# Министерство науки и высшего образования Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра теоретической и прикладной информатики

### Лабораторная работа № 2 по дисциплине «Объектно-ориентированное программирование»

**Классы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| сигма градиент синий1 | Факультет: | ПМИ |  |  |
| Группа: | ПМИ-81 |  |  |
| Студенты: | Демидович Е.Ю.,  Муравьев М.И. |  |  |
| Вариант: | 7 |  |  |
| Преподаватель: | Еланцева И.Л., Лисицин Д.В. |  |  |

Новосибирск

2019

1. **Постановка задачи**

Разработать класс, реализующий понятие треугольника в графической системе:

-Определить ответственность класса;

-Определить атрибуты, поместить их в закрытую часть класса;

-Определить функции класса, выделить и поместить интерфейс в открытую

часть;

-Обработку ошибок осуществлять с использованием механизма возбуждения и обработки исключительных ситуаций;

-Включить следующие функции в класс:

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;

Чтения и записи набора атрибутов;

1. **Описание разработанного класса**

**Атрибуты класса:**

POINT points[3] – координаты точек первого треугольника

POINT Cpoints[3] – координаты точек второго(малого) треугольника

int style – стиль пера

int bstyle – стиль кисти

int border – ширина обводки

int mode – режим отрисовки

COLORREF color – цвет пера

COLORREF bcolor – цвет кисти

**Процедуры класса:**

void drawTriangle(HDC hdc); //Рисование треугольника

void drawCounter(HDC hdc); //Контура

void drawCut(HDC hdc); //Треугольника с вырезом

void mainDraw(HDC hdc); //Рисование фигуры

void setMode(int new\_mode); //Тип фигуры

void getMode(int &curr\_mode);

void setPoints(POINT \*new\_points); //Точки треугольника

void getPoints(POINT \*curr\_points)

void setCPoints(POINT \*new\_Cpoints); //Точки треугольника для выреза

void getCPoints(POINT \*curr\_Cpoints);

void setBorder(int new\_border); //Толщина контура

void getBorder(int &curr\_border);

void setStyle(int new\_style); //Стиль пера(контура)

void getStyle(int &curr\_style);

void setColor(COLORREF new\_tcolor); //Цвет пера(контура)

void getColor(COLORREF &curr\_tcolor);

void setBStyle(int new\_bstyle); //Стиль кисти (заливки)

void getBStyle(int &curr\_bstyle);

void setBColor(COLORREF new\_bcolor); //Цвет кисти(заливки)

void getBColor(COLORREF &curr\_bcolor);

void checkSizeTC(POINT \*new\_Cpoints); //Входит ли вырез в треугольник

void checkSizeT( int rtx, int rty); //Входит ли треугольник в окно

void checkSizeT( HDC hdc ); //Входит ли треугольник в окно

void readInf(const char name[]); //Ввод треугольника из файла

void saveInf(const char name[]); //Вывод треугольника в файл

void move\_horizontally(int delta); //Сдвиг фигуры по горизонтали

void move\_vertically(int delta); //Сдвиг фигуры по вертикали

1. **Анализ задачи**

**Чтение из файла:**

Файл с исходной информацией имеет следующую структуру:

mode режим (0 – закрашенная фигура, 1 – контур, 2 – фигура с вырезом)

x[0] y[0]

x[1] y[1] координаты первого треугольника

x[2] y[2]

style стиль заливки (0-6)

r1 g1 b1 цвет заливки (три значения, каждое <256)

border bstyle ширина обводки (>=0), стиль пера(0-6)

r2 g2 b2 цвет пера(контура) (три значения, каждое <256)

x[0] y[0]

x[1] y[1] координаты второго треугольника(выреза)

x[2] y[2] (не читается если m != 2)

Запись в выходной файл происходит в том же формате что и чтение.

Запись в выходной файл, изображённой в консоли фигуры, производится при

нажатии клавиши «1».

**Проверка корректности данных:**

Если выполняется следующее равенство, то заданные точки лежат на одной прямой:

(x[2] – x[0]) \* (y[1] - y[0]) == (y[2] - [0].y) \* (x[1] - x[0]);

Если для каждой точки не выполняется следующее условие, то треугольник находится в пределах окна: i = [0,2]

x[i] < 0 || x[i] > rtx || y[i] < 0 || y[i] > rty , где rtx и rty – соответствующие размеры окна;

Если для каждой точки треугольника для выреза выполняется следующее условие, то треугольник не выходит за рамки первого треугольника:

a = (x[0] - x[i]) \* (y[1] - y[0]) - (x[1] - x[0]) \* (y[0] - y[i]);

b = (x[1] - x[i]) \* (y[2] - y[1]) - (x[2] - x[1]) \* (y[1] - y[i]);

c = (x[2] - x[i]) \* (y[0] - y[2]) - (x[0] - x[2]) \* (y[2] – y[i]);

((a >= 0 && b >= 0 && c >= 0) || (a <= 0 && b <= 0 && c <= 0))

x[i],y[i] – координаты точек второго треугольника, i = [0,2];

Так же, при изменении свойств объекта, проводится проверка данных.

