Министерство образования и науки РФ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра теоретической и прикладной информатики

Практическое задание № 2

по дисциплине «Объектно-ориентированное программирование»

**Факультет:** ПМИ

**Группа:**  ПМ-92

**Студенты:**  Иванов В., Попов А.

**Преподаватель:**  Неделько В.М.

Новосибирск

2020

***Постановка задачи***

Разработать программу, в которой был бы определен класс-контейнер (т.е. класс объектов, служащих для хранения объектов класса), реализующий понятие **«открытой» очереди**.

Контейнер должен быть реализован как динамическая структура данных. Разработать следующие функции-члены класса: конструктор, деструктор, функции для помещения объектов-фигур в контейнер, их возвращения (удаления), поиска в контейнере, «распечатки содержимого» контейнера – вывода информации о содержащихся в объекте-контейнере объектах и/или их графических образов, а также функции, обеспечивающие сохраняемость контейнера с использованием файла.

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

***Описание класса container***

struct queue

{

queue \*next; // **элемент списка очереди**

queue \*pred;

triangle k;

};

struct list

{

queue \*head; // **указатели на начало и конец очереди**

queue \*tail;

};

class container

**Для реализации класса определены атрибуты:**

queue \*q; // **указатель на элемент очереди**

list l; // **указатель на структуру list**

**Интерфейс класса:**

container(); // **конструктор класса (инициализация очереди)**

void add (triangle K); // **добавление элемента**

void del(); // **удаление элемента**

void get(triangle \*q); // **взять элемент из очереди (передает в функцию main)**

void text(); // **просмотр элементов в текстовом варианте**

void save(FILE \*F); // **функция сохранения элементов в файл**

~container(); // **деструктор класса (использует функцию del)**

void view(HDC,RECT rt); // **просмотр элементов в графическом варианте**

void empty(); // **проверка очереди на пустоту**

int find();}; // **поиск элемента в очереди**

***Описание класса triangle***

class triangle

**Для реализации класса определены атрибуты:**

point a1, b1, c1, a2, b2, c2; // **координаты вершин треугольников (большого и малого)**

int type, l\_width1, l\_width2, num; // **тип фигуры, ширина границ большого и малого треугольников, номер фигуры в очереди**

color linecol, introcol; // **цвет границы треугольника и цвет заливки**

int reconwindow(HWND hwnd, point a); // **проверка входит ли фигура в область окна**

float init\_point(point to[],int t1,int t2,int t3); // **псевдоскалярное произведение для 3-х точек**

int checkencl(point to[6]); // **проверка вложенности**

int checkpoint(point to[3]); // **проверка является ли фигура треугольником**

**Интерфейс класса:**

void paint(HDC hdc); // **функция рисования фигуры**

inline point get\_a1(){ return a1;} // **получение из класса координат вершин треугольников (большого и малого)**

inline point get\_b1(){ return b1;}

inline point get\_c1(){ return c1;}

inline point get\_a2(){ return a2;}

inline point get\_b2(){ return b2;}

inline point get\_c2(){ return c2;}

inline int get\_type(){ return type;} // **получение из класса типа фигуры**

inline int get\_l\_width1(){return l\_width1;} // **получение ширины контура большого треугольника**

inline int get\_l\_width2(){return l\_width2;} // **получение ширины контура малого треугольника**

inline color get\_linecol(){return linecol;} // **получение цвета контура**

inline color get\_introcol(){return introcol;} // **получение цвета заливки**

void set\_a1(HWND hwnd, point n\_a1); // **установка координат вершин треугольников в классе (большого и малого)**

void set\_b1(HWND hwnd, point n\_b1);

void set\_c1(HWND hwnd, point n\_c1);

void set\_a2(HWND hwnd, point n\_a2);

void set\_b2(HWND hwnd, point n\_b2);

void set\_c2(HWND hwnd, point n\_c2);

void set\_type(int n\_type); // **установка в классе типа фигуры**

void set\_l\_width1(int n\_l\_width1); // **установка ширины контура большого треугольника**

void set\_l\_width2(int n\_l\_width2); // **установка ширины контура малого треугольника**

