Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ПМт

Лабораторная работа №3

«Объекты»

|  |  |
| --- | --- |
| Факультет: | ПМИ |
| Группа: | ПМ-22 |
| Студент: | Бауыржанулы К. |
| Преподаватель: | Лисицин Д.В. |

**1. Постановка задачи**

1. Ознакомиться с вопросами использования конструкторов и деструкторов для создания и уничтожения объектов.

2. По предложенному преподавателем варианту разработать программу на языке С++, в которой был бы определен класс-контейнер, т.е. класс объектов, служащих для хранения объектов класса, разработанного в предыдущей лабораторной работе. Среди различных типов контейнеров упомянем списки, стеки, очереди, деревья, таблицы. 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.

3. Подготовить текстовые файлы с разработанной программой, оттранслировать их, собрать и выполнить программу с учетом требований операционных систем и программных оболочек, в которых эта программа выполняется. При необходимости исправить ошибки и вновь повторить технологический процесс решения задачи.

4. Оформить отчет по лабораторной работе. Отчет должен содержать постановку задачи, описание разработанных классов, алгоритм, текст разработанной программы и результаты тестирования.

5. Защитить лабораторную работу, ответив на вопросы преподавателя.

**2 Алгоритм решения**

1. Помещение элемента в контейнер:

1. Высчитываем хеш значение помещаемой фигуры.

2. Проверяем занята ли ячейка с этим хеш значением. Если нет, то вставляем, если да, то вставляем в конец строки.

1. Поиск элемента в контейнере:

1. Высчитываем хеш значение фигуры, которую ищем.

2. Проверяем, есть ли элемента в ячейке с этим хеш значением. Если нет, то такого элемента в таблице нет. Если элемента один, то выводим его. Если элементов в этой строке несколько, то надо уточнить оставшиеся параметры фигуры, после чего ищем её, если нашли, то выводим, иначе такой фигуры нету

1. Возвращение элемента с удалением:

1. Высчитываем хеш значение фигуры, которую ищем.

2. Проверяем, есть ли элемента в ячейке с этим хеш значением. Если нет, то такого элемента в таблице нет. Если элемента один, то выводим его. Если элементов в этой строке несколько, то надо уточнить оставшиеся параметры фигуры, после чего ищем её, если нашли, то выводим, иначе такой фигуры нет

3. Удаляем найденную фигуру из таблицы.

4. Если элемент в строке таблицы был не один, то смещаем элементы, следующие за нашей фигурой на один влево.

**3. Описание функций**

**3.1 main**

Прототип: void main ( )

Назначение: Инициализация входных данных программы. Реализация меню работы программы.

**3.2 prov**

Прототип: int prov(POINT b[4])

Назначение: Проверка: лежит ли второй параллелограмм в первом.

Входные параметры:

b[4]-массив координат диагонали.

Выходные параметры:

1 – если второй квадрат лежит в первом

0 – если второй квадрат не лежит в первом

**3.3 print\_kord**

Прототип: void print\_kord(POINT a[2])

Назначение: Вывод на экран координат.

Входные параметры:

a – массив координат.

**3.4 print\_lin**

Прототип: void print\_lin(int m[5])

Назначение: Вывод на экран характеристик линии.

Входные параметры:

m – массив характеристик.

**3.5 print\_str**

Прототип: void print\_str(int m[5])

Назначение: Вывод на экран характеристик штриховки.

Входные параметры:

m – массив характеристик.

**3.6 read\_kord**

Прототип: void read\_kord(POINT (\*a)[4])

Назначение: Считывание с консоли координат.

Входные параметры:

а – массив координат.

Выходные параметры:

а – массив координат.

**3.7 read\_lin**

Прототип: void read\_lin(int (\*m)[5])

Назначение: Считывание с консоли характеристик линии.

Входные параметры:

m – массив характеристик.

Выходные параметры:

m – массив характеристик.

**3.8 read\_str**

Прототип: void read\_str(int (\*m)[5])

Назначение: Считывание с консоли характеристик штриховки .

Входные параметры:

m – массив характеристик.

Выходные параметры:

m – массив характеристик.

Прототип: void strn ()

Назначение: Считывание новых характеристик штриховки.

**4. Строение класса:**

class box

{

private:

int lintip,lintol;//тип линии, толщина линии

int color[2][3]; // цвет линии и штриховки

int strtip; //тип штриховки

POINT kor[2]; //координаты

public:

void input(ifstream \*f);//ввод данных

void output(ofstream \*f);//вывод данных

void dvig(int k,int c);//смещение квадрата

int prov();//проверка: является ли квадратом

void fon();//отрисовка фона

void one();//контур

void two();//заштрихованный квадрат

void set\_str(int m[5]);//изменение характеристик штриховки

void set\_kord(POINT a[4]);//изменение координат

void set\_lin(int m[5]);//изменяет характеристики линии

void get\_lin(int (\*m)[5]);//возврат характеристик линии

void get\_str(int (\*m)[5]);//возврат характеристик штриховки

POINT get\_kord1();//возврат верхней координаты диагонали квадрата

POINT get\_kord2();//возврат нижней координаты диагонали квадрата

};

**4.1 input**

Прототип: void input(ifstream \*f)

Назначение: Считывание из файла данных.

Входные параметры:

f – указатель на файл.

**4.2 output**

Прототип: void output(ofstream \*f)

Назначение: Вывод в файл данных.

Входные параметры:

f – указатель на файл.

**4.3 dvig**

Прототип: void dbig(int k,int c)

Назначение: Смещение фигуры.

Входные параметры:

c– кол-во пикселей.

k – направление смещения.