Для обработки исключительных ситуаций, т.е. ситуаций, в которых входные данные имеют недопустимые значения, разработаны коды исключений:

|  |  |  |
| --- | --- | --- |
| Код | Нотация | Комментарий |
| -2 | ERR\_CANNOT\_OPEN\_INPUT\_FILE | Ошибка открытия входного файла |
| -1 | ERR\_WRONG\_SIZE | Недостаточно большой размер экрана |
| 0 | ERR\_NEGATIVE\_POINTS | Хотя бы одна точка треугольника имеет хотя бы одну отрицательную координату |
| 1 | ERR\_POINTS\_ONLINE | Точки треугольника лежат на одной прямой |
| 2 | ERR\_NEGATIVE\_BORDER | Отрицательная ширина пера |
| 3 | ERR\_WRONG\_STYLE | Неверный стиль кисти |
| 4 | ERR\_WRONG\_BSTYLE | Неверный стиль пера |
| 5 | ERR\_WRONG\_COLOR | Неверный цвет кисти |
| 6 | ERR\_WRONG\_BCOLOR | Неверный цвет пера |
| 7 | ERR\_WRONG\_CUT\_POINTS | Вырез частично или полностью не попадает в область заданного треугольника |
| 8 | ERR\_WRONG\_MODE | Неверный режим |

**Рисование фигур:**

Закрашенной:

Построение фигуры по заданным координатам, стилям и цветам заливки и обводки.

Контура:

Построение замкнутой линии по заданным точкам, стилю и цвету обводки.

С вырезом:

Сначала строится закрашенная фигура, затем внутри неё фигура с чёрным (так как фон контекста отображения консоли – чёрный) цветом обводки и заливки.

**Сдвиг фигуры:**

Сдвиг осуществляется изменением координат соответствующих точек (горизонтальное движение – x координаты, вертикальное – y координаты и т.д.) треугольника/треугольников на одинаковую величину, равную 2 пикселям, при нажатии соответствующей клавиши:

W – сдвиг вверх;

A – влево;

S – вниз;

D – вправо;

Q – в сторону левого верхнего угла;

E – в сторону правого верхнего угла;

Z – левого нижнего угла;

C – правого нижнего угла.

**Стили рисования фигур:**

Стиль кисти:

Определяется по следующим параметрам:

|  |  |  |
| --- | --- | --- |
| № Стиля | Описание | Пример |
| 0 | Сплошная |  |
| 1 | Горизонтальные линии | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-47-758.jpg |
| 2 | Вертикальные линии | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-48-988.jpg |
| 3 | Диагональная, с наклоном в 45 градусов вниз | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-53-272.jpg |
| 4 | Диагональная, с наклоном в 45 градусов вверх | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-54-689.jpg |
| 5 | Клеточная | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-56-097.jpg |
| 6 | Клеточная диагональная | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-57-425.jpg |

Стиль пера (работает только при толщине в 1 пиксел):

Определяется по следующим параметрам:

|  |  |  |
| --- | --- | --- |
| № Стиля | Описание | Пример |
| 0 | Сплошное | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-23-887.jpg |
| 1 | Пунктир | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-25-285.jpg |
| 2 | Точка | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-27-795.jpg |
| 3 | Пунктир-точка | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-29-289.jpg |
| 4 | Пунктир-точка-точка | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-31-027.jpg |
| 5 | Прозрачное | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-33-768.jpg |

1. **Алгоритм**

**Класс Triangle:**

**Закрытая часть:**

Массив точек points[3] – координаты точек треугольника

Массив точек Cpoints[3] – координаты точек внутреннего треугольника

Целое число style – стиль заливки

Цветовой тип color – цвет заливки

Целое число border, bstyle – ширина обводки, стиль обводки

Цветовой тип bcolor – цвет обводки

**Открытая часть:**

**drawTriangle(hdc – контекст отображения) – рисование закрашенного треугольника**

hMyPen = Создание пера(bstyle, border, bcolor)

hMyBrush = Создание кисти(style, style-1, color)

Выбор пера(hdc, hMyPen)

Выбор кисти(hdc, hMyBrush)

Построение фигуры(hdc, points, 3);

Удаление пера(hMyPen);

Удаление кисти(hMyBrush);

**drawCounter(hdc) – рисование контура**

hMyPen = Создание пера(bstyle, border, bcolor)

Массив точек ppt[4] = { points[0], points[1], points[2], points[0]}

Построение линии(hdc, ppt, 4);

Удаление пера(hMyPen);

**drawCut(hdc) – рисование треугольника с вырезом**

hMyPen = Создание пера(0, 0, RGB(0,0,0))

hMyBrush = Создание кисти(RGB(0,0,0))

Выбор пера(hdc, hMyPen)

