# 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 № 3 по дисциплине «Объектно-ориентированное программирование»

**Объекты**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| сигма градиент синий1 | Факультет: | ПМИ |  |  |
| Группа: | ПМИ-81 |  |  |
| Студенты: | Демидович Е.Ю.,  Муравьев М.И. |  |  |
| Вариант: | 7;8а |  |  |
| Преподаватель: | Еланцева И.Л., Лисицин Д.В. |  |  |

Новосибирск

2019

1. **Постановка задачи**

Разработать класс-контейнер – «открытый» стек. хранящий объекты класса треугольник.

Разработать следующие функции-члены класса:

Конструктор;

Деструктор;

Функции для помещения объектов-фигур в контейнер;

Функции для возвращения (удаления) объектов-фигур из контейнера

Поиска в контейнере;

«Распечатки содержимого» контейнера – вывода графических образов содержащихся в объекте-контейнере объектов;

Функции, обеспечивающие сохраняемость контейнера с использованием файла.

Реализацию класса-контейнера поместить в отдельный файл. Разработать функцию, демонстрирующую поведение объекта-контейнера с несколькими объектами-фигурами.

1. **Описание разработанного класса**

**класс Triangle:**

**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

**Функции класса:**

void drawTriangle(HDC hdc); //Рисование треугольника

void drawCounter(HDC hdc); //Контура

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

void mainDraw(HWND hwnd); //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

**Класс stack:**

**Атрибуты класса:**

Triangle \*arr – динамический массив, хранящий объекты-фигуры класса Triangle

int top – указатель на вершину стека;

int n – максимальное количество элементов в стеке.

**Функции класса:**

Stack(int N) – конструктор класса;

~Stack() – деструктор класса;

void Push(Triangle d) – помещение элемента в стек;

void Print(HWND hwnd) – распечатка стека;

Triangle Pop() – взятие(удаление) элемента из стека;

Triangle find(int \*key) – поиск элемента в стеке;

void Save(const char name[]) – сохранение содержимого стека в файл;

bool isEmpty() – проверка стека на пустоту;

bool isFull() – проверка стека на переполненность;

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(hwnd – идентификатор окна) – Основная функция рисования**

Получить контекст отображения (hwnd, hdc)

int oldx = 0, oldy = 0;

int rtx = 0, rty = 0;

char c = 0;

Делать

Получить размер контекста отображения (hdc, rt)

rtx = rt.right;

rty = rt.bottom;

Если (oldx != rtx || oldy != rty)

Очистить консоль

oldx = rtx;

oldy = rty;

Пытаться

Если (mode)

0:

drawTriangle(hdc);

break;

1:

drawCounter(hdc);

break;

2:

drawTriangle(hdc);

drawCut(hdc);

break;

Поймать (исключение)

Вывод("Increase window size(current is rtx : rty”)

Если (Нажата клавиша) {

c = Код клавиши;

}

} Пока (c != 13);

Освободить контекст отображения(hwnd, hdc);

**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**

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**

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**

curr\_border = border

**setStyle(new\_Style) – Стиль пера**

Если new\_style < 0 || new\_style > 6

throw ERR\_WRONG\_STYLE

**getStyle (&curr\_Style)**

curr\_style = style

**setColor(new\_Color) – Цвет пера**

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)**

curr\_color = color

**setBStyle(new\_BStyle) – Стиль кисти**

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

throw ERR\_WRONG\_BSTYLE

**getBStyle (&curr\_BStyle)**

curr\_bstyle = bstyle

**setBColor(new\_BColor) – Цвет кисти**

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)**

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 (color[i])

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

Чтение bstyle

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

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

Чтение border

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 bstyle

brgb[0] = bcolor(Красный)

brgb[1] = bcolor(Зеленый)

brgb[2] = bcolor(Синий)

Для (i = 0; i < 3; i++)

Запись brgb[i]

Запись border

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**

**Класс stack**

**Открытая часть:**

Массив объектов-фигур arr

Целочисленная top

Целочисленная n

**Закрытая часть:**

**Stack(N – размер стека)**

top = -1

n = N

Выделение памяти для arr под N-ое количество объектов Triangle.

**~Stack()**

Очищение памяти, выделенной под arr.

