МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ТЕОРЕТИЧЕСКОЙ И ПРИКЛАДНОЙ ИНФОРМАТИКИ

Лабораторная работа №5
по дисциплине «Объектно-ориентированное программирование»

Факультет: ПМИ
Группа: ПМИ-03
Студенты: Малыгин С. А, Сидоров Д. И.
Преподаватель: Лисицын Д. В., Неделько В. М.

НОВОСИБИРСК
2021

1) Условие задачи:

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

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

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

Описание функций:

```
template <class Type> Table <Type>::Table(int a)
{
	maxsize = a;
	data = new Tab[maxsize];
	size = 0;
}
```

Конструктор

```
template <class Type> Table <Type>::~Table()
{
	delete data;
}
```

Деструктор

```
template <class Type> bool Table<Type>::find(int key)
{
	for (inti = 0; i < size; i++)
		Если (data[i].key == key) return true;
	return false;
}
```

Поиск элемента по ключу

```
template <class Type> void Table<Type>::add_elem(int key, Type rec)
{
	if (size == maxsize) throw 4;
	Type x = rec;
	data[size].key = key;
	data[size].elem = x;
	size++;
}
```

Добавить элемент в таблицу

```
template <class Type> void Table<Type>::Удалить_elem(int key)
{
	if (!find(key)) throw 5;
	for (int i = 0; i < size; i++)
	{
		if (data[i].key == key)
			for (int j(i); j + 1 <= size; j++)
				data[j] = data[j + 1];
	}
	size--;
	data[size].key = NULL;
	data[size].elem = NULL;
}
```

Удалить элемент из таблицы

```
template <class Type> vector<int> Table<Type>::available_key()
{
```

```
	vector<int> keys;
	for (int i = 0; i < size; i++)
		keys.push_back(data[i].key);
	return keys;
}
```
Получить доступные ключи
```
template <class Type> Type Table<Type>::get_elem(int key)
{
	if (!find(key)) throw 5;
	for (int i = 0; i < size; i++)
		if (data[i].key == key) return data[i].elem;
}
```
Получить элемент из таблицы
```
template <class Type> void Table<Type>::Paint(int key, HDC hdc, RECT rt)
{
	if (!find(key)) throw 5;
	for (int i = 0; i < size; i++)
		if (data[i].key == key)
			data[i].elem->Draw(hdc, rt);
}
```
Нарисовать выбранный элемент
```
template <class Type> void Table<Type>::PaintALL(HDC hdc, RECT rt)
{
	for (int i = 0; i < size; i++)
	{
		data[i].elem->Draw(hdc, rt);
		system("cls");
	}
}
```
Нарисовать все элементы таблицы

```
template <class Type> void Table<Type>::saving(std::ofstream &fout)
{
	for (int i = 0; i < size; i++)
	{
		data[i].elem->save(fout);
	}
}
```
Сохранить элементы таблицы

2) Программа:

```
#pragma once
#include "Quadrangle.h"
#include <vector>

template <class Type> class Table
{
private:
   struct Tab
   {
		int key;
		Type elem;
   };
   Tab *data;
   int size;
   int maxsize;
public:
   Table(int a);
   ~Table();
   void add_elem(int key, Type rec);
   void delete_elem(int key);
   Type get_elem(int key);
```

```
   vector<int> available_key();
   void Paint(int key, HDC hdc, RECT rt);
   void PaintALL(HDC hdc, RECT rt);
   bool find(int key);
   void saving(std::ofstream &fout);
};
template <class Type> Table <Type>::Table(int a)
{
   maxsize = a;
   data = new Tab[maxsize];
   size = 0;
}
template <class Type> Table <Type>::~Table()
{
   delete data;
}
template <class Type> bool Table<Type>::find(int key)
{
   for (int i = 0; i < size; i++)
          if (data[i].key == key) return true;
   return false;
}
template <class Type> void Table<Type>::add_elem(int key, Type rec)
{
   if (size == maxsize) throw 4;
   Type x = rec;
   data[size].key = key;
   data[size].elem = x;
   size++;
}
template <class Type> void Table<Type>::delete_elem(int key)
{
   if (!find(key)) throw 5;
   for (int i = 0; i < size; i++)
   {
          if (data[i].key == key)
                 for (int j(i); j + 1 <= size; j++)
                        data[j] = data[j + 1];
   }
   size--;
   data[size].key = NULL;
   data[size].elem = NULL;
}
template <class Type> vector<int> Table<Type>::available_key()
{
   vector<int> keys;
   for (int i = 0; i < size; i++)
          keys.push_back(data[i].key);
   return keys;
}
template <class Type> Type Table<Type>::get_elem(int key)
{
   if (!find(key)) throw 5;
   for (int i = 0; i < size; i++)
          if (data[i].key == key) return data[i].elem;
}
template <class Type> void Table<Type>::Paint(int key, HDC hdc, RECT rt)
{
   if (!find(key)) throw 5;
   for (int i = 0; i < size; i++)
          if (data[i].key == key)
                 data[i].elem->Draw(hdc, rt);
}
template <class Type> void Table<Type>::PaintALL(HDC hdc, RECT rt)
{
```

```
   for (int i = 0; i < size; i++)
   {
          data[i].elem->Draw(hdc, rt);
          system("cls");
   }
}
template <class Type> void Table<Type>::saving(std::ofstream &fout)
{
   for (int i = 0; i < size; i++)
   {
          data[i].elem->save(fout);
   }
}
```

