10, 1140, 60);
   for(int i = 0; i < 13; i++)
   {
      butclear[i].draw();
   }

   setcolor(BLACK);
   rectangle(1040, 10, 1090, 60);
   for(int i = 0; i < 3; i++)
   {
      butback[i].draw();
   }
```

```
lineColor(10, 810, col, 0, imager);
lineColor(10, 840, col, 1, imageg);
lineColor(10, 870, col, 2, imageb);

if(abs(mousey()-35) <= 25 and mousebuttons())
{
   if(abs(mousex()-35) <= 25)
   {
      mode = 1;
   }
   if(abs(mousex()-85) <= 25)
   {
      mode = 2;
   }
   if(abs(mousex()-135) <= 25)
   {
      mode = 3;
   }
   if(abs(mousex()-185) <= 25)
   {
      mode = 4;
   }
   if(abs(mousex()-235) <= 25)
   {
      mode = 5;
   }
   if(abs(mousex()-285) <= 25)
   {
      mode = 6;
   }
}

if(abs(mousex()-630) <= 560 and abs(mousey()-450) <= 350 and mo
{
   switch(mode)
   {
      case 1:
      {
         addLine(acts, countDo, COLOR(col[0][0], col[1][0], col
         break;
      }
      case 2:
      {
         addStar(acts, countDo, COLOR(col[0][0], col[1][0], col
         break;
      }
      case 3:
      {
         addCircle(acts, countDo, COLOR(col[0][0], col[1][0], c
         break;
      }
```

```
case 4:
{
   int x = mousex();
   int y = mousey();
   fillIn(x, y, COLOR(col[0][0], col[1][0], col[2][0]), g
   while(mousebuttons());
   acts[countDo].num = 4;
   acts[countDo].fil.x = x;
   acts[countDo].fil.y = y;
   acts[countDo].fil.color = COLOR(col[0][0], col[1][0],
   countDo++;
   break;
}
case 5:
{
   addLineBez(acts, countDo, COLOR(col[0][0], col[1][0],
}
case 6:
{
   Point all[100];
   int brea = 0;
   int count = 0;
   int numer = -1;
   int mousepos[2];
   acts[countDo].num = 6;
   acts[countDo].lineCh = lineCha(COLOR(col[0][0], col[1]
   while(1)
   {
      p = 1-p;
      setactivepage(p);

      drawAll(acts, countDo);
      mousepos[0] = mousex();
      mousepos[1] = mousey();
      while(1-mousebuttons() or abs(mousepos[0]-600) > 59
      {
         mousepos[0] = mousex();
         mousepos[1] = mousey();
         numer = -1;
         if(kbhit()) if(getch() == 13)
         {
            brea = 1;
            break;
         }
      }
      if(brea)
      {
         countDo++;
         delay(200);
         break;
      }
      for(int i = 0; i < count; i++)
```

```
                {
                    if(abs(mousepos[0]-all[i].x) <= 7 and abs(mousep
                    {
                        numer = i;
                        break;
                    }
                }
                if(numer == -1)
                {
                    all[count] = {mousepos[0], mousepos[1]};
                    count++;
                    while(mousebuttons());
                }
                else
                {
                    all[numer].x = mousepos[0];
                    all[numer].y = mousepos[1];
                }
                acts[countDo].lineCh.delRoot();
                for(int i = 0; i < count; i++)
                {
                    circl dot = circl(all[i].x, all[i].y, 5);
                    dot.draw();
                    acts[countDo].lineCh.add(all[i].x, all[i].y);
                }
                acts[countDo].lineCh.draw();
                setvisualpage(p);
                delay(10);
            }
        }
    }
}
t = t+1;
t = t%8;

setvisualpage(p);
if(abs(mousey()-35) <= 25 and mousebuttons())
{
    if(abs(mousex()-1165) <= 25)
    {
        save(10, 70, 1190, 800);
    }
    if(abs(mousex()-1115) <= 25)
    {
        countDo = 0;
    }
    if(abs(mousex()-1065) <= 25 and countDo > 0)
    {
        countDo = countDo-1;
        while(mousebuttons());
    }
}
```