**4.4 prov**

Прототип: void prov()

Назначение: Проверка: является ли квадратом.

**4.5 fon**

Прототип: void fon()

Назначение: Отрисовка фона.

**4.6 one**

Прототип: void one()

Назначение: Отрисовка контура.

**4.7 two**

Прототип: void two()

Назначение: Отрисовка закрашенного квадрата.

**4.8 set\_str**

Прототип: void set\_str(int m[5])

Назначение: Изменение характеристик штриховки.

Входные параметры:

m– массив характеристик.

**4.9 set\_kord**

Прототип: void set\_kord(POINT a[4])

Назначение: Изменение координат.

Входные параметры:

a– массив координат.

**4.10 set\_lin**

Прототип: void set\_lin(int m[5])

Назначение: Изменение характеристик линии.

Входные параметры:

m– массив характеристик.

**4.11 get\_lin**

Прототип: void get\_lin(int (\*m)[5])

Назначение: Возврат характеристик линии.

Входные параметры:

m– массив характеристик.

Выходные параметры:

m– массив характеристик.

**4.12 get\_str**

Прототип: void get\_str(int (\*m)[5])

Назначение: Возврат характеристик штриховки.

Входные параметры:

m– массив характеристик.

Выходные параметры:

m– массив характеристик.

**4.13 get\_kord1**

Прототип: POINT get\_kord1();

Назначение: Возврат верхней координаты диагонали квадрата.

Выходные параметры:

координаты верхней точки диагонали квадрата.

**4.14 get\_kord2**

Прототип: POINT get\_kord2();

Назначение: Возврат нижней координаты диагонали квадрата.

Выходные параметры:

координаты нижней точки диагонали квадрата.

**5. Строение контейнера:**

struct elem

{

box \*el;

elem \*next;

};

struct table

{

elem \*el;

elem \*first;

};

class container

{

private:

table \*T; //указатель на таблицу

int n; //размер таблицы

int keys; //что является ключом

public:

container(int kolvo,int k); //конструктор

~container(); //деструктор

box \*find(); //поиск элемента,возвращает указатель на элемент

box \*pop(); //возвращение элемента с удалением

void push(box a); //помещение элемента в контейнер

void output(string f); //вывод содержимого контейнера в файл f

void print(); //вывод содержимого контейнера на экран

int check(int m[5],int m1[5],

int k[5],int k1[5], //проверка

POINT a[2],POINT b[2]

);

};

**5.1 container**

Прототип: container(int kolvo,int k)

Назначение: Создание хеш-таблицы.

Входные параметры:

kolvo – размер таблицы.

k – что является ключом

**5.2 ~container**

Прототип: ~container()

Назначение: Удаление хеш-таблицы.

**5.3 filling\_container\_data**

Прототип: void filling\_container\_data();

Назначение: Заполнение N и Keys

**5.4 find**

Прототип: box \*find()

Назначение: Поиск элемента в таблицы

Выходные параметры:

box \*– указатель на элемент.

**5.5. pop**

Прототип: box \*pop()

Назначение: Возвращение элемента с удалением из таблицы

Выходные параметры:

box \*– указатель на элемент.

**5.6 push**

Прототип: void push(box a)

Назначение: Помещение элемента в контейнер

Входные параметры:

a – элемент типа класса Box.

**5.7 output**

Прототип: void output(string f)

Назначение: Сохранение данных в файл.

Входные параметры:

f – название файла.

**5.8 print**

Прототип: void print()

Назначение: Вывод содержимого хеш-таблицы на экран.

**5.9 check**

Прототип: int check(int m[5],int m1[5],int k[5],int k1[5],POINT a[2],

POINT b[2])

Назначение: Сравнивает характеристики фигур.

Входные параметры:

m– массив характеристик линии 1-ой фигуры,

m1 – массив характеристик штриховки 1-ой фигуры,

a – координаты 1-ой фигуры,

k – массив характеристик линии 2-ой фигуры,

k1 – массив характеристик штриховки 2-ой фигуры,

b – координаты 2-ой фигуры.

Выходные параметры:

0 – фигуры разные,

1 – фигуры одинаковые.

**6. Текст программы**

**Файл “main.cpp”**

#include "container.h"

void main()

{setlocale(LC\_CTYPE,"Russian");

container a(20,1); //переменная для контейнера

box \*b; //переменная для характеристик квадрата

b=new box;

int key; //переменная для ключа

int i=0;

do{system("cls");

cout<<"------------------------------------------"<<endl;

cout<<"| 1-Считывание информации из файла | "<<endl;

cout<<"| 2-Вывод данных на экран | "<<endl;

cout<<"| 3-Нарисовать контур фигуры | "<<endl;

cout<<"| 4-Нарисовать заштрихованную фигуру | "<<endl;

cout<<"| 5-Нарисовать фигуру в фигуре | "<<endl;

cout<<"| 6-Изменение данных | "<<endl;

cout<<"| 7-Смещение фигур | "<<endl;

cout<<"| 8-Сохранение данных в файл | "<<endl;

cout<<"| 9-Выход | "<<endl;

cout<<"------------------------------------------"<<endl;

cin>>i;

switch(i){

case 1:{

system("cls");

cout<<"Введите название файла"<<endl;

string t\_fin;

cin>>t\_fin;

ifstream fin (t\_fin);

int er=0;

if(fin==NULL) cout<<"Ошибка в открытие файла"<<endl;

else

{while(!fin.eof())

