Министерство науки и высшего образования РФ
ФГАОУ ВО ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИУ)
Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования

«Растровый графический редактор»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по дисциплине «Объектно-ориентированное программирование»

ЮУрГУ–02.03.01.2023.313.ПЗ КР

Руководитель,

_____Демидов А.К.

«___» _____ 2023г.

Автор работы:

Студентка группы: ЕТ – 211

_____ Савони М.В.

«___» _____ 2023г.

Работа защищена с оценкой

_____

«___» _____2023 г.

Челябинск – 2023

Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Прикладная математика и программирование»
Направление _____

УТВЕРЖДАЮ
Заведующий кафедрой ПМиП
_____А.А.Замышляева
_____ 2023 г.

ЗАДАНИЕ
на курсовую работу студента
_____*Савонин М.В.*_____
Группа __*ЕТ-211*____

1. Дисциплина _____*Объектно-ориентированное программирование*_____
2. Тема работы ____*Растровый графический редактор*_____
_____
3. Срок сдачи студентом законченной работы *28 января 2023 г.*
4. Перечень вопросов, подлежащих разработке
1) разработка иерархии и интерфейса классов;
2) реализация программы (библиотеки классов) на языке C++
3) оформление программной документации (описание программы (библиотеки классов), руководство пользователя, листинг кода) и отчета по курсовой работе
4) презентация проектных решений для защиты КР (иерархия и интерфейсы классов, особенности реализации)
5. Календарный план

Наименование разделов (этапов) курсовой работы	**Срок выполнения разделов (этапов) работы**	**Отметка о выполнении руководителя**
анализ предметной области	01.09.2022-10.10.2022	
разработка иерархии и интерфейса классов	20.09.2022-07.11.2022	
реализация основных классов, функций	01.10.2022-20.11.2022	
тестирование программы и/или классов, улучшение и исправление ошибок	20.10.2022-10.12.2022	
оформление программной документации и отчета по курсовой работе	30.10.2022-20.12.2022	
защита курсовой работы	20.12.2022-28.12.2022	

Руководитель работы _____ /_____
Студент _____ /_____
(подпись) (расшифровка)

АННОТАЦИЯ

Савонин М.В. Растровый графический редактор. – Челябинск: ЮУрГУ, ЕТ-211, 2023. – 37с., 4 ил., библиографический список – 3 наим., 1 прил.

В курсовой работе описывается разработка растрового графического редактора с помощью объектно-ориентированного подхода. Работа содержит результаты объектно-ориентированного анализа и проектирования, инструкции по установке и использованию программы.

В результате работы был разработан растровый графический редактор, код которого приводится в приложении.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы. Объектно-ориентированный подход является наиболее прогрессивной технологией разработки программных систем, позволяет разрабатывать более сложные системы.
Цель работы – разработать растровый графический редактор Paint
Задачи работы:
– изучить приемы объектно-ориентированного анализа;
– научиться разрабатывать программы в объектно-ориентированном стиле;
– овладеть технологиями объектно-ориентированного анализа и проектирования [1];
– изучить концепции объектно-ориентированного программирования; изучить особенности объектной модели языка программирования C++ [2]
– научиться самостоятельно и творчески использовать знания и полученные практические навыки;
– овладеть навыками самостоятельного получения новых знаний по теории и практике объектного подхода в программировании.

Объект работы – программа для рисования
Предмет работы – применение объектно-ориентированного подхода для разработки программы.
Результаты работы можно использовать в процессе последующего обучения в соответствии с учебным планом подготовки бакалавров по направлению «Прикладная математика и информатика»

1 ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать растровый графический редактор Paint со следующими возможностями:

– рисование точек (свободное рисование);

– рисование линий;

– рисование прямоугольников (заполненных и нет);

– выбор цветов рисования и заполнения из 16 или более;

– чтение и запись рисунка в стандартном (.BMP, .PCX, .JPG) или собственном формате.

При движении мыши высвечивать координаты.

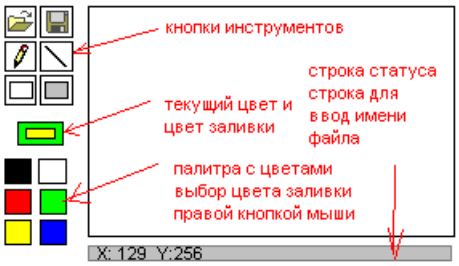

Рисунок 1 – Примерный интерфейс программы

Анализ предметной области выявляет следующие объекты:

– кнопки выбора режима рисования, имеющие квадратную форму и отличающиеся изображением;

– объект, отображающий текущие цвета рисования и сохранённые цвета;

– объект, отображающий текущие координаты мыши и вводимое имя файла при загрузке и сохранении;

– поле для рисования, реакция которого на движение мыши зависит от текущего выбранного цвета и инструмента.

Кнопки управления по поведению можно разделить на 3 группы, различающиеся по поведению:

– ползунки выбора цвета, изменяющие состояние объекта, отображающего сохранённые цвета

– кнопки выбора инструмента;

– 2 кнопки загрузки и сохранения рисунка.

Требования к программе:

– графический интерфейс;

– работа с мышью и клавиатурой.

2 ОПИСАНИЕ ПРОГРАММЫ

2.1 Для разработки программы были использованы:

– компилятор MinGW GNU C/C++ 7.2

– библиотека winBGIm [3]

2.2 Программа состоит из 2 модулей:

1 Модуль **main** (файл main.cpp) содержит следующие функции:

```
int pole(int left, int top, int widht, int height, string
&text, int t); // поле ввода текста с его записью в
переменную text
void tick(int left, int top, int widht, int height, string
text, int &fill); // галочка закрашивать ли фигуру
void lineColor(int x, int y, int col[3][2], int num, IMAGE
*image); // отображение и обработка одной из линий цвета
void addLine(vector<Figure*> &acts, int color); //
добавление линии
void addRect(vector<Figure*> &acts, int fill, int color,
int fillcolor); // добавления прямоугольника
void addCircle(vector<Figure*> &acts, int fill, int color,
int fillcolor); // добавление окружности
void addCurve(vector<Figure*> &acts, int color); //
добавление кривой
void addPicture(vector<Figure*> &acts); // добавление
изоброжения
void save(int left = 0, int top = 0, int width = 0, int
height = 0); // сохранение изображения
void chooseColor(int x, int y, int n, int colors[], int
col[3][2], int &choosed); // выбор цвета рисования
```