```
        delay(10);
    }

    getch();
}
```

Файл draw.h

```
#ifndef DRAW_H
#define DRAW_H

struct Point;
Point *addPoint(int, int, Point*);

class lineB
{
   private:
      int x1, y1, x2, y2;
      int color = WHITE, width = 1;

   public:
      lineB(){}
      lineB(int, int, int, int);
      lineB(int, int, int, int, int);
      lineB(int, int, int, int, int, int);
      void setColor(int);
      void setStart(int, int);
      void setEnd(int, int);
      void setWidth(int);
      void draw();
};

class star
{
   private:
      int x, y, r;
      int n = 5, colorLine = WHITE, colorFill = WHITE;
      float fi;
      lineB lines[50];
      void creatStar();
      void lineColor();

   public:
      star(){}
      star(int, int, int, int);
      star(int, int, int, int, float);
      star(int, int, int, int, float, int);
      star(int, int, int, int, float, int, int);
      star(const star &obj);
      ~star(){/*delete[] lines;*/}
      void setRadius(int);
      void setCenter(int, int);
      void setNumCorners(int);
      void setColorLine(int);
      void setColorFill(int);
      void setCorners(int);
      void draw();
```

```
        void fill();
};

class circl
{
    private:
        int x, y, r;
        int colorLine = WHITE, colorFill = WHITE;

    public:
        circl(){}
        circl(int, int, int);
        circl(int, int, int, int);
        circl(int, int, int, int, int);
        void setCenter(int, int);
        void setRadius(int);
        void setColorLine(int);
        void setColorFill(int);
        void draw();
        void fill();
};

class lineBez
{
    private:
        int n = 0, color = WHITE, width = 1;
        int data[100][2];
        double step = 0.02;
        void pefog(int[100], int, int);

    public:
        lineBez();
        lineBez(int);
        lineBez(int, int);
        void add(int, int);
        void setColor(int);
        void setWidth(int);
        void setStep(double);
        void draw();
};

class lineCha
{
    private:
        int color = WHITE, width = 1;
        Point *root;
        Point *chaikin(Point *);
        void drawL(Point *);
        void del(Point *);

    public:
        lineCha();
```

```
        lineCha(int);
        lineCha(int, int);
        ~lineCha();
        void delRoot();
        void add(int, int);
        void setColor(int);
        void setWidth(int);
        void draw();
};

struct Fill;
struct act;
void fillIn(int, int, int, int);
void drawAll(act[1000], int);
void addLine(act[1000], int&, int);
void addStar(act[1000], int&, int);
void addCircle(act[1000], int&, int);
void addLineBez(act[1000], int&, int);
void save(int, int, int, int);

#endif
```

Файл draw.cpp

```
#include <math.h>
#include <iostream>
#include "graphics.h"
#include "draw.h"

using namespace std;

struct Point
{
   int x, y;
   Point *next = NULL;
};

Point *addPoint(int x, int y, Point *root)
{
   if(root == NULL)
   {
      root = new Point;
      root->x = x;
      root->y = y;
      root->next = NULL;
      return root;
   }
   root->next = addPoint(x, y, root->next);
   return root;
}
```

```
// line ------------------------------------------------------------------


lineB::lineB(int x, int y, int x0, int y0)
{
   x1 = x;
   y1 = y;
   x2 = x0;
   y2 = y0;
   color = WHITE;
   width = 1;
}

lineB::lineB(int x, int y, int x0, int y0, int color0)
{
   x1 = x;
   y1 = y;
   x2 = x0;
   y2 = y0;
   color = color0;
   width = 1;
}

lineB::lineB(int x, int y, int x0, int y0, int color0, int width0)
{
   x1 = x;
   y1 = y;
   x2 = x0;
   y2 = y0;
   color = color0;
   width = width0;
}

