Выполнили:

студенты группы 19ВВ2

Гусев Д.О.

Кубасов И.М.

Приняли:

Синев М.П.

Дорошенко И.Н.

Пенза 2022

ОТЧЕТ

по лабораторной работе №10

по курсу «Технологии программирования»

на тему «STL»

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Цель работы

Изучение стандартной библиотеки шаблонов STL (Standard Template Library).

### Лабораторное задание

Лабораторное задание выполняется на базе лабораторной работы №1.

Для указанного в лабораторной работе №1 базового класса реализовать структуру данных в соответствии с лабораторным заданием.

Требования к классу:

* Класс из лабораторной работы №1 должен быть дополнен всеми необходимыми функциями (конструкторами) и операторами для обеспечения работы STL контейнеров и функций.
* В классе все строковые данные представить с использованием классов string (или wstring).

Требования к программе:

* Программа должна реализовать контейнер объектов класса в соответствии с вариантом задания.
* Для работы с контейнером должны быть реализованы циклы с операциями:
  + добавление элемента в контейнер;
  + удаление элемента из контейнера;
  + вывод контейнера на экран.

### Варианты лабораторных заданий

### 

### Листинг

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <conio.h>

#include <set>

#include <vector>

#include <string>

#include <Windows.h>

#include <fstream>

using namespace std;

//4. Класс «Книга» (тема, автор, издательство, год выпуска и т.д.).

class Interface

{

public:

virtual void Input() = NULL;

virtual void Output() = NULL;

};

class Phone : public Interface

{

protected:

int tmp\_size = 500;

int size = 50;

string m\_producer;

string m\_model;

int m\_size;

int m\_price;

int flag = 0;

public:

Phone()

{

m\_producer.clear();

m\_model.clear();

m\_size = 0;

m\_price = 0;

}

Phone(const char\* producer, const char\* model = "EVA-01", int size = 20, int price = 10000)

{

m\_producer = (string)producer;

m\_model = (string)model;

m\_size = size;

m\_price = price;

}

Phone(const Phone& phone) //конструктор копирования с ссылкой

{

m\_producer = phone.m\_producer;

m\_model = phone.m\_model;

m\_size = phone.m\_size;

m\_price = phone.m\_price;

}

Phone(const Phone\* phone) //конструктор копирования с указателем

{

m\_producer = phone->m\_producer;

m\_model = phone->m\_model;

m\_size = phone->m\_size;

m\_price = phone->m\_price;

flag = 1;

}

void Input() override

{

char\* tmp = new char[tmp\_size];

cout << "Введите производителя " << endl;

cin.getline(tmp, tmp\_size);

int len = strlen((const char\*)tmp);

while (len > size)

{

cout << "Длина строки больше 50 символов!!! " << endl;

cin.getline(tmp, tmp\_size);

len = strlen(tmp);

}

string strTmpProd(tmp);

m\_producer = strTmpProd;

cout << "Введите модель" << endl;

cin.getline(tmp, tmp\_size);

len = strlen((const char\*)tmp);

while (len > size)

{

cout << "Длина строки больше 50 символов!!! " << endl;

cin.getline(tmp, tmp\_size);

len = strlen(tmp);

}

string strTmpMod(tmp);

m\_model = strTmpMod;

cout << "Введите размер экрана " << endl;

cin >> m\_size;

cin.get();

cout << "Введите цену" << endl;

cin >> m\_price;

cin.get();

delete[] tmp;

system("cls");

}

void Output() override

{

get\_producer();

get\_model();

get\_size();

get\_price();

cout << endl;

system("pause");

}

void get\_producer()

{

cout << "Автор: " << m\_producer << endl;

}

void get\_model()

{

cout << "Theme: " << m\_model << endl;

}

void get\_size()

{

cout << "Publisher: " << m\_size << endl;

}

void get\_price()

{

cout << "Year: " << m\_price << endl;

}

bool operator > (const Phone& obj) const noexcept

{

return this->m\_price > obj.m\_price;

}

bool operator < (const Phone& obj) const noexcept

{

return this->m\_price < obj.m\_price;

}

Phone& operator= (const Phone& obj)

{

m\_producer = obj.m\_producer;

m\_model = obj.m\_model;

m\_size = obj.m\_size;

m\_price = obj.m\_price;

return \*this;

}

bool operator <= (const Phone obj)

{

return this->m\_price <= obj.m\_price;

}

friend std::ostream& operator<< (std::ostream& out, const Phone& phone);

~Phone()

{

}

};

std::ostream& operator << (std::ostream& out, Phone& obj)

{

obj.Output();

return out;

}

template <typename T> void OutputSet(set <T>& setName)

{

cout << "Вывод элементов set" << endl;

for (auto element : setName)

{

cout << element;

}

}

template <typename T> void InputSet(set <T>& setName, T\* mas, int size)

{

cout << "Заполнение set" << endl;

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

{

setName.insert(mas[i]);

}

}

template <typename T> void DeleteSetElement(set <T>& setName, T\* mas, int element)

{

auto delResult = setName.erase(mas[element]);

if (delResult == 1)

{

cout << "Элемент удален" << endl << endl;

}

else

{

cout << "Не удалось удаить элемент" << endl;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

const int PhoneMasLen = 3;

Phone PhoneMas[PhoneMasLen];

