МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ»

КАФЕДРА ТЕОРЕТИЧЕСКОЙ И ПРИКЛАДНОЙ ИНФОРМАТИКИ

Лабораторная работа №2

по дисциплине «Объектно-ориентированное программирование»

Факультет: ПМИ

Группа: ПМИ-03

Студенты: Малыгин С. А, Сидоров Д. И.

Преподаватель: Лисицын Д. В., Неделько В. М.

НОВОСИБИРСК

2021

1. Условие задачи:

1.Модифицировать программу, разработанную в лабораторной работе № 1, так чтобы в ней был определен класс, реализующий понятие геометрической фигуры в графической системе.

2.1. Определить ответственность класса. Учесть, что общение с пользователем (включая ввод данных с клавиатуры и вывод данных на экран, если их предполагается использовать), а также реакцию на возникающие при работе с функциями класса ошибки следует производить вне функций класса.

2.2. Определить атрибуты, необходимые для реализации класса. Поместить атрибуты в закрытую часть описания.

2.3. Определить функции, необходимые для реализации класса. Выделить интерфейс класса и поместить его в открытую часть описания. Включить в разработанный класс функции:

-устанавливающие и изменяющие геометрические и графические характеристики фигуры (set-функции);

- возвращающие геометрические и графические характеристики фигуры (get-функции);

-рисующие фигуру на экране;

- изменяющие положение фигуры на экране;

- обеспечивающие сохраняемость объекта:

- сохранение набора атрибутов объекта класса в файле и считывание его из файла (файлы для сохранения и считывания должны иметь один формат).

2.4 Обработку ошибок осуществлять с использованием механизма возбуждения и обработки исключительных ситуаций.

2.5. Разработать функцию, демонстрирующую поведение класса. Поместить реализацию класса в один файл, а демонстрационную функцию – в другой. Обеспечить необходимые межмодульные связи.

3. Подготовить текстовые файлы с разработанной программой, оттранслировать их, собрать и выполнить программу с учетом требований операционных систем и программных оболочек, в которых эта программа выполняется.

1. Описание функций:

void save();

Сохраняет результат работы класса в файл

void position(int x, int y);

Изменить положение фигуры

void get\_data();

Получить координаты фигуры

void set\_data();

Изменить координаты фигуры

void contour();

Нарисовать контур фигуры

void flood();

Нарисовать закрашенную фигуру

void fif();

Нарисовать фигуру в фигуре

bool Out\_of\_Window(int num);

Проверка на нахождение фигуры на экране

bool Test\_Convex(const List &figure);

Проверка на выпуклость фигуры

int AB\_D(int x, int y, int xA, int yA, int xB, int yB);

Формула для проверки нахождения фигуры в фигуре

bool Test\_In\_Figure(const List &outside, const List &inside);

Проверка на нахождение фигуры в фигуре

1. Алгоритм:

drawing.cpp:

Логическая функция drawing::Out\_of\_Window(целая num)

{

логическая test(true);

for (целая i(0); (i < 4) && (test == истина); i++)

{

test = ((rt.right >= Figure[num].x[i]) && (rt.bottom >= Figure[num].y[i]) && (0 <= Figure[num].x[i]) && (0 <= Figure[num].y[i]));

}

вернуть test;

}

Логическая функция drawing::Test\_Convex(константа List &figure)

{

Vector ab =

{

figure.x[1] - figure.x[0],

figure.y[1] - figure.y[0]

};

Vector bc =

{

figure.x[2] - figure.x[1],

figure.y[2] - figure.y[1]

};

Vector cd =

{

figure.x[3] - figure.x[2],

figure.y[3] - figure.y[2]

};

Vector da =

{

figure.x[0] - figure.x[3],

figure.y[0] - figure.y[3]

};

целая product1 = ab.x \* bc.y - ab.y \* bc.x;

целая product2 = bc.x \* cd.y - bc.y \* cd.x;

целая product3 = cd.x \* da.y - cd.y \* da.x;

целая product4 = da.x \* ab.y - da.y \* ab.x;

Вернуть ((product1 >= 0) && (product2 >= 0) && (product3 >= 0) && (product4 >= 0));

}

Целая функция drawing::AB\_D(целая x, целая y, целая xA, целая yA, целая xB, целая yB)

