Министерство образования и науки РФ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра теоретической и прикладной информатики

Практическое задание №3

по дисциплине «Объектно-ориентированное программирование»

**Факультет:** ПМИ

**Группа:**  ПМ-92

**Студенты:**  Иванов В., Попов А.

**Преподаватель:**  Неделько В.М.

Новосибирск

2020

***Постановка задачи***

**ЛР4**

Путем модификации программ, разработанных в лабораторных работах № 1, 2, разработать программу такую, чтобы в ней были определены несколько классов, реализующих понятие геометрической фигуры в графической системе:

– абстрактный класс «Фигура», содержащий чисто виртуальные функции;

– класс «Закрашенный», позволяющий задать кисть, ее параметры и, возможно, осуществить закраску;

– класс «Фигура-контур» потомок класса «Фигура»;

– класс «Закрашенная фигура» потомок класса «Фигура-контур», класс «Закрашенный» при этом использовать либо как второго родителя (множественное наследование), либо как часть класса «Закрашенная фигура» (агрегация);

– класс «Комбинированная фигура», реализующий две вложенные фигуры с закраской между ними.

Реализацию классов поместить в отдельный файл. Разработать функцию, демонстрирующую поведение разработанных классов, включая демонстрацию механизма позднего связывания.

**ЛР5**

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

Преобразовать класс-контейнер, разработанный в лабораторной работе №3, в шаблон, так чтобы элементами контейнеров могли быть различные классы, разработанные в лабораторной работе №4 (при различном инстанцировании шаблона).

Разработать функцию, демонстрирующую поведение разработанного шаблона: провести инстанцирование шаблона для каждого из классов-фигур, продемонстрировать их функционирование.

***Меню консольного окна***

**Q** - загрузить из файла элемент (фигура-контур)

**W** - загрузить из файла элемент (закрашенная фигура)

**E** - загрузить из файла элемент (вложенные треугольники)

**R** - сохранить в файл элемент (фигура-контур)

**T** - сохранить в файл элемент (закрашенная фигура)

**Y** - сохранить в файл элемент (вложенные фигуры)

**U** - загрузить из файла очередь, элементы которой: фигура-контур

**I** - загрузить из файла очередь, элементы которой: закрашенная фигура

**O** - загрузить из файла очередь, элементы которой: вложенные треугольники

**P** - сохранить в файл очередь, элементы которой: фигура-контур

**A** - сохранить в файл очередь, элементы которой: закрашенная фигура

**S** - сохранить в файл очередь, элементы которой: вложенные треугольники

**D** - добавить элемент фигура-контур в очередь

**F** - добавить элемент закрашенная фигура в очередь

**G** - добавить элемент вложенные треугольники в очередь

**H** - удалить элемент фигура-контур из очереди

**J** - удалить элемент закрашенная фигура из очереди

**K** - удалить элемент вложенные треугольники из очереди

**L** - показать графически очередь фигур-контуров

**Z** - показать графически очередь закрашенных фигур

**X** - показать графически очередь вложенных фигур

**C** - найти элемент фигура-контур по цвету контура

**V** - найти элемент закрашенная фигура по цвету контура

**B** - найти элемент вложенные треугольники по цвету контура

**N** - взять элемент фигура-контур из очереди

**M** - взять элемент закрашенная фигура из очереди

**,** - взять элемент вложенные треугольники из очереди

**Esc** - выход

***Текст программы***

**main.cpp**

#include <iostream>

#include "ucontainer.h"

void fill(HWND hwnd, HDC hdc){

HPEN bp = CreatePen(1, 1, RGB(0, 0, 0));

HBRUSH bb = GetStockBrush(BLACK\_BRUSH);

HPEN pen = SelectPen(hdc, bp);

HBRUSH brush = SelectBrush(hdc, bb);

RECT rect;

GetClientRect (hwnd, &rect);

Rectangle(hdc, -1, -1, rect.right + 1, rect.bottom + 1);

SelectPen(hdc, bp);

