# 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 № 5 по дисциплине «Объектно-ориентированное программирование»

**Шаблоны**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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**

Путем модификации программ, разработанных в лабораторных работах № 3, 4, разработать шаблон контейнера для хранения объектов классов, реализующих геометрические фигуры.

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

1. **Описание разработанного класса**

**Класс Triangle - "Фигура"**

**Открытая часть:**

Чисто виртуальная функция Draw(HDC hdc) - функция рисования

Чисто виртуальная функция load(const char name[]) - функция чтения из файла

Чисто виртуальная функция save(const char name[]) - функция сохранения в файл

**Класс brush - "Закрашенный"**

**Защищенная часть:**

Массив цветов bcolor - цвет закраски

Целочисленная bstyle - стиль закраски

**Открытая часть:**

Конструктор brush()

setBStyle(int new\_bstyle) - задание стиля закраски

getBStyle(int &curr\_bstyle) - взятие стиля закраски

setBColor(COLORREF new\_bcolor) - взятие цвета закраски

getBColor(COLORREF &curr\_bcolor) - задание цвета закраски

save(const char name[])

load(const char name[])

**Класс contour - "Фигура-контур" - наследник Triangle**

**Защищенная часть:**

Массив цветов color – цвет контура

int style, border – стиль и толщина контура

POINT points[3] – массив координат контура

**Открытая часть:**

Конструктор contour()

setStyle(int new\_bstyle) - задание стиля пера

getStyle(int &curr\_bstyle) - взятие стиля пера

setColor(COLORREF new\_bcolor) задание цвета пера

getColor(COLORREF &curr\_bcolor) - взятие цвета пера

setBorder(int new\_border) - задание ширины контура

getBorder(int &curr\_border) - взятие ширины контура

setPoints(POINT \*new\_points) - задание точек фигуры-контура

getPoints(POINT \*curr\_points) - взятие точек фигуры-контура

Draw(HDC hdc) - функция отрисовки фигуры-контура

save(const char name[]);

load(const char name[]);

checkSizeT(HDC hdc) - проверка вместимости фигуры в текущее окно

**Класс painted - "Закрашенная фигура" - наследник brush и contour**

**Открытая часть:**

Конструктор painted()

Конструктор painted(brush Brush, contour Contour)

save(const char name[]);

load(const char name1[], const char name2[]);

Draw(HDC hdc) - функция отрисовки закрашенной фигуры

**Класс combined - "Комбинированная фигура"**

**Закрытая часть:**

painted ext – внешний треугольник

painted intern – внутренний треугольник

**Открытая часть:**

Конструктор combined()

addExt(painted Ext) – добавление внешнего треугольника

save(const char name[]);

load(const char name1[], const char name2[]);

Draw(HDC hdc) - функция отрисовки комбинированной фигуры

checkSizeTC(POINT \*new\_Cpoints) - проверка вместимости второй фигуры в первую

getIntern(painted &t) – получение данных о внутреннем треугольнике

getExt(painted &t) – получение данных о внешнем треугольнике

**Шаблонный класс Stack с параметром T – «Открытый» стек**

**Закрытая часть:**

T \*arr – массив из элементов типа T

int top – указатель на вершину

int n – размер стека

**Открытая часть:**

Конструктор Stack(int N)

Деструктор ~Stack()

Push(T d) – добавление элемента в стек

Pop() – взятие элемента из стека

Print(HDC hdc) – распечатка содержимого стека

fint(int \*key) – поиск в стеке по цвету контура треугольника

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

isEmpty() – проверка стека на пустоту

isFull() – проверка стека на полноту

Специализированная функция find(int \*key) для класса combined – поиск в стеке по цвету закраски внутреннего треугольника