Выбор кисти(hdc, hMyBrush)

Построение фигуры(hdc, Cpoints, 3);

Удаление пера(hMyPen);

Удаление кисти(hMyBrush);

**mainDraw(hdc) – определение режима рисования по mode**

Если mode

равен 0, то drawTriangle(hdc); break;

равен 1, то drawCounter(hdc); break;

равен 2, то drawTriangle(hdc); drawCut(hdc); break;

**setMode(new\_mode - режим) – задание нового режима работы**

Если new\_mode < 0 || new\_mode > 2, то throw ERR\_WRONG\_MODE

mode = new\_mode

**getMode(&curr\_mode) – получение текущего режима**

curr\_mode = mode

**setPoints(\*new\_points – массив координат) – задание точек основного треугольника**

Для(i = 0; i < 3; i++)

Если new\_points[i].x < 0 || new\_points[i].y < 0

throw ERR\_NEGATIVE\_POINTS

Если ((new\_points[2].x - new\_points[0].x)\*(new\_points[1].y - new\_points[0].y) == (new\_points[2].y - new\_points[0].y)\*(new\_points[1].x - new\_points[0].x))

throw ERR\_POINTS\_ONLINE

Для(i = 0; i < 3; i++)

points[i] = new\_points[i]

**getPoints(\*curr\_points) – взятие текущих точек основного треугольника**

Для(i = 0; i < 3; i++)

curr\_points[i] = points[i]

**setCPoints(\*new\_Cpoints) – задание точек малого треугольника**

Для(i = 0; i < 3; i++)

Если new\_Cpoints[i].x < 0 || new\_Cpoints[i].y < 0

throw ERR\_NEGATIVE\_POINTS

Если ((new\_Cpoints[2].x - new\_Cpoints[0].x)\*(new\_Cpoints[1].y - new\_Cpoints[0].y) == (new\_Cpoints[2].y - new\_Cpoints[0].y)\*(new\_Cpoints[1].x - new\_Cpoints[0].x))

throw ERR\_POINTS\_ONLINE

Для(i = 0; i < 3; i++)

Cpoints[i] = new\_Cpoints[i]

**getCPoints(\*curr\_Cpoints) – взятие текущих точек малого треугольника**

Для(i = 0; i < 3; i++)

curr\_Cpoints[i] = Cpoints[i]

**setBorder(new\_border) – задание толщины контура**

Если new\_border < 0

throw ERR\_NEGATIVE\_BORDER

border = new\_border

**getBorder(&curr\_border) – взятие текущей толщины контура**

curr\_border = border

**setStyle(new\_Style) – задание стиля кисти**

Если new\_style < 0 || new\_style > 6

throw ERR\_WRONG\_STYLE

**getStyle (&curr\_Style) – взятие текущего стиля кисти**

curr\_style = style

**setColor(new\_Color) – задание цвета кисти**

Если (new\_color(Красный) > 255 || new\_color(Красный) < 0 || new\_color(Зеленый) > 255 || new\_color(Зеленый) < 0 || new\_color(Синий) > 255 || new\_color(Синий) < 0)

throw ERR\_WRONG\_COLOR

color = new\_color

**getColor (&curr\_Color) – взятие текущего цвета кисти**

curr\_color = color

**setBStyle(new\_BStyle) – задание стиля пера**

Если new\_bstyle < 0 || new\_bstyle > 6

throw ERR\_WRONG\_BSTYLE

**getBStyle (&curr\_BStyle) – взятие текущего стиля пера**

curr\_bstyle = bstyle

**setBColor(new\_BColor) – задание цвета пера**

Если (new\_bcolor(Красный) > 255 || new\_bcolor(Красный) < 0 || new\_bcolor(Зеленый) > 255 || new\_bcolor(Зеленый) < 0 || new\_bcolor(Синий) > 255 || new\_bcolor(Синий) < 0)

throw ERR\_WRONG\_BCOLOR

bcolor = new\_bcolor

**getBColor(&curr\_BColor) – взятие текущего цвета пера**

curr\_bcolor = bcolor

**checkSizeTC(\*new\_Cpoints) – проверка вместимости малого треугольника в большой**

Для (i = 0; i < 3; i++)

a = (points[0].x - new\_Cpoints[i].x)\*(points[1].y - points[0].y) - (points[1].x - points[0].x)\*(points[0].y - new\_Cpoints[i].y)

b = (points[1].x - new\_Cpoints[i].x)\*(points[2].y - points[1].y) - (points[2].x - points[1].x)\*(points[1].y - new\_Cpoints[i].y)

c = (points[2].x - new\_Cpoints[i].x)\*(points[0].y - points[2].y) - (points[0].x - points[2].x)\*(points[2].y - new\_Cpoints[i].y)

Если (!((a >= 0 && b >= 0 && c >= 0) || (a <= 0 && b <= 0 && c <= 0)))

throw ERR\_WRONG\_CUT\_POINTS

**checkSizeT(rtx, rty – размер окна) – проверка вместимости треугольника в окно**

Для (i = 0; i < 3; i++)