DeletePen(pen);}

void processing(int error){

switch(error){case 0:{ printf("**\n**Reading/writing error!");

getch();

break;}

case 1:{ printf("**\n**Window is too small to contain the triangle!");

getch();

break;}

2:{printf("**\n**Not a triangle!");

getch();

break;}

case 3:{printf("**\n**Wrong type!");

getch();

break;}

case 4:{printf("**\n**Wrong width of a line!");

getch();

break;}

case 5:{printf("**\n**Color error!");

getch();

break;}

case 6:{printf("**\n**Figures aren’t nested!");

getch();

break;}}}

void main(){

FILE\* F;

setlocale(LC\_ALL, "Russian");

printf("Растяните окно и нажмите любую клавишу");

getch();

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

RECT rt;

GetClientRect(hwnd, &rt);

SetBkColor(hdc,RGB(0,0,0));

char c = 0;

circuit cc;

painted pp;

shape \*sh;

multi mm;

container<circuit> c1;

container<painted> c2;

container<multi> c3;

char inp[20];

do

{try{

system("cls");

fill(hwnd,hdc);

printf("q. Load a circuit of triangle**\n**");

printf("w. Load a fill triangle**\n**");

printf("e. Load a double of triangle**\n**");

printf("r. Save a circuit of triangle**\n**");

printf("t. Save a fill triangle**\n**");

printf("y. Save a double of triangle**\n**");

printf("u. Load a queue of circuits**\n**");

printf("i. Load a queue of fill triangles**\n**");

printf("o. Load a queue of double triangles**\n**");

printf("p. Save the queue of circuits**\n**");

printf("a. Save the queue of fill triangles**\n**");

printf("s. Save the queue of double triangles**\n**");

printf("d. Add the circuit of triangle to the queue**\n**");

printf("f. Add the fill triangle to the queue**\n**");

printf("g. Add the double of triangle to the queue**\n**");

printf("h. Delete a circuit of triangle out off the queue**\n**");

printf("j. Delete a fill triangle out off the queue**\n**");

printf("k. Delete a double of triangle out off the queue**\n**");

printf("l. View the queue of circuits**\n**");

printf("z. View the queue of fill triangles**\n**");

printf("x. View the queue of double triangles**\n**");

printf("c. Find a circuit of triangle in the queue**\n**");

printf("v. Find a fill triangle in the queue**\n**");

printf("b. Find a double of triangle in the queue**\n**");

printf("n. Get a circuit of triangle out off the queue**\n**");

printf("m. Get a fill triangle out off the queue**\n**");

printf(",. Get a double of triangle out off the queue**\n**");

printf("Esc. Exit**\n**");

c = getch();

switch(c)

{

case 'q':{

system("cls");

fill(hwnd,hdc);

l:

printf("Введите имя файла: ");

scanf("%s",&inp);

strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto l;

cc.fload(F);

sh = &cc;

sh->fill();

fclose(F);

c = 0;

break;}

case 'w':{

system("cls");

fill(hwnd,hdc);

q:

printf("Введите имя файла: ");

scanf("%s",&inp);

strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto q;

pp.fload(F);

sh = &pp;

sh->fill();

fclose(F);

c = 0;

break;}

case 'e':{

system("cls");

fill(hwnd,hdc);

w:

printf("Введите имя файла: ");

scanf("%s",&inp); strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto w;

mm.fload(F);

sh = &mm;

sh->fill();

fclose(F);

c = 0;

break;}

case 'r':{

system("cls");

fill(hwnd,hdc);

e:

printf("Введите имя файла: ");

scanf("%s",&inp); strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto e;

cc.fsave(F);

sh = &cc;

sh->fill();

fclose(F);

c = 0;

break;}

case 't':{

system("cls");

fill(hwnd,hdc);

r:

printf("Введите имя файла: ");

scanf("%s",&inp); strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto r;

pp.fsave(F);

sh = &pp;

sh->fill();

fclose(F);

c = 0;

break;}

'y':{

system("cls");

fill(hwnd,hdc);

t:

printf("Введите имя файла: ");

scanf("%s",&inp);

strcpy(&inp[strlen(inp)],".txt"); if(!(F = fopen(inp,"r"))) goto t;