1. **Алгоритм**

**Класс brush:**

**brush(): bcolor(0), bstyle(0) {}**

**void setBStyle(new\_bstyle):**

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

throw ERR\_WRONG\_BSTYLE

bstyle = new\_bstyle

**void getBStyle (&curr\_bstyle):**

curr\_bstyle = bstyle

**void setBColor(new\_bcolor):**

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

**void getBColor(&curr\_bcolor):**

curr\_bcolor = bcolor

**void save(const char name[]):**

Создание файла name

Запись bstyle

brgb[0] = bcolor(Красный)

brgb[1] = bcolor(Зеленый)

brgb[2] = bcolor(Синий)

Для (i = 0; i < 3; i++)

Запись brgb[i]

Закрытие файла

**void load(const char name[]):**

Открытие файла name

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

Чтение bstyle

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

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

try

Задание bstyle

Задание bcolor

catch(исключение)

throw исключение

Закрытие файла

**Класс contour:**

**contour(): color(0), style(0), border(0), points{0, 0, 0} {}**

**void setStyle(new\_Style):**

Если new\_style < 0 || new\_style > 6

throw ERR\_WRONG\_STYLE

style = new\_Style

**void getStyle (&curr\_Style):**

curr\_style = style

**void setColor(new\_Color):**

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

**void getColor (&curr\_Color):**

curr\_color = color

**void setBorder(new\_border):**

Если new\_border < 0

throw ERR\_NEGATIVE\_BORDER

border = new\_border

**void getBorder(&curr\_border):**

curr\_border = border

**void setPoints(\*new\_points – массив координат):**

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]

**void getPoints(\*curr\_points):**

Для(i = 0; i < 3; i++)

curr\_points[i] = points[i]

**void Draw(hdc):**

Вызов chechSizeT(hdc)

hMyPen = Создание пера(style, border, color)

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

Построение линии(hdc, ppt, 4);

Удаление пера(hMyPen);

**void save(const char name[]):**

Создание файла name

Для (i = 0; i < 3; i++)

Запись (points[i].x, points[i].y)

Запись style

Запись border

rgb[0] = color(Красный)

rgb[1] = color(Зеленый)

rgb[2] = color(Синий)

Для (i = 0; i < 3; i++)

Запись rgb[i]

Закрытие файла

**void load(const char name[]):**

Открытие файла name

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

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

Чтение style

Чтение border

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

try

Задание points классу

Задание style

Задание color

Задание border

catch(исключение)

throw исключение

Закрытие файла

**void checkSizeT(hdc):**

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

rtx = rt.right

rty = rt.bottom

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

**Класс painted:**

**painted(): {}**

**painted(brush Brush, contour Contour):**

Brush.getBColor(текущий объект класса->bcolor)

Brush.getBStyle(текущий объект класса ->bstyle)

Contour.getPoints(текущий объект класса ->points)

Contour.getBorder(текущий объект класса ->border)

Contour.getColor(текущий объект класса ->color)

Contour.getStyle(текущий объект класса ->style)

**void Draw(HDC hdc) - функция отрисовки закрашенной фигуры**

Вызов checkSizeT(hdc)

hMyPen = Создание пера(style, border, color)

hMyBrush = Создание кисти(bstyle, bstyle-1, bcolor)

Выбор пера(hdc, hMyPen)

Выбор кисти(hdc, hMyBrush)

Построение фигуры(hdc, points, 3);

Удаление пера(hMyPen);

Удаление кисти(hMyBrush);

**void save(const char name[]):**

Фунцкия класса brush::save(name)

Функция класса contour::save(name)

**void load(const char name1[], const char name2[]):**

Функция класса brush::load(name1)

Функция класса contour::load(name2)

**Класс combined:**

**combined() {}**

**void Draw(HDC hdc):**

Вызов checkSizeT(hdc) для атрибута ext

Вызов checkSizeTC(hdc)

Вызов функции отрисовки Draw(hdc) для ext

Вызов функции отрисовки Draw(hdc) для intern

**void save(const char name[]):**

Вызов save(name) для intern из класса brush

Вызов save(name) для intern из класса contour

**void load(const char name1[], const char name2[]):**

Вызов load(name1) для intern из класса brush

Вызов load(name2) для intern из класса contour

**void addExt(painted Ext):**

ext = Ext

**void getIntern(painted &t):**

t = intern

**void getExt(painted &t):**

t = ext

**void checkSizeTC(\*new\_Cpoints):**

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

**Класс Stack:**

**Stack(int N – задаваемый размер стека)**

top = -1

n = N

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

**~Stack()**

Очищение памяти, выделенной под arr.

