МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННО БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Учебный Центр Информационных Технологий «Информатика»

Лабораторная работа № 2

по дисциплине «Объектно-ориентированное программирование»

Направление подготовки: 230105 - «Программное обеспечение вычислительной техники и автоматизированных систем»

Выполнил слушатель: Сокотущенко Иван Игоревич
Вариант: 2
Дата сдачи:
Преподаватель: Силов Я. В.

Новосибирск, 2017г.

1. Задание

Изучить понятие объекта, средства его создания и уничтожения.

По предложенному преподавателем варианту разработать про-грамму на языке C++, 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.

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.

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

Защитить лабораторную работу, ответив на вопросы преподавателя.

Вариант задания №2 – выпуклый четырёхугольник.

Пусть все объекты класса, реализующего геометрическую фигуру, разделены на несколько категорий по значению некоторого признака (ввести в рассмотрение некоторый «нетривиальный» признак, либо считать отнесение фигуры к конкретной категории прерогативой пользователя). Разработать класс, реализующий понятие таблицы с дубликатами (повторяющимися ключами), в качестве ключа использовать номер категории. Функция поиска ищет элементы определенной категории. Таблицу реализовать на основе двунаправленного списка (обеспечить «распечатку» содержимого как в прямом, так и в обратном направлении);

2. Структурное описание

В данной лабораторной работе мы реализуем класс **’’Container.h’’**. Для удобства будем называть его «класс». Также из предыдущей лабораторной работы имеем два класса - ’’Quadrangle.h’’ и ’’View.h’’.

Для реализации данного класса ’’Container.h’’ созданы три файла. Заголовочный файл и два файла исходного кода - ’’Container_methods.cpp’’ и ’’ main.cpp’’. В заголовочном файле ’’Container.h’’ описывается интерфейс (тело) класса, в файле ’’Container_methods.cpp’’ реализованы методы класса, в классе main.cpp’’ реализуется демонстрацию работы класса.

Класс содержит в себе закрытые (private) члены класса:

- атрибуты, необходимые для реализации класса:
  - структура **link** (имеет три поля Quadrangle *quadrangle, link *next, link *prev), в которой хранятся объекты класса и указатели на другие объекты класса – следующий и предыдущий;
  - int **count –** целочисленное значение, хранящее количество объектов в классе Container.

В открытые (public) члены класса входят:

- конструктор класса по умолчанию **Container()**;
- деструктор **~ Container()**;
- функция, добавляющая объекты в контейнер **addFigure()**;
- функция, удаляющая объекты в контейнере **removeFigure()**;
- функция, поиска объекта по его порядковому номеру **searchFigureOnNumber()**;
- функция, поиска объекта по некоторому «нетривиальному» признаку **searchFigureOnSign()**;
- функция, обеспечивающая «распечатку» содержимого в прямом направлении **printInfo()**;
- функция, обеспечивающая «распечатку» содержимого в обратном направлении **printInfoBack()**;
- функция сохранения данных (координат четырёхугольника) в файл **saveInFile()**.

В файле “main.cpp” размещена функция int main(), которая демонстрирует поведение класса.

3. Функциональное описание

Рассмотрим реализацию методов класса.

Подключаем заголовочный файл ’’Container.h’’.

Public:

- конструктор класса по умолчанию **Container()** – запускается автоматически при создании нового объекта класса без параметров, инициализирует поля класса значениями по умолчанию. Т. к. контейнер пуст, количество объектов равняется нулю (count =0), голова (head = NULL) и хвост (tail = NULL) двунаправленного списка равняются нулю;
- деструктор **~Container()** – запускается автоматически при удалении контейнера и освобождает выделенную для создания контейнера память. Для этого создаётся цикл while, условием завершения которого является наличие одного объекта. На каждом шаге цикла хвостом (tail) списка считается объект, следующий за головой (head), после установки хвоста (tail) на этот следующий объект, голова (head) удаляется посредством вызова оператора delete и головой (head) становится хвост (tail) списка. То есть происходит по объектное удаление элементов списка, начиная с его головы (head);
- **void addFigure(Quadrangle &quadrangle) -** принимает в качестве параметра ссылку на объект типа Quadrangle. Далее метод создаёт структуру типа link и инициализирует поле quadrangle этой структуры значением (объектом), переданным в качестве фактического параметра метода. При этом устанавливаются связи списка - хвост (tail) инициализируется значением NULL, а при инициализации головы (head) возможны два варианта. Первый – когда список пуст и добавляемый объект является первым, тогда голова (head) является одновременно и хвостом (tail) списка и она же хранит переданный по ссылке объект. Второй вариант – когда в списке уже есть объекты, тогда хвост (tail) списка направляется на последний элемент структуры link, затем следующему за хвостом (tail) элементу присваивается значение переданного по ссылке объекта (структуру link) и, наконец, хвостом (tail) становится этот объект. После всех проделанных действий значение счётчика объектов в контейнере count увеличивается на один;
- **void removeFigure(int number)** – принимает в качестве параметра целочисленное значение номера удаляемого объекта. При этом перед выполнением метода происходит проверка на корректность заданного номера – если переданное значение не лежит в диапазоне от 1 до текущего count, то генерируется исключение «Такого номера не существует.». Метод работает следующим образом: сначала устанавливается голова (head) списка на начало структуры link. Возможны четыре случая:

а) удаляется первый, но не единственный, элемент в списке, тогда голова (head) перенаправляется на следующий элемент, первый (удаляемый) элемент инициализируется NULL, объект удаляется с помощью оператора delete, счётчик количества объектов в контейнере count уменьшается на единицу и происходит выход из метода.

б) второй случай – когда удаляется первый элемент в списке и он же является единственным, тогда голова (head) инициализируется нулём, происходит удаление объекта с помощью оператора delete, счётчик количества объектов count обнуляется и происходит выход из метода.

в) третий случай – когда удаляемый объект является последним в списке. Для определения этого вводится целочисленное значение pos, которое инициализируется значением количества элементов в списке (т.е. с помощью цикла while пробегаем по всем элементам списка и на каждом элементе значение pos увеличиваем на единицу). Алгоритм работы в этом случае следующий – хвост (tail) устанавливается на последний элемент списка, потом перенаправляется на предыдущий, а последний инициализируется NULL, а объект удаляется посредством оператора delete. Счётчик количества объектов count уменьшается на единицу и происходит выход из метода.

г) наконец, случай удаления объекта из середины списка. Алгоритм удаления следующий: записать адрес узла, следующего за удаляемым узлом, в указатель на следующий узел узла, являющегося предыдущим для удаляемого узла. Затем, записать адрес узла, являющегося предыдущим для удаляемого, в указатель на предыдущий узел узла, следующего за удаляемым узлом. И наконец, удалить узел, предназначенный для удаления.

- **void searchFigureOnNumber(int number)** – принимает в качестве параметра целочисленное значение номера нужного объекта. При этом перед выполнением метода происходит проверка на корректность заданного номера – если переданное значение не лежит в диапазоне от 1 до текущего count, то генерируется исключение «Такого номера не существует.». Сначала устанавливается голова (head) списка на начало структуры link. Вводится целочисленное значение id. Далее, с помощью цикла while пробегаемся по всем объектам в контейнере, на каждом шаге увеличивая значение id. Если фактиче-

ское значение параметра, переданного на вход метода совпадёт со значением id, то нужный элемент найдет, происходит вывод на консоль его геометрических параметров (берутся из класса Quadrangle) и происходит выход из метода.

- **void searchFigureOnSign(string sign) -** принимает в качестве параметра строку со значением «нетривиального» признака. В качестве такого признака принимается значение «big» или «small». Четырёхугольник считается «small», если его площадь менее 5000 пикселей. Введём дополнительно в класс Quadrangle поле с таким «нетривиальным» признаком – для этого в файле «Quadrangle.h» добавим поле «string notTrivialSign», а в файле «Qudrangle_methods.cpp» добавим целочисленное значение площади S, которое вычисляет значение площади по координатам вершин четырёхугольника. В случае, если значение площади S конкретного четырёхугольника меньше 5000, notTrivialSign = "small", в противном случае notTrivialSign = "big".

Перед выполнением метода searchFigureOnSign() происходит проверка на корректность заданного признака – если переданное значение не является "small" или "big", то генерируется исключение «Такой категории не существует.». Алгоритм работы данного метода аналогичен алгоритму работы метода searchFigureOnNumber(), только условием вывода на консоль геометрической информации об объектах является совпадение по признаку.

- **void printInfo()** – выводит на консоль информацию, об объектах, хранящихся в контейнере. Для этого голова (head) устанавливается на начало структуры link, задаётся цикл while, условием завершения которого является достижение пустого объекта (хвоста) списка. На каждом шаге цикла происходит движение к следующему элементу списка и вывод информации о геометрических параметрах текущего объекта (четырёхугольника), которая хранится в классе quadrangle.
- **void printInfoBack()** – выводит на консоль информацию, об объектах, хранящихся в контейнере, но в обратном порядке. Алгоритм работы метода аналогичен алгоритму работы метода printInfo(), только хвост (tail) устанавливается на конец структуры link, и на каждом шаге цикла while движение идёт в обратном направлении (к предыдущему элементу списка).
- **void saveInFile(string fd) -** метод сохраняет данные (геометрические параметры четырёхугольников) в файл, название которого задаётся в качестве фактического параметра. Для этого с помощью функций из стандартной библиотеки <fstream> файл создаётся, в него записываются координаты через пробел, и файл закрывается.