void set\_linecol(color n\_linecol); // **установка цвета контура**

void set\_introcol(color n\_introcol);}; // **установка цвета заливки**

void setfull(point \*n\_a1, point \*n\_b1, point \*n\_c1, point \*n\_a2, point \*n\_b2, point \*n\_c2, int \*n\_type, int \*n\_l\_width1, int \*n\_l\_width2, color \*n\_linecol, color \*n\_introcol); // **установка всех параметров фигуры**

void getfull(point \*n\_a1, point \*n\_b1, point \*n\_c1, point \*n\_a2, point \*n\_b2, point \*n\_c2, int \*n\_type, int \*n\_l\_width1, int \*n\_l\_width2, color \*n\_linecol, color \*n\_introcol); // **получение всех параметров фигуры**

void fload(FILE \*F, HWND hwnd); // **загрузить очередь из файла**

void fsave(FILE \*F); // **сохранить измененную очередь в файл**

void numr(int \*n);}; // **получения номера позиции искомой фигуры в очереди**

**Функции-нечлены:**

void processing(int error); // **функция выдачи сообщений об ошибках**

***Текст программы***

**main.cpp**

#include <iostream>

#include "class\_container.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;}

case 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;}}}

int main(){

FILE \*in,\*out;

char inp[20];

USES\_CONVERSION;

HBRUSH xBr0=CreateSolidBrush(RGB(0,0,0));

int k = 1;

HWND hwnd = FindWindow (A2W("ConsoleWindowClass"), NULL);

HDC hdc = GetDC (hwnd);

RECT rt;

GetClientRect(hwnd, &rt);

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

point ga1, gb1, gc1, ga2, gb2, gc2;

int gtype, gl\_width1, gl\_width2;

color glinecol, gintrocol;

try{triangle treugolnik;

container m;

container();

l:

printf("Name of the file: **\n**");

scanf("%s",&inp);

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

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

while(!feof(in)){

triangle q;

q.fload(in, hwnd);

m.add(q);}

fclose(in);

do{system("cls");

printf(" 0 - Add element **\n** 1 - Print out **\n** 2 - Show graphically**\n** 3 - Delete the triangle **\n** 4 - Save **\n** 5 - Destructor**\n** 6 - Is the queue empty? **\n** 7 - Search for element **\n** 8 - Exit **\n**");

switch(getch()){case '0':{int a1x, a1y, b1x, b1y, c1x, c1y, a2x, a2y, b2x, b2y, c2x, c2y, t, lw1, lw2,lr, lg, lb, ir, ig, ib;

point n\_a1, n\_b1, n\_c1, n\_a2, n\_b2, n\_c2;

color line, intro;

triangle q; printf("Enter 12 coordinates of 6 points, shape type, line width and 2 colors (six parameters).");

scanf("%i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i", &a1x, &a1y, &b1x, &b1y, &c1x, &c1y, &a2x, &a2y, &b2x, &b2y, &c2x, &c2y, &t, &lw1, &lw2, &lr, &lg, &lb, &ir, &ig, &ib);

n\_a1.x = a1x;

n\_a1.y = a1y;

n\_b1.x = b1x;

n\_b1.y = b1y;

n\_c1.x = c1x;

n\_c1.y = c1y;

n\_a2.x = a2x;

n\_a2.y = a2y;

n\_b2.x = b2x;

n\_b2.y = b2y;

n\_c2.x = c2x;

n\_c2.y = c2y;

line.r = lr;

line.g = lg;

line.b = lb;

intro.r = ir;

intro.g = ig;

intro.b = ib;

q.set\_a1(hwnd, n\_a1);

q.set\_b1(hwnd, n\_b1);

q.set\_c1(hwnd, n\_c1);

q.set\_a2(hwnd, n\_a2);

q.set\_b2(hwnd, n\_b2);

q.set\_c2(hwnd, n\_c2);

q.set\_linecol(line);

q.set\_introcol(intro);

q.set\_type(t);

q.set\_l\_width1(lw1);

q.set\_l\_width2(lw2);

m.add(q);

break;}

case '1':{m.text();

getch();

break;}

case '2':{system("cls");

m.view(hdc,rt);

fill(hwnd, hdc);