Если points[i].x < 0 || points[i].y < 0

throw ERR\_NEGATIVE\_POINTS

Или Если (points[i].x > rtx || points[i].y > rty)

throw ERR\_WRONG\_SIZE

**checkSizeT(hdc)**

Взятие размеров окна(WindowFromDC(hdc), &rt)

rtx = rt.right

rty = rt.bottom

checkSizeT(rtx, rty)

**readInf(const char name[] – имя файла для чтения) – чтение данных из файла**

Открытие файла name

Если не открыт, вывод «Can not open input file»; вернуть -1

Чтение mod

Для(i = 0; i < 3; i++) чтение (points[i].x, points[i].y)

Чтение style

Для(i = 0; i < 3; i++) чтение (rgb[i])

color = RGB(rgb[0], rgb[1], rgb[2])

Чтение border

Чтение bstyle

Для(i = 0; i < 3; i++) чтение (brgb[i])

bcolor = RGB(brgb[0], brgb[1], brgb[2])

Если mode == 2

Для(i = 0; i < 3; i++) чтение (Cpoints[i].x, Cpoints[i].y)

try

Задание points классу

Задание style

Задание color

Задание bstyle

Задание bcolor

Задание border

Если mode == 2

Задание Cpoints

catch(исключение)

throw исключение

Закрытие файла

**saveInf(const char name[] – имя файла для сохранения) – запись данных в файл**

Создание файла name

Запись mode в файл

Для (i = 0; i < 3; i++)

Запись (points[i].x, points[i].y)

Запись style

rgb[0] = color(Красный)

rgb[1] = color(Зеленый)

rgb[2] = color(Синий)

Для (i = 0; i < 3; i++)

Запись rgb[i]

Запись border

Запись bstyle

brgb[0] = bcolor(Красный)

brgb[1] = bcolor(Зеленый)

brgb[2] = bcolor(Синий)

Для (i = 0; i < 3; i++)

Запись brgb[i]

Если mode == 2

Для (i = 0; i < 3; i++)

Запись (Cpoints[i].x, Cpoints[i].y)

Закрытие файла

**move\_horizontally(delta - смещение) – сдвиг координат по горизонтали**

Для (i = 0; i < 3; i++)

points[i].x += delta

Если mode == 2

Cpoints[i].x += delta

**move\_vertically(delta) – сдвиг координат по вертикали**

Для (i = 0; i < 3; i++)

points[i].y += delta

Если mode == 2

Cpoints[i].y += delta

**hidecursor() – скрытие приглашения к вводу в консоли**

**ClearConsole() – очищение консоли**

**main() – основная программа**

Задание контекста отображения(hdc)

RECT rt

hOldPen = Взятие стандартного пера(BLACK\_PEN)

hOldBrush = Взятие стандартной кисти(WHITE\_BRUSH)

oldx = 0; oldy = 0

rtx = 0; rty = 0

Triangle triangle = Triangle()

shift = 2

tosleep = false

hidecursor()

try

triangle.readInf(“input.txt”)

triangle.getMode(mode)

c = 0;

Делать {

tosleep = true

Получение размеров окна(hwnd, &rt)

rtx = rt.right

rty = rt.bottom

Если (oldx != rtx || oldy != rty)

system("CLS")

hidecursor()

ClearConsole()

oldx = rtx

oldy = rty

try

Sleep(50)

tosleep = false

triangle.mainDraw(hdc)

catch(исключение)

Вывод(“Increase window size(current is %d : %d)”, rtx, rty)

Если (\_kbhit())

c = \_getch();

Если (c)

49:

triangle.saveInf("output.txt")

Вывод("Preset saved.")

break;

119:

triangle.move\_vertically(-shift);

break;

115:

triangle.move\_vertically(shift);

break;

97:

triangle.move\_horizontally(-shift);

break;

100:

triangle.move\_horizontally(shift);

break;

113:

triangle.move\_vertically(-shift);

triangle.move\_horizontally(-shift);

break;

122:

triangle.move\_vertically(shift);

triangle.move\_horizontally(-shift);

break;

101:

triangle.move\_vertically(-shift);

triangle.move\_horizontally(shift);

break;

99:

triangle.move\_vertically(shift);

triangle.move\_horizontally(shift);

break;

try

if (c==49 || c == 119 || c == 115 || c == 97 || c == 100 || c == 113 || c == 122 || c == 101 || c == 99)

clearConsole()

Sleep(50)

tosleep = false

triangle.mainDraw(hdc)

catch(исключение)

Если (исключение)

ERR\_NEGATIVE\_POINTS:

Вывод(“Negative coords”);

break;

ERR\_WRONG\_SIZE:

Вывод(“Increase window size(current is %d : %d)”, rtx, rty);

break;

default:

throw исключение

Если(tosleep)

sleep(50)