**void Push(d – элемент – объект-фигура)**

Если (isFull()) кинуть ERR\_FULL

top++

arr[top] = d

**void Print(hwnd – идентификатор окна)**

Если (isEmpty()) кинуть ERR\_EMPTY

temp\_arr = arr

temp\_top = top

Пока(temp\_top != -1)

temp\_arr[temp\_top].mainDraw(hwnd)

temp\_top—

**Triangle Pop()**

Если (isEmpty()) кинуть ERR\_EMPTY

top—

Вернуть arr[top+1]

**Triangle find(key – значение ключа для поиска)**

Если (isEmpty()) кинуть ERR\_EMPTY

temp\_arr = arr

temp\_top = top

Цветовой массив color, t\_color

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

temp\_arr[temp\_top].getColor(t\_color)

Пока (t\_color != color)

Если (temp\_top == -1) кинуть ERR\_NOT\_FOUND

temp\_top—

temp\_arr[temp\_top].getColor(t\_color)

Вернуть temp\_arr[temp\_top]

**void Save(const char name[] – имя файла для сохранения)**

temp\_arr = arr

temp\_top = top

Пока(temp\_top != -1)

temp\_arr[temp\_top].saveInf(name)

temp\_top—

**bool isEmpty()**

Если (top == -1) вернуть true

вернуть false

**bool isFull()**

Если (top == n - 1) вернуть true

вернуть false

**main():**

Взятие идентификатора окна hwnd

Вывод("Menu:\n1 - Push triangle into stack from buffer\n2 - Pop triangle from stack into buffer\n3 - Print stack(push Enter for every element)\n4 - Find in stack by color and buffer it\n5 - Save stack into file\n6 - Image triangle in buffer\n7 - Load triangle from file\n")

Делать

Пытаться

Если (Нажата клавиша)

Очистить консоль

Вывод ("Menu:\n1 - Push triangle into stack from buffer\n2 - Pop triangle from stack into buffer\n3 - Print stack(push Enter for every element)\n4 - Find in stack by color and buffer it\n5 - Save stack into file\n6 - Image triangle in buffer\n7 - Load triangle from file\n")

c = код клавиши

Если (c)

49:

stack.Push(triangle)

Вывод ("Element pushed successfully!\n")

break

50:

triangle = stack.Pop()

Вывод ("Element popped successfully!\n")

break

51:

Очистить консоль

stack.Print(hwnd)

break

52:

Вывод ("Set color in format R G B : \n")

Для (i = 0; i < 3; i++) Ввод("%d", &color[i])

triangle = stack.find(color)

Вывод ("Triangle found and buffered!\n")

break

53:

Открыть файл("stackout.txt”, ”w”)

Закрыть файл(f)

stack.Save("stackout.txt")

Вывод ("Your stack saved into stackout.txt\n")

break

54:

Очистить консоль

triangle.mainDraw(hwnd)

clearConsole()

break

55:

triangle.readInf("input.txt")

Вывод ("Triangle loaded successfully!\n")

break

Поймать (исключение)

Если (исключение)

ERR\_EMPTY:

Вывод ("Can't pop/print any element! Your stack is empty.\n")

break

ERR\_FULL:

Вывод ("Can't add an element! Your stack is full.\n")

break

ERR\_NOT\_FOUND:

Вывод ("There is no triangle with such color.\n")

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

Пока (c != 27)

1. **Текст программы**

**oop3lab.cpp**

#include "external.h"

#include "triangle.h"

#include "stack.h"

#define N 30

void main()

{

HWND hwnd = GetConsoleWindow();

Stack stack = Stack(N);

Triangle triangle = Triangle();

char c = 0;

printf\_s("Menu:\n1 - Push triangle into stack from buffer\n2 - Pop triangle from stack into buffer\n3 - Print stack(push Enter for every element)\n4 - Find in stack by color and buffer it\n5 - Save stack into file\n6 - Image triangle in buffer\n7 - Load triangle from file\n");

do

{

try

{

if (\_kbhit()) {

clearConsole();

printf\_s("Menu:\n1 - Push triangle into stack from buffer\n2 - Pop triangle from stack into buffer\n3 - Print stack(push Enter for every element)\n4 - Find in stack by color and buffer it\n5 - Save stack into file\n6 - Image triangle in buffer\n7 - Load triangle from file\n");

c = \_getch();

switch (c)

{

case 49: //1 - push

stack.Push(triangle);

printf\_s("Element pushed successfully!\n");

break;

case 50: //2 - pop

triangle = stack.Pop();

printf\_s("Element popped successfully!\n");

break;

case 51: //3 - print

clearConsole();

stack.Print(hwnd);

break;

case 52: //4 - find

int color[3];

printf\_s("Set color in format R G B : \n");

for (int i = 0; i < 3; i++) scanf\_s("%d", &color[i]);

triangle = stack.find(color);

printf\_s("Triangle found and buffered!\n");

break;

case 53: //5 - save

FILE \*f;

fopen\_s(&f, "stackout.txt", "w");

fclose(f);

stack.Save("stackout.txt");

printf\_s("Your stack saved into stackout.txt\n");

break;

case 54: //6 - рисование текущего треугольника

clearConsole();

triangle.mainDraw(hwnd);

clearConsole();

break;

case 55: //7 - загрузить треугольник из файла

triangle.readInf("input.txt");

printf\_s("Triangle loaded successfully!\n");

break;

}

}

}

catch (int exception)