4. Описание работы программы

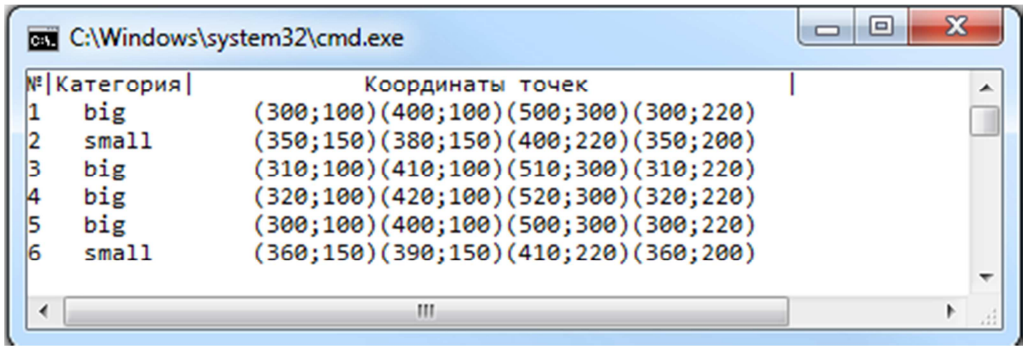

Рис. 1 – Вывод в консоль информации об объектах, хранящихся в контейнере.

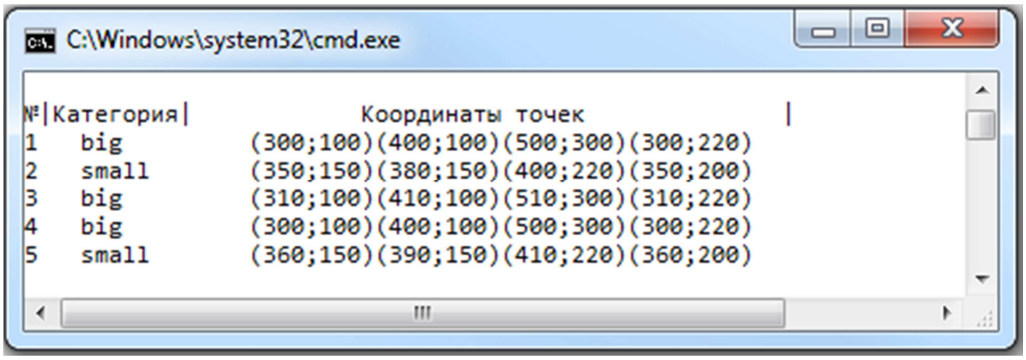

Рис. 2 – Демонстрация работы метода удаления объекта по заданному порядковому номеру.

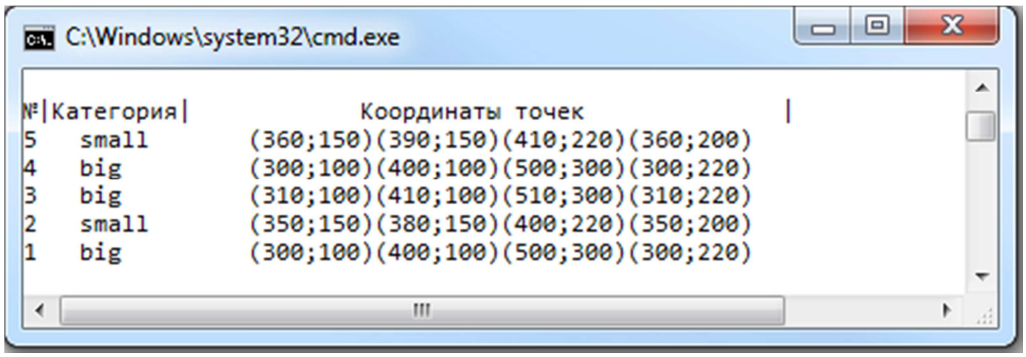

Рис. 3 - Вывод в консоль информации об объектах, хранящихся в контейнере (в обратном порядке).

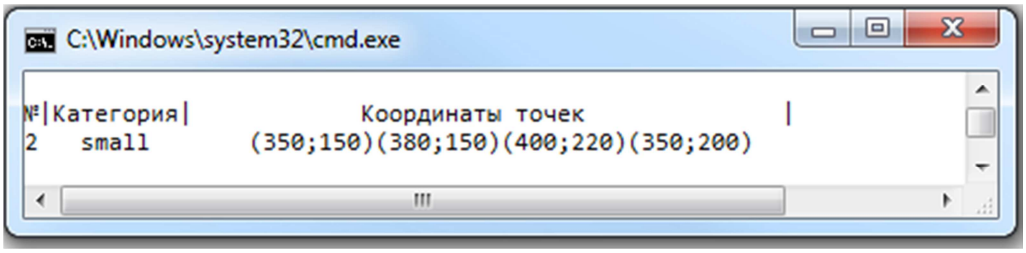

Рис. 4 - Демонстрация работы метода поиска объекта по заданному номеру.