{try{b->input(&fin);}catch(int error){er++;if(error==0)std::cout<<"Error in"<<er<<" box"<<std::endl;else a->push(\*b);}}

fin.close();

cout<<"Считывание из файла было произведено"<<endl;}

getch();}break;

case 2:{system("cls");a->print();getch();}break;

case 3:{system("cls");if((b=(a->find()))!=NULL){system("cls");b->one();getch();b->fon();}getch();}break;

case 4:{system("cls");if((b=(a->find()))!=NULL){system("cls");b->two();getch();b->fon();}getch();}break;

case 5:{box \*b1;b1=new box;system("cls");if((b=(a->find()))!=NULL && (b1=(a->find()))!=NULL){system("cls");

three(\*b,\*b1);}getch();}break;

case 6:{system("cls");

cout<<"---------------------------------"<<endl;

cout<<"| 1-Добавить новую фигуру | "<<endl;

cout<<"| 2-Изменить данные | "<<endl;

cout<<"| 3-Выход | "<<endl;

cout<<"---------------------------------"<<endl;

int number;cin>>number;

switch(number){

case 1:{system("cls");POINT c[4];int m[5],m1[5];try{read\_all(&c,&m,&m1);}

catch(int error){if(error==1){b->set\_kord(c);b->set\_lin(m);b->set\_str(m1);a->push(\*b);cout<<"Новая фигура была добавлена"<<endl;}}

getch();}break;

case 2:{system("cls");box \*d=new box;;if((d=a->pop())!=NULL){

system("cls");

cout<<"----------------------------------------"<<endl;

cout<<"| 1-Изменение координат | "<<endl;

cout<<"| 2-Изменение характеристик линии | "<<endl;

cout<<"| 3-Изменение характеристик линии | "<<endl;

cout<<"| 4-Выход | "<<endl;

cout<<"----------------------------------------"<<endl;

int number1;cin>>number1;switch(number1){

case 1:{POINT c[4];try{read\_kord(&c);}

catch(int error){if(error==1){d->set\_kord(c);a->push(\*d);}

else std::cout<<"Ошибка ввода координат"<<std::endl;}}break;

case 2:{int m[5];try{read\_lin(&m);}catch(int error){if(error==1){d->set\_lin(m);a->push(\*d);}else std::cout<<"Ошибка ввода характеристик линии"<<std::endl;}}break;

case 3:{int m[5];try{read\_str(&m);}catch(int error){if(error==1){d->set\_str(m);a->push(\*d);}else std::cout<<"Ошибка ввода характеристик штриховки"<<std::endl;}

}break;}}}break;}}break;

case 7:{system("cls");box \*c=new box;if((c=a->pop())!=NULL){int j,k;

cout<<"1-Смещение вверх"<<endl;

cout<<"2-Смещение вправо"<<endl;

cout<<"3-Смещение вниз"<<endl;

cout<<"4-Смещение влево"<<endl;

cout<<"Через пробел введите на сколько пикселей смещать"<<endl;

cin>>j>>k;c->dvig(j,k);a->push(\*c);

cout<<"Смещение было произведено"<<endl;}

getch();system("cls");}break;

case 8:{

system("cls");

cout<<"------------------------------------------"<<endl;

cout<<"| 1-Сохранить данные всего контейнера | "<<endl;

cout<<"| 2-Сохранить данные отдельной фигуры | "<<endl;

cout<<"| 3-Выход | "<<endl;

cout<<"------------------------------------------"<<endl;

int number;cin>>number;

switch(number){

case 1:{system("cls");out(a);cout<<"Данные успешно сохранены"<<endl;getch();}break;

case 2:{system("cls");if((b=(a->find()))!=NULL){cout<<"Введите название файла"<<endl;string c;cin>>c;

ofstream f (c);b->output(&f);cout<<"Данные успешно сохранены"<<endl;}getch();}break;}}break;}}while(i!=9);

delete a;}

**Файл “box.h”**

#pragma once

#ifndef Box

#define Box

#include <windows.h>

#include <windowsx.h>

#include <fstream>

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include <iostream>

using namespace std;

class box

{

private:

int lintip,lintol;//тип линии, толщина линии

int color[2][3]; // цвет линии и штриховки

int strtip; //тип штриховки

POINT kor[2]; //координаты

public:

void input(ifstream \*f);//ввод данных

void output(ofstream \*f);//вывод данных

void dvig(int k,int c);//смещение квадрата

void fon();//отрисовка фона

void one();//контур

void two();//заштрихованный квадрат

void set\_str(int m[5]);//изменение характеристик штриховки

void set\_kord(POINT a[4]);//изменение координат

void set\_lin(int m[5]);//изменяет характеристики линии

void get\_lin(int (\*m)[5]);//возврат характеристик линии

void get\_str(int (\*m)[5]);//возврат характеристик штриховки

POINT get\_kord1();//возврат верхней координаты диагонали квадрата

POINT get\_kord2();//возврат нижней координаты диагонали квадрата

};

void three(box a,box a1);//фигура в фигуре

int prov(box a,box a1);//проверка на вложенность

void print\_kord(POINT a[2]);//показывает координаты

void print\_lin(int m[5]);//вывод характеристик линии

void print\_str(int m[5]);//вывод характеристик штриховки

void read\_kord(POINT (\*a)[4]);//ввод координат

void read\_lin(int (\*m)[5]);//ввод характеристик линии

void read\_str(int (\*m)[5]);//ввод характеристик штриховки

void read\_all(POINT (\*a)[4],int (\*m)[5],int (\*m1)[5]);//ввод новой фигуры

#endif

**Файл “box.cpp”**

#include "box.h"

//считывание из файла

void box::input(ifstream \*f)

{RECT rt;POINT a[2];int m[5],m1[4];

HWND hwnd = GetConsoleWindow ( );