{

switch (exception)

{

case ERR\_EMPTY:

printf\_s("Can't pop/print any element! Your stack is empty.\n");

break;

case ERR\_FULL:

printf\_s("Can't add an element! Your stack is full.\n");

break;

case ERR\_NOT\_FOUND:

printf\_s("There is no triangle with such color.\n");

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;

}

}

} while (c != 27);

}

**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(HWND hwnd); //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(HWND hwnd) {

HDC hdc = GetDC(hwnd);

RECT rt;

int oldx = 0, oldy = 0;

int rtx = 0, rty = 0;

char c = 0;

do {

GetClientRect(hwnd, &rt);

rtx = rt.right;

rty = rt.bottom;

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

{

system("cls");

oldx = rtx;

oldy = rty;

try

{

switch (mode)

{

case 0:

drawTriangle(hdc);

break;

case 1:

drawCounter(hdc);

break;

case 2:

drawTriangle(hdc);

drawCut(hdc);

break;

}

}

catch (int exception)

{

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

}

}

if (\_kbhit()) {

c = \_getch();

}

} while (c != 13);

ReleaseDC(hwnd, hdc);

}

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, "at");

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;

}

}

**stack.h**

#ifndef STACK\_H

#define STACK\_H

#define ERR\_FULL 10

#define ERR\_EMPTY 20

#define ERR\_NOT\_FOUND 30

#include "Triangle.h"

class Stack {

private:

Triangle \*arr;

int top;

int n;

public:

Stack(int N);

~Stack();

void Push(Triangle d);

void Print(HWND hwnd);

Triangle Pop();

Triangle find(int \*key);

void Save(const char name[]);

bool isEmpty();

bool isFull();

};

#endif

**stack.cpp**

#include "stack.h"

Stack::Stack(int N) {

top = -1;

n = N;

arr = new Triangle[N];

}

Stack::~Stack() {

delete[] arr;

}

void Stack::Push(Triangle d) {

if (isFull()) throw ERR\_FULL;

top++;

arr[top] = d;

}

Triangle Stack::Pop() {

if (isEmpty()) throw ERR\_EMPTY;

top--;

return arr[top + 1];

}

void Stack::Print(HWND hwnd) {

if (isEmpty()) throw ERR\_EMPTY;

Triangle \*temp\_arr = arr;

int temp\_top = top;

while (temp\_top != -1) {

temp\_arr[temp\_top].mainDraw(hwnd);

temp\_top--;

}

}

Triangle Stack::find(int \*key) {

if (isEmpty()) throw ERR\_EMPTY;

Triangle \*temp\_arr = arr;

int temp\_top = top;

COLORREF color, t\_color;

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

temp\_arr[temp\_top].getColor(t\_color);

while (t\_color != color) {

if (temp\_top == -1) throw ERR\_NOT\_FOUND;

temp\_top--;

temp\_arr[temp\_top].getColor(t\_color);

}

return temp\_arr[temp\_top];

}

void Stack::Save(const char name[]) {

Triangle \*temp\_arr = arr;

int temp\_top = top;

while (temp\_top != -1) {

temp\_arr[temp\_top].saveInf(name);

temp\_top--;

}

}

bool Stack::isEmpty() {

if (top == -1) return true;

return false;

}

bool Stack::isFull() {

if (top == n - 1) return true;

return false;

}

**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** |
| Содержание стека  0  100 20  30 50  60 50  0  255 0 0  0  255 255 0  1  0  10 20  30 50  60 50  0  255 0 255  0  255 255 0  1  Поиск цвета  1)255 0 0  2)255 255 0  3)1 1 1  Последовательность нажатий  7 1 7 1  1)4 6 ENTER ENTER  2)4 6 ENTER ENTER  3)4 | Тест поиска  Корректные входные данные.  Поиск по трем разным данным:  1)Корректные  2)Корректные  3)Некорректные |  |
| Содержание стека  0  100 20  30 50  60 50  0  255 0 0  0  255 255 0  1  0  10 20  30 50  60 50  0  255 0 255  0  255 255 0  1  Последовательность нажатий  7 1 7 1  3 ENTER ENTER  2 3 ENTER ENTER | Корректные входные данные.  Распечатка стека  ДО взятия и  ПОСЛЕ взятия одного элемента |  |
| Содержание стека  **-**  Последовательность нажатий  1)2/3  2)6  3)4 | 1)Взятие элемента из пустого стека  ИЛИ  Распечатка содержимого стека  2)Рисование треугольника из пустого буфера  (Будет выведено пустое изображение) |  |
| Содержание стека  0  100 20  30 50  60 50  0  255 0 0  0  255 255 0  1  0  10 20  30 50  60 50  0  255 0 255  0  255 255 0  1  Последовательность нажатий  7 1 7 1  5 | Проверка сохраняемости содержимого стека в файле. |  |