2 Модуль **draw** (интерфейсная часть в файле draw.h, реализация в файле draw.cpp) содержит следующие классы:

```
class Figure // Абстрактный класс Figure
{
   protected:
      int x0 ,y0; // основная точка фигуры
      int color, thickness; // цвет и толщина линии
   public:
      Figure(int x0, int y0, int color = WHITE, int thickness =
1): x0(x0), y0(y0), color(color), thickness(thickness) {} //
конструктор
      void setCoord(int x0, int y0){this->x0 = x0; this->y0 = y0;}
//сетер координат
      void setColor(int color){this->color = color;} //сетер цвета
лини
      void setThickness(int thickness){this->thickness =
thickness;} // сетер толщины линии
      virtual void draw(); // виртуальный метод отрисовки
```

```
};

class Line: public Figure // Класс Линии
{
   private:
      int x1, y1; // координаты конечной точки линии

   public:
      Line(int x0, int y0, int x1, int y1, int color = WHITE, int
thickness = 1): Figure(x0, y0, color, thickness), x1(x1), y1(y1){}
// конструктор
      Line(const Line &a): Figure(a.x0, a.y0, a.color,
a.thickness), x1(a.x1), y1(a.y1){} // конструктор копии
      void setCoordEnd(int x1, int y1){this->x1 = x1; this->y1 =
y1;} // сетер конечной точки линии
      void draw(); //отрисовка линии
};

class Rect: public Figure // класс Прямоугольник
{
   protected:
      int x1, y1; // вторая точка прямоугольника
   public:
      Rect(int x0, int y0, int x1, int y1, int color = WHITE, int
thickness = 1): Figure(x0, y0, color, thickness), x1(x1), y1(y1){}
// Конструктор
      Rect(const Rect &a): Figure(a.x0, a.y0, a.color,
a.thickness), x1(a.x1), y1(a.y1){}// конструктор копии
      void setCoordEnd(int x1, int y1){this->x1 = x1; this->y1 =
y1;} // сетер второй точки прямоугольника
      void draw(); // отрисовка прямоугольника
};

class FillRec: public Rect // Класс закрашенный прямоугольник
{
   private:
      int fillColor; // цвет закраски
   public:
      FillRec(int x0, int y0, int x1, int y1, int fillColor, int
color = WHITE, int thickness = 1): Rect(x0, y0, x1, y1, color,
thickness), fillColor(fillColor){} // конструктор
      FillRec(const FillRec &a): Rect(a.x0, a.y0, a.x1, a.y1,
a.color, a.thickness), fillColor(a.fillColor){} //конструктор
копии
      void setFillColor(int fillColor){this->fillColor =
fillColor;} // сетер цвет закраски
      void draw(); // отрисовка
};

class Circl: public Figure // Класс окружность
```

```
{
   protected:
      int r; // радиус окружности

   public:
      Circl(int x, int y, int r, int color = WHITE, int thickness
= 1): Figure(x, y, color, thickness), r(r){} // конструктор
      Circl(const Circl &a): Figure(a.x0, a.y0, a.color,
a.thickness), r(a.r){} // конструктор копии
      void setRadius(int r){this->r = r;} // сетер радиуса
      void draw(); // отрисовка окружности
};

class FillCircl: public Circl // Класс закрашенная окружность
{
   private:
      int fillColor; // цвет закраски
   public:
      FillCircl(int x, int y, int r, int fillColor, int color =
WHITE, int thickness = 1): Circl(x, y, r, color, thickness),
fillColor(fillColor){} // конструктор
      FillCircl(const FillCircl &a): Circl(a.x0, a.y0, a.r,
a.color, a.thickness), fillColor(a.fillColor){} // конструктор
копии
      void setFillColor(int fillColor){this->fillColor =
fillColor;} //  сетер цвета закраски
      void draw(); // отрисовка
};

class Curve: public Figure // Класс кривой Безье
{
   private:
      vector<vector<int>> data; // массив точек
      double step = 0.02; // шаг искривления
      void pefog(int[100], int, int); // вычисление
последовательности степеней

   public:
      Curve(): Figure(0, 0, WHITE, 1) {} // конструктор по
умолчанию
      Curve(int data[][2], int n, int color = WHITE, int thickness
= 1) : Figure(0, 0, color, thickness) // конструктор по массиву
точек
      {
          for (int i = 0; i < n; i++)
          {
              this->data.push_back({0, 0});
              this->data[i][0] = data[i][0];
              this->data[i][1] = data[i][1];
          }
```

```
        }
        Curve(vector<vector<int>> data, int color = WHITE, int
thickness = 1) : Figure(0, 0, color, thickness), data(data) {} //
конструктор по вектору точек
        Curve(const Curve &a): Figure(a.x0, a.y0, a.color,
a.thickness), data(a.data), step(a.step){} // конструктор копии
        void add(int x, int y){data.push_back({x, y});} //
добавление точки
        void setStep(double a){step = a;} // сетер шага
        void setPoints(vector<vector<int>>); // сутер точек
        vector<vector<int>> getPoints(){return data;} // гетер точек
        void draw(); // отрисовка
};

class Picture: public Figure // Класс картинки
{
   private:
      IMAGE *file; //загруженная картинка
   public:
      Picture(int x, int y, string fileName): Figure(x, y),
file(loadBMP(fileName.c_str())){} // конструктор
      void draw() {putimage(x0, y0, file, COPY_PUT);} // отрисовка
};
```

2.3 Иерархия классов показана рисунке 2.