{

вернуть (x - xA) \* (yB - yA) - (y - yA) \* (xB - xA);

}

Логическая функция drawing::Test\_In\_Figure(константа List &outside, константа List &inside)

{

for (целая i(0); i < 4; i++)

{

for (целая j(0); j < 3; j++)

{

если (AB\_D(inside.x[i], inside.y[i], outside.x[j], outside.y[j], outside.x[j + 1], outside.y[j + 1]) > 0)

вернуть ложь;

}

}

вернуть истину;

}

Процедура drawing::save()

{

открыть файл("save.txt");

вывод << count << endl << endl;

for (целая i(0); i < count; i++)

{

for (целая j(0); j < 4; j++)

{

вывод << Figure[i].x[j] << " " << Figure[i].y[j] << endl;

}

вывод << endl;

}

Закрыть файл();

}

Целая функция drawing::choose()

{

Беззнаковая целая num(-1);

пока ((0 > num) || (num >= count))

{

Очистить консоль;

вывод << "Доступны фигуры: ";

for (целая i(1); i <= count; i++)

вывод << i << " ";

вывод << endl << "Введите номер фигуры: ";

ввод >> num;

num--;

}

вернуть num;

}

Процедура drawing::position(целая x,целая y)

{

целая i = choose();

for (целая j(0); j < 4; j++)

{

Figure[i].x[j] = Figure[i].x[j] + x;

Figure[i].y[j] = Figure[i].y[j] + y;

вывод << endl;

}

Очистить консоль;

save();

}

Процедура drawing::get\_data()

{

целая i = choose();

вывод << endl;

for (целая j(0); j < 4; j++)

{

вывод << "Точка № " << (j + 1) << endl << "X: " << Figure[i].x[j] << " Y: " << Figure[i].y[j] << endl << endl;

}

}

Процедура drawing::set\_data()

{

целая i = choose();

for (целая j(0); j < 4; j++)

{

вывод << "X = ";

ввод >> Figure[i].x[j];

вывод << "Y = ";

ввод >> Figure[i].y[j];

вывод << endl;

}

Очистить консоль;

save();

}

Процедура drawing::create()

{

for (целая j(0); j < 4; j++)

{

вывод << "X = ";

ввод >> Figure[count].x[j];

вывод << "Y = ";

ввод >> Figure[count].y[j];

вывод << endl;

}

count++;

Очистить консоль;

save();

}

Процедура drawing::delete\_f()

{

целая i = choose();

for (i; i + 1 < count; i++)

for (целая j(0); j < 4; j++)

{

Figure[i].x[j] = Figure[i + 1].x[j];

Figure[i].y[j] = Figure[i + 1].y[j];

}

count--;

Очистить консоль;

save();

}

Процедура drawing::contour()

{

целая num = choose();

Получить размер окна;

если (!Test\_Convex(Figure[num])) вызвать исключение 1;

если (!Out\_of\_Window(num)) вызвать исключение 2;

Очистить консоль;

вывод << "Введите цвет контура(RGB): " << endl;

целая R(0), G(0), B(0);

вывод << "R = ";

ввод >> R;

вывод << endl << "G = ";

ввод >> G;

вывод << endl << "B = ";

ввод >> B;

HPEN hPen1 = Создать перо (PS\_SOLID, 3, RGB(R, G, B));

do

{

Очистить консоль;

Получить размер окна;

POINT points[5];

for (int i = 0; i < 4; i++)

{

points[i] = { Figure[num].x[i], Figure[num].y[i] };

}

points[4] = { Figure[num].x[0], Figure[num].y[0] };

Выбрать перо(hdc, hPen1);

Нарисовать четырехугольник(hdc, points, 5);

} пока (\_getch() != 27);

Удалить перо (hPen1);

}

Процедура drawing::flood()

{

целая num = choose();

Получить размер окна;

Если (!Test\_Convex(Figure[num])) вызвать исключение 1;

Если (!Out\_of\_Window(num)) вызвать исключение 2;

вывод << "Введите цвет контура фигуры(RGB): " << endl;

целая R(0), G(0), B(0);

вывод << "R = ";

ввод >> R;

вывод << endl << "G = ";

ввод >> G;

вывод << endl << "B = ";