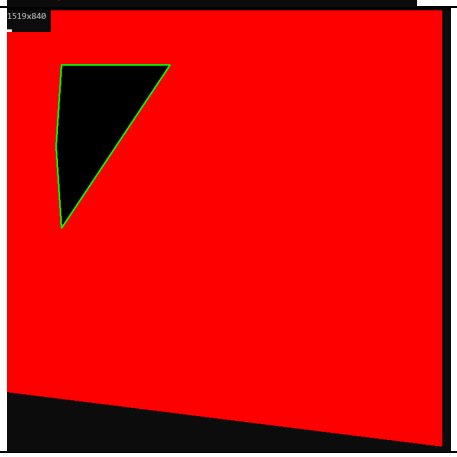
3) Алгоритм:

```
template <class Type> Table <Type>::Table(Целая a)
{
   maxsize = a;
   data = new Tab[maxsize];
   size = 0;
}

template <class Type> Table <Type>::~Table()
{
   Удалить data;
}
template <class Type> Логическая Table<Type>::find(Целая key)
{
   for (Целая i = 0; i < size; i++)
          Если (data[i].key == key) Вернуть истину;
   Вернуть ложь;
}
template <class Type> void Table<Type>::add_elem(Целая key, Type rec)
{
   Если (size == maxsize) throw 4;
   Type x = rec;
   data[size].key = key;
   data[size].elem = x;
   size++;
}
template <class Type> void Table<Type>::Удалить_elem(Целая key)
{
   Если (!find(key)) throw 5;
   for (Целая i = 0; i < size; i++)
   {
          Если (data[i].key == key)
                 for (Целая j(i); j + 1 <= size; j++)
                        data[j] = data[j + 1];
   }
   size--;
   data[size].key = ПУСТО;
   data[size].elem = ПУСТО;
}
template <class Type> вектор<Целая> Table<Type>::available_key()
{
   вектор<Целая> keys;
   for (Целая i = 0; i < size; i++)
          keys.push_back(data[i].key);
   Вернуть keys;
}
template <class Type> Type Table<Type>::get_elem(Целая key)
{
```

```
   Если (!find(key)) throw 5;
   for (Целая i = 0; i < size; i++)
          Если (data[i].key == key) Вернуть data[i].elem;
}
template <class Type> void Table<Type>::Paint (Целая key, HDC hdc, RECT rt)
{
   Если (!find(key)) throw 5;
   for (Целая i = 0; i < size; i++)
          Если (data[i].key == key)
                 data[i].elem->Draw(hdc, rt);
}
template <class Type> void Table<Type>::PaintALL(HDC hdc, RECT rt)
{
   for (Целая i = 0; i < size; i++)
   {
          data[i].elem->Draw(hdc, rt);
          system("cls");
   }
}


template <class Type> void Table<Type>::saving(std::ofstream &fout)
{
   for (Целая i = 0; i < size; i++)
   {
          data[i].elem->save(fout);
   }
}
```