mm.fsave(F);

sh = &mm;

sh->fill();

fclose(F);

c = 0;

break;}

case 'u':{

system("cls");

fill(hwnd,hdc);

y:

printf("Введите имя файла: ");

scanf("%s",&inp); strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto y;

c1.load(F, hwnd);

fclose(F);

c = 0;

break;}

case 'i':{

system("cls");

fill(hwnd,hdc);

u:

printf("Введите имя файла: ");

scanf("%s",&inp);

strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto u;

c2.load(F, hwnd);

fclose(F);

c = 0;

break;}

case 'o':{

system("cls");

fill(hwnd,hdc);

i:

printf("Введите имя файла: ");

scanf("%s",&inp); strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto i;

c3.load(F, hwnd);

fclose(F);

c = 0;

break;}

case 'p':{

system("cls");

fill(hwnd,hdc);

o:

printf("Введите имя файла: ");

scanf("%s",&inp);

strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto o;

c1.save(F);

fclose(F);

c = 0;

break;}

case 'a':{

system("cls");

fill(hwnd,hdc);

p:

printf("Введите имя файла: ");

scanf("%s",&inp); strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto p;

c2.save(F);

fclose(F);

c = 0;

break;}

case 's':{

system("cls");

fill(hwnd,hdc);

a:

printf("Введите имя файла: ");

scanf("%s",&inp);

strcpy(&inp[strlen(inp)],".txt");

if(!(F = fopen(inp,"r"))) goto a;

c3.save(F);

fclose(F);

c = 0;

break;}

case 'd':{

system("cls");

fill(hwnd,hdc);

c1.add(cc);

c = 0;

break;}

case 'f':{

system("cls");

fill(hwnd,hdc);

c2.add(pp);

c = 0;

break;}

case 'g':{

system("cls");

fill(hwnd,hdc);

c3.add(mm);

c = 0;

break;}

case 'h':{

system("cls");

fill(hwnd,hdc);

c1.del();

c = 0;

break;}

case 'j':{

system("cls");

fill(hwnd,hdc);

c2.del();

c = 0;

break;}

case 'k':{

system("cls");

fill(hwnd,hdc);

c3.del();

c = 0;

break;}

case 'l':{

system("cls");

fill(hwnd,hdc);

c1.view(hdc, rt);

c = 0;

break;}

case 'z':{

system("cls");

fill(hwnd,hdc);

c2.view(hdc, rt);

c = 0;

break;}

case 'x':{

system("cls");

fill(hwnd,hdc);

c3.view(hdc, rt);

c = 0;

break;}

case 'c':{

system("cls");

fill(hwnd,hdc);

c1.find();

c = 0;

getch();

break;}

case 'v':{

system("cls");

fill(hwnd,hdc);

c2.find();

c = 0;

getch();

break;}

case 'b':{

system("cls");

fill(hwnd,hdc);

c3.find();

c = 0;

getch();

break;}

case 'n':{

system("cls");

fill(hwnd,hdc);

cc = c1.get();

cc.paint();

c = 0;

break;}

case 'm':{

system("cls");

fill(hwnd,hdc);

pp = c2.get();

pp.paint();

c = 0;

break;}

case ',':{

system("cls");

fill(hwnd,hdc);

mm = c3.get();

mm.paint();

c = 0;

break;}}}

catch(int error){processing(error);}}

while(c != 27);

ReleaseDC (hwnd, hdc);

getch();}

**c\_brush.h**

#ifndef C\_BRUSH\_H\_

#define C\_BRUSH\_H\_

#include "figure.h"

class c\_brush

{

protected:

color c\_brush\_color;

HBRUSH so\_brush;

public:

inline color get\_color(void){return c\_brush\_color;}

void set\_c\_brush(color new\_brush);

c\_brush& operator=(const c\_brush& those);

HBRUSH get\_so\_brush(void);

};