**void Push(T d)**

Если (isFull()) кинуть ERR\_FULL

top++

arr[top] = d

**T Pop()**

Если (isEmpty()) кинуть ERR\_EMPTY

top—

Вернуть arr[top+1]

**void Print(HDC hdc)**

Если (isEmpty()) кинуть ERR\_EMPTY

temp\_arr = arr

temp\_top = top

Пока(temp\_top != -1)

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

temp\_top—

**T find(int \*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[])**

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

**Stack<combined>::find(int \*key):**

Если (isEmpty()) кинуть ERR\_EMPTY

combined \*temp\_arr = arr

temp\_top = top

Цветовой массив color, t\_color

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

temp\_arr[temp\_top].getIntern(temp)

temp.getColor(t\_color)

Пока (t\_color != color)

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

temp\_arr[temp\_top].getIntern(temp)

temp.getColor(t\_color)

Вернуть temp\_arr[temp\_top]

**main():**

Взятие идентификатора окна hwnd

Вывод("Menu:1 - Push into stack 2 - Pop from stack 3 - Print stack 4 - Find figure in stack by color 5 - Save stack into file 6 - Draw figure in buffer 7 - Load figure into buffer from file Esc - Exit")

Делать

Пытаться

Если (Нажата клавиша)

Очистить консоль

Вывод ("Menu:1 - Push into stack 2 - Pop from stack 3 - Print stack 4 - Find figure in stack by color 5 - Save stack into file 6 - Draw figure in buffer 7 - Load figure into buffer from file Esc - Exit")

c = код клавиши

Если (c)

49:

Очистить консоль

Вывод ("Stack for 1 – Contours 2 - Painted triangles 3 - Combined figures”)

c = код клавиши

Очистить консоль

Если (c)

49:

stack\_c.Push(Contour)

Вывод ("Element pushed successfully!\n")

break

50:

stack\_p.Push(MainPainted)

Вывод ("Element pushed successfully!\n")

break

51:

stack\_cb.Push(Combined)

Вывод ("Element pushed successfully!\n")

break

50:

Очистить консоль

Вывод ("Stack for 1 – Contours 2 - Painted triangles 3 - Combined figures”)

c = код клавиши

Очистить консоль

Если (c)

49:

Contour = stack\_c.Pop()

Вывод ("Element popped successfully!\n")

break

50:

MainPainted = stack\_p.Pop()

Вывод ("Element popped successfully!\n")

break

51:

Combined = stack\_cb.Pop()

Вывод ("Element popped successfully!\n")

break

51:

Очистить консоль

Вывод ("Stack for 1 – Contours 2 - Painted triangles 3 - Combined figures”)

c = код клавиши

Очистить консоль

Если (c)

Очистить консоль

49:

Очистить консоль

stack\_c.Print(hdc)

break

50:

Очистить консоль

stack\_p.Print(hdc)

break

51:

Очистить консоль

stack\_cb.Print(hdc)

break

52:

Очистить консоль

Вывод ("Stack for 1 – Contours 2 - Painted triangles 3 - Combined figures”)

c = код клавиши

Очистить консоль

Если (c)

49:

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

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

Contour = stack\_c.find(color)

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

break

50:

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

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

MainPainted = stack\_p.find(color)

Вывод ("Painted triangle found and buffered!\n")

break

51:

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

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

Combined = stack\_cb.find(color)

Вывод ("Combined figure found and buffered!\n")

break

53:

Очистить консоль

Вывод ("Stack for 1 – Contours 2 - Painted triangles 3 - Combined figures”)

c = код клавиши

Очистить консоль

Если (c)

49:

Открыть файл("stackout\_cont.txt”, ”w”)

Закрыть файл(f)

stack\_c.Save("stackout\_cont.txt")

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

break

50:

Открыть файл("stackout\_painted.txt”, ”w”)

Закрыть файл(f)

stack\_p.Save("stackout\_painted.txt")

Вывод ("Your stack with painted triangles saved into stackout\_painted.txt\n")

break

51:

Открыть файл("stackout\_comb.txt”, ”w”)

Закрыть файл(f)

stack\_cb.Save("stackout\_comb.txt")

Вывод ("Your stack with combined figures saved into stackout\_comb.txt\n")

break

54:

Очистить консоль

Вывод("Stack for 1 – Contours 2 - Painted triangles 3 - Combined figures");

c = код клавиши

Очистить консоль

Делать

Очистить консоль

Получение размеров окна

Если размеры окна поменялись

Очистить консоль

Обновить размеры окна

Пытаться

Если (c)

49:

triangle = &Contour;

triangle->Draw(hdc);

break;

50:

triangle = &MainPainted;

triangle->Draw(hdc);

break;

51:

Combined.Draw(hdc);

break;

Поймать (исключение)

Если (исключение)

ERR\_WRONG\_SIZE:

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

break;

ERR\_WRONG\_CUT\_POINTS:

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

break;

Если нажата клавиша{

c = код клавиши

Пока (c != 13);

Очистить консоль

break;

55:

Очистить консоль

Вывод ("1 – Contour 2 – Brush for painted triangle 3 – Internal triangle for combined figure”)

c = код клавиши

Очистить консоль

Если (c)

49:

Contour.load(“incontour.txt”)

Вывод(“Contour loaded successfully!”)

break

50:

Brush.load(“inbrush.txt”)

Вывод(“Brush loaded successfully!”)

break

51:

Combined.load(“inbrush\_combined.txt”, “incontour\_combined.txt”)

Вывод(“Combined figure loaded successfully!”)

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\_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\_CANNOT\_OPEN\_INPUT\_FILE:

Вывод ("Cannot open input file")

break

Вывод(“Press any key to go back to menu”)

Пока (c != 27)

Освободить контекст отображения

1. **Текст программы**

**triangle.h**

#ifndef TRIANGLE\_H

#define TRIANGLE\_H

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <conio.h>

#include "error.h"

class Triangle {

public:

virtual void Draw(HDC hdc) = 0;

virtual void load(const char name[]) = 0;

virtual void save(const char name[]) = 0;

};

#endif

**brush.h**

#ifndef BRUSH\_H

#define BRUSH\_H

#include "triangle.h"

#include "error.h"

class brush { //Кисть

protected:

COLORREF bcolor;

int bstyle;

public:

brush() : bcolor(NULL), bstyle(NULL) {};

void setBStyle(int new\_bstyle);

void getBStyle(int &curr\_bstyle);

void setBColor(COLORREF new\_bcolor);

void getBColor(COLORREF &curr\_bcolor);

void save(const char name[]);

void load(const char name[]);

};

#endif

**brush.cpp**

#include "brush.h"

#include "error.h"

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

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

bstyle = new\_bstyle;

}

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

curr\_bstyle = bstyle;

}

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

curr\_bcolor = bcolor;

}

void brush::save(const char name[]) {

FILE \*f;

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

int brgb[3];

fprintf\_s(f, "%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]);

fclose(f);

}

void brush::load(const char name[]) {

FILE \*f;

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

if (f == NULL)

throw ERR\_CANNOT\_OPEN\_INPUT\_FILE;

int brgb[3], bstyle;

COLORREF bcolor;

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]);

try

{

setBStyle(bstyle);

setBColor(bcolor);