break;}

case '3':{treugolnik.getfull(&ga1, &gb1, &gc1, &ga2, &gb2, &gc2, &gtype, &gl\_width1, &gl\_width2, &glinecol, &gintrocol);

m.get(&treugolnik);

treugolnik.paint(hdc);

getch();

break;}

case '4':{out=fopen("out.txt","w");

m.save(out);

getch();

fclose(out);

break;}

case '5':{m.~container();

break;}

case '6':{m.empty();

break;}

case '7':{m.find();

getch();

break;}

case '8':{k=0;

break;}}}

while(k);

return 1;}

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

**triangle.cpp**

void triangle::set\_a1(HWND hwnd, point n\_a1){

if(!reconwindow(hwnd, n\_a1)) throw 1;

else{a1.x = n\_a1.x; a1.y = n\_a1.y;}}

void triangle::set\_b1(HWND hwnd, point n\_b1){

if(!reconwindow(hwnd, n\_b1)) throw 1;

else{b1.x = n\_b1.x; b1.y = n\_b1.y;}}

void triangle::set\_c1(HWND hwnd, point n\_c1){

point pt[3];

pt[0].x = a1.x;

pt[0].y = a1.y;

pt[1].x = b1.x;

pt[1].y = b1.y;

pt[2].x = n\_c1.x;

pt[2].y = n\_c1.y;

if(!reconwindow(hwnd, n\_c1)) throw 1;

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

else{c1.x = n\_c1.x; c1.y = n\_c1.y;}}}

void triangle::set\_a2(HWND hwnd, point n\_a2){

if(!reconwindow(hwnd, n\_a2)) throw 1;

else{a2.x = n\_a2.x; a2.y = n\_a2.y;}}

void triangle::set\_b2(HWND hwnd, point n\_b2){

if(!reconwindow(hwnd, n\_b2)) throw 1;

else{b2.x = n\_b2.x; b2.y = n\_b2.y;}}

void triangle::set\_c2(HWND hwnd, point n\_c2){

point pt[3];

pt[0].x = a2.x;

pt[0].y = a2.y;

pt[1].x = b2.x;

pt[1].y = b2.y;

pt[2].x = n\_c2.x;

pt[2].y = n\_c2.y;

point ptt[6];

ptt[0].x = a1.x;

ptt[0].y = a1.y;

ptt[1].x = b1.x;

ptt[1].y = b1.y;

ptt[2].x = c1.x;

ptt[2].y = c1.y;

ptt[3].x = a2.x;

ptt[3].y = a2.y;

ptt[4].x = b2.x;

ptt[4].y = b2.y;

ptt[5].x = n\_c2.x;

ptt[5].y = n\_c2.y;

if(!reconwindow(hwnd, n\_c2)) throw 1;

else{if(!checkpoint(pt)) throw 2;else{if(!checkencl(ptt)) throw 6;

else{c2.x = n\_c2.x; c2.y = n\_c2.y;}}}}

void triangle::set\_type(int n\_type){

if((n\_type > 3) || (n\_type < 1)) throw 3; else{ ype = n\_type;}}

void triangle::set\_l\_width1(int n\_l\_width1){

if((n\_l\_width1 < 1) || (n\_l\_width1 > 10)) throw 4; else l\_width1 = n\_l\_width1;}

void triangle::set\_l\_width2(int n\_l\_width2){

if((n\_l\_width2 < 1) || (n\_l\_width2 > 10)) throw 4; else l\_width2 = n\_l\_width2;}

void triangle::set\_linecol(color n\_linecol){

if((n\_linecol.r > 255)||(n\_linecol.r < 0)||(n\_linecol.g > 255)||(n\_linecol.g < 0) || (n\_linecol.b > 255) || (n\_linecol.b < 0)) throw 5;

else{linecol.r = n\_linecol.r;

linecol.g = n\_linecol.g;

linecol.b = n\_linecol.b;}}

void triangle::set\_introcol(color n\_introcol){

if((n\_introcol.r > 255) || (n\_introcol.r < 0) ||

(n\_introcol.g > 255) || (n\_introcol.g < 0) ||

(n\_introcol.b > 255) || (n\_introcol.b < 0)) throw 5;

else{introcol.r = n\_introcol.r;

introcol.g = n\_introcol.g;

introcol.b = n\_introcol.b;}}

void triangle::paint(HDC hdc){

switch(type){

case 1:{

HPEN Pen1 = CreatePen(0, l\_width1, RGB(linecol.r, linecol.g, linecol.b));