#endif

**figure.h**

#ifndef FIGURE\_H\_

#define FIGURE\_H\_

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <conio.h>

struct point{int x; int y;};

struct color{int r; int g; int b;};

class figure{

protected:

point a, b, c;

int reconwindow(HWND hwnd, point a);

float init\_point(point to[],int t1,int t2,int t3);

int checkpoint(point to[3]);

public:

inline point get\_a(){ return a;} // получение из класса координат вершин треугольников (большого и малого)

inline point get\_b(){ return b;}

inline point get\_c(){ return c;}

void set\_a(HWND hwnd, point n\_a); // установка координат вершин треугольников в классе (большого и малого)

void set\_b(HWND hwnd, point n\_b);

void set\_c(HWND hwnd, point n\_c);

virtual void paint() = 0;

virtual void fload(FILE \*F) = 0;

virtual void fsave(FILE \*F) = 0;};

#endif

**circuit.h**

#ifndef CIRCUIT\_H\_

#define CIRCUIT\_H\_

#include "figure.h"

class circuit: public figure

{

protected:

color c\_color;

int c\_width;

public:

circuit(point p) { set\_pos(p);}

inline int get\_c\_width(void){return c\_width;}

void set\_c\_width(int c\_w);

inline color get\_c\_color(void){return c\_color;}

void set\_c\_color(color c\_col);

void paint();

void fload(FILE \*F);

void fsave(FILE \*F);

circuit& operator = (const circuit& those);

};

#endif

**multi.h**

#ifndef MULTI\_H\_

#define MULTI\_H\_

#include"painted.h"

class multi: public painted

{

protected:

int checkencl(point to[6]);

painted interside;

public:

multi(point p) { set\_pos(p);}

void paint();

void fload(FILE \*F);

void fsave(FILE \*F);

multi&operator=(const multi& those);};

#endif

**painted.h**

#ifndef PAINTED\_H\_

#define PAINTED\_H\_

#include "circuit.h"

#include "c\_brush.h"

class painted: public circuit, public c\_brush

{

protected:

public:

painted(point p) { set\_pos(p);}

void paint();

void fload(FILE \*F);

void fsave(FILE \*F);

painted& operator=(const painted& those);};

#endif

**ucontainer.h**

#ifndef UCONTAINER\_H\_

#define UCONTAINER\_H\_

#include "multi.h"

template<class Type>

struct queue

{queue \*next;

queue \*pred;

Type k;};

template<class Type>

struct list

{queue<Type> \*head;

queue<Type> \*tail;};

template<class Type>

class container

{private:

queue<Type> \*q;

list<Type> l;

public:

container();

void add (Type K);

void del();

Type get();

void save(FILE \*F);

void load(FILE \*F, HWND hwnd);

~container();

void view(HDC,RECT rt);

void empty();

int find();};

template<class Type>

container<Type>::container(){

l.head = NULL;

l.tail = l.head;}

template<class Type>

void container<Type>::add(Type K){

queue<Type> \*a;

a = new queue<Type>;

if(!l.head){

l.head = new queue<Type>;

l.head -> next = a;

a -> k = K;

l.tail = a;

l.tail -> next = NULL;}

l.tail -> next = a;

a -> pred = l.tail;

a -> k = K;

a -> next = l.head->next;

l.tail = a;

l.head -> pred = l.tail;}

template<class Type>

void container<Type>::del(){queue<Type> \*a = l.head;

if(a){a = l.head -> next;

if(!a) printf("Очередь пуста");

else if(a -> next == l.head->next){

delete a;

l.tail = NULL;

l.head = NULL;

printf("Треугольник удален **\n**");}

else{l.head -> next = a -> next;

l.tail -> next = a -> next;

a -> next -> pred = l.tail;

delete a;

printf("Треугольник удален");}}

else printf("Очередь пуста");}

template<class Type>

void container<Type>::save(FILE \*F){

queue<Type> \*a=l.head;

if(!a) printf("Очередь пуста");

else{a = l.head -> next;

while(a -> next != l.head -> next){

a -> k.fsave(F);

a = a -> next;}

a -> k.fsave(F);

printf("Данные сохранены");}}

template<class Type>

void container<Type>::load(FILE \*F, HWND hwnd){

while(!feof(F)){

Type q;

q.fload(F);

add(q);}

template<class Type>

container<Type>::~container(){

while(l.head) del();}

template<class Type>

void container<Type>::view(HDC hdc,RECT rt){

queue<Type> \*a;