void lineB::setColor(int color0)
{
   color = color0;
}

void lineB::setStart(int x, int y)
{
   x1 = x;
   y1 = y;
}

void lineB::setEnd(int x, int y)
{
   x2 = x;
   y2 = y;
}

void lineB::setWidth(int w)
{
```

```
    width = w;
}

void lineB::draw()
{
    int dx = abs(x2-x1);
    int dy = abs(y2-y1);
    int err = 0;
    if(dy < dx)
    {
        if(x1 > x2)
        {
            int a = x1;
            x1 = x2;
            x2 = a;
            a = y1;
            y1 = y2;
            y2 = a;
        }
        int d = dy == 0 ? 0 : (y2-y1)/dy;
        int y = y1;
        for(int x = x1; x <= x2; x++)
        {
            for(int i = 0; i < width; i++)
            {
                int y0;
                y0 = (i+1)/2*pow(-1, i);
                putpixel(x, y+y0, color);
            }
            err = err+dy;
            if(err >= dx)
            {
                y += d;
                err -= dx;
            }
        }
    }
    else
    {
        if(y1 > y2)
        {
            int a = x1;
            x1 = x2;
            x2 = a;
            a = y1;
            y1 = y2;
            y2 = a;
        }
        int d = dx == 0 ? 0 : (x2-x1)/dx;
        int x = x1;
        for(int y = y1; y <= y2; y++)
        {
```

```
            for(int i = 0; i < width; i++)
            {
                int x0;
                x0 = (i+1)/2*pow(-1, i);
                putpixel(x+x0, y, color);
            }
            err = err+dx;
            if(err >= dy)
            {
                x += d;
                err -= dy;
            }
        }
    }
}

// star ---------------------------------------------------------------
void star::creatStar()
{
    int x0, y0;
    int m = (n-1)/2;
    m = m<2 ? 2 : m;
    x0 = x+int(r*(cos(3.14/n*m)/cos(3.14/n*(m-1)))*cos(fi));
    y0 = y+int(r*(cos(3.14/n*m)/cos(3.14/n*(m-1)))*sin(fi));
    for(int i = 1; i < n*2+1; i++)
    {
        int y1, x1;
        if(i%2)
        {
            x1 = x+int(r*cos(i*3.14*1/n+fi));
            y1 = y+int(r*sin(i*3.14*1/n+fi));
        }
        else
        {
            x1 = x+int(r*(cos(3.14/n*m)/cos(3.14/n*(m-1)))*cos(i*3.14*1/
            y1 = y+int(r*(cos(3.14/n*m)/cos(3.14/n*(m-1)))*sin(i*3.14*1/
        }
        lines[i-1].setWidth(1);
        lines[i-1].setStart(x0, y0);
        lines[i-1].setEnd(x1, y1);
        x0 = x1;
        y0 = y1;
    }
    lineColor();
}

void star::lineColor()
{
    for(int i = 0; i < n*2; i++)
    {
        lines[i].setColor(colorLine);
    }
```

```
}

star::star(int x0, int y0, int r0, int n0)
{
   x = x0;
   y = y0;
   r = r0;
   n = n0;
   fi = 0;
   colorLine = WHITE;
   colorFill = WHITE;
   creatStar();
}

star::star(int x0, int y0, int r0, int n0, float fi0)
{
   x = x0;
   y = y0;
   r = r0;
   n = n0;
   fi = fi0;
   colorLine = WHITE;
   colorFill = WHITE;
   creatStar();
}

star::star(int x0, int y0, int r0, int n0, float fi0, int colorLine0)
{
   x = x0;
   y = y0;
   r = r0;
   n = n0;
   fi = fi0;
   colorLine = colorLine0;
   colorFill = WHITE;
   creatStar();
}

star::star(int x0, int y0, int r0, int n0, float fi0, int colorLine0,
{
   x = x0;
   y = y0;
   r = r0;
   n = n0;
   fi = fi0;
   colorLine = colorLine0;
   colorFill = colorFill0;
   creatStar();
}

star::star(const star &obj)
{
```

```
    x = obj.x;
    y = obj.y;
    r = obj.r;
    n = obj.n;
    fi = obj.fi;
    colorLine = obj.colorLine;
    colorFill = obj.colorFill;
    for(int i = 0; i < n*2; i++) lines[i] = obj.lines[i];
    creatStar();