HPEN hOldPen = SelectPen (hdc, Pen1);

POINT ppt[4]={ {a1.x, a1.y}, {b1.x, b1.y}, {c1.x , c1.y}, {a1.x, a1.y} };

Polyline (hdc, ppt, 4);

SelectPen (hdc, hOldPen);

DeletePen (Pen1);

break;}

case 2:{

HPEN Pen1 = CreatePen(0, l\_width1, RGB(linecol.r, linecol.g, linecol.b));

HPEN hPen = SelectPen (hdc, Pen1);

HBRUSH Brush1 = CreateSolidBrush(RGB(introcol.r, introcol.g, introcol.b));

HBRUSH hBrush = SelectBrush (hdc, Brush1);

POINT ppt1[4] = { {a1.x, a1.y}, {b1.x, b1.y}, {c1.x , c1.y}, {a1.x, a1.y} };

Polygon(hdc, ppt1, 4);

SelectBrush(hdc, hBrush);

DeleteBrush(Brush1);

SelectPen (hdc, hPen);

DeletePen (Pen1);}

case 3:{

HPEN Pen1 = CreatePen(0, l\_width1, RGB(linecol.r, linecol.g, linecol.b));

HPEN Pen2 = CreatePen(0, l\_width2, RGB(linecol.r, linecol.g, linecol.b));

HPEN hPen = SelectPen (hdc, Pen1);

HBRUSH Brush1 = CreateSolidBrush(RGB(introcol.r, introcol.g, introcol.b));

HBRUSH Brush2 = GetStockBrush(BLACK\_BRUSH);

HBRUSH hBrush = SelectBrush (hdc, Brush1);

POINT ppt1[4] = { {a1.x, a1.y}, {b1.x, b1.y}, {c1.x , c1.y}, {a1.x, a1.y} };

POINT ppt2[4] = { {a2.x, a2.y}, {b2.x, b2.y}, {c2.x , c2.y}, {a2.x, a2.y} };

Polygon(hdc, ppt1, 4);

SelectBrush(hdc, hBrush);

DeleteBrush(Brush1);

SelectPen(hdc, hPen);

DeletePen(Pen1);

hPen = SelectPen (hdc, Pen2);

hBrush = SelectBrush (hdc, Brush2);

Polygon(hdc, ppt2, 4);

SelectPen (hdc, hPen);

DeletePen (Pen1);}}}

void triangle::setfull(point \*n\_a1, point \*n\_b1, point \*n\_c1, point \*n\_a2, point \*n\_b2, point \*n\_c2, int \*n\_type, int \*n\_l\_width1, int \*n\_l\_width2, color \*n\_linecol, color \*n\_introcol){a1 = \*n\_a1;

b1 = \*n\_b1;

c1 = \*n\_c1;

a2 = \*n\_a2;

b2 = \*n\_b2;

c2 = \*n\_c2;

type = \*n\_type;

l\_width1 = \*n\_l\_width1;

l\_width2 = \*n\_l\_width2;

linecol = \*n\_linecol;

introcol = \*n\_introcol;}

void triangle::getfull(point \*n\_a1, point \*n\_b1, point \*n\_c1, point \*n\_a2, point \*n\_b2, point \*n\_c2, int \*n\_type, int \*n\_l\_width1, int \*n\_l\_width2, color \*n\_linecol, color \*n\_introcol){\*n\_a1 = a1;

\*n\_b1 = b1;

\*n\_c1 = c1;

\*n\_a2 = a2;

\*n\_b2 = b2;

\*n\_c2 = c2;

\*n\_type = type;

\*n\_l\_width1 = l\_width1;

\*n\_l\_width2 = l\_width2;

\*n\_linecol = linecol;