*/\*int n = 0;\*/*

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

if(!l.head) printf("Очередь пуста");

else{a = l.head -> next;

while(a -> next != l.head -> next){

*/\*a -> k.numr(&n);\*/*

a -> k.paint();

a = a -> next; getch();

FillRect(hdc,&rt,xBr);}

a -> k.paint();getch();}

FillRect(hdc,&rt,xBr);}

template<class Type>

void container<Type>::empty(){

if(l.head == NULL) printf("Очередь пуста");

else printf("Очередь не пуста"); getch();}

template<class Type>

int container<Type>::find()

{int flag = 0;

queue<Type> \*qe = l.head;

int a, b, c, i = 1;

color col;

if(qe){qe = l.head -> next;

printf("Введите цвет линии **\n**");

scanf("%d %d %d", &a, &b, &c);

while(qe -> next != l.head -> next){col = qe->k.get\_c\_color();

if(a == col.r && b == col.g && c == col.b){flag = flag + 1;

printf("**\n**Элемент найден #%d",i);}

qe=qe->next;i++;} col = qe->k.get\_c\_color();

if(a == col.r && b == col.g && c == col.b){flag = flag + 1;

printf("**\n**Элемент найден #%d",i);}

if(flag) return 1;

printf("Элемент не найден");return 0;}

else printf("Очередь пуста");return 0;}

template<class Type>

Type container<Type>::get(){

queue<Type> \*a = l.head;

if(a){a = l.head->next;

if(a -> next == l.head -> next){

delete a;

l.tail = NULL;

l.head = NULL;}

else{l.head -> next = a -> next;

l.tail -> next = a -> next;

a -> next -> pred = l.tail;

delete a;}return a->k else printf("Очередь пуста");}

#endif

**c\_brush.cpp**

#include "c\_brush.h"

void c\_brush::set\_c\_brush(color new\_brush){

if((new\_brush.r > 255) || (new\_brush.r < 0) ||

(new\_brush.g > 255) || (new\_brush.g < 0) ||

(new\_brush.b > 255) || (new\_brush.b < 0)) throw 5;

else{c\_brush\_color.r = new\_brush.r;

c\_brush\_color.g = new\_brush.g;

c\_brush\_color.b = new\_brush.b;}}

c\_brush& c\_brush::operator = (const c\_brush& those){if(this == &those) return \*this;

else{c\_brush\_color.r == those.c\_brush\_color.r;

c\_brush\_color.g == those.c\_brush\_color.g;

c\_brush\_color.b == those.c\_brush\_color.b;}

return \*this;}

HBRUSH c\_brush::get\_so\_brush(void){so\_brush = CreateSolidBrush(RGB(c\_brush\_color.r,c\_brush\_color.g,c\_brush\_color.b));

return so\_brush;}

**figure.cpp**

#include "figure.h"

void figure::set\_a(HWND hwnd, point n\_a){

if(!reconwindow(hwnd, n\_a)) throw 1;

else{a.x = n\_a.x; a.y = n\_a.y;}}

void figure::set\_b(HWND hwnd, point n\_b){

if(!reconwindow(hwnd, n\_b)) throw 1;

else{b.x = n\_b.x; b.y = n\_b.y;}}

void figure::set\_c(HWND hwnd, point n\_c){

point pt[3];

pt[0].x = a.x;

pt[0].y = a.y;

pt[1].x = b.x;

pt[1].y = b.y;

pt[2].x = n\_c.x;

pt[2].y = n\_c.y;

if(!reconwindow(hwnd, n\_c)) throw 1;

else{if(!checkpoint(pt)) throw 2;

else{c.x = n\_c.x; c.y = n\_c.y;}}}

int figure::reconwindow(HWND hwnd, point a){

RECT rect;