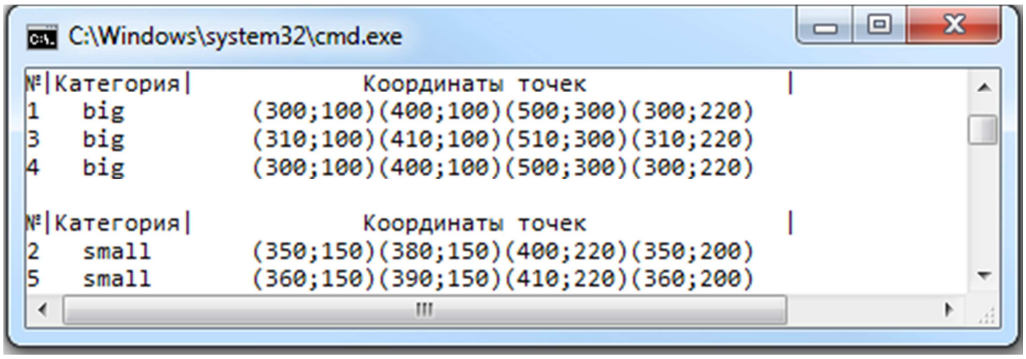

Рис. 5 - Демонстрация поиска объектов в контейнере по «нетривиальному» признаку.

Выводы

В ходе выполнения лабораторной работы были изучены понятия объекта, средства его создания и уничтожения.

Ссылка на github.com: **https://github.com/IvanSokotushenko/OOPlab2/tree/master/Lab2var2v3**.