}

void star::setRadius(int r0)
{
    r = r0;
    creatStar();
}

void star::setCenter(int x0, int y0)
{
    x = x0;
    y = y0;
    creatStar();
}

void star::setNumCorners(int n0)
{
    n = n0;
    creatStar();
}

void star::setColorLine(int color0)
{
    colorLine = color0;
}

void star::setColorFill(int color0)
{
    colorFill = color0;
}

void star::setCorners(int fi0)
{
    fi = fi0;
    creatStar();
}

void star::draw()
{
    for(int i = 0; i < n*2; i++)
    {
        lines[i].draw();
```

```
    }
}

void star::fill()
{
    setfillstyle(SOLID_FILL, colorFill);
    fillIn(x, y, colorFill, getpixel(x, y));
}

// circle  ------------------------------------------------------------

circl::circl(int x0, int y0, int r0)
{
    x = x0;
    y = y0;
    r = r0;
    colorLine = WHITE;
    colorFill = WHITE;
}

circl::circl(int x0, int y0, int r0, int colorLine0)
{
    x = x0;
    y = y0;
    r = r0;
    colorLine = colorLine0;
    colorFill = WHITE;
}

circl::circl(int x0, int y0, int r0, int colorLine0, int colorFill0)
{
    x = x0;
    y = y0;
    r = r0;
    colorLine = colorLine0;
    colorFill = colorFill0;
}

void circl::setCenter(int x0, int y0)
{
    x = x0;
    y = y0;
}

void circl::setRadius(int r0)
{
    r = r0;
}

void circl::setColorLine(int color0)
{
    colorLine = color0;
```

```
}

void circl::setColorFill(int color0)
{
   colorFill = color0;
}

void circl::draw()
{
   int thickness = 1;
   for(int i = 1; i < thickness+1; i++)
   {
      int x1 = 0;
      int y1 = r+i/2*pow(-1, i-1);
      int delta = 1-2*(r+i/2*pow(-1, i-1));
      int err = 0;
      while(y1 >= x1)
      {
         putpixel(x+x1, y+y1, colorLine);
         putpixel(x+x1, y-y1, colorLine);
         putpixel(x-x1, y+y1, colorLine);
         putpixel(x-x1, y-y1, colorLine);
         putpixel(x+y1, y+x1, colorLine);
         putpixel(x+y1, y-x1, colorLine);
         putpixel(x-y1, y+x1, colorLine);
         putpixel(x-y1, y-x1, colorLine);
         err = 2*(delta+y1)-1;
         if(delta < 0 && err <= 0)
         {
            delta += 2*++x1+1;
            continue;
         }
         if(delta > 0 && err > 0)
         {
            delta -= 2*--y1+1;
            continue;
         }
         delta += 2*(++x1- --y1);
      }
   }
}

void circl::fill()
{
   setfillstyle(SOLID_FILL, colorFill);
   fillIn(x, y, colorFill, getpixel(x, y));
}

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

void lineBez::pefog(int pef[100], int k, int n)
{
```

```
    if(n <= k) return;
    int a[100] = {0};
    for(int i = 0; i < k; i++) a[i] = pef[i];
    for(int i = 1; i < k+1; i++)
    {
       pef[i] = a[i-1]+a[i];
    }
    pefog(pef, k+1, n);
}

lineBez::lineBez()
{
    color = WHITE;
    width = 1;
}

lineBez::lineBez(int color0)
{
    color = color0;
    width = 1;
}

lineBez::lineBez(int color0, int width0)
{
    color = color0;
    width = width0;

}

void lineBez::add(int x0, int y0)
{
    data[n][0] = x0;
    data[n][1] = y0;
    n++;
}

void lineBez::setColor(int color0)
{
    color = color0;
}

void lineBez::setWidth(int width0)
{
    width = width0;
}

void lineBez::setStep(double step0)
{
    step = step0;
}

void lineBez::draw()
```

```
{
   if(n>1)
   {
      int pef[100] = {0};
      pef[0] = 1;
      pef[1] = 1;
      pefog(pef, 2, n);
      Point b = {data[0][0], data[0][1]};
      for(double t = step; t <= 1; t += step)
      {
         Point a = {0, 0};
         for(int i = 0; i < n; i++)
         {
            a.x += pef[i]*pow(1-t, n-i-1)*pow(t, i)*data[i][0];
            a.y += pef[i]*pow(1-t, n-i-1)*pow(t, i)*data[i][1];
         }
         lineB c = lineB(b.x, b.y, a.x, a.y, color, width);
         c.draw();
         b = a;
      }
      lineB c = lineB(b.x, b.y, data[n-1][0], data[n-1][1], color, wi
      c.draw();
   }
}