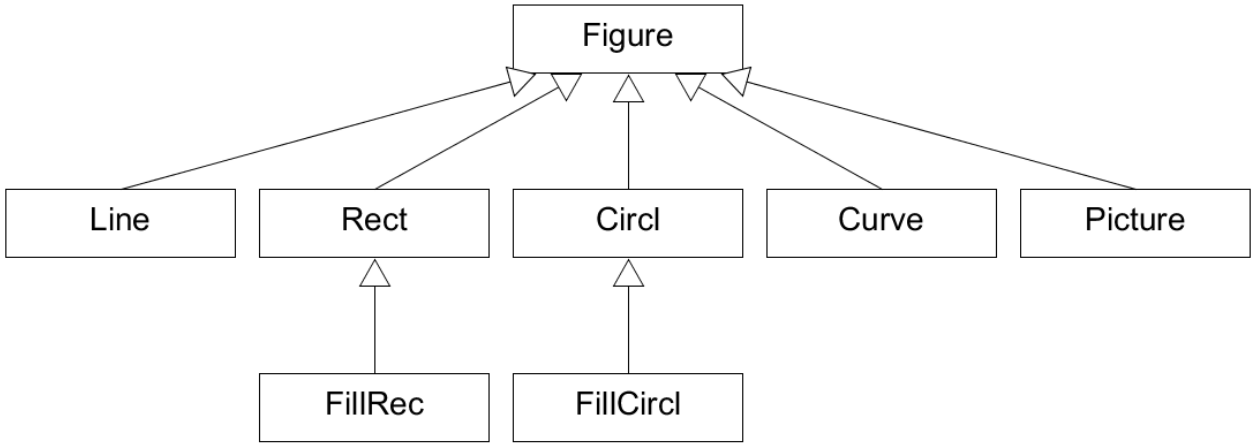

Рисунок 2 — Иерархия классов

2.4 Пояснения по алгоритму и особенностям реализации

Для рисования можно выбрать любой цвет который возможен в области RGB. Чтобы добавить цвет в палитру нужно нажать на любую ячейку палитры и выставить нужное значение на полосках RGB, после чего цвет будет доступен для рисования.

Так же кнопки режимов рисования отрисовываются своими же методами классов, что даёт возможность маштабирования интерфейса.

2.5 Используемые внешние файлы

Программа использует файлы в формате BMP:

– Red.bmp – картинка для рисования полоски выбора насыщенности красного цвета

– Green.bmp – картинка для рисования полоски выбора насыщенности зелёного цвета

– Blue.bmp – картинка для рисования полоски выбора насыщенности синего цвета

Все картинки имеют размер 256x20 и 24-битный цвет и должны находиться в текущей папке.

Также программа может в процессе работы сохранять, а потом загружать файлы в BMP формате, размером 1181x731, цветовое разрешение которых зависит от текущих настроек Windows.

3 ИНСТРУКЦИЯ ПО УСТАНОВКЕ И ТРЕБОВАНИЯ К СИСТЕМЕ

3.1 Требования к компьютеру:
Процессор: Intel Pentium 4 1ГГц или выше
Память: 512+ МБ
Видеокарта: разрешение 1200x900 или выше, 24-битный цвет
Свободное место на диске: 10 Мб
Операционная система: Windows XP или выше
Дополнительное устройство: мышь

3.2 Для установки скопировать следующие файлы на диск:
– paint.exe;
– Red.bmp, Green.bmp, Blue.bmp.

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске программы откроется окно редактора (см. рисунок 3).