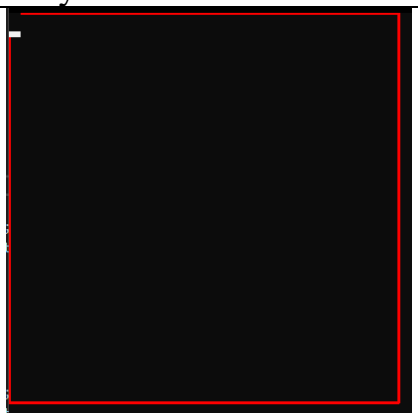
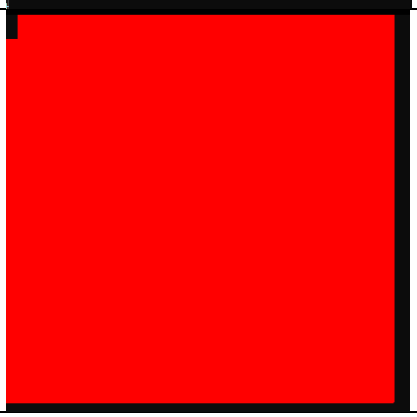
4) Набор тестов:

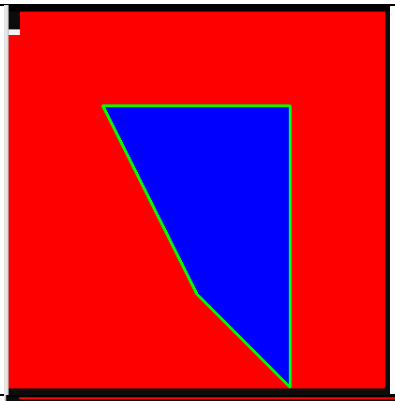
№	Set.txt	Результат	Примечание
1	700 700 1000 1000 800 1500 500 1500	Фигура выходит за края экрана!	Фигура выходит за края экрана!
2		Файл пуст!	Файл пуст!
3	1 1 200 1 200 200 1 200		Контур фигуры

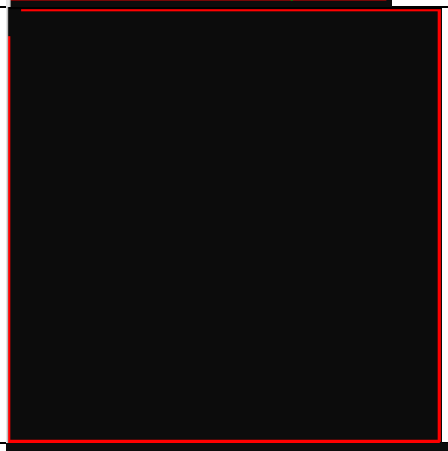
4	100 100 300 100 100 400 90 250		Заливка фигуры
5	0 0 800 0 800 800 0 700 100 100 300 100 100 400 90 250		Фигура в фигуре
6	100 100 300 100 150 250 175 400	Фигура не выпуклая!	Четырехугольник невыпуклый
7	100 100 300 100 100 400 90 250 0 0 400 0 400 400 0 400	Внутренняя фигура выходит за края внешней фигуры!	Фигура не в фигуре

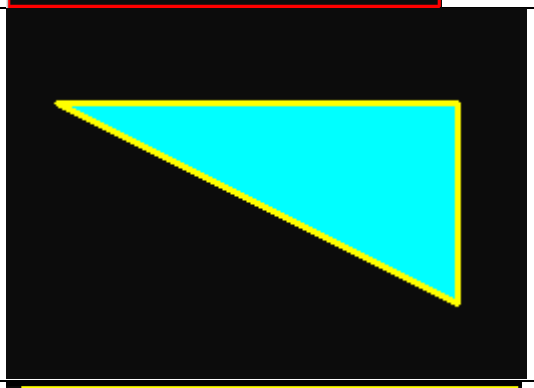
№	Функция	Было	Стало	Примечание
1	get_data	Выбрана фигура 1	Точка № 1 X: 1 Y: 1 Точка № 2 X: 200 Y: 1 Точка № 3 X: 200 Y: 200 Точка № 4	Функция работает правильно