}

catch (int exception)

{

throw exception;

}

fclose(f);

}

**contour.h**

#ifndef CONTOUR\_H

#define CONTOUR\_H

#include "triangle.h"

class contour : public Triangle { //Фигура-контур..Перо

protected:

COLORREF color;

int style, border;

POINT points[3];

public:

contour() : color(NULL), style(NULL), border(NULL), points{NULL, NULL, NULL} {}

void setStyle(int new\_style);

void getStyle(int &curr\_style);

void setColor(COLORREF new\_color);

void getColor(COLORREF &curr\_color);

void setBorder(int new\_border);

void getBorder(int &curr\_border);

void setPoints(POINT \*new\_points);

void getPoints(POINT \*curr\_points);

void Draw(HDC hdc);

void save(const char name[]);

void load(const char name[]);

void checkSizeT(HDC hdc);

};

#endif

**contour.cpp**

#include "contour.h"

#include "error.h"

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

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

style = new\_style;

}

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

curr\_style = style;

}

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

curr\_color = color;

}

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

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

border = new\_border;

}

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

curr\_border = border;

}

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

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

}

void contour::Draw(HDC hdc) {

checkSizeT(hdc);

HPEN hMyPen = CreatePen(style, border, color);

SelectPen(hdc, hMyPen);

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

Polyline(hdc, ppt, 4);

DeletePen(hMyPen);

}

void contour::save(const char name[]) {

FILE \*f;

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

int rgb[3];

fprintf\_s(f, "\n");

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

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

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

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

rgb[0] = GetRValue(color);

rgb[1] = GetGValue(color);

rgb[2] = GetBValue(color);

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

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

fclose(f);

}

void contour::load(const char name[]) {

FILE \*f;

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

if (f == NULL) throw ERR\_CANNOT\_OPEN\_INPUT\_FILE;

int rgb[3], style, border;

COLORREF color;

POINT points[3];

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

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

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

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

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

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

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

try

{

setPoints(points);

setStyle(style);

setColor(color);

setBorder(border);

}

catch (int exception)

{

throw exception;

}

fclose(f);

}

void contour::checkSizeT(HDC hdc) {

RECT rt;

GetClientRect(WindowFromDC(hdc), &rt);

int rtx = rt.right;

int rty = rt.bottom;

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;

}

**painted.h**

#ifndef PAINTED\_H

#define PAINTED\_H

#include "brush.h"

#include "contour.h"

class painted : public brush, public contour { // Закрашенная фигура

public:

painted() {}

painted(brush Brush, contour Contour);

void save(const char name[]);

void load(const char name1[], const char name2[]);

void Draw(HDC hdc);

};

#endif

**painted.cpp**

#include "painted.h"

painted::painted(brush Brush, contour Contour) {

Brush.getBColor(this->bcolor);

Brush.getBStyle(this->bstyle);

Contour.getPoints(this->points);

Contour.getBorder(this->border);

Contour.getColor(this->color);

Contour.getStyle(this->style);

}

void painted::Draw(HDC hdc) {

checkSizeT(hdc);

COLORREF buffcolor = color;

if (border == 0)

buffcolor = bcolor;

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

LOGBRUSH\* logBrush = new LOGBRUSH;

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

logBrush->lbHatch = bstyle - 1;

logBrush->lbColor = bcolor;

HBRUSH hMyBrush = CreateBrushIndirect(logBrush);

SelectPen(hdc, hMyPen);

SelectBrush(hdc, hMyBrush);

Polygon(hdc, points, 3);

DeletePen(hMyPen);

delete logBrush;

DeleteBrush(hMyBrush);

}

void painted::save(const char name[]) {

brush::save(name);

contour::save(name);

}

void painted::load(const char name1[], const char name2[]) {

brush::load(name1);

contour::load(name2);