ввод >> B;

HPEN hPen1 = Создать перо(PS\_SOLID, 3, RGB(R, G, B));

Очистить консоль;

вывод << "Введите цвет фигуры(RGB): " << endl;

вывод << "R = ";

ввод >> R;

вывод << endl << "G = ";

ввод >> G;

вывод << endl << "B = ";

ввод >> B;

HBRUSH hBrush1 = Создать кисть(RGB(R, G, B));

do

{

Очистить консоль;

Получить размер окна;

POINT points[4];

for (целая i = 0; i < 4; i++)

{

points[i] = { Figure[num].x[i], Figure[num].y[i] };

}

Выбрать перо (hdc, hPen1);

Выбрать кисть(hdc, hBrush1);

Нарисовать четырехугольник(hdc, points, 4);

} пока (\_getch() != 27);

Удалить кисть(hBrush1);

Удалить перо(hPen1);

}

Процедура drawing::fif()

{

целая num1 = choose();

целая num2 = choose();

Получить размер окна;

целая (!(Out\_of\_Window(num1) && Out\_of\_Window(num2))) вызвать исключение 1;

если (!(Test\_Convex(Figure[num1]) && Test\_Convex(Figure[num2]))) вызвать исключение 2;

если (!Test\_In\_Figure(Figure[num1], Figure[num2])) вызвать исключение 3;

Очистить консоль;

BOOL bxt;

вывод << "Введите цвет контура фигуры-1 (RGB): " << endl;

целая R(0), G(0), B(0);

вывод << "R = ";

ввод >> R;

вывод << endl << "G = ";

ввод >> G;

вывод << endl << "B = ";

ввод >> B;

HPEN hPen1 = создать перо(PS\_SOLID, 3, RGB(R, G, B));

Очистить консоль;

вывод << "Введите цвет фигуры-1 (RGB): " << endl;

вывод << "R = ";

ввод >> R;

вывод << endl << "G = ";

ввод >> G;

вывод << endl << "B = ";

ввод >> B;

HBRUSH hBrush1 = Создать кисть(RGB(R, G, B));

Очистить консоль;

вывод << "Введите цвет контура фигуры-2 (RGB): " << endl;

вывод << "R = ";

ввод >> R;

вывод << endl << "G = ";

ввод >> G;

вывод << endl << "B = ";

ввод >> B;

HPEN hPen2 = Создать перо(PS\_SOLID, 3, RGB(R, G, B));

HBRUSH hBrush2 = Получить кисть(BLACK\_BRUSH);

do

{

Очистить консоль;

Получить размер окна;

POINT points1[4];

for (целая i = 0; i < 4; i++)

{

points1[i] = { Figure[num1].x[i], Figure[num1].y[i] };

}

Выбрать перо(hdc, hPen1);

Выбрать кисть(hdc, hBrush1);

bxt = Нарисовать четырехугольник(hdc, points1, 4);

POINT points2[4];

for (целая i = 0; i < 4; i++)

{

points2[i] = { Figure[num2].x[i], Figure[num2].y[i] };

}

Выбрать перо(hdc, hPen2);

Выбрать кисть(hdc, hBrush2);

bxt = Нарисовать четырехугольник(hdc, points2, 4);

} пока (\_getch() != 27);

Удалить кисть(hBrush1);

Удалить кисть (hBrush2);

Удалить перо(hPen1);

Удалить перо(hPen2);

}

drawing::drawing()

{

};

drawing::~drawing()

{

};

Source.cpp:

drawing d;

целая counter;

структура List

{

int x[4];

int y[4];

};

List \*Figure = new List[15];

Процедура menu()

{

если (d.count > 0)

{

вывод << "Вам доступны " << d.count << "/15 фигур" << endl;

вывод << "Выберите функцию: " << endl;

вывод << endl << "1 - незакрашенная фигура(фигура-контур)" << endl;

вывод << "2 - закрашенная фигура" << endl;

вывод << "3 - две вложенные одна в другую фигуры" << endl;

вывод << "4 - получить параметры фигуры" << endl;

вывод << "5 - изменить параметры фигуры" << endl;

вывод << "6 - изменить положение фигуры" << endl;

вывод << "7 - Создать новую фигуру" << endl;