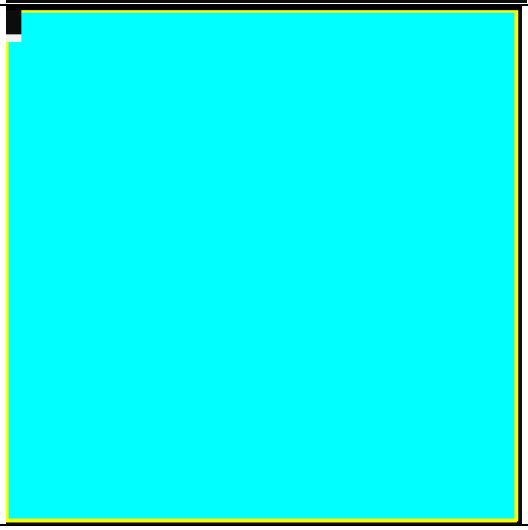
			X: 1 Y: 200	
2	set_data	Координаты 1 фигуры 1 1 200 1 200 200 1 200	Координаты 1 фигуры 0 100 200 100 200 200 1 200	Функция работает правильно
4	save	Файл save.txt пуст или занят устаревшей информацией	Файл заполнен новыми данными	Функция работает правильно
6	position	Координаты 1 фигуры 1 1 200 1 200 200 1 200 Сместить на 100 и 100	Координаты 1 фигуры 101 101 300 101 300 300 101 300	Функция работает правильно
7	reader	0 0 0 0 0 0 0 0	0 100 200 100 200 200 1 200	Функция работает правильно

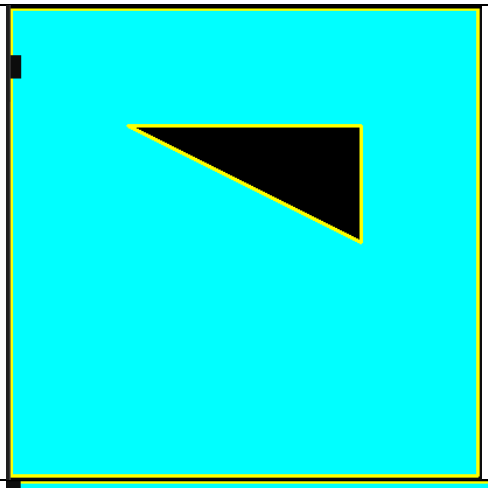
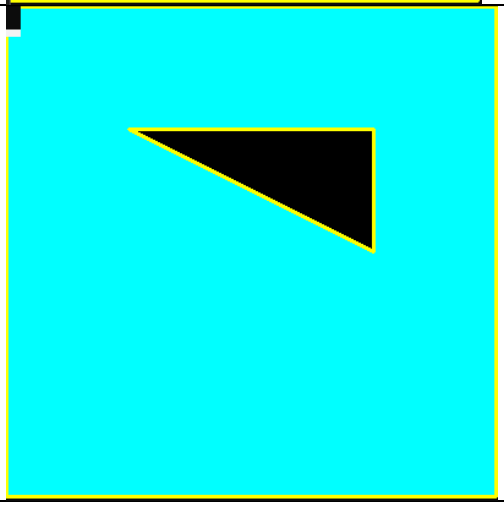
№	Функция	Результат	Примечание
1	Painted_Counter::Draw()		Функция работает правильно
2	Quadrangle_fill::Draw()		Функция работает правильно

3	`Two_Quadrangle::Draw()`		Функция работает правильно

4	`PC.reader(fin);`
	`TPC.add_elem(1, &PC);`
	`TPC.Paint(1,hdc,rt);`
			Функция работает правильно

5	`QF.set_data(first1, rgbc, rgbf);` `TQF.add_elem(1, &QF);TQF.Paint(1, hdc,rt);`		Функция работает правильно

6	`QF2.set_data(first, rgbc, rgbf);`
	`TQF.add_elem(2, &QF2)`
	`TQF.Paint(2, hdc, rt);`
			Функция работает правильно

7	GET = TTQ.get_elem(1); GET->Draw(hdc, rt);		
8	TQ.set_data(first,first1, rgbc, rgbf); TTQ.add_elem(1, &TQ); TTQ.Paint(1, hdc, rt);		Функция работает правильно
9	TQF.delete_elem(1); TQF.delete_elem(2); cout<<TQF.find(1)<<" "<< TQF.find(2)<<endl;	0 0	Функция работает правильно
10	TQF.saving(fout);	100 100 300 100 300 200 200 150 100 100 300 100 300 200 200 150 0 0 400 0 400 400 0 400 0 0 400 0 400 400 0 400	Функция работает правильно

5) Программа работает правильно, что подтверждают тесты.