\*n\_introcol = introcol;}

void triangle::fload(FILE \*F, HWND hwnd){

if(F == NULL) throw 0;

else{ int a1x, a1y, b1x, b1y, c1x, c1y, a2x, a2y, b2x, b2y, c2x, c2y, t, lw1, lw2,

lr, lg, lb, ir, ig, ib;

point n\_a1, n\_b1, n\_c1, n\_a2, n\_b2, n\_c2;

color line, intro;

fscanf(F, "%i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i", &a1x, &a1y, &b1x, &b1y, &c1x, &c1y, &a2x, &a2y, &b2x, &b2y, &c2x, &c2y, &t, &lw1, &lw2, &lr, &lg, &lb, &ir, &ig, &ib);

n\_a1.x = a1x;

n\_a1.y = a1y;

n\_b1.x = b1x;

n\_b1.y = b1y;

n\_c1.x = c1x;

n\_c1.y = c1y;

n\_a2.x = a2x;

n\_a2.y = a2y;

n\_b2.x = b2x;

n\_b2.y = b2y;

n\_c2.x = c2x;

n\_c2.y = c2y;

line.r = lr;

line.g = lg;

line.b = lb;

intro.r = ir;

intro.g = ig;

intro.b = ib;

set\_a1(hwnd, n\_a1);

set\_b1(hwnd, n\_b1);

set\_c1(hwnd, n\_c1);

set\_a2(hwnd, n\_a2);

set\_b2(hwnd, n\_b2);

set\_c2(hwnd, n\_c2);

set\_linecol(line);

set\_introcol(intro);

set\_type(t);

set\_l\_width1(lw1);

set\_l\_width2(lw2);}}

void triangle::fsave(FILE \*F){if(F == NULL) throw 0;

else{ fprintf(F, "%i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i", a1.x, a1.y, b1.x, b1.y, c1.x, c1.y, a2.x, a2.y, b2.x, b2.y, c2.x, c2.y, type, l\_width1, l\_width2, linecol.r, linecol.g, linecol.b, introcol.r, introcol.g, introcol.r);}}

void triangle::numr(int \*n){\*n=num;}

int triangle::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 triangle::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 triangle::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;}

int triangle::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;}

**triangle.h**

#ifndef TRIANGLE\_H\_

#define TRIANGLE\_H\_

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <conio.h>

#include <AtlConv.h>

struct point{int x; int y;};

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

class triangle{

private:

point a1, b1, c1, a2, b2, c2;

int type, l\_width1, l\_width2, num;

color linecol, introcol;

int reconwindow(HWND hwnd, point a);

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

int checkencl(point to[6]);

int checkpoint(point to[3]);

public:

void paint(HDC hdc);

inline point get\_a1(){ return a1;}

inline point get\_b1(){ return b1;}

inline point get\_c1(){ return c1;}

inline point get\_a2(){ return a2;}

inline point get\_b2(){ return b2;}

inline point get\_c2(){ return c2;}

inline int get\_type(){ return type;}

inline int get\_l\_width1(){ return l\_width1;}

inline int get\_l\_width2(){ return l\_width2;}

inline color get\_linecol(){ return linecol;}

inline color get\_introcol(){ return introcol;}

void set\_a1(HWND hwnd, point n\_a1);

void set\_b1(HWND hwnd, point n\_b1);

void set\_c1(HWND hwnd, point n\_c1);

void set\_a2(HWND hwnd, point n\_a2);

void set\_b2(HWND hwnd, point n\_b2);

void set\_c2(HWND hwnd, point n\_c2);

void set\_type(int n\_type);

void set\_l\_width1(int n\_l\_width1);

void set\_l\_width2(int n\_l\_width2);

void set\_linecol(color n\_linecol);

void set\_introcol(color n\_introcol);

void setfull(point \*n\_a1, point \*n\_b1, point \*n\_c1, point \*n\_a2, point \*n\_b2, point \*n\_c2, int \*n\_type, int \*n\_l\_width1, int \*n\_l\_width2, color \*n\_linecol, color \*n\_introcol);

void getfull(point \*n\_a1, point \*n\_b1, point \*n\_c1, point \*n\_a2, point \*n\_b2, point \*n\_c2, int \*n\_type, int \*n\_l\_width1, int \*n\_l\_width2, color \*n\_linecol, color \*n\_introcol);

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

void fsave(FILE \*F);

void numr(int \*n);};