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

{

PhoneMas[i].Input();

}

set <Phone> setPhone;

InputSet(setPhone, PhoneMas, PhoneMasLen);

OutputSet(setPhone);

int delChoose;

cout << "Удаление элемента set" << endl << "Введите номер элемента для удаления: ";

cin >> delChoose;

delChoose -= 1;

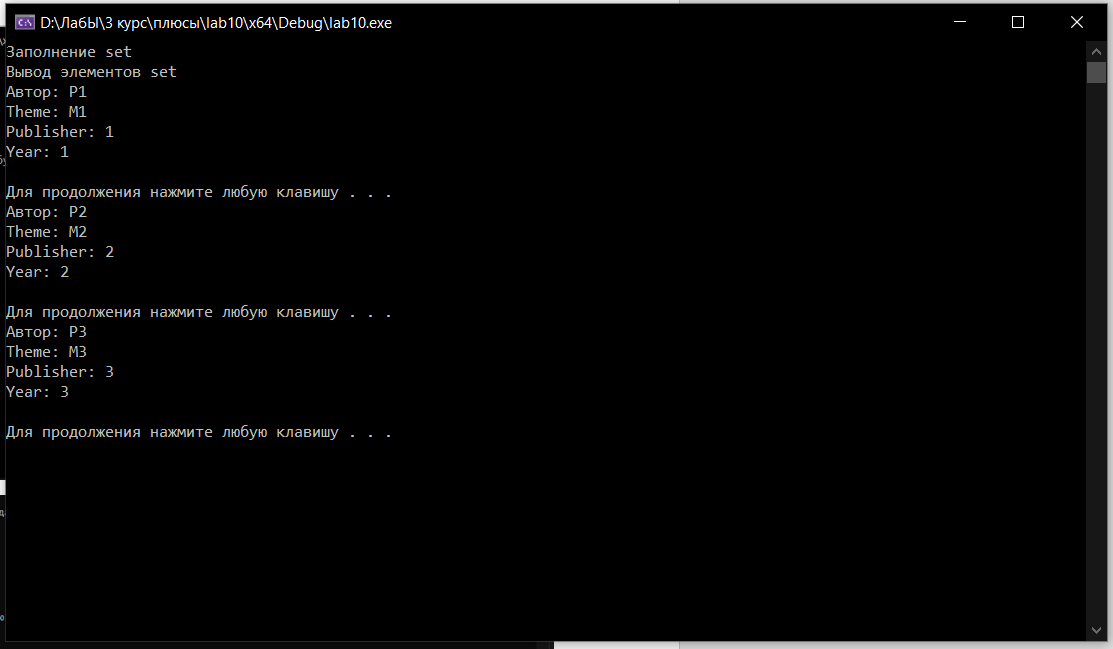
DeleteSetElement(setPhone, PhoneMas, delChoose);

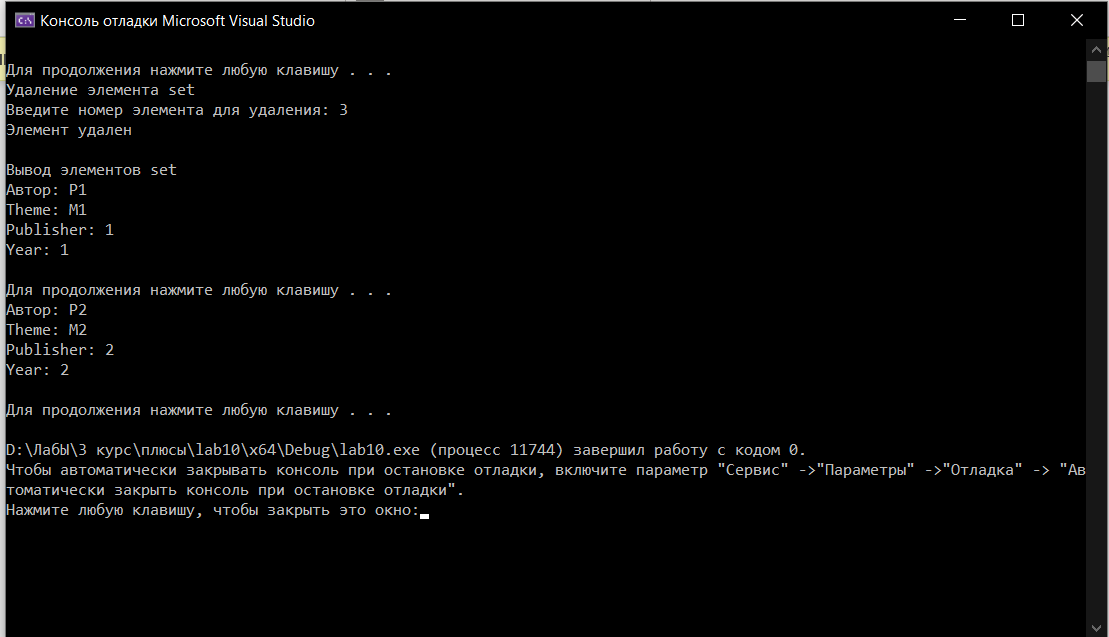
OutputSet(setPhone);

return 0;

}

**Результаты работы программы**





### Вывод

Мы изучили стандартную библиотеку шаблонов STL (Standard Template Library)