вывод << "8 - Удалить фигуру" << endl;

вывод << "9 - выход из программы" << endl;

целая Flag(-1);

пока ((1 > Flag) || (Flag > 9))

{

ввод >> Flag;

Flag;

Переключатель (Flag)

{

Кейс 1:

Защищенный код

{

d.contour();

}

Обработка исключений (целая a)

{

если (a == 1)

вывод << "---Фигура не выпуклая!--"<<endl;

если (a == 2)

вывод << "---Фигура выходит за края экрана!---" << endl;

}

menu();

прекратить;

Кейс 2:

Защищенный код

{

d.flood();

}

Обработка исключений (целая a)

{

если (a == 1)

вывод << "---Фигура не выпуклая!---" << endl;

если (a == 2)

вывод << "---Фигура выходит за края экрана!---" << endl;

}

menu();

прекратить;

Кейс 3:

Защищенный код

{

d.fif();

}

Обработка исключений (целая a)

{

если (a == 1)

вывод << "---Фигура выходит за края экрана!---" << endl;

если (a == 2)

вывод << "---Фигура(ы) не выпуклая(ые)!---" << endl;

если (a == 3)

вывод << "---Внутренняя фигура выходит за края внешней фигуры!---" << endl;

}

menu();

прекратить;

Кейс 4:

d.get\_data();

menu();

прекратить;

Кейс 5:

d.set\_data();

menu();

прекратить;

Кейс 6:

Очистить консоль;

целая x, y;

вывод << "На сколько пикселей вы хотите сместить фигуру?" << endl;

вывод << "X: ";

ввод >> x;

вывод << endl << "Y: ";

ввод >> y;

d.position(x,y);

menu();

прекратить;

Кейс 7:

Очистить консоль;

вывод << "Введите координаты новой фигуры: " << endl;

d.create();

menu();

прекратить;

Кейс 8:

d.delete\_f();

menu();

прекратить;

Кейс 9:

выход;

прекратить;

}

}

}

иначе

{

вывод << "Вам доступны " << d.count << "/15 фигур" << endl;

вывод << "Выберите функцию: " << endl;

вывод << endl << "1 - Создать новую фигуру" << endl;

вывод << "2 - выход из программы" << endl;

беззнаковая целая Flag(-1);

пока ((0 > Flag) || (Flag > 2))

{

вывод << ">>>>";

ввод >> Flag;

Переключатель (Flag)

{

Кейс 1:

Очистить консоль

d.create();

menu();

Прекратить;

Кейс 2:

выход;

Прекратить;

}

}

}

}

Процедура start()

{

целая flag(-1);

пока ((1 > flag) || (flag > 3))

{

вывод << "1-Ввод из файла" << endl;

вывод << "2-Ввод c клавиатуры" << endl;

вывод << "3-Продолжить работу с сохраненными данными" << endl;

ввод >> flag;

Переключатель (flag)

{

Кейс 1:

Открыть файл("set.txt");

если (файл не открылся)

{

вывод << "Не удается открыть файл" << endl;

}

ввод >> d.count;

если (d.count > 0)

{

for (целая i(0); i < d.count; i++)

for (целая j(0); j < 4; j++)

{

ввод >> d.Figure[i].x[j];

ввод >> d.Figure[i].y[j];

}

Закрыть файл();

}

прекратить;

Кейс 2:

вывод << "Сколько фигур необходимо?" << endl;

ввод >> d.count;

Очистить консоль;

если (d.count > 0)

{

for (целая i(0); i < d.count; i++)

{

вывод << "Введите координаты фигуры №" << (i + 1) << endl;

for (целая j(0); j < 4; j++)

{

вывод << "X = ";

ввод >> d.Figure[i].x[j];

вывод << "Y = ";

ввод >> d.Figure[i].y[j];

вывод << endl;

}

Очистить консоль;

}

}

прекратить;

Кейс 3:

Открыть файл("save.txt");

если (файл не открывается)

{

вывод << "Не удается открыть файл" << endl;

}

ввод >> d.count;

если (d.count > 0)

{

for (целая i(0); i < d.count; i++)

for (целая j(0); j < 4; j++)

{

ввод >> d.Figure[i].x[j];

ввод >> d.Figure[i].y[j];