Пока (c != 27);

clearConsole()

catch(исключение)

clearConsole();

Если(исключение)

ERR\_CANNOT\_OPEN\_INPUT\_FILE:

Вывод ("Can not open input file");

break;

ERR\_WRONG\_MODE:

Вывод ("Wrong mode (should be in range [0,2])");

break;

ERR\_NEGATIVE\_POINTS:

Вывод ("Negative coords");

break;

ERR\_POINTS\_ONLINE:

Вывод ("Points belong to line");

break;

ERR\_NEGATIVE\_BORDER:

Вывод ("Negative border value");

break;

ERR\_WRONG\_STYLE:

Вывод ("Wrong brush style value(should be in range[0,6])");

break;

ERR\_WRONG\_BSTYLE:

Вывод ("Wrong pen style value(should be in range[0, 6])");

break;

ERR\_WRONG\_COLOR:

Вывод ("Wrong brush color value (each color should be in range[0,255])");

break;

ERR\_WRONG\_BCOLOR:

Вывод ("Wrong pen color value (each color should be in range[0,255])");

break;

ERR\_WRONG\_CUT\_POINTS:

Вывод ("Second triangle cannot be placed in the first one");

break;

\_getch();

Выбрать перо(hdc, hOldPen);

Выбрать кисть (hdc, hOldBrush);

Освободить контекст отображения(hwnd, hdc);

1. **Текст программы**

**oop2lab.cpp**

#include "external.h"

#include "triangle.h"

void main()

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

RECT rt;

HPEN hOldPen = GetStockPen(BLACK\_PEN);

HBRUSH hOldBrush = GetStockBrush(WHITE\_BRUSH);

int oldx = 0, oldy = 0;

int rtx = 0, rty = 0;

Triangle triangle = Triangle();

int mode;

int shift = 2;

bool tosleep = false;

hidecursor();

try

{

triangle.readInf("input.txt");

triangle.getMode(mode);

char c = 0;

do {

tosleep = true;

GetClientRect(hwnd, &rt);

rtx = rt.right;

rty = rt.bottom;

if (oldx != rtx || oldy != rty)

{

hidecursor();

clearConsole();

oldx = rtx;

oldy = rty;

try

{

Sleep(50);

tosleep = false;

triangle.mainDraw(hdc);

}

catch (int exception)

{

printf\_s("Increase window size(current is %d : %d)", rtx, rty);

}

}

if (\_kbhit()) {

c = \_getch();

switch (c)

{

case 49:

triangle.saveInf("output.txt");

printf\_s("\t\tPreset saved.");

break;

case 119:

triangle.move\_vertically(-shift);

break;

case 115:

triangle.move\_vertically(shift);

break;

case 97:

triangle.move\_horizontally(-shift);

break;

case 100:

triangle.move\_horizontally(shift);

break;

case 113:

triangle.move\_vertically(-shift);

triangle.move\_horizontally(-shift);

break;

case 122:

triangle.move\_vertically(shift);

triangle.move\_horizontally(-shift);

break;

case 101:

triangle.move\_vertically(-shift);

triangle.move\_horizontally(shift);

break;

case 99:

triangle.move\_vertically(shift);

triangle.move\_horizontally(shift);

break;

}

try

{

if (c==49 || c == 119 || c == 115 || c == 97 || c == 100 || c == 113 || c == 122 || c == 101 || c == 99)

{

clearConsole();

Sleep(50);

tosleep = false;

triangle.mainDraw(hdc);

}

}

catch (int exception)

{

switch (exception)

{

case ERR\_NEGATIVE\_POINTS:

printf\_s("Negative coords");

break;

case ERR\_WRONG\_SIZE:

printf\_s("Increase window size(current is %d : %d)", rtx, rty);

break;

default:

throw exception;

}

}

}

if (tosleep)

Sleep(50);

} while (c != 27);

clearConsole();

}

catch (int exception)

{

clearConsole();

switch (exception)

{

case ERR\_CANNOT\_OPEN\_INPUT\_FILE:

printf\_s("Can not open input file");

break;

case ERR\_WRONG\_MODE:

printf\_s("Wrong mode (should be in range [0,2])");

break;

case ERR\_NEGATIVE\_POINTS:

printf\_s("Negative coords");

break;

case ERR\_POINTS\_ONLINE:

printf\_s("Points belong to line");

break;

case ERR\_NEGATIVE\_BORDER:

printf\_s("Negative border value");

break;

case ERR\_WRONG\_STYLE:

printf\_s("Wrong brush style value(should be in range[0,6])");

break;

case ERR\_WRONG\_BSTYLE:

printf\_s("Wrong pen style value(should be in range[0, 6])");

break;

case ERR\_WRONG\_COLOR:

printf\_s("Wrong brush color value (each color should be in range[0,255])");

break;

case ERR\_WRONG\_BCOLOR:

printf\_s("Wrong pen color value (each color should be in range[0,255])");

break;

case ERR\_WRONG\_CUT\_POINTS:

printf\_s("Second triangle cannot be placed in the first one");

break;