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

Point *lineCha::chaikin(Point *root)
{
   if(root == NULL) return NULL;
   if(root->next == NULL) return root;
   Point *r = root->next;
   Point *a, *b;
   a = new Point;
   b = new Point;
   a->x = root->x+(r->x-root->x)/4;
   a->y = root->y+(r->y-root->y)/4;

   b->x = root->x+(r->x-root->x)*3/4;
   b->y = root->y+(r->y-root->y)*3/4;
   b->next = chaikin(r);
   a->next = b;
   return a;
}

void lineCha::drawL(Point *root)
{
   if(root == NULL) return;
   if(root->next == NULL) return;
   Point *a = root->next;
   lineB l = lineB(root->x, root->y, a->x, a->y, color, width);
   l.draw();
```

```
    drawL(a);
}

void lineCha::del(Point *tree)
{
    if(tree == NULL) return;
    del(tree->next);
    delete tree;
}

lineCha::lineCha()
{
    color = WHITE;
    width = 1;
    root = NULL;
}

lineCha::lineCha(int color0)
{
    color = color0;
    width = 1;
    root = NULL;
}

lineCha::lineCha(int color0, int width0)
{
    color = color0;
    width = width0;
    root = NULL;
}

lineCha::~lineCha()
{
    del(root);
}

void lineCha::delRoot()
{
    del(root);
    root = NULL;
}

void lineCha::add(int x, int y)
{
    root = addPoint(x, y, root);
}

void lineCha::setColor(int color0)
{
    color = color0;
}
```

```
void lineCha::setWidth(int width0)
{
   width = width0;
}

void lineCha::draw()
{
   Point *r = chaikin(root);
   r->x = root->x;
   r->y = root->y;
   for(int i = 0; i < 5; i++)
   {
      r = chaikin(r);
      r->x = root->x;
      r->y = root->y;
   }
   setcolor(WHITE);
   drawL(r);
}

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

struct Fill
{
   int x, y;
   int color;
};

struct act
{
   int num = 0;
   lineB lin;
   star sta;
   circl circ;
   Fill fil;
   lineBez linBz;
   lineCha lineCh;
};

void fillIn(int x, int y, int colorFill, int colorNow)
{
   if(getpixel(x, y) == colorNow)
   {
      putpixel(x, y, colorFill);
      fillIn(x+1, y, colorFill, colorNow);
      fillIn(x-1, y, colorFill, colorNow);
      fillIn(x, y+1, colorFill, colorNow);
      fillIn(x, y-1, colorFill, colorNow);
   }
}

void drawAll(act acts[1000], int countDo)
```

```
{
   setfillstyle(SOLID_FILL, BLACK);
   bar(10, 70, 1190, 800);
   for(int i = 0; i < countDo; i++)
   {
      switch(acts[i].num)
      {
         case 1: acts[i].lin.draw(); break;
         case 2: acts[i].sta.draw(); break;
         case 3: acts[i].circ.draw(); break;
         case 4: fillIn(acts[i].fil.x, acts[i].fil.y, acts[i].fil.col
         case 5: acts[i].linBz.draw(); break;
         case 6: acts[i].lineCh.draw(); break;
      }
   }
}

void addLine(act acts[1000], int &countDo, int color)
{
   int p = 0;
   int x = mousex();
   int y = mousey();
   acts[countDo].num = 1;
   acts[countDo].lin = lineB(x, y, mousex(), mousey(), color);
   while(mousebuttons() and abs(mousex()-600) <= 590 and abs(mousey()
   {
      p = 1-p;
      setactivepage(p);