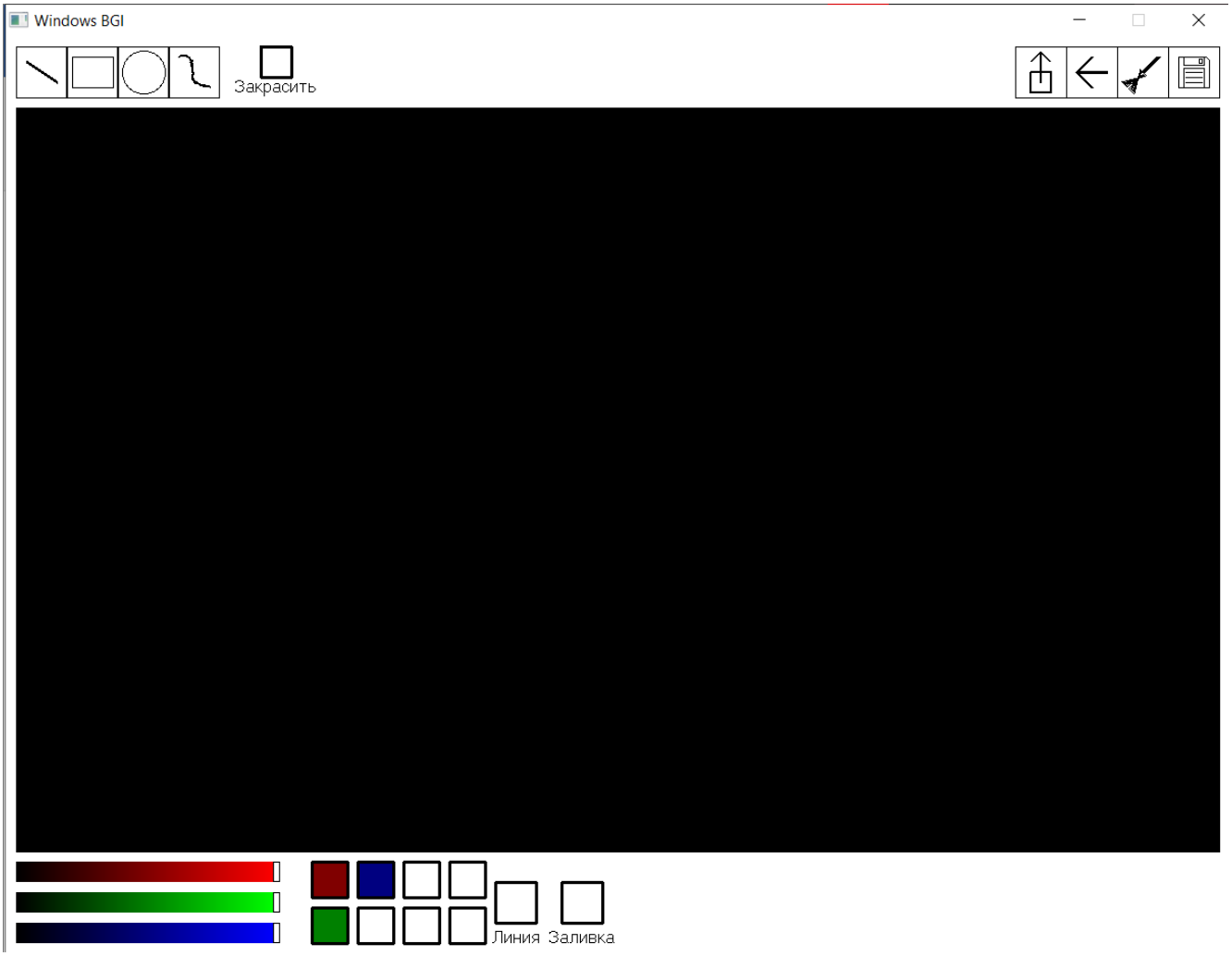

Рисунок 3 – Окно редактора

Чёрное поле в центре – поле для рисования. При перемещении мыши по полю в правом нижнем углу в строке состояния высвечиваются текущие координаты курсора в системе координат: ось X направлена из верхнего левого угла белого поля вправо, ось Y – вниз. При выходе за границы поля для рисования координаты исчезают.

Снизу пользователь видит ползунки RGB, палитру из 8 цветов и цвета рисования. Нажатием правой кнопки мыши выбирается ячейку которую хотим заполнить нужным нам цветом с помозью ползунков RGB. Эти цвета можно выбрать нажав на квадратик “Линия” или “Заливка” и затем нажав на нужный цвет из палитры. При запуске программы цвет линий и заливки белый.

Рассмотрим по порядку кнопки инструментов слева направо (см. рисунок 4).

Рисунок 4 — Палитра инструментов

Для рисования линий необходимо щелкнуть по первой кнопке в левой верхней части экрана. При нажатии левой кнопки мыши в поля для рисования начинается рисование линии выбранным цветом. При отпускании — линии фиксируется и рисуется выбранным цветом.

Для рисования прямоугольника необходимо щелкнуть по второй кнопке в левой верхней части экрана. Для его закраски надо нажать на квадратик “Закрасить”. Аналогично, рисование прямоугольников начинается при нажатии левой кнопки мыши в поле для рисования. При отпускании — рисуется прямоугольник.

Для рисования окружности необходимо щелкнуть по третьей кнопке в левой верхней части экрана. Для её закраски надо нажать на квадратик “Закрасить”. Аналогично, рисование окружности начинается при нажатии левой кнопки мыши в поле для рисования. При отпускании — рисуется окружность.

Для рисования кривой линии необходимо щелкнуть по четвёртой кнопке в левой верхней части экрана. Для добавления точек изгиба нужно нажать на любую точку поля для рисования (первые точки выставляются аналогично). Для изменения положения этих точек нужно нажать и удерживать левую кнопку мыши и перенести курсор на нужное место, после чего отпустить кнопку. Для завершения рисования нужно нажать Enter.

Первая кнопка в верхней правой части экрана — загрузка изображения. При нажатии на эту кнопку левее неё появится поле для ввода. С помощью клавиш Backspace и алфавитно-цифровых клавиш можно ввести и изменить путь и имя файла. При нажатии Enter файл будет загружен. Если файл не существует, загрузка не производится.

Для отмены действия можно нажать на вторую кнопку в верхней правой части, а для отчистки нарисованных фигур можно нажать на третью кнопку.

Четвёртая кнопка в верхней правой части — сохранение изображения. Аналогично, пользователь должен ввести путь и имя файла для сохранения и нажать Enter.

Для завершения работы с программой необходимо щелкнуть по кнопке с крестиком в верхнем правом углу.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были выявлены объекты предметной области и определена система классов для них, разработан интерфейс программы. После объектно-ориентированного проектирования классы были реализованы на языке C++. Разработанный код был проверен на контрольных тестах и в код были внесены необходимые исправления. Для программы была разработана документация, описывающая её установку и использование. Таким образом, цель работы была достигнута, задачи – решены.

Результаты работы можно использовать в процессе последующего обучения в форме навыков практического применения объектно-ориентированного подхода для разработки сложных программных систем, понимания порядка этапов разработки программного обеспечения и достигаемых на каждом этапе результатов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Гамма, Э. Приемы объектно ориентированного проектирования. Паттерны проектирования. [Электронный ресурс] / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. — Электрон. дан. — М. : ДМК Пресс, 2007. — 368 с. — Режим доступа: http://e.lanbook.com/book/1220

2 Липман, С. Язык программирования C++. Полное руководство. [Электронный ресурс] / С. Липман, Ж. Лажойе. — Электрон. дан. — М. : ДМК Пресс, 2006. — 1105 с. — Режим доступа: http://e.lanbook.com/book/1216

3 Графическая библиотека WinBGIm. – URL: https://ipc.susu.ru/20786.html (дата обращения: 01.04.2019).