}

\_getch();

}

SelectPen(hdc, hOldPen);

SelectBrush(hdc, hOldBrush);

ReleaseDC(hwnd, hdc);

}

**triangle.h**

#ifndef TRIANGLE\_H

#define TRIANGLE\_H

#define ERR\_CANNOT\_OPEN\_INPUT\_FILE -2

#define ERR\_WRONG\_SIZE -1

#define ERR\_NEGATIVE\_POINTS 0

#define ERR\_POINTS\_ONLINE 1

#define ERR\_NEGATIVE\_BORDER 2

#define ERR\_WRONG\_STYLE 3

#define ERR\_WRONG\_BSTYLE 4

#define ERR\_WRONG\_COLOR 5

#define ERR\_WRONG\_BCOLOR 6

#define ERR\_WRONG\_CUT\_POINTS 7

#define ERR\_WRONG\_MODE 8

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <conio.h>

class Triangle {

private:

POINT points[3], Cpoints[3];

int style, bstyle, border, mode;

COLORREF color, bcolor;

public:

void drawTriangle(HDC hdc); //Рисование треугольника

void drawCounter(HDC hdc); //Контура

void drawCut(HDC hdc); //Треугольника с вырезом

void mainDraw(HDC hdc); //Рисование фигуры

void setMode(int new\_mode); //Тип фигуры

void getMode(int &curr\_mode);

void setPoints(POINT \*new\_points); //Точки треугольника

void getPoints(POINT \*curr\_points);

void setCPoints(POINT \*new\_Cpoints); //Точки треугольника для выреза

void getCPoints(POINT \*curr\_Cpoints);

void setBorder(int new\_border); //Толщина контура

void getBorder(int &curr\_border);

void setStyle(int new\_style); //Стиль пера(контура)

void getStyle(int &curr\_style);

void setColor(COLORREF new\_tcolor); //Цвет пера(контура)

void getColor(COLORREF &curr\_tcolor);

void setBStyle(int new\_bstyle); //Стиль кисти (заливки)

void getBStyle(int &curr\_bstyle);

void setBColor(COLORREF new\_bcolor); //Цвет кисти(заливки)

void getBColor(COLORREF &curr\_bcolor);

void checkSizeTC(POINT \*new\_Cpoints); //Входит ли вырез в треугольник

void checkSizeT( int rtx, int rty); //Входит ли треугольник в окно

void checkSizeT( HDC hdc ); //Входит ли треугольник в окно

void readInf(const char name[]); //Ввод треугольника из файла

void saveInf(const char name[]); //Вывод треугольника в файл

void move\_horizontally(int delta); //Сдвиг фигуры по горизонтали

void move\_vertically(int delta); //Сдвиг фигуры по вертикали

};

#endif

**triangle.cpp**

#include "triangle.h"

void Triangle::drawTriangle(HDC hdc) {

checkSizeT(hdc);

COLORREF buffcolor = bcolor;

if (border == 0)

buffcolor = color;

HPEN hMyPen = CreatePen(bstyle, border, buffcolor);

LOGBRUSH\* logBrush = new LOGBRUSH;

logBrush->lbStyle = (style == 0) ? BS\_SOLID : BS\_HATCHED;

logBrush->lbHatch = style - 1;

logBrush->lbColor = color;

HBRUSH hMyBrush = CreateBrushIndirect(logBrush);

SelectPen(hdc, hMyPen);

SelectBrush(hdc, hMyBrush);

Polygon(hdc, points, 3);

DeletePen(hMyPen);

delete logBrush;

DeleteBrush(hMyBrush);

}

void Triangle::drawCounter(HDC hdc) {

checkSizeT(hdc);

HPEN hMyPen = CreatePen(bstyle, border, bcolor);

SelectPen(hdc, hMyPen);

POINT ppt[4] = { points[0],points[1],points[2],points[0] };

Polyline(hdc, ppt, 4);

DeletePen(hMyPen);

}

void Triangle::drawCut(HDC hdc) {

checkSizeT(hdc);

HPEN hMyPen = CreatePen(0, 0, RGB(0, 0, 0));

HBRUSH hMyBrush = CreateSolidBrush(RGB(0, 0, 0));

SelectPen(hdc, hMyPen);

SelectBrush(hdc, hMyBrush);

Polygon(hdc, Cpoints, 3);

DeletePen(hMyPen);

DeleteBrush(hMyBrush);

}

void Triangle::mainDraw(HDC hdc) {

switch (mode)

{

case 0:

drawTriangle(hdc);

break;

case 1:

drawCounter(hdc);

break;

case 2:

drawTriangle(hdc);

drawCut(hdc);

break;

}

}

void Triangle::setMode(int new\_mode) {

if (mode < 0 || mode > 2) throw ERR\_WRONG\_MODE;

mode = new\_mode;