      drawAll(acts, countDo);

      acts[countDo].lin.setStart(x, y);
      acts[countDo].lin.setEnd(mousex(), mousey());

      acts[countDo].lin.draw();

      setvisualpage(p);
      delay(10);
   }
   countDo++;
}

void addStar(act acts[1000], int &countDo, int color)
{
   int p = 0;
   int x = mousex();
   int y = mousey();
   acts[countDo].num = 2;
   acts[countDo].sta = star(mousex(), mousey(), 0, 5, 0, color);
   while(mousebuttons() and abs(mousex()-600) <= 590 and abs(mousey()
   {
      p = 1-p;
```

```
        setactivepage(p);

        drawAll(acts, countDo);

        acts[countDo].sta.setCenter(x+(mousex()-x)/2, y+(mousey()-y)/2)
        acts[countDo].sta.setRadius(fmin(abs(mousex()-x)/2, abs(mousey(

        acts[countDo].sta.draw();

        setvisualpage(p);
        delay(10);
    }
    countDo++;
}

void addCircle(act acts[1000], int &countDo, int color)
{
    int p = 0;
    int x = mousex();
    int y = mousey();
    acts[countDo].num = 3;
    acts[countDo].circ = circl(mousex(), mousey(), 0, color);
    while(mousebuttons() and abs(mousex()-600) <= 590 and abs(mousey()
    {
        p = 1-p;
        setactivepage(p);

        drawAll(acts, countDo);

        acts[countDo].circ.setCenter(x+(mousex()-x)/2, y+(mousey()-y)/2
        acts[countDo].circ.setRadius(fmin(abs(mousex()-x)/2, abs(mousey

        acts[countDo].circ.draw();

        setvisualpage(p);
        delay(10);
    }

    countDo++;
}

void addLineBez(act acts[1000], int &countDo, int color)
{
    Point all[100];
    int brea = 0, p = 0;
    int count = 0;
    int numer = -1;
    while(1)
    {
        lineBez a = lineBez(color);
        p = 1-p;
        setactivepage(p);
```

```
drawAll(acts, countDo);

while(1-mousebuttons() or abs(mousex()-600) > 590 or abs(mousey
{
   numer = -1;
   if(kbhit()) if(getch() == 13)
   {
      brea = 1;
      break;
   }
}
if(brea)
{
   for(int i = 0; i < count; i++)
   {
      a.add(all[i].x, all[i].y);
   }
   acts[countDo].num = 5;
   acts[countDo].linBz = a;
   countDo++;
   delay(200);
   break;
}
for(int i = 0; i < count; i++)
{
   if(abs(mousex()-all[i].x) <= 7 and abs(mousey()-all[i].y) <=
   {
      numer = i;
      break;
   }
}
if(numer == -1)
{
   all[count] = {mousex(), mousey()};
   count++;
   while(mousebuttons());
}
else
{
   all[numer].x = mousex();
   all[numer].y = mousey();
}
for(int i = 0; i < count; i++)
{
   circl dot = circl(all[i].x, all[i].y, 5);
   dot.draw();
   a.add(all[i].x, all[i].y);
}
a.draw();

setvisualpage(p);
```

```
        delay(10);
    }
}

void save(int left = 0, int top = 0, int width = 0, int height = 0)
{
    IMAGE *output;

    if(width == 0)
    {
        width = getmaxx();
        height = getmaxy();
    }
    width++;
    height++;
    output = createimage(width-left, height-top);

    getimage(left, top, width-1, height-1, output);
    saveBMP("output.bmp", output);
    freeimage(output);
}
```

Файл control.h

```
#ifndef CONTROL_H
#define CONTROL_H

int pole(int, int, int, int, std::string, int, int&, int);
void lineColor(int, int, int[3][2], int, IMAGE*);
IMAGE *saveScr(int, int, int, int);
bool butSave(int, int);

#endif
```

Файл control.cpp

```
#include <string>
#include <graphics.h>

#include "control.h"

using namespace std;

int pole(int left, int top, int widht, int height, string Text, int d
{
    if(wrt)
    {
        if(kbhit())
        {
            int key = getch();
```