}

Закрыть файл();

}

прекратить;

}

если (d.count < 0) d.count = 0;

d.save();

Очистить консоль;

}

}

Процедура save\_main()

{

counter = d.count;

если (d.count > 0)

{

for (целая i(0); i < d.count; i++)

for (целая j(0); j < 4; j++)

{

Figure[i].x[j]=d.Figure[i].x[j];

Figure[i].x[j]=d.Figure[i].y[j];

}

}

}

int main()

{

setlocale(LC\_ALL, "rus");

start();

menu();

save\_main();

}

1. Код программы:

Source.cpp:

#include "drawing.h"

#include <locale.h>

#include <iostream>

#include <fstream>

using namespace std;

drawing d;

int counter;

struct List

{

int x[4];

int y[4];

};

List \*Figure = new List[15];

void menu()

{

if (d.count > 0)

{

cout << "Вам доступны " << d.count << "/15 фигур" << endl;

cout << "Выберите функцию: " << endl;

cout << endl << "1 - незакрашенная фигура(фигура-контур)" << endl;

cout << "2 - закрашенная фигура" << endl;

cout << "3 - две вложенные одна в другую фигуры" << endl;

cout << "4 - получить параметры фигуры" << endl;

cout << "5 - изменить параметры фигуры" << endl;

cout << "6 - изменить положение фигуры" << endl;

cout << "7 - Создать новую фигуру" << endl;

cout << "8 - Удалить фигуру" << endl;

cout << "9 - выход из программы" << endl;

int Flag(-1);

while ((1 > Flag) || (Flag > 9))

{

cin >> Flag;

Flag;

switch (Flag)

{

case 1:

try

{

d.contour();

}

catch (int a)

{

if (a == 1)

cout << "---Фигура не выпуклая!--"<<endl;

if (a == 2)

cout << "---Фигура выходит за края экрана!---" << endl;

}

menu();

break;

case 2:

try

{

d.flood();

}

catch (int a)

{

if (a == 1)

cout << "---Фигура не выпуклая!---" << endl;

if (a == 2)

cout << "---Фигура выходит за края экрана!---" << endl;

}

menu();

break;

case 3:

try

{

d.fif();

}

catch (int a)

{

if (a == 1)

cout << "---Фигура выходит за края экрана!---" << endl;

if (a == 2)

cout << "---Фигура(ы) не выпуклая(ые)!---" << endl;

if (a == 3)

cout << "---Внутренняя фигура выходит за края внешней фигуры!---" << endl;

}

menu();

break;

case 4:

d.get\_data();

menu();

break;

case 5:

d.set\_data();

menu();

break;

case 6:

system("cls");

int x, y;

cout << "На сколько пикселей вы хотите сместить фигуру?" << endl;

cout << "X: ";

cin >> x;

cout << endl << "Y: ";

cin >> y;

d.position(x,y);

menu();

break;

case 7:

system("cls");

cout << "Введите координаты новой фигуры: " << endl;

d.create();

menu();

break;

case 8:

d.delete\_f();

menu();

break;

case 9:

exit;

break;

}

}

}

else

{

cout << "Вам доступны " << d.count << "/15 фигур" << endl;

cout << "Выберите функцию: " << endl;

cout << endl << "1 - Создать новую фигуру" << endl;

cout << "2 - выход из программы" << endl;

unsigned int Flag(-1);

while ((0 > Flag) || (Flag > 2))

{

cout << ">>>>";

cin >> Flag;

switch (Flag)

{

case 1:

system("cls");

d.create();

menu();

break;

case 2:

exit;

break;

}

}

}

}

void start()

{

int flag(-1);

while ((1 > flag) || (flag > 3))

{

cout << "1-Ввод из файла" << endl;

cout << "2-Ввод c клавиатуры" << endl;

cout << "3-Продолжить работу с сохраненными данными" << endl;

cin >> flag;

ifstream fin;

switch (flag)

{

case 1:

fin.open("set.txt");

if (!fin.is\_open())

{

cout << "Не удается открыть файл" << endl;

}

fin >> d.count;

if (d.count > 0)

{

for (int i(0); i < d.count; i++)

for (int j(0); j < 4; j++)

{

fin >> d.Figure[i].x[j];

fin >> d.Figure[i].y[j];