ПРИЛОЖЕНИЕ А

А.1 Файл main.cpp

```
#define _USE_MATH_DEFINES

#include <iostream>
#include <random>
#include <time.h>
#include <math.h>
#include <vector>

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

using namespace std;

int pole(int left, int top, int widht, int height, string &text,
int t) // поле ввода текста с его записью в переменную text
{
   if(kbhit())
   {
      int key = getch();
      switch(key)
      {
         case 8: text.erase(text.size()-1, 1); break;
         case 13: return 1;
         default:
            text += key;
            break;
      }
   }

   setfillstyle(SOLID_FILL, WHITE);
   bar(left, top, left+widht, top+height);
   setcolor(BLACK);
   rectangle(left, top, left+widht, top+height);
   settextjustify(LEFT_TEXT, CENTER_TEXT);
   settextstyle(TRIPLEX_FONT, HORIZ_DIR, USER_CHAR_SIZE);
   setusercharsize(1, 2, 1, 1);
   if(t > 3)
   {
      char str[30];
      sprintf(str, "%s", text.c_str());
      outtextxy(left+5, top+height/2+5, str);
   }
   else
   {
      char str[30];
```

```
      sprintf(str, "%s|", text.c_str());
      outtextxy(left+5, top+height/2+5, str);
   }
   settextstyle(3, HORIZ_DIR, 1);
   return 0;
}

void tick(int left, int top, int widht, int height, string text,
int &fill) // галочка закрашивать ли фигуру
{
   setlinestyle(SOLID_LINE, 0, 3);
   setcolor(BLACK);
   settextjustify(CENTER_TEXT, CENTER_TEXT);
   outtextxy(left+widht/2, top+height+15, text.c_str());
   rectangle(left, top, left+widht, top+height);
   if(abs(mousex()-left-widht/2) <= widht/2 and abs(mousey()-top-
height/2) <= height/2 and mousebuttons())
   {
      fill = 1-fill;
      while(mousebuttons());
   }
   if(fill)
   {
      line(left, top, left+widht, top+height);
      line(left, top+height, left+widht, top);
   }
   setlinestyle(SOLID_LINE, 0, 1);
}

void lineColor(int x, int y, int col[3][2], int num, IMAGE *image)
// отображение и обработка одной из линий цвета
{
   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)
< 10 and col[(num+1)%3][1] == 0 and col[(num+2)%3][1] == 0)
      {
         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];
   }
}

void addLine(vector<Figure*> &acts, int color) // добавление линии
{
   int p = 0;
   int x = mousex();
   int y = mousey();
   Line a(x, y, mousex(), mousey(), color);
   while(mousebuttons() and abs(mousex()-600) <= 590 and
abs(mousey()-435) <= 365)
   {
      p = 1-p;
      setactivepage(p);

      drawAll(acts);

      a.setCoord(x, y);
      a.setCoordEnd(mousex(), mousey());

      a.draw();

      setvisualpage(p);
      delay(10);
   }
   acts.push_back(new Line(x, y, mousex(), mousey(), color));
}

void addRect(vector<Figure*> &acts, int fill, int color, int
fillcolor) // добавления прямоугольника
{
   int p = 0;
   int x = mousex();
   int y = mousey();
   Rect *a;
   if(fill) a = new FillRec(x, y, mousex(), mousey(), fillcolor,
color);
   else a = new Rect(x, y, mousex(), mousey(), color);

   while(mousebuttons() and abs(mousex()-600) <= 590 and
abs(mousey()-435) <= 365)
```

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

      drawAll(acts);

      a->setCoord(x, y);
      a->setCoordEnd(mousex(), mousey());
      a->draw();

      setvisualpage(p);
      delay(10);
   }
   if(fill)
      acts.push_back(new FillRec(x, y, mousex(), mousey(),
fillcolor, color));
   else
      acts.push_back(new Rect(x, y, mousex(), mousey(), color));
}

void addCircle(vector<Figure*> &acts, int fill, int color, int
fillcolor) // добавление окружности
{
   int p = 0;
   int x = mousex();
   int y = mousey();
   int r = 0;
   Circl *a;
   if(fill) a = new FillCircl(mousex(), mousey(), r, fillcolor,
color);
   else a = new Circl(mousex(), mousey(), r, color);
   while(mousebuttons() and abs(mousex()-600) <= 590 and
abs(mousey()-435) <= 365)
   {
      p = 1-p;
      setactivepage(p);

      drawAll(acts);

      r = fmin(abs(mousex()-x)/2, abs(mousey()-y)/2);
      a->setCoord(x+(mousex()-x)/2, y+(mousey()-y)/2);
      a->setRadius(r);

      a->draw();

      setvisualpage(p);
      delay(10);
   }
```

```
   acts.push_back(a);
}

void addCurve(vector<Figure*> &acts, int color) // добавление криво
й
{
   vector<vector<int>> all;
   int brea = 0, p = 0;
   int numer = -1;
   while (1)
   {
      p = 1 - p;
      setactivepage(p);

      drawAll(acts);

      while (1 - mousebuttons() or abs(mousex() - 600) > 590 or
abs(mousey() - 435) > 365)
      {
         numer = -1;
         if (kbhit()) if (getch() == 13)
         {
             brea = 1;
             break;
         }
      }
      if (brea)
      {
         acts.push_back(new Curve(all, color, 1));
         delay(200);
         break;
      }
      for (int i = 0; i < all.size(); i++)
      {
         if (abs(mousex() - all[i][0]) <= 7 and abs(mousey() -
all[i][1]) <= 7)
         {
             numer = i;
             break;
         }
      }
      if (numer == -1)
      {
         all.push_back({mousex(), mousey()});
         while (mousebuttons());
      }
      else
      {
         all[numer][0] = mousex();
```

```
            all[numer][1] = mousey();
        }
        for (int i = 0; i < all.size(); i++)
        {
            Circl dot = Circl(all[i][0], all[i][1], 5);
            dot.draw();
        }
        Curve a(all, color, 1);
        a.draw();

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

void addPicture(vector<Figure*> &acts) // добавление изоброжения
{
    string text;
    int t = 0, p = 0;
    while(1)
    {
        p = 1 - p;
        setactivepage(p);
        setcolor(BLACK);
        if(pole(800, 15, 150, 40, text, t))
        {
            acts.push_back(new Picture(10, 70, text));
            break;
        }
        t++;
        t = t%8;
        setvisualpage(p);
        delay(50);
    }
}

void save(int left = 0, int top = 0, int width = 0, int height =
0) // сохранение изображения
{
    string text;
    int t = 0, p = 0;
    while(1)
    {
        p = 1 - p;
        setactivepage(p);
        setcolor(BLACK);
        if(pole(800, 15, 150, 40, text, t))
        {
            break;
```

```
      }
      t++;
      t = t%8;
      setvisualpage(p);
      delay(50);
   }
   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(text.c_str(), output);
   freeimage(output);
}

void chooseColor(int x, int y, int n, int colors[], int col[3][2],
int &choosed) // выбор цвета рисования
{
   IMAGE *imager;
   imager = loadBMP("Red.bmp");
   IMAGE *imageg;
   imageg = loadBMP("Green.bmp");
   IMAGE *imageb;
   imageb = loadBMP("Blue.bmp");

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

   freeimage(imager);
   freeimage(imageg);
   freeimage(imageb);

   for(int i = 0; i < n; i++)
   {
      FillRec a(x+i/2*45, y+i%2*45, x+35+i/2*45, y+35+i%2*45,
colors[i+2], BLACK, 3);
      a.draw();
   }
   for(int i = 0; i < 2; i++)
   {
```