}

**combined.h**

#ifndef COMBINED\_H

#define COMBINED\_H

#include "painted.h"

class combined {

private:

painted ext;

painted intern;

public:

combined() {};

void Draw(HDC hdc);

void addExt(painted Ext);

void save(const char name[]);

void load(const char name1[], const char name2[]);

void checkSizeTC();

void getIntern(painted &t);

void getExt(painted &t);

};

#endif

**combined.cpp**

#include "combined.h"

#include "error.h"

void combined::Draw(HDC hdc) {

ext.checkSizeT(hdc);

checkSizeTC();

ext.Draw(hdc);

intern.Draw(hdc);

}

void combined::save(const char name[]) {

intern.brush::save(name);

intern.contour::save(name);

}void combined::load(const char name1[], const char name2[]) {

intern.brush::load(name1);

intern.contour::load(name2);

}

void combined::addExt(painted Ext) {

ext = Ext;

}

void combined::checkSizeTC() {

POINT points[3], new\_Cpoints[3];

ext.getPoints(points);

intern.getPoints(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 combined::getIntern(painted &t) {

t = intern;

}

void combined::getExt(painted &t) {

t = ext;

}

**stack.h**

#ifndef STACK\_H

#define STACK\_H

#define ERR\_FULL 10

#define ERR\_EMPTY 20

#define ERR\_NOT\_FOUND 30

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include "combined.h"

template <class T>

class Stack {

private:

T \*arr;

int top;

int n;

public:

Stack(int N) {

top = -1;

n = N;

arr = new T[N];

}

~Stack() {

delete[] arr;

}

void Push(T d) {

if (isFull()) throw ERR\_FULL;

top++;

arr[top] = d;

}

T Pop() {

if (isEmpty()) throw ERR\_EMPTY;

top--;

return arr[top + 1];

}

void Print(HDC hdc) {

if (isEmpty()) throw ERR\_EMPTY;

T \*temp\_arr = arr;

int temp\_top = top;

while (temp\_top != -1) {

temp\_arr[temp\_top].Draw(hdc);

temp\_top--;

}

}

T find(int \*key) {

if (isEmpty()) throw ERR\_EMPTY;

T \*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 Save(const char name[]) {

T \*temp\_arr = arr;

int temp\_top = top;

while (temp\_top != -1) {

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

temp\_top--;

}

}

bool isEmpty() {

if (top == -1) return true;

return false;

}

bool isFull() {

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

return false;

}

};

template<>

combined Stack<combined>::find(int \*key) {

if (isEmpty()) throw ERR\_EMPTY;

combined \*temp\_arr = arr;

int temp\_top = top;

COLORREF color, t\_color;

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

painted temp;

temp\_arr[temp\_top].getIntern(temp);

temp.getColor(t\_color);

while (t\_color != color) {

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

temp\_top--;

temp\_arr[temp\_top].getIntern(temp);

temp.getColor(t\_color);

}

return temp\_arr[temp\_top];

}

#endif

**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"

void hidecursor()

{

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

CONSOLE\_CURSOR\_INFO info;

info.dwSize = 25;

info.bVisible = FALSE;

SetConsoleCursorInfo(consoleHandle, &info);

}

void clear()

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

RECT rt;

GetClientRect(hwnd, &rt);

system("cls");

HPEN hCurrPen, hOldPen;

HBRUSH hCurrBrush, hOldBrush;

hCurrPen = CreatePen(PS\_SOLID, 1, COLOR\_BACKGROUND);

hCurrBrush = CreateSolidBrush(COLOR\_BACKGROUND);

hOldPen = SelectPen(hdc, hCurrPen);

hOldBrush = SelectBrush(hdc, hCurrBrush);

GetClientRect(hwnd, &rt);

Rectangle(hdc, 0, 0, rt.right, rt.bottom);

SelectPen(hdc, hOldPen);

DeletePen(hCurrPen);

SelectBrush(hdc, hOldBrush);

DeleteBrush(hCurrBrush);

}

void clearConsole() {

clear();