Приложение. Текст программы

```
#pragma once
#include "Quadrangle.h"
#include "View.h"
#include <fstream>
using namespace std;
struct link
{
	Quadrangle *quadrangle;
	link *next, *prev;
};
class Container
{
	int count;
	link *head, *tail;
public:
	Container();
	~Container();
	void addFigure(Quadrangle &quadrangle);
	void removeFigure(int number);
	void searchFigureOnNumber(int number);
	void searchFigureOnSign(string sign);
	void printInfo();
	void printInfoBack();
	void saveInFile(string fd);
};
```

```
#include "Container.h"
using namespace std;
Container::Container()
{
	head = NULL;
	tail = NULL;
	count = 0;
}
Container::~Container()
{
	while (head)
	{
		tail = head->next;
		delete head;
		head = tail;
	}
}
void Container::addFigure(Quadrangle &quadrangle)
{
	link *temp = new link;
	temp->next = NULL;
	temp->quadrangle = &quadrangle;
	if (head != NULL)
	{
		temp->prev = tail;
		tail->next = temp;
		tail = temp;
	}
	else
	{
		temp->prev = NULL;
		head = tail = temp;

	}
	count++;
}
```

```
void Container::printInfo()
{
	link *temp = head;
	int id = 1;
	cout << "№" << "|Категория" << "|\t	Координаты точек\t	|";
	cout << "\n";
	while (temp != NULL)
	{
cout << id;
cout << "	" << temp->quadrangle->notTrivialSign << " ";
cout << "\t(" << temp->quadrangle->points[0].x << ";" << temp->quadrangle->points[0].y << ")";
cout << "(" << temp->quadrangle->points[1].x << ";" << temp->quadrangle->points[1].y << ")";
cout << "(" << temp->quadrangle->points[2].x << ";" << temp->quadrangle->points[2].y << ")";
cout << "(" << temp->quadrangle->points[3].x << ";" << temp->quadrangle->points[3].y << ")";
cout << "\n";
id++;
temp = temp->next;
	}
	cout << "\n";
}

void Container::printInfoBack()
{
	link *temp = tail;
	int id = count;
	cout << "№" << "|Категория" << "|\t	Координаты точек\t	|";
	cout << "\n";
	while (temp != NULL)
	{
cout << id;
cout << "	" << temp->quadrangle->notTrivialSign << " ";
cout << "\t(" << temp->quadrangle->points[0].x << ";" << temp->quadrangle->points[0].y << ")";
cout << "(" << temp->quadrangle->points[1].x << ";" << temp->quadrangle->points[1].y << ")";
cout << "(" << temp->quadrangle->points[2].x << ";" << temp->quadrangle->points[2].y << ")";
cout << "(" << temp->quadrangle->points[3].x << ";" << temp->quadrangle->points[3].y << ")";
cout << "\n";
id--;
temp = temp->prev;
	}
	cout << "\n";
}

void Container::saveInFile(string fd)
{
	ofstream out(fd, ios::out);
	link *temp = head;
	int id = 1;
	out << "№" << "|Категория" << "|\t	Координаты точек\t	|";
	out << "\n";
	while (temp != NULL)
	{
out << id;
out << "	" << temp->quadrangle->notTrivialSign << " ";
out << "\t(" << temp->quadrangle->points[0].x << ";" << temp->quadrangle->points[0].y << ")";
out << "(" << temp->quadrangle->points[1].x << ";" << temp->quadrangle->points[1].y << ")";
out << "(" << temp->quadrangle->points[2].x << ";" << temp->quadrangle->points[2].y << ")";
out << "(" << temp->quadrangle->points[3].x << ";" << temp->quadrangle->points[3].y << ")";
out << "\n";
id++;
temp = temp->next;
	}
	out << "\n";
	out.close();
}
```

```
void Container::removeFigure(int number)
{
	if (number<1 || number > count)
		throw exception("Такого номера не существует.\n");
	link *temp = head;
	if (number == 1 && head->next)
	{
		head = head->next;
		head->prev = NULL;
		delete temp;
		count--;
		return;
	}
	else if (count == 1 && head == tail)
	{
		head->next = NULL;
		head = NULL;
		delete head;
		count = 0;
		return;
	}

	int pos = 0;
	while (temp != NULL)
	{
		temp = temp->next;
		pos++;
	}

	if (number == pos)
	{
		link *temp = tail;
		tail = tail->prev;
		tail->next = NULL;
		delete temp;
		count--;
		return;
	}
	link *tempNew = head;
	link *tempDel;
	for (int i = 0; i < number - 1; i++)
	{
		tempNew = tempNew->next;
	}
	tempDel = tempNew;
	tempDel->prev->next = tempNew->next;
	tempDel->next->prev = tempNew->prev;
	delete tempNew;
	count--;
}

void Container::searchFigureOnNumber(int number)
{
	if (number<1 || number > count)
		throw exception("Такого номера не существует.\n");
	link *temp = head;
	int id = 1;
	while (temp != NULL)
	{
		if (id == number)
		{
cout << "№" << "|Категория" << "|\t	Координаты точек\t	|";
cout << "\n";
cout << id;
cout << "	" << temp->quadrangle->notTrivialSign << " ";
cout << "\t(" << temp->quadrangle->points[0].x << ";" << temp->quadrangle->points[0].y << ")";
cout << "(" << temp->quadrangle->points[1].x << ";" << temp->quadrangle->points[1].y << ")";
cout << "(" << temp->quadrangle->points[2].x << ";" << temp->quadrangle->points[2].y << ")";