```
      FillRec a(x+(n/2+i)*45+i*20, y+20, x+40+(n/2+i)*45+i*20,
y+60, colors[i], BLACK, 3);
      a.draw();
   }

   settextjustify(CENTER_TEXT, CENTER_TEXT);
   outtextxy(x+n/2*45+20, y+80, "Линия");
   outtextxy(x+n/2*45+85, y+80, "Заливка");

   if(mousebuttons())
   {
      if(choosed < 2 and choosed != -1)
      {
         int a = -1;
         for(int i = 0; i < n; i++)
         {
            if(abs(mousex()-(x+i/2*45+17)) < 18 and abs(mousey()-
(y+i%2*45+17)) < 18)
            {
               a = i+2;
               while(mousebuttons());
            }
         }
         if(a != -1)
         {
            colors[choosed] = colors[a];
            choosed = -1;
         }
      }
      else
      {
         if(choosed != -1) colors[choosed] = COLOR(col[0][0],
col[1][0], col[2][0]);
         for(int i = 0; i < n; i++)
         {
            if(abs(mousex()-(x+i/2*45+17)) < 18 and abs(mousey()-
(y+i%2*45+17)) < 18)
            {
               choosed = i+2;
               while(mousebuttons());
            }
         }
         if(abs(mousex()-(x+n/2*45+20)) < 20 and abs(mousey()-
(y+40)) < 20)
         {
            choosed = 0;
            while(mousebuttons());
         }
```

```
         if(abs(mousex()-(x+n/2*45+85)) < 20 and abs(mousey()-
(y+40)) < 20)
         {
            choosed = 1;
            while(mousebuttons());
         }
      }
   }
}

int main() // основная функция
{
   setlocale(LC_ALL,"Russian");
   setlocale(LC_ALL,"rus");
   srand(time(0));
   initwindow(1200, 900);

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

   srand(time(0));

   int col[3][2] = { {255, 0}, {255, 0}, {255, 0} }; // цвет выбра
нный на полосках RGB и флаг держим ли мы ползунок мышкой
   int colors[10] = { WHITE, WHITE, RED, GREEN, BLUE, WHITE,
WHITE, WHITE, WHITE, WHITE }; // массив цветов рисования и сохранённ
ых цветов
   int p = 0, t = 0; // отоброжаемая страница и таймер для отоброжен
ия | при вводе названия файла
   int mode = 0, fill = 0, choosed = -1; // режим рисования, закраш
иваем ли мы фигуру, выбранный квадратик цветов
   vector<Figure*> acts; // все отоброжаемые фигуры

   const int numButton = 4; // количество кнопок для выбора режима р
исования
   Figure* button[numButton][2]; // массив иконок кнопок

   for(int i = 0; i < numButton; i++)
      button[i][0] = new Rect(10+i*50, 10, 59+i*50, 60, BLACK);
   button[0][1] = new Line(20, 25, 50, 45, BLACK, 3);
   button[1][1] = new Rect(65, 20, 105, 50, BLACK);
   button[2][1] = new Circl(135, 35, 21, BLACK);
```

```
   int pointsbut[4][2] = { {170, 20}, {200, 20}, {170, 50}, {200,
50} };
   button[3][1] = new Curve(pointsbut, 4, BLACK, 2);

   Line butsave[18] = { Line(1150, 20, 1177, 20, BLACK, 1), // ико
нка кнопка сохранения
                                    Line(1177, 20, 1180, 23, BLACK,
1),
                                    Line(1180, 23, 1180, 50, BLACK,
1),
                                    Line(1180, 50, 1150, 50, BLACK,
1),
                                    Line(1150, 50, 1150, 20, BLACK,
1),
                                    Line(1155, 20, 1155, 28, BLACK,
1),
                                    Line(1155, 28, 1175, 28, BLACK,
1),
                                    Line(1175, 28, 1175, 20, BLACK,
1),
                                    Line(1170, 22, 1173, 22, BLACK,
1),
                                    Line(1173, 22, 1173, 26, BLACK,
1),
                                    Line(1173, 26, 1170, 26, BLACK,
1),
                                    Line(1170, 26, 1170, 22, BLACK,
1),
                                    Line(1155, 50, 1155, 31, BLACK,
1),
                                    Line(1155, 31, 1175, 31, BLACK,
1),
                                    Line(1175, 31, 1175, 50, BLACK,
1),
                                    Line(1157, 35, 1173, 35, BLACK,
1),
                                    Line(1157, 40, 1173, 40, BLACK,
1),
                                    Line(1157, 45, 1173, 45, BLACK,
1) };

   Line butclear[13] = { Line(1130, 20, 1115, 35, BLACK, 5), // ик
онка кнопки отчистки
                                   Line(1116, 36, 1099, 49, BLACK,
1),
                                   Line(1114, 34, 1101, 51, BLACK,
1),
```

```
                                    Line(1116, 36, 1098, 48, BLACK,
1),
                                    Line(1114, 34, 1102, 52, BLACK,
1),
                                    Line(1117, 37, 1097, 47, BLACK,
1),
                                    Line(1113, 33, 1103, 53, BLACK,
1),
                                    Line(1117, 37, 1096, 46, BLACK,
1),
                                    Line(1113, 33, 1104, 54, BLACK,
1),
                                    Line(1118, 38, 1095, 45, BLACK,
1),
                                    Line(1112, 32, 1105, 55, BLACK,
1),
                                    Line(1118, 38, 1094, 44, BLACK,
1),
                                    Line(1112, 32, 1106, 56, BLACK,
1) };

   Line butback[3] = { Line(1080, 35, 1050, 35, BLACK, 3), // икон
ки кнопки отмены действия
                                    Line(1050, 35, 1065, 20, BLACK,
3),
                                    Line(1050, 35, 1065, 50, BLACK,
3) };

   Figure *butload[4] = { new Rect(1005, 35, 1025, 55, BLACK, 2),
// иконка кнопки загрузки изоброжения
                                       new Line(1015, 45, 1015, 15,
BLACK, 2),
                                       new Line(1015, 15, 1006, 24,
BLACK, 2),
                                       new Line(1015, 15, 1024, 24,
BLACK, 2)};
   settextstyle(3, HORIZ_DIR, 1);
   while(1)
   {
      p = 1-p;
      setactivepage(p);
      clearviewport();
      for(int i = 0; i < numButton; i++)
      {
         for(int k = 0; k < 2; k ++)
         {
            button[i][k]->draw();
         }
```