}

void Triangle::getMode(int &curr\_mode) {

curr\_mode = mode;

}

void Triangle::setPoints(POINT \*new\_points) {

for (int i = 0; i < 3; i++)

if (new\_points[i].x < 0 || new\_points[i].y < 0)

throw ERR\_NEGATIVE\_POINTS;

if ((new\_points[2].x - new\_points[0].x)\*(new\_points[1].y - new\_points[0].y) == (new\_points[2].y - new\_points[0].y)\*(new\_points[1].x - new\_points[0].x))

throw ERR\_POINTS\_ONLINE;

for (int i = 0; i < 3; i++)

points[i] = new\_points[i];

}

void Triangle::getPoints(POINT \*curr\_points) {

for (int i = 0; i < 3; i++) curr\_points[i] = points[i];

}

void Triangle::setCPoints(POINT \*new\_Cpoints) {

checkSizeTC(new\_Cpoints);

for (int i = 0; i < 3; i++)

if (new\_Cpoints[i].x < 0 || new\_Cpoints[i].y < 0)

throw ERR\_NEGATIVE\_POINTS;

if ((new\_Cpoints[2].x - new\_Cpoints[0].x)\*(new\_Cpoints[1].y - new\_Cpoints[0].y) == (new\_Cpoints[2].y - new\_Cpoints[0].y)\*(new\_Cpoints[1].x - new\_Cpoints[0].x))

throw ERR\_POINTS\_ONLINE;

for (int i = 0; i < 3; i++)

Cpoints[i] = new\_Cpoints[i];

}

void Triangle::getCPoints(POINT \*curr\_Cpoints) {

for (int i = 0; i < 3; i++) curr\_Cpoints[i] = Cpoints[i];

}

void Triangle::setBorder(int new\_border) {

if (new\_border < 0) throw ERR\_NEGATIVE\_BORDER;

border = new\_border;

}

void Triangle::getBorder(int &curr\_border) {

curr\_border = border;

}

void Triangle::setStyle(int new\_style) {

if (new\_style < 0 || new\_style > 6) throw ERR\_WRONG\_STYLE;

style = new\_style;

}

void Triangle::getStyle(int &curr\_style) {

curr\_style = style;

}

void Triangle::setColor(COLORREF new\_color) {

if (GetRValue(new\_color) > 255 || GetRValue(new\_color) < 0 || GetGValue(new\_color) > 255 || GetGValue(new\_color) < 0 || GetBValue(new\_color) > 255 || GetBValue(new\_color) < 0) throw ERR\_WRONG\_COLOR;

color = new\_color;

}

void Triangle::getColor(COLORREF &curr\_color) {

curr\_color = color;

}

void Triangle::setBStyle(int new\_bstyle) {

if (new\_bstyle < 0 || new\_bstyle > 6) throw ERR\_WRONG\_BSTYLE;

bstyle = new\_bstyle;

}

void Triangle::getBStyle(int &curr\_bstyle) {

curr\_bstyle = bstyle;

}

void Triangle::setBColor(COLORREF new\_bcolor) {

if (GetRValue(new\_bcolor) > 255 || GetRValue(new\_bcolor) < 0 || GetGValue(new\_bcolor) > 255 || GetGValue(new\_bcolor) < 0 || GetBValue(new\_bcolor) > 255 || GetBValue(new\_bcolor) < 0) throw ERR\_WRONG\_BCOLOR;

bcolor = new\_bcolor;

}

void Triangle::getBColor(COLORREF &curr\_bcolor) {

curr\_bcolor = bcolor;

}

void Triangle::checkSizeTC(POINT \*new\_Cpoints) {

int a, b, c;

for (int i = 0; i < 3; i++) {

a = (points[0].x - new\_Cpoints[i].x)\*(points[1].y - points[0].y) - (points[1].x - points[0].x)\*(points[0].y - new\_Cpoints[i].y);

b = (points[1].x - new\_Cpoints[i].x)\*(points[2].y - points[1].y) - (points[2].x - points[1].x)\*(points[1].y - new\_Cpoints[i].y);

c = (points[2].x - new\_Cpoints[i].x)\*(points[0].y - points[2].y) - (points[0].x - points[2].x)\*(points[2].y - new\_Cpoints[i].y);

if (!((a >= 0 && b >= 0 && c >= 0) || (a <= 0 && b <= 0 && c <= 0)))

throw ERR\_WRONG\_CUT\_POINTS;

}

}

void Triangle::checkSizeT(int rtx, int rty) {

for (int i = 0; i < 3; i++)

if (points[i].x < 0 || points[i].y < 0)

throw ERR\_NEGATIVE\_POINTS;

else if (points[i].x > rtx || points[i].y > rty)

throw ERR\_WRONG\_SIZE;

}

void Triangle::checkSizeT(HDC hdc) {

RECT rt;