}

fin.close();

}

break;

case 2:

cout << "Сколько фигур необходимо?" << endl;

cin >> d.count;

system("cls");

if (d.count > 0)

{

for (int i(0); i < d.count; i++)

{

cout << "Введите координаты фигуры №" << (i + 1) << endl;

for (int j(0); j < 4; j++)

{

cout << "X = ";

cin >> d.Figure[i].x[j];

cout << "Y = ";

cin >> d.Figure[i].y[j];

cout << endl;

}

system("cls");

}

}

break;

case 3:

fin.open("save.txt");

if (!fin.is\_open())

{

cout << "Не удается открыть файл" << endl;

}

fin >> d.count;

if (d.count > 0)

{

for (int i(0); i < d.count; i++)

for (int j(0); j < 4; j++)

{

fin >> d.Figure[i].x[j];

fin >> d.Figure[i].y[j];

}

fin.close();

}

break;

}

if (d.count < 0) d.count = 0;

d.save();

system("cls");

}

}

void save\_main()

{

counter = d.count;

if (d.count > 0)

{

for (int i(0); i < d.count; i++)

for (int j(0); j < 4; j++)

{

Figure[i].x[j]=d.Figure[i].x[j];

Figure[i].x[j]=d.Figure[i].y[j];

}

}

}

int main()

{

setlocale(LC\_ALL, "rus");

start();

menu();

save\_main();

}

Drawing.h:

#include <windows.h>

#include <windowsx.h>

#pragma once

class drawing

{

public:

struct List

{

int x[4];

int y[4];

};

struct Vector

{

int x;

int y;

};

int count;

List \*Figure = new List[15];

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

RECT rt;

drawing();

~drawing();

void save();

void position(int x, int y);

void get\_data();

void set\_data();

void create();

void delete\_f();

void contour();

void flood();

void fif();

private:

bool Out\_of\_Window(int num);

bool Test\_Convex(const List &figure);

int AB\_D(int x, int y, int xA, int yA, int xB, int yB);

bool Test\_In\_Figure(const List &outside, const List &inside);

int choose();

};

Drawing.cpp:

#include "drawing.h"

#include <windows.h>

#include <windowsx.h>

#include <conio.h>

#include <fstream>

#include <iostream>

using namespace std;

bool drawing::Out\_of\_Window(int num)

{

bool test(true);

for (int i(0); (i < 4) && (test == true); i++)

{

test = ((rt.right >= Figure[num].x[i]) && (rt.bottom >= Figure[num].y[i]) && (0 <= Figure[num].x[i]) && (0 <= Figure[num].y[i]));

}

return test;

}

bool drawing::Test\_Convex(const List &figure)

{

Vector ab =

{

figure.x[1] - figure.x[0],

figure.y[1] - figure.y[0]

};

Vector bc =

{

figure.x[2] - figure.x[1],

figure.y[2] - figure.y[1]

};

Vector cd =

{

figure.x[3] - figure.x[2],

figure.y[3] - figure.y[2]

};

Vector da =

{

figure.x[0] - figure.x[3],

figure.y[0] - figure.y[3]

};

int product1 = ab.x \* bc.y - ab.y \* bc.x;

int product2 = bc.x \* cd.y - bc.y \* cd.x;

int product3 = cd.x \* da.y - cd.y \* da.x;

int product4 = da.x \* ab.y - da.y \* ab.x;

return ((product1 >= 0) && (product2 >= 0) && (product3 >= 0) && (product4 >= 0));

}

int drawing::AB\_D(int x, int y, int xA, int yA, int xB, int yB)

{

return (x - xA) \* (yB - yA) - (y - yA) \* (xB - xA);

}

bool drawing::Test\_In\_Figure(const List &outside, const List &inside)

{

for (int i(0); i < 4; i++)

{

for (int j(0); j < 3; j++)

{

if (AB\_D(inside.x[i], inside.y[i], outside.x[j], outside.y[j], outside.x[j + 1], outside.y[j + 1]) > 0)

return false;

}

}

return true;

}

void drawing::save()

{

ofstream fout;

fout.open("save.txt");

fout << count << endl << endl;

for (int i(0); i < count; i++)

{

for (int j(0); j < 4; j++)