GetClientRect (hwnd, &rt);

char buf[100];(\*f)>>a[0].x;(\*f)>>a[0].y;(\*f)>>a[1].x;(\*f)>>a[1].y;(\*f)>>m[0];(\*f)>>m[1];(\*f)>>m[2];(\*f)>>m[3];

(\*f)>>m[4];(\*f)>>m1[0];(\*f)>>m1[1];(\*f)>>m1[2];(\*f)>>m1[3];

sprintf (buf, "Размер окна %d на %d пикселей",rt.right, rt.bottom);

if(abs(a[0].x-a[1].x)==abs(a[0].y-a[1].y) && a[1].x<=rt.right && a[1].y<=rt.bottom

&& m[0]<8 && m[0]>0 && m[1]>0 && m[2]<=255 && m[2]>=0 && m[3]<=255 && m[3]>=0 && m[4]<=255 && m[4]>=0

&& m1[0]>=0 && m1[0]<7 && m1[1]<=255 && m1[1]>=0 && m1[2]<=255 && m1[2]>=0 && m1[3]<=255 && m1[3]>=0)

{kor[0].x=a[0].x;kor[0].y=a[0].y;kor[1].x=a[1].x;kor[1].y=a[1].y;lintip=m[0];lintol=m[1];color[0][0]=m[2];

color[0][1]=m[3];color[0][2]=m[4];strtip=m1[0];color[1][0]=m1[1];color[1][1]=m1[2];color[1][2]=m1[3];

throw 1;}else throw 0;}

//отрисовка фона

void box::fon()

{RECT rt;HWND hwnd = GetConsoleWindow ( );

GetClientRect (hwnd, &rt);

HDC hdc = GetDC (hwnd);

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

HBRUSH Brush0 = CreateSolidBrush (RGB(0, 0, 0));

HBRUSH hOldBrush = SelectBrush (hdc, Brush0);

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

SelectBrush (hdc, Brush0);

DeleteBrush (Brush0);

system("cls");}

//отрисовка контура

void box::one()

{HPEN Pen;HPEN hOldPen;HBRUSH Brush1;HBRUSH hOldBrush;RECT rt;

HWND hwnd = GetConsoleWindow ( );

GetClientRect (hwnd, &rt);

HDC hdc = GetDC (hwnd);

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

switch(lintip)

{case 1:{Pen = CreatePen (PS\_SOLID,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen =SelectPen (hdc, Pen);}break;

case 2:{Pen = CreatePen (PS\_DASH,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 3:{Pen = CreatePen (PS\_DOT,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 4:{Pen = CreatePen (PS\_DASHDOT,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 5:{Pen = CreatePen (PS\_DASHDOTDOT,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 6:{Pen = CreatePen (PS\_NULL,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 7:{Pen = CreatePen (PS\_INSIDEFRAME,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;}

Brush1 = CreateSolidBrush (RGB(0, 0, 0));

hOldBrush = SelectBrush (hdc, Brush1);

Rectangle (hdc,kor[0].x,kor[0].y,kor[1].x,kor[1].y);

SelectPen (hdc, hOldPen);DeletePen (Pen);SelectPen (hdc, hOldBrush);DeletePen (Brush1);ReleaseDC (hwnd, hdc);}

//отрисовка закрашенной фигуры

void box::two()

{HPEN Pen;HPEN hOldPen;HBRUSH Brush2;HBRUSH hOldBrush;RECT rt;

HWND hwnd = GetConsoleWindow ( );GetClientRect (hwnd, &rt);

HDC hdc = GetDC (hwnd);SetBkColor (hdc, RGB(0, 0, 0));

switch(lintip)

{case 1:{Pen = CreatePen (PS\_SOLID,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen =SelectPen (hdc, Pen);}break;

case 2:{Pen = CreatePen (PS\_DASH,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 3:{Pen = CreatePen (PS\_DOT,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 4:{Pen = CreatePen (PS\_DASHDOT,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 5:{Pen = CreatePen (PS\_DASHDOTDOT,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 6:{Pen = CreatePen (PS\_NULL,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;

case 7:{Pen = CreatePen (PS\_INSIDEFRAME,lintol, RGB(color[0][0],color[0][1],color[0][2]));hOldPen = SelectPen (hdc, Pen);}break;}

switch(strtip)

{case 0:{Brush2 = CreateSolidBrush (RGB(color[1][0],color[1][1],color[1][2]));hOldBrush = SelectBrush (hdc,Brush2);}break;

case 1:{Brush2 = CreateHatchBrush(HS\_BDIAGONAL,RGB(color[1][0],color[1][1],color[1][2]));hOldBrush = SelectBrush (hdc,Brush2);}break;

case 2:{Brush2 = CreateHatchBrush(HS\_CROSS,RGB(color[1][0],color[1][1],color[1][2]));hOldBrush = SelectBrush (hdc,Brush2);}break;

case 3:{Brush2 = CreateHatchBrush(HS\_DIAGCROSS,RGB(color[1][0],color[1][1],color[1][2]));hOldBrush = SelectBrush (hdc,Brush2);}break;

case 4:{Brush2 = CreateHatchBrush(HS\_FDIAGONAL,RGB(color[1][0],color[1][1],color[1][2]));hOldBrush = SelectBrush (hdc,Brush2);}break;

case 5:{Brush2 = CreateHatchBrush(HS\_HORIZONTAL,RGB(color[1][0],color[1][1],color[1][2]));hOldBrush = SelectBrush (hdc,Brush2);}break;

case 6:{Brush2 = CreateHatchBrush(HS\_VERTICAL,RGB(color[1][0],color[1][1],color[1][2]));hOldBrush = SelectBrush (hdc,Brush2);}break;}