#endif

**container\_functions.cpp**

#include "class\_container.h"

container::container(){

l.head = NULL;

l.tail = l.head;}

void container::add(triangle K){

queue \*a;

a = new queue;

if(!l.head){

l.head = new queue;

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;}

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

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

if(!a) printf("The queue is empty.");

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

delete a;

l.tail = NULL;

l.head = NULL;

printf("Triangle removed. **\n**");}

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

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

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

delete a;

printf("Triangle removed.");}}

else printf("The queue is empty.");}

void container::text(){

queue \*a = l.head;

if(!a) printf("The queue is empty.");

else{a = a -> next;

point ga1, gb1, gc1, ga2, gb2, gc2;

int gtype, gl\_width1, gl\_width2;

color glinecol, gintrocol;

int ga1x, ga1y, gb1x, gb1y, gc1x, gc1y, ga2x, ga2y, gb2x, gb2y, gc2x, gc2y, lcr, lcg, lcb, icr, icg, icb;

int i = 1;

while(a -> next != l.head -> next){a->k.getfull(&ga1, &gb1, &gc1, &ga2, &gb2, &gc2, &gtype, &gl\_width1, &gl\_width2, &glinecol, &gintrocol);

ga1x = ga1.x;

ga1y = ga1.y;

gb1x = gb1.x;

gb1y = gb1.y;

gc1x = gc1.x;

gc1y = gc1.y;

ga2x = ga2.x;

ga2y = ga2.y;

gb2x = gb2.x;

gb2y = gb2.y;

gc2x = gc2.x;

gc2y = gc2.y;

lcr = glinecol.r;

lcg = glinecol.g;

lcb = glinecol.b;

icr = gintrocol.r;

icg = gintrocol.g;

icb = gintrocol.b;

printf("%d Coordinates: %d %d %d %d %d %d %d %d %d %d %d %d **\n** Type: %d **\n** Line width: %d %d **\n** Pen color: %d %d %d **\n** Brush colors: %d %d %d **\n**",i,ga1x, ga1y, gb1x, gb1y, gc1x, gc1y, ga2x, ga2y, gb2x, gb2y, gc2x, gc2y, gtype, gl\_width1, gl\_width2, lcr, lcg, lcb, icr, icg, icb); a = a -> next; i++;}

a -> k.getfull(&ga1, &gb1, &gc1, &ga2, &gb2, &gc2, &gtype, &gl\_width1, &gl\_width2, &glinecol, &gintrocol);

ga1x = ga1.x;

ga1y = ga1.y;

gb1x = gb1.x;

gb1y = gb1.y;

gc1x = gc1.x;

gc1y = gc1.y;

ga2x = ga2.x;

ga2y = ga2.y;

gb2x = gb2.x;

gb2y = gb2.y;

gc2x = gc2.x;

gc2y = gc2.y;

lcr = glinecol.r;

lcg = glinecol.g;

lcb = glinecol.b;

icr = gintrocol.r;

icg = gintrocol.g;

icb = gintrocol.b;

printf("%d Coordinates: %d %d %d %d %d %d %d %d %d %d %d %d **\n** Type: %d **\n** Line width: %d %d **\n** Pen color: %d %d %d **\n** Brush colors: %d %d %d **\n**",i,ga1x, ga1y, gb1x, gb1y, gc1x, gc1y, ga2x, ga2y, gb2x, gb2y, gc2x, gc2y, gtype, gl\_width1, gl\_width2, lcr, lcg, lcb, icr, icg, icb);}}

void container::save(FILE \*F){

queue \*a=l.head;

if(!a) printf("The queue is empty.");

else{a = l.head -> next;

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

a -> k.fsave(F);

a = a -> next;}

a -> k.fsave(F);

printf("Saved.");}}

container::~container(){

while(l.head) del();}

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

queue \*a;

int n = 0;

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

if(!l.head) printf("The queue is empty.");

else{a = l.head -> next;

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

a -> k.numr(&n);

a -> k.paint(hdc);

a = a -> next; getch();

FillRect(hdc,&rt,xBr);}

a -> k.paint(hdc);

getch();}

FillRect(hdc,&rt,xBr);}

void container::empty(){

if(l.head == NULL) printf("The queue is empty.");

else printf("The queue isn’t empty."); getch();}

int container::find()