{

fout << Figure[i].x[j] << " " << Figure[i].y[j] << endl;

}

fout << endl;

}

fout.close();

}

int drawing::choose()

{

unsigned int num(-1);

while ((0 > num) || (num >= count))

{

system("cls");

cout << "Доступны фигуры: ";

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

cout << i << " ";

cout << endl << "Введите номер фигуры: ";

cin >> num;

num--;

}

return num;

}

void drawing::position(int x,int y)

{

int i = choose();

for (int j(0); j < 4; j++)

{

Figure[i].x[j] = Figure[i].x[j] + x;

Figure[i].y[j] = Figure[i].y[j] + y;

cout << endl;

}

system("cls");

save();

}

void drawing::get\_data()

{

int i = choose();

cout << endl;

for (int j(0); j < 4; j++)

{

cout << "Точка № " << (j + 1) << endl << "X: " << Figure[i].x[j] << " Y: " << Figure[i].y[j] << endl << endl;

}

}

void drawing::set\_data()

{

int i = choose();

for (int j(0); j < 4; j++)

{

cout << "X = ";

cin >> Figure[i].x[j];

cout << "Y = ";

cin >> Figure[i].y[j];

cout << endl;

}

system("cls");

save();

}

void drawing::create()

{

for (int j(0); j < 4; j++)

{

cout << "X = ";

cin >> Figure[count].x[j];

cout << "Y = ";

cin >> Figure[count].y[j];

cout << endl;

}

count++;

system("cls");

save();

}

void drawing::delete\_f()

{

int i = choose();

for (i; i + 1 < count; i++)

for (int j(0); j < 4; j++)

{

Figure[i].x[j] = Figure[i + 1].x[j];

Figure[i].y[j] = Figure[i + 1].y[j];

}

count--;

system("cls");

save();

}

void drawing::contour()

{

int num = choose();

GetClientRect(hwnd, &rt);

if (!Test\_Convex(Figure[num])) throw 1;

if (!Out\_of\_Window(num)) throw 2;

system("cls");

cout << "Введите цвет контура(RGB): " << endl;

int R(0), G(0), B(0);

cout << "R = ";

cin >> R;

cout << endl << "G = ";

cin >> G;

cout << endl << "B = ";

cin >> B;

HPEN hPen1 = CreatePen(PS\_SOLID, 3, RGB(R, G, B));

do

{

system("cls");

GetClientRect(hwnd, &rt);

POINT points[5];

for (int i = 0; i < 4; i++)

{

points[i] = { Figure[num].x[i], Figure[num].y[i] };

}

points[4] = { Figure[num].x[0], Figure[num].y[0] };

SelectPen(hdc, hPen1);

Polyline(hdc, points, 5);

} while (\_getch() != 27);

DeletePen(hPen1);

}

void drawing::flood()

{

int num = choose();

GetClientRect(hwnd, &rt);

if (!Test\_Convex(Figure[num])) throw 1;

if (!Out\_of\_Window(num)) throw 2;

cout << "Введите цвет контура фигуры(RGB): " << endl;

int R(0), G(0), B(0);

cout << "R = ";

cin >> R;

cout << endl << "G = ";

cin >> G;

cout << endl << "B = ";

cin >> B;

HPEN hPen1 = CreatePen(PS\_SOLID, 3, RGB(R, G, B));

system("cls");

cout << "Введите цвет фигуры(RGB): " << endl;

cout << "R = ";

cin >> R;

cout << endl << "G = ";

cin >> G;

cout << endl << "B = ";

cin >> B;

HBRUSH hBrush1 = CreateSolidBrush(RGB(R, G, B));

do

{

system("cls");

GetClientRect(hwnd, &rt);

POINT points[4];

for (int i = 0; i < 4; i++)

{

points[i] = { Figure[num].x[i], Figure[num].y[i] };

}

SelectPen(hdc, hPen1);

SelectBrush(hdc, hBrush1);

Polygon(hdc, points, 4);

} while (\_getch() != 27);

DeleteBrush(hBrush1);

DeletePen(hPen1);

}

void drawing::fif()

{

int num1 = choose();

int num2 = choose();