Rectangle (hdc, kor[0].x,kor[0].y,kor[1].x,kor[1].y);SelectPen (hdc, hOldPen);DeletePen (Pen);

SelectPen (hdc, hOldBrush);DeletePen (Brush2);ReleaseDC (hwnd, hdc);}

//вывод в файл

void box::output(ofstream \*f)

{(\*f)<<kor[0].x<<" ";(\*f)<<kor[0].y<<endl;(\*f)<<kor[1].x<<" ";(\*f)<<kor[1].y<<endl;(\*f)<<lintip<<" ";

(\*f)<<lintol<<" ";(\*f)<<color[0][0]<<" ";(\*f)<<color[0][1]<<" ";(\*f)<<color[0][2]<<endl;(\*f)<<strtip<<" ";

(\*f)<<color[1][0]<<" ";(\*f)<<color[1][1]<<" ";(\*f)<<color[1][2]<<"\n"<<endl;}

//сдвиг фигуры

void box::dvig(int k,int c)

{switch(k)

{case 1:{kor[0].y-=c;kor[1].y-=c;}break;

case 2:{kor[0].x+=c;kor[1].x+=c;}break;

case 3:{kor[0].y+=c;kor[1].y+=c;}break;

case 4:{kor[0].x-=c;kor[1].x-=c;}break;}}

//возврат значений 1-ой координаты

POINT box::get\_kord1()

{return kor[0];}

//возврат значений 2-ой координаты

POINT box::get\_kord2()

{return kor[1];}

//возврат значений штриховки

void box::get\_str(int (\*m)[5])

{(\*m)[0]=strtip;(\*m)[2]=color[1][0];(\*m)[3]=color[1][1];(\*m)[4]=color[1][2];}

//возврат значений линии

void box::get\_lin(int (\*m)[5])

{(\*m)[0]=lintip;(\*m)[1]=lintol;(\*m)[2]=color[0][0];(\*m)[3]=color[0][1];(\*m)[4]=color[0][2];}

//изменение координат

void box::set\_kord(POINT a[2])

{kor[0].x=a[0].x;kor[0].y=a[0].y;kor[1].x=a[1].x;kor[1].y=a[1].y;}

//изменение характеристик штриховки

void box::set\_str(int m[5])

{strtip=m[0];color[1][0]=m[2];color[1][1]=m[3];color[1][2]=m[4];}

//изменение характеристик линии

void box::set\_lin(int m[5])

{lintip=m[0];lintol=m[1];color[0][0]=m[2];color[0][1]=m[3];color[0][2]=m[4];}

//рисует фигуру в фигуре

void three(box a,box a1)

{if(prov(a,a1)){a.two();a1.two();}else cout<<"Фигуры не вложены"<<endl;getch();a.fon();}

//вывод на экран координат

void print\_kord(POINT a[4])

{cout<<"x1= "<<a[0].x;cout<<"\ny1= "<<a[0].y;cout<<"\nx2= "<<a[1].x;cout<<"\ny2= "<<a[1].y;}

//вывод на экран характеристик линии

void print\_lin(int m[5])

{cout<<"Тип линии "<<m[0];cout<<"\nТолщина линии "<<m[1];

cout<<"\nЦвет RGB "<<m[2]<<" "<<m[3]<<" "<<m[4];}

//вывод на экран характеристик штриховки

void print\_str(int m[5])

{cout<<"Тип штриховки "<<m[0];cout<<"\nЦвет RGB "<<m[2]<<" "<<m[3]<<" "<<m[4];}

//проверка на вложенность

int prov(box a,box a1)

{POINT b[4];b[0]=a.get\_kord1();b[1]=a.get\_kord2();b[2]=a1.get\_kord1();b[3]=a1.get\_kord2();

if(b[0].x<b[2].x && b[1].y>b[3].y && b[1].x>b[3].x && b[0].y<b[2].y)return 1;else return 0;}

//чтение координат

void read\_kord(POINT (\*a)[4])

{cout<<"\nВведите координаты верхней точки диагонали"<<endl;

cin>>(\*a)[0].x>>(\*a)[0].y;

cout<<"\nВведите координаты нижней точки диагонали"<<endl;

cin>>(\*a)[1].x>>(\*a)[1].y;RECT rt;

HWND hwnd = GetConsoleWindow ( );

GetClientRect (hwnd, &rt);char buf[100];

sprintf (buf, "Размер окна %d на %d пикселей",rt.right, rt.bottom);

if(abs((\*a)[0].x-(\*a)[1].x)==abs((\*a)[0].y-(\*a)[1].y) && (\*a)[1].x<=rt.right && (\*a)[1].y<=rt.bottom)throw 1;else throw 0;}

//чтение характеристик линии

void read\_lin(int (\*m)[5])

{cout<<"Введите тип линии"<<endl;cin>>(\*m)[0];

cout<<"Введите толщину линии"<<endl;cin>>(\*m)[1];

cout<<"Введите цвет RGB"<<endl;cin>>(\*m)[2]>>(\*m)[3]>>(\*m)[4];

if((\*m)[0]>7 && (\*m)[0]<1 && (\*m)[1]<1 && (\*m)[2]>255 && (\*m)[2]<0 && (\*m)[3]>255 && (\*m)[3]<0 && (\*m)[4]>255 && (\*m)[4]<0)throw 0;else throw 1;}

//чтение характеристик штриховки

void read\_str(int (\*m)[5])