cout << "(" << temp->quadrangle->points[3].x << ";" << temp->quadrangle->points[3].y << ")";
```

```
cout << "\n";
cout << "\n";
return;
		}
		id++;
		temp = temp->next;
	}
}

void Container::searchFigureOnSign(string sign)
{
	if (sign != "big  " && sign != "small")
		throw exception("Такой категории не существует.\n");
	link *temp = head;
	int id = 1;
	cout << "№" << "|Категория" << "|\t	Координаты точек\t	|";
	cout << "\n";
	while (temp != NULL)
	{
		if (sign == temp->quadrangle->notTrivialSign)
		{
cout << id;
cout << "    " << temp->quadrangle->notTrivialSign << " ";
cout << "\t(" << temp->quadrangle->points[0].x << ";" << temp->quadrangle->points[0].y << ")";
cout << "(" << temp->quadrangle->points[1].x << ";" << temp->quadrangle->points[1].y << ")";
cout << "(" << temp->quadrangle->points[2].x << ";" << temp->quadrangle->points[2].y << ")";
cout << "(" << temp->quadrangle->points[3].x << ";" << temp->quadrangle->points[3].y << ")";
cout << "\n";
		}
		id++;
		temp = temp->next;
	}
	cout << "\n";
}


#pragma once
#include <windows.h>
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
struct node
{
	int x, y;
};

class Quadrangle
{
	POINT *points;
	node leftTop, rightTop, rightBottom, leftBottom, change;
	string notTrivialSign;
	bool isBulge(node leftTop, node rightTop, node rightBottom, node leftBottom);
	bool isOnTheNested(Quadrangle *internalQuadrangle);
	friend class View;
	friend class Container;
public:
	Quadrangle();
	~Quadrangle();
	Quadrangle(node leftTop, node rightnTop, node rightBottom, node leftBottom);
	POINT *setQuadrangle(node leftTop, node rightTop, node rightBottom, node leftBottom);
	POINT *getQuadrangle();
	void enterCoordinatesFromFiles(string fd);
	void saveCoordinatesToFile(string fd);
	void changePosition(node *change);
};
```

```cpp
#include "Quadrangle.h"
using namespace std;
bool Quadrangle::isBulge(node leftTop, node rightTop, node rightBottom, node leftBottom)
{
	return (((rightTop.x - leftTop.x)*(rightBottom.y - leftTop.y) -
		(rightTop.y - leftTop.y)*(rightBottom.x - leftTop.x))*
		((rightTop.x - leftTop.x)*(leftBottom.y - leftTop.y) -
		(rightTop.y - leftTop.y)*(leftBottom.x - leftTop.x))<0 ||
		((leftBottom.x - rightBottom.x)*(rightTop.y - rightBottom.y) -
		(leftBottom.y - rightBottom.y)*(rightTop.x - rightBottom.x))*
		((leftBottom.x - rightBottom.x)*(leftTop.y - rightBottom.y) -
		 (leftBottom.y - rightBottom.y)*(leftTop.x - rightBottom.x))<0);
}

bool Quadrangle::isOnTheNested(Quadrangle *internalQuadrangle)
{
	POINT *points1 = this->getQuadrangle();
	POINT *points2 = internalQuadrangle->getQuadrangle();
	return ((points2[0].x < points1[0].x) || (points2[0].y < points1[0].y) ||
		(points2[1].x > points1[1].x) || (points2[1].y < points1[1].y) ||
		(points2[2].x > points1[2].x) || (points2[2].y > points1[2].y) ||
		(points2[3].x < points1[3].x) || (points2[3].y > points1[3].y));
}

POINT *Quadrangle::setQuadrangle(node leftTop, node rightTop, node rightBottom, node leftBottom)
{
	if (isBulge(leftTop, rightTop, rightBottom, leftBottom))
		throw exception("Четырёхугольник не является выпуклым!");
	points[0] = { leftTop.x, leftTop.y };
	points[1] = { rightTop.x, rightTop.y };
	points[2] = { rightBottom.x, rightBottom.y };
	points[3] = { leftBottom.x, leftBottom.y };
}

POINT* Quadrangle::getQuadrangle()
{
	return this->points;
}


Quadrangle::Quadrangle()
{
	points = new POINT[4];
	points[0] = { 300, 100 };
	points[1] = { 400, 100 };
	points[2] = { 300, 200 };
	points[3] = { 500, 300 };
}

Quadrangle::~Quadrangle()
{
	delete[] getQuadrangle();
}

Quadrangle::Quadrangle(node leftTop, node rightTop, node rightBottom, node leftBottom)
{
	if (isBulge(leftTop, rightTop, rightBottom, leftBottom))
		throw exception("Четырёхугольник не является выпуклым!");
	points = new POINT[4];
	points[0] = { leftTop.x, leftTop.y };
	points[1] = { rightTop.x, rightTop.y };
	points[2] = { rightBottom.x, rightBottom.y };
	points[3] = { leftBottom.x, leftBottom.y };
	int S = ((leftTop.x - rightTop.x)*(leftTop.y + rightTop.y) +
		(rightTop.x - rightBottom.x)*(rightTop.y + rightBottom.y) +
		(rightBottom.x - leftBottom.x)*(rightBottom.y + leftBottom.y) +
		(leftBottom.x - leftTop.x)*(leftBottom.y + leftTop.y));
```

```
	if (S >= 5000)
		notTrivialSign = "big  ";
	else
	{
		notTrivialSign = "small";
	}
}

void Quadrangle::enterCoordinatesFromFiles(string fd)
{
	ifstream in(fd, ios::in);
	if (in.is_open())
	{
		POINT *points = this->getQuadrangle();