GetClientRect(hwnd, &rt);

if (!(Out\_of\_Window(num1) && Out\_of\_Window(num2))) throw 1;

if (!(Test\_Convex(Figure[num1]) && Test\_Convex(Figure[num2]))) throw 2;

if (!Test\_In\_Figure(Figure[num1], Figure[num2])) throw 3;

system("cls");

BOOL bxt;

cout << "Введите цвет контура фигуры-1 (RGB): " << endl;

int R(0), G(0), B(0);

cout << "R = ";

cin >> R;

cout << endl << "G = ";

cin >> G;

cout << endl << "B = ";

cin >> B;

HPEN hPen1 = CreatePen(PS\_SOLID, 3, RGB(R, G, B));

system("cls");

cout << "Введите цвет фигуры-1 (RGB): " << endl;

cout << "R = ";

cin >> R;

cout << endl << "G = ";

cin >> G;

cout << endl << "B = ";

cin >> B;

HBRUSH hBrush1 = CreateSolidBrush(RGB(R, G, B));

system("cls");

cout << "Введите цвет контура фигуры-2 (RGB): " << endl;

cout << "R = ";

cin >> R;

cout << endl << "G = ";

cin >> G;

cout << endl << "B = ";

cin >> B;

HPEN hPen2 = CreatePen(PS\_SOLID, 3, RGB(R, G, B));

HBRUSH hBrush2 = GetStockBrush(BLACK\_BRUSH);

do

{

system("cls");

GetClientRect(hwnd, &rt);

POINT points1[4];

for (int i = 0; i < 4; i++)

{

points1[i] = { Figure[num1].x[i], Figure[num1].y[i] };

}

SelectPen(hdc, hPen1);

SelectBrush(hdc, hBrush1);

bxt = Polygon(hdc, points1, 4);

POINT points2[4];

for (int i = 0; i < 4; i++)

{

points2[i] = { Figure[num2].x[i], Figure[num2].y[i] };

}

SelectPen(hdc, hPen2);

SelectBrush(hdc, hBrush2);

bxt = Polygon(hdc, points2, 4);

} while (\_getch() != 27);

DeleteBrush(hBrush1);

DeleteBrush(hBrush2);

DeletePen(hPen1);

DeletePen(hPen2);

}

drawing::drawing()

{

};

drawing::~drawing()

{

};

1. Набор тестов:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Set.txt | Результат | Примечание |
| 1 | 700 700  1000 1000  800 1500  500 1500 | Фигура выходит за края экрана! | Фигура выходит за края экрана! |
| 2 |  | Файл пуст! | Файл пуст! |
| 3 | 1 1  200 1  200 200  1 200 |  | Контур фигуры |
| 4 | 100 100  300 100  100 400  90 250 |  | Заливка фигуры |
| 5 | 0 0  800 0  800 800  0 700  100 100  300 100  100 400  90 250 |  | Фигура в фигуре |
| 6 | 100 100  300 100  150 250  175 400 | Фигура не выпуклая! | Четырехугольник невыпуклый |
| 7 | 100 100  300 100  100 400  90 250  0 0  400 0  400 400  0 400 | Внутренняя фигура выходит за края внешней фигуры! | Фигура не в фигуре |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Функция | Было | Стало | Примечание |
| 1 | get\_data | Выбрана фигура 1 | Точка № 1  X: 1 Y: 1  Точка № 2  X: 200 Y: 1  Точка № 3  X: 200 Y: 200  Точка № 4  X: 1 Y: 200 | Функция работает правильно |
| 2 | set\_data | Координаты 1 фигуры  1 1  200 1  200 200  1 200 | Координаты 1 фигуры  0 100  200 100  200 200  1 200 | Функция работает правильно |
| 3 | create | 10/15 есть | Добавлена новая фигура  11/15 есть | Функция работает правильно |
| 4 | save | Файл save.txt пуст  или занят устаревшей информацией | Файл заполнен новыми данными | Функция работает правильно |
| 5 | delete\_f | 10/15 есть | Удалена выбранная фигура  9/15 есть | Функция работает правильно |
| 6 | position | Координаты 1 фигуры  1 1  200 1  200 200  1 200  Сместить на 100 и 100 | Координаты 1 фигуры  101 101  300 101  300 300  101 300 | Функция работает правильно |

1. Программа работает правильно, что подтверждают тесты.