{cout<<"\nВведите тип штриховки"<<endl;cin>>(\*m)[0];

cout<<"\nВведите цвет RGB"<<endl;cin>>(\*m)[2]>>(\*m)[3]>>(\*m)[4];

if((\*m)[0]>6 && (\*m)[0]<0 && (\*m)[2]>255 && (\*m)[2]<0 && (\*m)[3]>255 && (\*m)[3]<0 && (\*m)[4]>255 && (\*m)[4]<0)throw 0;else throw 1;}

//ввод новой фигуры

void read\_all(POINT (\*a)[4],int (\*m)[5],int (\*m1)[5])

{try{read\_kord(a);}catch(int error){if(error==1)try{read\_lin(m);}catch(int error){if(error==1)

try{read\_str(m1);}catch(int error){if(error==1)throw 1;if(error==0){std::cout<<"Ошибка в характеристиках штриховки"<<std::endl;throw 0;}}else std::cout<<"Ошибка в характеристиках линии"<<std::endl;}

else std::cout<<"Ошибка в координатах"<<std::endl;}}

**Файл “container.h”**

#pragma once

#ifndef Container

#define Container

#include "box.h"

#include <string>

struct elem{box \*el;elem \*next;};

struct table{elem \*el;elem \*first;};

class container

{private:

table \*T; //указатель на таблицу

int n; //размер таблицы

int keys; //что является ключом

public:

container(int kolvo,int k); //конструктор

~container(); //деструктор

box \*find(); //поиск элемента,возвращает указатель на элемент

box \*pop(); //возвращение элемента с удалением

void push(box a); //помещение элемента в контейнер

void output(string f); //вывод содержимого контейнера в файл f

void print(); //вывод содержимого контейнера на экран

int check(int m[5],int m1[5],

int k[5],int k1[5], //проверка

POINT a[2],POINT b[2]

);};

void out(container \*a); //вывод элементов в файл

#endif

**Файл “container.cpp”**

include "container.h"

//конструктор

container::container(int kolvo,int k)

{T=NULL;keys=k;n=kolvo;T=new table[n];for(int i=0;i<n;i++)T[i].first=NULL;}

//деструктор

container::~container()

{if(T!=NULL)

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

if(T[i].first!=NULL)

while(T[i].el->next!=NULL)

{delete T[i].el->el;

T[i].el=T[i].el->next;}

delete T;}

//помещение элемента в контейнер

void container::push(box a)

{table \*P=T;elem \*c;c=new elem;c->next=NULL;c->el=new box;\*(c->el)=a;int key;int position;

switch(keys){

case 1:{POINT b[2];b[0]=a.get\_kord1();b[1]=a.get\_kord2();key=b[0].x+b[0].y+b[1].x+b[1].y;position=(key\*17+1)%n;

if(P[position].first==NULL)

{P[position].el=c;P[position].first=P[position].el;}else

{while(P[position].el->next!=NULL){P[position].el=P[position].el->next;}

P[position].el->next=new elem;P[position].el->next=c;P[position].el=P[position].first;}

}break;

case 2:{int m[5];a.get\_lin(&m);key=m[0]+m[1]+m[2]+m[3]+m[4];position=(key\*17+1)%n;if(P[position].first==NULL)

{P[position].el=c;P[position].el->next=NULL;P[position].first=P[position].el;}else

{while(P[position].el->next!=NULL)

{P[position].el=P[position].el->next;}P[position].el->next=new elem;P[position].el->next=c;P[position].el=P[position].first;}}break;

case 3:{int m[5];a.get\_str(&m);key=m[0]+m[2]+m[3]+m[4];position=(key\*17+1)%n;if(P[position].first==NULL){

P[position].el=c;P[position].first=P[position].el;}else

{while(P[position].el->next!=NULL){P[position].el=P[position].el->next;}

P[position].el->next=new elem;P[position].el->next=c;P[position].el=P[position].first;}}break;}}

//проверка нужную ли фигуру мы нашли

int container::check(int m[5],int m1[5],int k[5],int k1[5],POINT a[2],POINT b[2])

{for(int i=0;i<5;i++){if(m[i]!=k[i] || k1[i]!=m1[i])return 0;if(i<2) if(a[i].x!=b[i].x || a[i].y!=b[i].y)return 0;}return 1;}

//поиск элемента,возвращает указатель на элемент

box \*container::find()

{int key;int position;

switch(keys){

case 1:{POINT b[2];

cout<<"Введите координаты"<<endl;cin>>b[0].x>>b[0].y>>b[1].x>>b[1].y;

key=b[0].x+b[0].y+b[1].x+b[1].y;position=(key\*17+1)%n;

if(T[position].first!=NULL)

{if(T[position].el->next==NULL)return T[position].el->el;else {

cout<<"Элементов с таким ключом несколько"<<endl;

cout<<"Введите характеристики линии"<<endl;

int m[5];cin>>m[0]>>m[1]>>m[2]>>m[3]>>m[4];

cout<<"Введите характеристики штриховки"<<endl;

int m1[5];cin>>m1[0]>>m1[2]>>m1[3]>>m1[4];

int k[5],k1[5];POINT c[2];

T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();

while(check(m,m1,k,k1,b,c)!=1 && T[position].el->next!=NULL)

{T[position].el=T[position].el->next;T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();}

if(check(m,m1,k,k1,b,c)!=1) {cout<<"Такого элемента нету"<<endl; return NULL;}else return T[position].el->el;}