		in >> points[0].x;
		in >> points[0].y;
		in >> points[1].x;
		in >> points[1].y;
		in >> points[2].x;
		in >> points[2].y;
		in >> points[3].x;
		in >> points[3].y;
		if (isBulge(leftTop, rightTop, rightBottom, leftBottom))
			throw exception("Четырёхугольник не является выпуклым!");
		in.close();
	}
	else
		throw exception("В этом файле нет нужных данных!");
}

void Quadrangle::saveCoordinatesToFile(string fd)
{
	ofstream out(fd, ios::out);
	out << this->points[0].x;
	out << " ";
	out << this->points[0].y;
	out << " ";
	out << this->points[1].x;
	out << " ";
	out << this->points[1].y;
	out << " ";
	out << this->points[2].x;
	out << " ";
	out << this->points[2].y;
	out << " ";
	out << this->points[3].x;
	out << " ";
	out << this->points[3].y;
	out << " ";
	out.close();
}

void Quadrangle::changePosition(node *change)
{
	this->change = *change;
	setQuadrangle({ this->points[0].x + this->change.x, points[0].y + this->change.y },
	{ points[1].x + this->change.x, points[1].y + this->change.y },
	{ points[2].x + this->change.x, points[2].y + this->change.y },
	{ points[3].x + this->change.x, points[3].y + this->change.y });
}
```

```
#pragma once
#pragma once
#include <windowsx.h>
#include "Quadrangle.h"
using namespace std;
struct color
{
	short R;
	short G;
	short B;
};

class View
{
	HWND hwnd;
	HDC hdc;
	RECT rt;
	HPEN hPen;
	HBRUSH hBrush;
	color colorPen, colorBrush;
	short width;

	bool isAffiliation(Quadrangle *quadrangle);
	bool isRGB(color colorPenOrBrush);
	bool isWidth(short *width);
	friend class Container;
public:
	View();
	~View();
	View(HWND, HDC, short *width, color *colorPen, color *colorBrush);
	void setCharacteristicsView(short *width, color *colorPen, color *colorBrush);
	short *getWidth();
	color *getColorPen();
	color *getColorBrush();
	void viewUnpaintQudrangle(Quadrangle *quadrangle);
	void viewPaintQudrangle(Quadrangle *quadrangle);
	void viewTwoNestedPaintQuadrangle(Quadrangle *externalQuadrangle, Quadrangle
*internalQuadrangle);
	void enterCharacteristicsFromFiles(string fd);
	void saveCharacteristicsToFile(string fd);
};

#include "View.h"
using namespace std;
bool View::isAffiliation(Quadrangle *quadrangle)
{
	GetClientRect(this->hwnd, &(this->rt));
	POINT *points = quadrangle->getQuadrangle();
	return ((points[0].x < rt.left) || (points[0].y < rt.top) || (points[0].x > rt.right) ||
	(points[0].y > rt.bottom) ||(points[1].x > rt.right) || (points[1].y < rt.top) ||
	(points[1].x < rt.left) || (points[1].y > rt.bottom) ||(points[2].x > rt.right) ||
	(points[2].y > rt.bottom) || (points[2].x < rt.left) || (points[2].y < rt.top) ||
	(points[3].x < rt.left) || (points[3].y > rt.bottom) || (points[3].x > rt.right) ||
	(points[3].y < rt.top));
}

bool View::isRGB(color colorPenOrBrush)
{
	return ((colorPenOrBrush.R < 0 || colorPenOrBrush.R > 255) ||
		(colorPenOrBrush.G < 0 || colorPenOrBrush.G > 255) ||
		(colorPenOrBrush.B < 0 || colorPenOrBrush.B> 255));
}

bool View::isWidth(short *width)
{
	return (*width < 0 || *width > 9);
}
```

```
View::View()
{
	hwnd = GetConsoleWindow();
	hdc = GetDC(hwnd);
	colorPen = { 0, 0, 0 };
	colorBrush = { 255, 255, 255 };
	hPen = CreatePen(PS_SOLID, 5, RGB(this->colorPen.R, this->colorPen.G, this->colorPen.B));
	hBrush = CreateSolidBrush(RGB(this->colorBrush.R, this->colorBrush.G, this->colorBrush.B));
}

View::~View()
{
	SelectPen(hdc, hPen);
	SelectBrush(hdc, hBrush);
	DeletePen(hPen);
	DeleteBrush(hBrush);
	ReleaseDC(this->hwnd, this->hdc);
}

View::View(HWND hwnd, HDC hdc, short *width, color *colorPen, color *colorBrush)
{
this->hwnd = hwnd;
this->hdc = hdc;
this->width = *width;
if (isWidth(width))
	throw exception("Такую толщину линий задать нельзя!\nМаксимальная толщина - 9 пикселей.");
this->colorPen = *colorPen;
if (isRGB(*colorPen))
	throw exception("Такого цвета не существует!");
this->colorBrush = *colorBrush;
if (isRGB(*colorBrush))
	throw exception("Такого цвета не существует!");
hPen = CreatePen(PS_SOLID, this->width, RGB(this->colorPen.R, this->colorPen.G, this->colorPen.B));
hBrush = CreateSolidBrush(RGB(this->colorBrush.R, this->colorBrush.G, this->colorBrush.B));
}

void View::setCharacteristicsView(short *width, color *colorPen, color *colorBrush)
{
this->width = *width;
if (isWidth(width))
	throw exception("Такую толщину линий задать нельзя!\nМаксимальная толщина - 9 пикселей.");
this->colorPen = *colorPen;
if (isRGB(*colorPen))
	throw exception("Такого цвета не существует!");
this->colorBrush = *colorBrush;
if (isRGB(*colorBrush))