GetClientRect (hwnd, &rect);

if((a.x > 0)&&(a.y > 0)&&(a.x < rect.right)&&(a.y < rect.bottom)) return 1;

else return 0;}

float figure::init\_point(point to[],int t1,int t2,int t3){float res;

res=(to[t3].x-to[t1].x)\*(to[t2].y-to[t1].y)-(to[t3].y-to[t1].y)\*(to[t2].x-to[t1].x);

return res;}

int figure::checkpoint(point to[3]) {if(((to[1].x - to[0].x) \* (to[2].y - to[0].y))==((to[1].y - to[0].y) \* (to[2].x - to[0].x)))return 0; else return 1;}

**circuit.cpp**

#include "circuit.h"

void circuit::set\_c\_color(color c\_col){

if((c\_col.r > 255) || (c\_col.r < 0) ||

(c\_col.g > 255) || (c\_col.g < 0) ||

(c\_col.b > 255) || (c\_col.b < 0)) throw 5;

else{c\_color.r = c\_col.r;

c\_color.g = c\_col.g;

c\_color.b = c\_col.b;}}

void circuit::set\_c\_width(int c\_w){

if((c\_w < 1) || (c\_w > 10)) throw 4;

else c\_width = c\_w;}

void circuit::paint(){

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

RECT wsize;

HPEN Pen1 = CreatePen(0, c\_width, RGB(c\_color.r, c\_color.g, c\_color.b));

HPEN hOldPen = SelectPen (hdc, Pen1);

POINT ppt[4]={ {a.x, a.y}, {b.x, b.y}, {c.x , c.y}, {a.x, a.y} };

Polyline (hdc, ppt, 4);

SelectPen (hdc, hOldPen);

DeletePen (Pen1);}

void circuit::fload(FILE \*F){

HWND hwnd = GetConsoleWindow();

int ax,ay,bx,by,cx,cy,w,cr,cg,cb;

point na,nb,nc;

color col;

fscanf(F, "%d %d %d %d %d %d %d %d %d %d",&ax,&ay,&bx,&by,&cx,&cy,&w,&cr,&cg,&cb);

na.x = ax;

na.y = ay;

nb.x = bx;

nb.y = by;

nc.x = cx;

nc.y = cy;

col.r = cr;

col.g = cg;

col.b = cb;

set\_a(hwnd, na);

set\_b(hwnd, nb);

set\_c(hwnd, nc);

set\_c\_width(w);

set\_c\_color(col);}

void circuit::fsave(FILE \*F){

if(F == NULL) throw 0;

else

fprintf(F, "%d %d %d %d %d %d %d %d %d %d", a.x, a.y, b.x, b.y, c.x, c.y, c\_width, c\_color.r, c\_color.g, c\_color.b);}

circuit& circuit:: operator=(const circuit& those){

if(this == &those) return \*this;

else{

a.x = those.a.x;

a.y = those.a.y;

b.x = those.b.x;

b.y = those.b.y;

c.x = those.c.x;

c.y = those.c.y;

c\_width = those.c\_width;

c\_color.r = those.c\_color.r;

c\_color.g = those.c\_color.g;

c\_color.b = those.c\_color.b;}

return \*this;}

**multi.cpp**

#include"multi.h"

int multi::checkencl(point to[6]){

if(((init\_point(to,0,1,3)<0) && (init\_point(to,0,1,4)<0) && (init\_point(to,0,1,5)<0) && (init\_point(to,1,2,3)<0) && (init\_point(to,1,2,4)<0) && (init\_point(to,1,2,5)<0) && (init\_point(to,2,0,3)<0) && (init\_point(to,2,0,4)<0) && (init\_point(to,2,0,5)<0)) ||

((init\_point(to,0,1,3)>0) && (init\_point(to,0,1,4)>0) && (init\_point(to,0,1,5)>0) &&

(init\_point(to,1,2,3)>0) && (init\_point(to,1,2,4)>0) && (init\_point(to,1,2,5)>0) && (init\_point(to,2,0,3)>0) && (init\_point(to,2,0,4)>0) && (init\_point(to,2,0,5)>0)))

return 1;else return 0;}

void multi::paint(){

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

HPEN Pen1 = CreatePen(0, c\_width, RGB(c\_color.r, c\_color.g, c\_color.b));