{int flag = 0;

queue \*qe = l.head;

int a, b, c, i = 1;

color col;

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

printf("Enter the line color: **\n**");

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

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

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

printf("**\n**Element found. #%d",i);}

qe=qe->next;i++;}

col = qe->k.get\_linecol();

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

printf("**\n**Element found. #%d",i);}

if(flag) return 1;

printf("Element not found.");return 0;}

else printf("The queue is empty.");return 0;}

void container::get(triangle \*q){

point ga1, gb1, gc1, ga2, gb2, gc2;

int gtype, gl\_width1, gl\_width2;

color glinecol, gintrocol;

queue \*a = l.head;

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

a -> k.getfull(&ga1, &gb1, &gc1, &ga2, &gb2, &gc2, &gtype, &gl\_width1, &gl\_width2, &glinecol, &gintrocol);

q -> setfull(&ga1, &gb1, &gc1, &ga2, &gb2, &gc2, &gtype, &gl\_width1, &gl\_width2, &glinecol, &gintrocol);

delete a;

l.tail = NULL;

l.head = NULL;}

else{a -> k.getfull(&ga1, &gb1, &gc1, &ga2, &gb2, &gc2, &gtype, &gl\_width1, &gl\_width2, &glinecol, &gintrocol);

q -> setfull(&ga1, &gb1, &gc1, &ga2, &gb2, &gc2, &gtype, &gl\_width1, &gl\_width2, &glinecol, &gintrocol);

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

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

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

delete a;}}

else printf("The queue is empty.");}

**class\_container.h**

#ifndef CLASS\_CONTAINER\_H\_

#define CLASS\_CONTAINER\_H\_

#include "triangle.h"

struct queue

{queue \*next;

queue \*pred;

triangle k;};

struct list

{queue \*head;

queue \*tail;};

class container {

public:

queue \*q;

list l;

container();

void add (triangle K);

void del();

void get(triangle \*q);

void text();

void save(FILE \*F);

~container();

void view(HDC,RECT rt);

void empty();

int find();

};

#endif

***Тестирование***

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Назначение |
| 1 | 20 5 300 15 150 200 25 10 270 20 150 100 1 5 1 255 255 0 0 255 0 20 5 300 15 150 200 25 10 270 20 150 100 2 5 1 255 255 0 0 255 255 | Вывести очередь в консоль графически |
| 2 | 20 5 300 15 150 200 25 10 270 20 150 100 1 5 1 255 255 0 0 255 0 20 5 300 15 150 200 25 10 270 20 150 100 2 5 1 255 255 0 0 255 255 | Распечатать параметры каждой из фигур очереди |
| 3 | 20 5 300 15 150 200 25 10 270 20 150 100 1 5 1 255 255 0 0 255 0 20 5 300 15 150 200 25 10 270 20 150 100 2 5 1 255 255 0 0 255 255 | Найти элемент в очереди (поиск по цвету линии 255 255 0) |
| 4 | 20 5 300 15 150 200 25 10 270 20 150 100 1 5 1 255 255 0 0 255 0 20 5 300 15 150 200 25 10 270 20 150 100 2 5 1 255 255 0 0 255 255 | Добавить в очередь элемент :  20 220 120 40 180 220 60 200 100 120 140 200 2 2 2 255 255 0 0 255 0 |
| 5 | 20 5 300 15 150 200 25 10 270 20 150 100 1 5 1 255 255 0 0 255 0 20 5 300 15 150 200 25 10 270 20 150 100 2 5 1 255 255 0 0 255 0 20 220 120 40 180 220 60 200 100 120 140 200 2 2 2 255 255 0 0 255 0 | Удалить элемент из очереди (удаляется первый элемент) |
| 6 | 20 5 300 15 150 200 25 10 270 20 150 100 1 5 1 255 255 0 0 255 0 | Проверить очередь на пустоту (проверка после очистки всего при помощи деструктора) |
| 7 | 20 5 300 15 150 200 25 10 270 20 150 100 1 5 1 255 255 0 0 255 0 20 5 300 15 150 200 25 10 270 20 150 100 2 5 1 255 255 0 0 255 255 | Найти несуществующий элемент (цвет линии 0 0 0) |