	throw exception("Такого цвета не существует!");
hPen = CreatePen(PS_SOLID, this->width, RGB(this->colorPen.R, this->colorPen.G, this->colorPen.B));
hBrush = CreateSolidBrush(RGB(this->colorBrush.R, this->colorBrush.G, this->colorBrush.B));
}

short *View::getWidth()
{
	return &this->width;
}

color *View::getColorPen()
{
	return &this->colorPen;
}

color *View::getColorBrush()
{
	return &this->colorBrush;
}
```

```
void View::viewUnpaintQudrangle(Quadrangle *quadrangle)
{
	if (isAffiliation(quadrangle))
		throw exception("Четырёхугольник выходит за границу экрана!");
	InvalidateRect(this->hwnd, 0, TRUE);
	UpdateWindow(this->hwnd);
	HPEN SelectPen(this->hdc, this->hPen);
	HBRUSH SelectBrush(this->hdc, GetStockBrush(NULL_BRUSH));
	POINT *points = quadrangle->getQuadrangle();
	Polygon(this->hdc, points, 4);
}

void View::viewPaintQudrangle(Quadrangle *quadrangle)
{
	if (isAffiliation(quadrangle))
		throw exception("Четырёхугольник выходит за границу экрана!");
	InvalidateRect(this->hwnd, 0, TRUE);
	UpdateWindow(this->hwnd);
	HPEN SelectPen(this->hdc, this->hPen);
	HBRUSH SelectBrush(this->hdc, this->hBrush);
	POINT *points = quadrangle->getQuadrangle();
	Polygon(this->hdc, points, 4);
}

void View::viewTwoNestedPaintQuadrangle(Quadrangle *externalQuadrangle, Quadrangle
*internalQuadrangle)
{
	if (isAffiliation(externalQuadrangle))
		throw exception("Четырёхугольник выходит за границу экрана!");
	InvalidateRect(this->hwnd, 0, TRUE);
	UpdateWindow(this->hwnd);
	if (externalQuadrangle->isOnTheNested(internalQuadrangle))
		throw exception("Внутренний четырёхугольник выходит за границы внешнего!");
	HPEN SelectPen(this->hdc, this->hPen);
	HBRUSH SelectBrush(this->hdc, GetStockBrush(NULL_BRUSH));
	POINT *points1 = externalQuadrangle->getQuadrangle();
	POINT *points2 = internalQuadrangle->getQuadrangle();
	Polygon(this->hdc, points1, 4);
	Polygon(this->hdc, points2, 4);
	HBRUSH SelectBrush(this->hdc, this->hBrush);
	FloodFill(this->hdc, points1[0].x + 10, points1[0].y + 10, RGB(this->colorPen.R,
		this->colorPen.G, this->colorPen.B));
}

void View::enterCharacteristicsFromFiles(string fd)
{
	ifstream in(fd, ios::in);
	if (in.is_open())
{
in >> this->width;
if (isWidth(&this->width))
	throw exception("Такую толщину линий задать нельзя!\nМаксимальная толщина - 9 пикселей.");
in >> this->colorPen.R;
in >> this->colorPen.G;
in >> this->colorPen.B;
this->colorPen = colorPen;
if (isRGB(colorPen))
	throw exception("Такого цвета не существует!");
in >> this->colorBrush.R;
in >> this->colorBrush.G;
in >> this->colorBrush.B;
if (isRGB(colorBrush))
	throw exception("Такого цвета не существует!");
this->hPen = CreatePen(PS_SOLID, 5, RGB(this->colorPen.R, this->colorPen.G, this->colorPen.B));
this->hBrush = CreateSolidBrush(RGB(this->colorBrush.R, this->colorBrush.G, this->colorBrush.B));
in.close();
	}	else
		throw exception("В этом файле нет нужных данных");
}
```

```
void View::saveCharacteristicsToFile(string fd)
{
	ofstream out(fd, ios::out);
	out << this->width;
	out << " ";
	out << this->colorPen.R;
	out << " ";
	out << this->colorPen.G;
	out << " ";
	out << this->colorPen.B;
	out << " ";
	out << this->colorBrush.R;
	out << " ";
	out << this->colorBrush.G;
	out << " ";
	out << this->colorBrush.B;
	out.close();
}
```

```
#include "Quadrangle.h"
#include "View.h"
#include "Container.h"
using namespace std;

int main()
{
	setlocale(LC_ALL, "rus");
	try {
		HWND hwnd = GetConsoleWindow();
		HDC hdc = GetDC(hwnd);

		Quadrangle first({ 300, 100 }, { 400, 100 }, { 500, 300 }, { 300, 220 });
		Quadrangle second({ 350, 150 }, { 380, 150 }, { 400, 220 }, { 350, 200 });
		Quadrangle third({ 310, 100 }, { 410, 100 }, { 510, 300 }, { 310, 220 });
		Quadrangle fourth({ 320, 100 }, { 420, 100 }, { 520, 300 }, { 320, 220 });
		Quadrangle fifth({ 300, 100 }, { 400, 100 }, { 500, 300 }, { 300, 220 });
		Quadrangle sixth({ 360, 150 }, { 390, 150 }, { 410, 220 }, { 360, 200 });
		Container container;

		container.addFigure(first);
		container.addFigure(second);
		container.addFigure(third);
		container.addFigure(fourth);
		container.addFigure(fifth);
		container.addFigure(sixth);

		container.printInfo();
		container.saveInFile("Container_save.txt");

		container.removeFigure(4);
		container.printInfo();
		container.printInfoBack();

		container.searchFigureOnNumber(2);
		container.searchFigureOnSign("big  ");
		container.searchFigureOnSign("small");
	}
	catch (exception exeption)
	{
		cout << exeption.what() << endl;
	}
	return 0;
}
```