HPEN hPen = SelectPen (hdc, Pen1);

HBRUSH hBrush = SelectBrush(hdc, get\_so\_brush());

POINT ppt1[3] = { {a.x, a.y}, {b.x, b.y}, {c.x , c.y} };

Polygon(hdc, ppt1, 3);

SelectPen(hdc, hPen);

DeletePen(Pen1);

Pen1 = CreatePen(0, interside.get\_c\_width(), RGB(interside.get\_c\_color().r, interside.get\_c\_color().g, interside.get\_c\_color().b));

hPen = SelectPen (hdc, Pen1);

hBrush = SelectBrush(hdc, interside.get\_so\_brush());

POINT ppt2[3] = { {interside.get\_a().x, interside.get\_a().y}, {interside.get\_b().x, interside.get\_b().y}, {interside.get\_c().x , interside.get\_c().y} };

Polygon(hdc, ppt2, 3);

SelectPen(hdc, hPen);

DeletePen(Pen1);}

void multi::fload(FILE \*F){

HWND hwnd = GetConsoleWindow();

int ax,ay,bx,by,cx,cy,w,cr,cg,cb,cir,cig,cib,iax,iay,ibx,iby,icx,icy,w2;

point na,nb,nc;

color col, intro;

fscanf(F, "%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d",&ax,&ay,&bx,&by,&cx,&cy,&w,&cr,&cg,&cb,&cir,&cig,&cib,&iax,&iay,&ibx,&iby,&icx,&icy,&w2);

na.x = ax;

na.y = ay;

nb.x = bx;

nb.y = by;

nc.x = cx;

nc.y = cy;

col.r = cr;

col.g = cg;

col.b = cb;

intro.r = cir;

intro.g = cig;

intro.b = cib;

set\_a(hwnd, na);

set\_b(hwnd, nb);

set\_c(hwnd, nc);

set\_c\_width(w);

set\_c\_color(col);

set\_c\_brush(intro);

na.x = iax;

na.y = iay;

nb.x = ibx;

nb.y = iby;

nc.x = icx;

nc.y = icy;

interside.set\_a(hwnd, na);

interside.set\_b(hwnd, nb);

interside.set\_c(hwnd, nc);

interside.set\_c\_width(w2);

interside.set\_c\_color(col);

intro.r = 0;

intro.g = 0;

intro.b = 0;

interside.set\_c\_brush(intro);

point ppt[6] = { {a.x, a.y}, {b.x, b.y}, {c.x , c.y}, {interside.get\_a().x, interside.get\_a().y}, {interside.get\_b().x, interside.get\_b().y}, {interside.get\_c().x , interside.get\_c().y} };

if(!checkencl(ppt))throw 6;}

void multi::fsave(FILE \*F){

if(F == NULL) throw 0;

else fprintf(F, "%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d",a.x,a.y,b.x,b.y,c.x,c.y,c\_width,c\_color.r,c\_color.g,c\_color.b,c\_brush\_color.r,c\_brush\_color.g,c\_brush\_color.b,interside.get\_a().x,interside.get\_a().y,interside.get\_b().x,interside.get\_b().y,interside.get\_c().x,interside.get\_c().y,interside.get\_c\_width());}

multi& multi:: operator=(const multi& those){

if(this == &those) return \*this;

else{a.x = those.a.x;

a.y = those.a.y;

b.x = those.b.x;

b.y = those.b.y;

c.x = those.c.x;

c.y = those.c.y;

c\_width = those.c\_width;

interside = those.interside;

c\_brush\_color;

c\_color.r = those.c\_color.r;

c\_color.g = those.c\_color.g;

c\_color.b = those.c\_color.b;

c\_brush\_color.r = those.c\_brush\_color.r;

c\_brush\_color.g = those.c\_brush\_color.g;

c\_brush\_color.b = those.c\_brush\_color.b;}

return \*this;}

**painted.cpp**

include "painted.h"

void painted::paint(){

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

HPEN Pen1 = CreatePen(0, c\_width, RGB(c\_color.r, c\_color.g, c\_color.b));