//system("cls");

//system("cls");

}

**error.h**

#ifndef ERROR\_H

#define ERROR\_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

#endif

**oop5lab.cpp**

#include "external.h"

#include "combined.h"

#include "error.h"

#include "stack.h"

#define N 30

void main()

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

bool tosleep = false;

Triangle \*triangle;

brush Brush;

painted SecPainted;

contour Contour;

painted MainPainted;

combined Combined;

Stack<contour> stack\_c = Stack<contour>(N);

Stack<painted> stack\_p = Stack<painted>(N);

Stack<combined> stack\_cb = Stack<combined>(N);

RECT rt;

int oldx = 0, oldy = 0;

int rtx = 0, rty = 0;

char c = 0;

printf\_s("Menu:\n1 - Push into stack\n2 - Pop from stack\n3 - Print stack\n4 - Find figure in stack by color\n5 - Save stack into file\n6 - Draw figure in buffer\n7 - Load figure into buffer from file\nEsc - Exit\n");

do

{

try

{

if (\_kbhit()) {

clearConsole();

printf\_s("Menu:\n1 - Push into stack\n2 - Pop from stack\n3 - Print stack\n4 - Find figure in stack by color\n5 - Save stack into file\n6 - Draw figure in buffer\n7 - Load figure into buffer from file\nEsc - Exit\n");

c = \_getch();

switch (c)

{

case 49: //1 - push

clearConsole();

printf\_s("Stack for\n1 - Contours\n2 - Painted triangles\n3 - Combined figures\n");

c = \_getch();

clearConsole();

switch (c)

{

case 49: //1 - push contour

stack\_c.Push(Contour);

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

break;

case 50: //2 - push painted

stack\_p.Push(MainPainted);

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

break;

case 51: //3 - push combined

stack\_cb.Push(Combined);

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

break;

}

break;

case 50: //2 - pop

clearConsole();

printf\_s("Stack for\n1 - Contours\n2 - Painted triangles\n3 - Combined figures\n");

c = \_getch();

clearConsole();

switch (c)

{

case 49: //1 - pop contour

Contour = stack\_c.Pop();

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

break;

case 50: //2 - pop painted

MainPainted = stack\_p.Pop();

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

break;

case 51: //3 - pop combined

Combined = stack\_cb.Pop();

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

break;

}

break;

case 51: //3 - print

clearConsole();

printf\_s("Stack for\n1 - Contours\n2 - Painted triangles\n3 - Combined figures\n");

c = \_getch();

clearConsole();

switch (c)

{

case 49: //1 - print contour

clearConsole();

stack\_c.Print(hdc);

break;

case 50: //2 - print painted

clearConsole();

stack\_p.Print(hdc);

break;

case 51: //3 - print combined

clearConsole();

stack\_cb.Print(hdc);

break;

}

break;

case 52: //4 - find

clearConsole();

printf\_s("Stack for\n1 - Contours\n2 - Painted triangles\n3 - Combined figures\n");

c = \_getch();

int color[3];

clearConsole();

switch (c)

{

case 49: //1 - find contour

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

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

Contour = stack\_c.find(color);

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

break;

case 50: //2 - find painted

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

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

MainPainted = stack\_p.find(color);

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

break;

case 51: //3 - find combined

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

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

Combined = stack\_cb.find(color);

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

break;

}

break;

case 53: //5 - save

clearConsole();

printf\_s("Stack for\n1 - Contours\n2 - Painted triangles\n3 - Combined figures\n");

c = \_getch();

FILE \*f;

clearConsole();

switch (c)

{

case 49: //1 - save contour

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

fclose(f);

stack\_c.Save("stackout\_cont.txt");

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

break;

case 50: //2 - save painted

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

fclose(f);

stack\_p.Save("stackout\_painted.txt");

printf\_s("Your stack with painted triangles saved into stackout\_painted.txt\n");

break;

case 51: //3 - save combined

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

fclose(f);

stack\_cb.Save("stackout\_comb.txt");

printf\_s("Your stack with combined figures saved into stackout\_comb.txt\n");

break;