```
    }

    tick(250, 10, 30, 30, "Закрасить", fill);
    chooseColor(300, 810, 8, colors, col, choosed);
    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();
    }

    setcolor(BLACK);
    rectangle(990, 10, 1040, 60);
    for (int i = 0; i < 4; i++)
    {
        butload[i]->draw();
    }

    drawAll(acts);
    if(abs(mousey()-35) <= 25 and mousebuttons())
    {
       for(int i = 0; i < numButton; i++)
          if(abs(mousex()-35-i*50) <= 25) mode = i+1;
    }
    if(abs(mousex()-630) <= 560 and abs(mousey()-450) <= 350 and
mousebuttons() and mode != 0)
    {
       switch(mode)
       {
          case 1:
          {
             addLine(acts, colors[0]);
             break;
          }
          case 2:
          {
```

```
                addRect(acts, fill, colors[0], colors[1]);
                break;
            }
            case 3:
            {
                addCircle(acts, fill, colors[0], colors[1]);
                break;
            }
            case 4:
            {
                addCurve(acts, colors[0]);
                break;
            }
        }
    }
    setvisualpage(p);
    if (abs(mousey() - 35) <= 25 and mousebuttons())
    {
        if (abs(mousex() - 1165) <= 25)
        {
            save(10, 70, 1190, 800);
        }
        if (abs(mousex() - 1115) <= 25)
        {
            acts.clear();
        }
        if (abs(mousex() - 1065) <= 25 and acts.size() > 0)
        {
            acts.pop_back();
            while (mousebuttons());
        }
        if (abs(mousex() - 1015) <= 25)
        {
           addPicture(acts);
        }
    }
    delay(20);
  }
  return 0;
}
```

A.2 Файл draw.h

```
#ifndef DRAW_H
#define DRAW_H

#include <vector>

using namespace std;

struct Point
```

```
{
   int x, y;
};

class Figure // Абстрактный класс Figure
{
   protected:
      int x0 ,y0; // основная точка фигуры
      int color, thickness; // цвет и толщина линии
   public:
      Figure(int x0, int y0, int color = WHITE, int thickness =
1): x0(x0), y0(y0), color(color), thickness(thickness) {} // конст
руктор
      void setCoord(int x0, int y0){this->x0 = x0; this->y0 = y0;}
//сетер координат
      void setColor(int color){this->color = color;} //сетер цвета
лини
      void setThickness(int thickness){this->thickness =
thickness;} // сетер толщины линии
      virtual void draw(); // виртуальный метод отрисовки
};

class Line: public Figure // Класс Линии
{
   private:
      int x1, y1; // координаты конечной точки линии

   public:
      Line(int x0, int y0, int x1, int y1, int color = WHITE, int
thickness = 1): Figure(x0, y0, color, thickness), x1(x1), y1(y1){}
// конструктор
      Line(const Line &a): Figure(a.x0, a.y0, a.color,
a.thickness), x1(a.x1), y1(a.y1){} // конструктор копии
      void setCoordEnd(int x1, int y1){this->x1 = x1; this->y1 =
y1;} // сетер конечной точки линии
      void draw(); //отрисовка линии
};

class Rect: public Figure // класс Прямоугольник
{
   protected:
```

```
      int x1, y1; // вторая точка прямоугольника
   public:
      Rect(int x0, int y0, int x1, int y1, int color = WHITE, int
thickness = 1): Figure(x0, y0, color, thickness), x1(x1), y1(y1){}
// Конструктор
      Rect(const Rect &a): Figure(a.x0, a.y0, a.color,
a.thickness), x1(a.x1), y1(a.y1){}// конструктор копии
      void setCoordEnd(int x1, int y1){this->x1 = x1; this->y1 =
y1;} // сетер второй точки прямоугольника
      void draw(); // отрисовка прямоугольника
};

class FillRec: public Rect // Класс закрашенный прямоугольник
{
   private:
      int fillColor; // цвет закраски
   public:
      FillRec(int x0, int y0, int x1, int y1, int fillColor, int
color = WHITE, int thickness = 1): Rect(x0, y0, x1, y1, color,
thickness), fillColor(fillColor){} // конструктор
      FillRec(const FillRec &a): Rect(a.x0, a.y0, a.x1, a.y1,
a.color, a.thickness), fillColor(a.fillColor){} //конструктор копии
      void setFillColor(int fillColor){this->fillColor =
fillColor;} // сетер цвет закраски
      void draw(); // отрисовка
};

class Circl: public Figure // Класс окружность
{
   protected:
      int r; // радиус окружности

   public:
      Circl(int x, int y, int r, int color = WHITE, int thickness
= 1): Figure(x, y, color, thickness), r(r){} // конструктор
      Circl(const Circl &a): Figure(a.x0, a.y0, a.color,
a.thickness), r(a.r){} // конструктор копии
      void setRadius(int r){this->r = r;} // сетер радиуса
      void draw(); // отрисовка окружности
};
```

```
class FillCircl: public Circl // Класс закрашенная окружность
{
   private:
      int fillColor; // цвет закраски
   public:
      FillCircl(int x, int y, int r, int fillColor, int color =
WHITE, int thickness = 1): Circl(x, y, r, color, thickness),
fillColor(fillColor){} // конструктор
      FillCircl(const FillCircl &a): Circl(a.x0, a.y0, a.r,
a.color, a.thickness), fillColor(a.fillColor){} // конструктор копи
и
      void setFillColor(int fillColor){this->fillColor =
fillColor;} //  сетер цвета закраски
      void draw(); // отрисовка
};

class Curve: public Figure // Класс кривой Безье
{
   private:
      vector<vector<int>> data; // массив точек
      double step = 0.02; // шаг искривления
      void pefog(int[100], int, int); // вычисление последовательност
и степеней

   public:
      Curve(): Figure(0, 0, WHITE, 1) {} // конструктор по умолчани
ю
      Curve(int data[][2], int n, int color = WHITE, int thickness
= 1) : Figure(0, 0, color, thickness) // конструктор по массиву точек
      {
          for (int i = 0; i < n; i++)
          {
              this->data.push_back({0, 0});
              this->data[i][0] = data[i][0];
              this->data[i][1] = data[i][1];
          }
      }
      Curve(vector<vector<int>> data, int color = WHITE, int
thickness = 1) : Figure(0, 0, color, thickness), data(data) {} //
конструктор по вектору точек
```