}else{cout<<"Такого элемента нету"<<endl; return NULL;}}break;

case 2:{int m[5];cout<<"Характеристики линии"<<endl;cin>>m[0]>>m[1]>>m[2]>>m[3]>>m[4];

key=m[0]+m[1]+m[2]+m[3]+m[4];position=(key\*17+1)%n;

if(T[position].first!=NULL){if(T[position].el->next==NULL)return T[position].el->el;

else {cout<<"Элементов с таким ключом несколько"<<endl;POINT b[2];

cout<<"Введите координаты"<<endl;cin>>b[0].x>>b[0].y>>b[1].x>>b[1].y;

cout<<"Введите характеристики штриховки"<<endl;int m1[5];cin>>m1[0]>>m1[2]>>m1[3]>>m1[4];

int k[5],k1[5];POINT c[2];T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();

while(check(m,m1,k,k1,b,c)!=1 && T[position].el->next!=NULL)

{T[position].el=T[position].el->next;T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();}

if(check(m,m1,k,k1,b,c)!=1) {cout<<"Такого элемента нету"<<endl; return NULL;}

else return T[position].el->el;}}else

{cout<<"Такого элемента нету"<<endl; return NULL;}}break;

case 3:{cout<<"Введите характеристики штриховки"<<endl;

int m1[5];cin>>m1[0]>>m1[2]>>m1[3]>>m1[4];

key=m1[0]+m1[2]+m1[3]+m1[4];position=(key\*17+1)%n;

if(T[position].first!=NULL)

{if(T[position].el->next==NULL)return T[position].el->el;else {

cout<<"Элементов с таким ключом несколько"<<endl;POINT b[2];

cout<<"Введите координаты"<<endl;cin>>b[0].x>>b[0].y>>b[1].x>>b[1].y;

cout<<"Введите характеристики линии"<<endl;

int m[5];cin>>m[0]>>m[1]>>m[2]>>m[3]>>m[4];

int k[5],k1[5];POINT c[2];T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();

while(check(m,m1,k,k1,b,c)!=1 && T[position].el->next!=NULL)

{T[position].el=T[position].el->next;T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();}

if(check(m,m1,k,k1,b,c)!=1) {cout<<"Такого элемента нету"<<endl; return NULL;}else return T[position].el->el;

}}else{cout<<"Такого элемента нету"<<endl; return NULL;}}break;}}

//возвращение элемента с удалением

box \*container::pop()

{box \*rez;int key;int position;

switch(keys){

case 1:{POINT b[2];cout<<"Введите координаты"<<endl;cin>>b[0].x>>b[0].y>>b[1].x>>b[1].y;

key=b[0].x+b[0].y+b[1].x+b[1].y;position=(key\*17+1)%n;

if(T[position].first!=NULL)

{if(T[position].el->next==NULL){rez=T[position].el->el;T[position].el=NULL;return rez;}

else {cout<<"Элементов с таким ключом несколько"<<endl;

cout<<"Введите характеристики линии"<<endl;

int m[5];cin>>m[0]>>m[1]>>m[2]>>m[3]>>m[4];

cout<<"Введите характеристики штриховки"<<endl;int m1[5];

cin>>m1[0]>>m1[2]>>m1[3]>>m1[4];int k[5],k1[5];POINT c[2];

T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();

if(check(m,m1,k,k1,b,c)==1)

{T[position].first=T[position].el->next;rez=T[position].el->el;delete T[position].el;T[position].el=T[position].first;

return rez;}

while(check(m,m1,k,k1,b,c)!=1 && T[position].el->next->next!=NULL)

{T[position].el->next->el->get\_lin(&k);T[position].el->next->el->get\_str(&k1);

c[0]=T[position].el->next->el->get\_kord1();c[1]=T[position].el->next->el->get\_kord2();

if(T[position].el->next->next!=NULL && check(m,m1,k,k1,b,c)!=1)T[position].el=T[position].el->next;}

if(check(m,m1,k,k1,b,c)!=1)

{T[position].el->next->el->get\_lin(&k);T[position].el->next->el->get\_str(&k1);

c[0]=T[position].el->next->el->get\_kord1();c[1]=T[position].el->next->el->get\_kord2();

if(check(m,m1,k,k1,b,c)!=1)

{cout<<"Такой фигуры нету"<<endl;return NULL;}

else {rez=T[position].el->next->el;T[position].el->next=NULL;return rez;}}

elseif(T[position].el->next!=NULL)

{rez=T[position].el->next->el;T[position].el->next=T[position].el->next->next;T[position].el=T[position].first;

return rez;}

else

{rez=T[position].el->el;delete T[position].el;T[position].el=T[position].first;return rez;}}

}else{cout<<"Такой фигуры нету"<<endl;return NULL;}}break;

case 2:{int m[5];

cout<<"Характеристики линии"<<endl;cin>>m[0]>>m[1]>>m[2]>>m[3]>>m[4];

key=m[0]+m[1]+m[2]+m[3]+m[4];position=(key\*17+1)%n;

if(T[position].first!=NULL)

{if(T[position].el->next==NULL){rez=T[position].el->el;T[position].el=NULL;return rez;}

else {cout<<"Элементов с таким ключом несколько"<<endl;POINT b[2];

cout<<"Введите координаты"<<endl;cin>>b[0].x>>b[0].y>>b[1].x>>b[1].y;

cout<<"Введите характеристики штриховки"<<endl;int m1[5];cin>>m1[0]>>m1[2]>>m1[3]>>m1[4];

int k[5],k1[5];POINT c[2];T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();

if(check(m,m1,k,k1,b,c)==1)

{T[position].first=T[position].el->next;rez=T[position].el->el;delete T[position].el;T[position].el=T[position].first;

return rez;}

while(check(m,m1,k,k1,b,c)!=1 && T[position].el->next->next!=NULL)

{T[position].el->next->el->get\_lin(&k);T[position].el->next->el->get\_str(&k1);

c[0]=T[position].el->next->el->get\_kord1();c[1]=T[position].el->next->el->get\_kord2();

if(T[position].el->next->next!=NULL && check(m,m1,k,k1,b,c)!=1)T[position].el=T[position].el->next;}

if(check(m,m1,k,k1,b,c)!=1)