}

break;

case 54: //6 - рисование текущего треугольника

clearConsole();

printf\_s("Stack for\n1 - Contours\n2 - Painted triangles\n3 - Combined figures\n");

c = \_getch();

clearConsole();

do {

tosleep = true;

GetClientRect(hwnd, &rt);

rtx = rt.right;

rty = rt.bottom;

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

hidecursor();

oldx = rtx;

oldy = rty;

Sleep(50);

tosleep = false;

}

try

{

switch (c)

{

case 49: //1 - Contour

triangle = &Contour;

triangle->Draw(hdc);

break;

case 50: //2 - Painted

triangle = &MainPainted;

triangle->Draw(hdc);

break;

case 51: //3 - Combined

Combined.Draw(hdc);

break;

}

}

catch (int exception)

{

switch (exception) {

case ERR\_WRONG\_SIZE:

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

break;

case ERR\_WRONG\_CUT\_POINTS:

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

break;

}

}

if (\_kbhit()) {

c = \_getch();

}

if (tosleep)

Sleep(50);

} while (c != 13);

clearConsole();

break;

case 55: //7 - загрузить треугольник из файла

clearConsole();

printf\_s("1 - Contour\n2 - Brush for painted triangle\n3 - Internal figure for combined figure\n");

c = \_getch();

clearConsole();

switch (c)

{

case 49: //1 - load contour

Contour.load("incontour.txt");

printf\_s("Contour loaded succesfully!\n");

break;

case 50: //2 - load painted

Brush.load("inbrush.txt");

printf\_s("Brush loaded succesfully!\n");

MainPainted = painted(Brush, Contour);

break;

case 51: //3 - load combined

Combined.load("inbrush\_combined.txt", "incontour\_combined.txt");

printf\_s("Combined figure loaded succesfully!\n");

Combined.addExt(MainPainted);

break;

}

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\_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\_CANNOT\_OPEN\_INPUT\_FILE:

printf\_s("Cannot open input file");

break;

}

printf\_s("\nPress any key to go back to menu");

\_getch();

}

} while (c != 27);

ReleaseDC(hwnd, hdc);

}

1. **Тесты**

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Пояснение** | **Результат** |
| Содержание стека  **Brush:**  **1)**  0  255 255 0  **2)**  0  255 0 255  **Contour:**  **1)**  100 20  30 50  60 50  0 1  255 0 0  **2)**  10 20  30 50  60 50  0 1  255 255 0  Поиск цвета  1)255 0 0  2)255 255 0  3)1 1 1 | Тест поиска  Корректные входные данные.  Поиск по трем разным данным:  1)Корректные  2)Корректные  3)Некорректные  Поиск производился среди закрашенных треугольников |  |
| Содержание стека  **Brush:**  **1)**  0  255 255 0  **2)**  0  255 0 255  **Contour:**  **1)**  100 20  30 50  60 50  0 1  255 0 0  **2)**  10 20  30 50  60 50  0 1  255 255 0 | Корректные входные данные.  Распечатка стека  ДО взятия и  ПОСЛЕ взятия одного элемента  Распечатка стека с контурами и закрашенными треугольниками |  |
| Содержание стека  **-** | 1)Взятие элемента из пустого стека  ИЛИ  Распечатка содержимого стека  2)Рисование треугольника из пустого буфера  (Будет выведено пустое изображение) |  |
| Содержание стека  **Brush:**  0  255 0 255  **Contour:**  10 20  30 50  60 50  0 1  255 255 0  **Combined brush:**  0  255 0 255  **Combined contour:**  300 440  250 350  500 485  0 1  0 255 255 | Проверка сохраняемости содержимого стека в файле. | Внутренний треугольник комбинированной фигуры    Контур:    Закрашенный треугольник: |