```
            if(key == 13) wrt = 0;
            if(key == 8) data = data/10;
            if(key > 47 && key < 58) data = data*10+key-48;
        }

        setfillstyle(SOLID_FILL, WHITE);
        bar(left-5, top-5, left+widht, top+height);
        setcolor(BLACK);
        if(t > 3)
        {
            char text[30];
            sprintf(text, "%s%d", Text.c_str(), data);
            outtextxy(left, top, text);
        }
        else
        {
            char text[30];
            sprintf(text, "%s%d|", Text.c_str(), data);
            outtextxy(left, top, text);
        }
    }
    else
    {
        setfillstyle(SOLID_FILL, WHITE);
        bar(left-5, top-5, left+widht, top+height);
        setcolor(BLACK);
        char text[30];
        sprintf(text, "%s%d", Text.c_str(), data);
        outtextxy(left, top, text);
    }
    return data;
}

void lineColor(int x, int y, int col[3][2], int num, IMAGE *image)
{
    putimage(x, y, image, COPY_PUT);

    setfillstyle(SOLID_FILL, WHITE);
    setcolor(BLACK);
    bar(col[num][0]+x-3, y, col[num][0]+x+3, y+19);
    rectangle(col[num][0]+x-3, y, col[num][0]+x+3, y+19);


    if(mousebuttons())
    {
        if(abs(mousex()-col[num][0]-x) < 4 and abs(mousey() - y-10) < 1
        {
            col[num][1] = 1;
        }
    }
    else
    {
```

```
        col[num][1] = 0;
    }

    if(col[num][1])
    {
        col[num][0] = mousex()-x;
        col[num][0] = (col[num][0]+abs(col[num][0]))/2;
        col[num][0] = col[num][0]>255 ? 255 : col[num][0];
    }
}

IMAGE *saveScr(int left, int top, int width, int height)
{
    IMAGE *output;

    output = createimage(width-left, height-top);

    getimage(left, top, width, height, output);
    return output;
}

bool butSave(int left, int top)
{
    setcolor(WHITE);
    bar(left, top, left+150, top+50);
    setcolor(BLACK);
    outtextxy(left+30, top+15, "Сохранит");
    return abs(mousex()-left-75) < 75 && abs(mousey()-top-25) < 25 &&
}
```

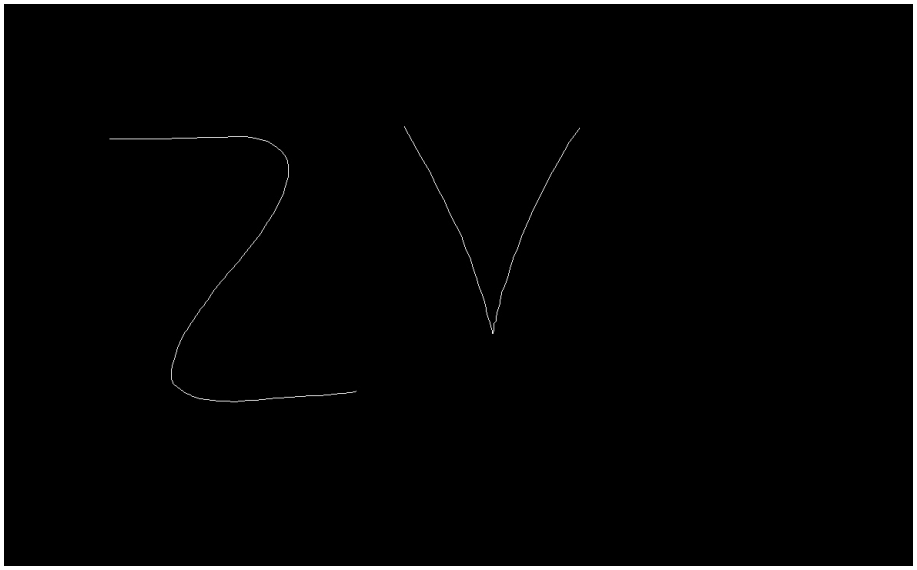

Рисунок 4.1 – Результат выполнения программы