HPEN hPen = SelectPen (hdc, Pen1);

HBRUSH hBrush = SelectBrush(hdc, get\_so\_brush());

POINT ppt1[3] = { {a.x, a.y}, {b.x, b.y}, {c.x , c.y} };

Polygon(hdc, ppt1, 3);

SelectPen(hdc, hPen);

DeletePen(Pen1);}

void painted::fload(FILE \*F){

HWND hwnd = GetConsoleWindow();

int ax,ay,bx,by,cx,cy,w,cr,cg,cb,cir,cig,cib;

point na,nb,nc;

color col, intro;

fscanf(F, "%d %d %d %d %d %d %d %d %d %d %d %d %d",&ax,&ay,&bx,&by,&cx,&cy,&w,&cr,&cg,&cb,&cir,&cig,&cib);

na.x = ax;

na.y = ay;

nb.x = bx;

nb.y = by;

nc.x = cx;

nc.y = cy;

col.r = cr;

col.g = cg;

col.b = cb;

intro.r = cir;

intro.g = cig;

intro.b = cib;

set\_a(hwnd, na);

set\_b(hwnd, nb);

set\_c(hwnd, nc);

set\_c\_width(w);

set\_c\_color(col);

set\_c\_brush(intro);}

void painted::fsave(FILE \*F){

if(F == NULL) throw 0;

else fprintf(F, "%d %d %d %d %d %d %d %d %d %d %d %d %d", a.x, a.y, b.x, b.y, c.x, c.y, c\_width, c\_color.r, c\_color.g, c\_color.b, c\_brush\_color.r, c\_brush\_color.g, c\_brush\_color.b);}

painted& painted:: operator=(const painted& those){

if(this == &those) return \*this;

else{a.x = those.a.x;

a.y = those.a.y;

b.x = those.b.x;

b.y = those.b.y;

c.x = those.c.x;

c.y = those.c.y;

c\_width = those.c\_width;

c\_brush\_color;

c\_color.r = those.c\_color.r;

c\_color.g = those.c\_color.g;

c\_color.b = those.c\_color.b;

c\_brush\_color.r = those.c\_brush\_color.r;

c\_brush\_color.g = those.c\_brush\_color.g;

c\_brush\_color.b = those.c\_brush\_color.b;}

return \*this;}

***Тестирование***

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Назначение |
| 1 | 10 130 70 40 160 130 3 255 255 255 255 0 0 60 70 90 70 90 120 3 | Проверка корректности отображения очереди из 3-х элементов (загружаем 3 элемента и добавляем их в очередь, затем выбираем пункт “показать графически”) |
| 2 | 10 130 70 40 160 130 3 255 255 255 255 0 0 60 70 90 70 90 120 3 | Проверка корректности взятия элемента из очереди (выбираем пункт “взятие элемента”, затем - “показать графически”) |
| 3 | 10 130 70 40 160 130 3 255 255 255 255 0 0 60 70 90 70 90 120 3 10 130 70 40 160 130 5 255 0 255 45 44 0 60 70 90 70 90 120 1 10 130 70 40 160 130 3 24 25 255 255 0 0 60 70 90 70 90 120 3 | Проверка корректности загрузки очереди из файла (из файла res2 считываем очередь из вложенных треугольников разных цветов, добавляем и выбираем пункт “просмотреть графически”) |
| 4 | 10 130 70 40 160 130 3 255 255 255 255 0 0 10 130 70 40 160 130 5 220 0 55 45 100 0 10 130 70 40 160 130 1 24 25 255 255 44 60 | Проверка корректности загрузки очереди из файла (из файла res3 считываем очередь из закрашенных треугольников разных цветов, добавляем и выбираем пункт “просмотреть графически”) |
| 5 | 10 130 70 40 160 130 3 255 255 255 255 0 0 10 130 70 40 160 130 5 220 0 55 45 100 0 10 130 70 40 160 130 1 24 25 255 255 44 60 | Проверка корректности добавления к очереди, загруженной из файла (из res3) элемента того же типа из другого файла (res1) |