{T[position].el->next->el->get\_lin(&k);T[position].el->next->el->get\_str(&k1);

c[0]=T[position].el->next->el->get\_kord1();c[1]=T[position].el->next->el->get\_kord2();

if(check(m,m1,k,k1,b,c)!=1){cout<<"Такой фигуры нету"<<endl;return NULL;}

else {rez=T[position].el->next->el;T[position].el->next=NULL;return rez;}}elseif(T[position].el->next!=NULL)

{rez=T[position].el->next->el;T[position].el->next=T[position].el->next->next;T[position].el=T[position].first;

return rez;}

else

{rez=T[position].el->el;delete T[position].el;T[position].el=T[position].first;return rez;}}}else

{cout<<"Такой фигуры нету"<<endl;return NULL;}}break;

case 3:{cout<<"Введите характеристики штриховки"<<endl;int m1[5];cin>>m1[0]>>m1[2]>>m1[3]>>m1[4];

key=m1[0]+m1[2]+m1[3]+m1[4];position=(key\*17+1)%n;

if(T[position].first!=NULL)

{if(T[position].el->next==NULL){rez=T[position].el->el;T[position].el=NULL;return rez;}

else {cout<<"Элементов с таким ключом несколько"<<endl;POINT b[2];cout<<"Введите координаты"<<endl;

cin>>b[0].x>>b[0].y>>b[1].x>>b[1].y;cout<<"Введите характеристики линии"<<endl;

int m[5];cin>>m[0]>>m[1]>>m[2]>>m[3]>>m[4];int k[5],k1[5];POINT c[2];

T[position].el->el->get\_lin(&k);T[position].el->el->get\_str(&k1);

c[0]=T[position].el->el->get\_kord1();c[1]=T[position].el->el->get\_kord2();

if(check(m,m1,k,k1,b,c)==1)

{T[position].first=T[position].el->next;rez=T[position].el->el;delete T[position].el;T[position].el=T[position].first;

return rez;}

while(check(m,m1,k,k1,b,c)!=1 && T[position].el->next->next!=NULL)

{T[position].el->next->el->get\_lin(&k);T[position].el->next->el->get\_str(&k1);

c[0]=T[position].el->next->el->get\_kord1();c[1]=T[position].el->next->el->get\_kord2();

if(T[position].el->next->next!=NULL && check(m,m1,k,k1,b,c)!=1)T[position].el=T[position].el->next;}

if(check(m,m1,k,k1,b,c)!=1)

{T[position].el->next->el->get\_lin(&k);T[position].el->next->el->get\_str(&k1);

c[0]=T[position].el->next->el->get\_kord1();c[1]=T[position].el->next->el->get\_kord2();

if(check(m,m1,k,k1,b,c)!=1){cout<<"Такой фигуры нету"<<endl;return NULL;}

else {rez=T[position].el->next->el;T[position].el->next=NULL;return rez;}}

else

if(T[position].el->next!=NULL)

{rez=T[position].el->next->el;T[position].el->next=T[position].el->next->next;T[position].el=T[position].first;

return rez;}

else

{rez=T[position].el->el;delete T[position].el;T[position].el=T[position].first;return rez;}}}else

{cout<<"Такой фигуры нету"<<endl;return NULL;}}break;}}

//вывод элементов на экран

void container::print()

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

if(T[i].first!=NULL)

{while(T[i].el!=NULL)

{POINT a[2];int b[5];a[0]=T[i].el->el->get\_kord1();a[1]=T[i].el->el->get\_kord2();T[i].el->el->get\_lin(&b);

print\_kord(a);cout<<endl;print\_lin(b);cout<<endl;T[i].el->el->get\_str(&b);print\_str(b);cout<<"\n\n\n\n\n";

T[i].el=T[i].el->next;}T[i].el=T[i].first;}}

//сохранение элементов в файл

void out(container \*a)

{string f;cout<<"Введите название файл"<<endl;cin>>f;a->output(f);}

//сохранение элементов

void container::output(string f)

{ofstream fout(f);for(int i=0;i<n;i++)

if(T[i].first!=NULL)

{while(T[i].el!=NULL){

POINT a[2];int b[5];a[0]=T[i].el->el->get\_kord1();a[1]=T[i].el->el->get\_kord2();T[i].el->el->get\_lin(&b);

print\_kord(a);fout<<endl;print\_lin(b);fout<<endl;T[i].el->el->get\_str(&b);print\_str(b);fout<<"\n\n\n\n\n";

T[i].el=T[i].el->next;}T[i].el=T[i].first;}

**7.Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Результат | Примечание |
| 1 | 30 30  150 150  3 1 255 0 0  4 0 0 255  50 50  100 100  4 1 0 255 0  0 0 0 0 |  | Все данные корректны |
| 2 | 50 50  200 200  3 1 255 0 0  4 0 0 255  50 50  200 200  4 1 0 255 0  0 0 0 0 | Error | Данные второго квадрата не корректны(имеет одинаковый размер с первым) |
| 3 | 50 700  700 50  3 1 255 0 0  4 0 0 255  50 700  700 50  4 1 0 255 0  0 0 0 0 | Error | Данные не корректны(размер квадрата выходит за размер окна) |
| 4 | 37 74  200 200  3 1 255 0 0  4 0 0 255  50 50  150 150  4 1 0 255 0  0 0 0 0 | Error | Данные не корректны(первая фигура не является квадратом) |

**8. Результаты отладки**

Программа работает верно, что подтверждается результатами тестов.