GetClientRect(WindowFromDC(hdc), &rt);

int rtx = rt.right;

int rty = rt.bottom;

checkSizeT(rtx, rty);

}

void Triangle::readInf(const char name[]) {

FILE \*f;

fopen\_s(&f, name, "rt");

if (f == NULL)

throw ERR\_CANNOT\_OPEN\_INPUT\_FILE;

int brgb[3], rgb[3], style, bstyle, border;

COLORREF bcolor, color;

POINT points[3], Cpoints[3];

fscanf\_s(f, "%d", &mode); //mode

for (int i = 0; i < 3; i++)

fscanf\_s(f, "%d%d", &(points[i].x), &(points[i].y)); //points

fscanf\_s(f, "%d", &(style)); //style

for (int i = 0; i < 3; i++) //color

fscanf\_s(f, "%d", &(rgb[i]));

color = RGB(rgb[0], rgb[1], rgb[2]);

fscanf\_s(f, "%d", &(bstyle)); //bstyle

for (int i = 0; i < 3; i++) //bcolor

fscanf\_s(f, "%d", &(brgb[i]));

bcolor = RGB(brgb[0], brgb[1], brgb[2]);

fscanf\_s(f, "%d", &(border)); //border

if (mode == 2)

for (int i = 0; i < 3; i++)

fscanf\_s(f, "%d%d", &(Cpoints[i].x), &(Cpoints[i].y));

try

{

setPoints(points);

setStyle(style);

setColor(color);

setBStyle(bstyle);

setBColor(bcolor);

setBorder(border);

if (mode == 2)

setCPoints(Cpoints);

}

catch (int exception)

{

throw exception;

}

fclose(f);

}

void Triangle::saveInf(const char name[]) {

FILE \*f;

fopen\_s(&f, name, "w");

int brgb[3], rgb[3];

fprintf\_s(f, "%d\n\n", mode); //mode

for (int i = 0; i < 3; i++)

fprintf\_s(f, "%d %d\n", points[i].x, points[i].y); //points

fprintf\_s(f, "\n%d\n", style); //style

rgb[0] = GetRValue(color);

rgb[1] = GetGValue(color);

rgb[2] = GetBValue(color);

for (int i = 0; i < 3; i++) //color

fprintf\_s(f, "%d ", rgb[i]);

fprintf\_s(f, "\n\n%d\n", bstyle); //bstyle

brgb[0] = GetRValue(bcolor);

brgb[1] = GetGValue(bcolor);

brgb[2] = GetBValue(bcolor);

for (int i = 0; i < 3; i++) //bcolor

fprintf\_s(f, "%d ", brgb[i]);

fprintf\_s(f, "\n\n%d\n\n", border); //border

if (mode == 2)

for (int i = 0; i < 3; i++) //Cpoints(ToCut)

fprintf\_s(f, "%d %d\n", Cpoints[i].x, Cpoints[i].y);

fclose(f);

}

void Triangle::move\_horizontally(int delta) {

for (int i = 0; i < 3; i++) {

points[i].x += delta;

if (mode == 2)

Cpoints[i].x += delta;

}

}

void Triangle::move\_vertically(int delta) {

for (int i = 0; i < 3; i++) {

points[i].y += delta;

if (mode == 2)

Cpoints[i].y += delta;

}

}

**external.h**

#ifndef EXTERNAL\_H

#define EXTERNAL\_H

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <conio.h>

#include "triangle.h"

void hidecursor();

void clearConsole();

#endif

**external.cpp**

#include "external.h"

#include "triangle.h"

void hidecursor()

{

HANDLE consoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO info;

info.dwSize = 25;

info.bVisible = FALSE;

SetConsoleCursorInfo(consoleHandle, &info);

}

void clearConsole() {

system("cls");

system("cls");

}

1. **Тесты**

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Пояснение** | **Результат** |
| 0  100 200  300 500  600 500  0  255 0 0  1 0  255 255 0 | Корректные входные данные.  Рисование закрашенного треугольника.  Изображение появляется после расширения окна. |  |
| 1  100 100  300 250  600 280  0  255 0 0  1 0  255 255 0 | Корректные входные данные.  Рисование контура треугольника. |  |
| 2  100 200  300 500  600 500  0  0 128 192  1 0  255 255 0  300 440  250 350  500 485 | Корректные входные данные.  Рисование треугольника с вырезом. |  |
| 0  100 200  100 500  100 400  0  0 128 192  1 0  255 255 0 | Некорректные входные данные.  Вершины треугольника лежат на одной прямой. |  |
| 2  100 200  300 500  600 500  0  255 0 0  1 0  255 255 0  50 440  250 40  500 485 | Некорректные входные данные.  Вырез частично или полностью не попадает в область заданного треугольника |  |
| 0  -10 20  30 50  60 50  0  255 0 0  0  255 255 0  1  300 440  250 350  500 485 | Некорректные входные данные.  Точка треугольника имеет отрицательную координату |  |