```
      Curve(const Curve &a): Figure(a.x0, a.y0, a.color,
a.thickness), data(a.data), step(a.step){} // конструктор копии
      void add(int x, int y){data.push_back({x, y});} // добавление
точки
      void setStep(double a){step = a;} // сетер шага
      void setPoints(vector<vector<int>>); // сутер точек
      vector<vector<int>> getPoints(){return data;} // гетер точек
      void draw(); // отрисовка
};

class Picture: public Figure // Класс картинки
{
   private:
      IMAGE *file; //загруженная картинка
   public:
      Picture(int x, int y, string fileName): Figure(x, y),
file(loadBMP(fileName.c_str())){} // конструктор
      void draw() {if(file) putimage(x0, y0, file, COPY_PUT);} //
отрисовка
};

void outcoord(); // вывод координат указателя мыши
void drawAll(vector<Figure*> acts); // отрисовка всех фигур

#endif
```

A.3 Файл draw.cpp

```
#define _USE_MATH_DEFINES

#include <iostream>
#include <random>
#include <time.h>
#include <math.h>
#include <vector>

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

using namespace std;

void Line::draw()
{
   int dx = abs(x1-x0);
```

```
int dy = abs(y1-y0);
int err = 0;
if(dy < dx)
{
   if(x0 > x1)
   {
      int a = x0;
      x0 = x1;
      x1 = a;
      a = y0;
      y0 = y1;
      y1 = a;
   }
   int d = dy == 0 ? 0 : (y1-y0)/dy;
   int y = y0;
   for(int x = x0; x <= x1; x++)
   {
      for(int i = 0; i < thickness; i++)
      {
         int yd;
         yd = (i+1)/2*pow(-1, i);
         putpixel(x, y+yd, color);
      }
      err = err+dy;
      if(err >= dx)
      {
         y += d;
         err -= dx;
      }
   }
}
else
{
   if(y0 > y1)
   {
      int a = x0;
      x0 = x1;
      x1 = a;
      a = y0;
      y0 = y1;
      y1 = a;
   }
   int d = dx == 0 ? 0 : (x1-x0)/dx;
   int x = x0;
   for(int y = y0; y <= y1; y++)
   {
      for(int i = 0; i < thickness; i++)
      {
         int xd;
         xd = (i+1)/2*pow(-1, i);
```

```
            putpixel(x+xd, y, color);
         }
         err = err+dx;
         if(err >= dy)
         {
            x += d;
            err -= dy;
         }
      }
   }
}
//-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------

void Rect::draw()
{
   Line a(x0, y0, x1, y0, color, thickness);
   a.draw();
   a.setCoord(x1, y0);
   a.setCoordEnd(x1, y1);
   a.draw();
   a.setCoord(x1, y1);
   a.setCoordEnd(x0, y1);
   a.draw();
   a.setCoord(x0, y0);
   a.setCoordEnd(x0, y1);
   a.draw();
}
//-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------

void FillRec::draw()
{
   Rect::draw();
   setfillstyle(SOLID_FILL, fillColor);
   floodfill((x1-x0)/2+x0, (y1-y0)/2+y0, color);
}

//-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------
void Circl::draw()
{
   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(x0+x1, y0+y1, color);
         putpixel(x0+x1, y0-y1, color);
         putpixel(x0-x1, y0+y1, color);
         putpixel(x0-x1, y0-y1, color);
         putpixel(x0+y1, y0+x1, color);
         putpixel(x0+y1, y0-x1, color);
         putpixel(x0-y1, y0+x1, color);
         putpixel(x0-y1, y0-x1, color);
         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 FillCircl::draw()
{
   Circl::draw();
   setfillstyle(SOLID_FILL, fillColor);
   floodfill(x0, y0, color);
}

//------------------------------------------------------------------
------------------------------------------------------------------
------------------------------------------------------------
void Curve::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);
```

```
}

void Curve::setPoints(vector<vector<int>> data)
{
   for(int i = 0; i < data.size(); i++)
   {
      this->data.push_back({0, 0});
      this->data[i][0] = data[i][0];
      this->data[i][1] = data[i][1];
   }
}

void Curve::draw()
{
   if(data.size()>1)
   {
      int pef[100] = {0};
      pef[0] = 1;
      pef[1] = 1;
      pefog(pef, 2, data.size());
      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 < data.size(); i++)
         {
            a.x += pef[i]*pow(1-t, data.size()-i-1)*pow(t,
i)*data[i][0];
            a.y += pef[i]*pow(1-t, data.size()-i-1)*pow(t,
i)*data[i][1];
         }
         Line c = Line(b.x, b.y, a.x, a.y, color, thickness);
         c.draw();
         b = a;
      }
      Line c = Line(b.x, b.y, data[data.size()-1][0],
data[data.size()-1][1], color, thickness);
      c.draw();
   }
   if(data.size() == 1) for(int i = 0; i < thickness; i++)
   {
      putpixel(data[0][0]+i, data[0][1], color);
      putpixel(data[0][0], data[0][1]+i, color);
      putpixel(data[0][0]-i, data[0][1], color);
      putpixel(data[0][0], data[0][1]-i, color);
   }
}

void outcoord()
{
```

```
   if(mousex() >= 10 and mousex() <= 1190 and mousey() >= 70 and
mousey() <= 800)
   {
      string str = '(' + to_string(mousex()-10) + ';' +
to_string(mousey()-70) + ')';
      outtextxy(1150, 880, str.c_str());
   }
}

void drawAll(vector<Figure*> acts)
{
   setfillstyle(SOLID_FILL, BLACK);
   bar(10, 70, 1190, 800);
   for(int i = 0; i < acts.size(); i++)
   {
      acts[i]->draw();
   }
   outcoord();
}
```