Выполнили:

студенты группы 19ВВ2

Гусев Д.О.

Кубасов И.М.

Приняли:

Синев М.П.

Дорошенко И.Н.

Пенза 2021

ОТЧЕТ

по лабораторной работе №9

по курсу «Технологии программирования»

на тему «Шаблоны»

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Цель работы

Изучение механизма параметризованных классов и функций предназначенных для реализации обобщенного программирования.

### Лабораторное задание

Лабораторная работа выполняется на базе работ №1 и №7. Из работы №1 берется базовый класс, из работы №7 берется динамическая структура.

В работе требуется реализовать две параметризованные функции и один (или более) параметризованный класс.

Требования к параметризованным функциям:

* Шаблонные функции должны выполнять сравнение двух объектов класса из лабораторной работы №1.
  + Первая функция должна сравнивать объекты на больше ( > ).
  + Вторая функция должна сравнивать объекты на меньше или равно ( <= ).

Требования к параметризованному классу:

* Шаблонный класс должен реализовывать функционал динамической структуры данных. Вариант структуры берется из лабораторной работы №7.
* Класс должен поддерживать формирование динамической структуры для фундаментальных (int, short, void\* и т.п.) и производных типов данных (классы, структуры). В качестве производного типа данных должен использоваться класс из лабораторной работы №1.

Требования к базовому классу:

* Класс из лабораторной работы №1 должен быть дополнен всеми необходимыми функциями (конструкторами) и операторами для обеспечения работы шаблонных функций и шаблонного класса.

Требования к программе:

* В программе должны несколько раз вызываться шаблонные функции для сравнения различных объектов класса.
* В программе должны быть реализованы две динамические структуры:
  + Первая, для объектов фундаментального типа (int, short, void\* и т.п.).
  + Вторая, для объектов производного типа. Класс берется из лабораторной работы №1.
* Для всех динамических структур должны быть реализованы циклы с операциями:
  + добавление элемента в структуру;
  + удаление элемента из структуры;
  + вывод структуры на экран.

### Варианты лабораторных заданий

Вариант реализуемого класса берется из задания для лабораторной работы №1.

Вариант реализуемой динамической структуры данных берется из задания для лабораторной работы №7.

### Листинг

**Phone.h**

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma once

#include <iostream>

using namespace std;

class Phone

{

protected:

double m\_size;

char\* m\_model = new char[50];

char\* m\_producer = new char[50];

int m\_price;

public:

Phone(Phone& Phone) {

strcpy\_s(m\_producer, 50, Phone.m\_producer);

strcpy\_s(m\_model, 50, Phone.m\_model);

m\_size = Phone.m\_size;

m\_price = Phone.m\_price;

}

Phone(Phone\* Phone) {

strcpy\_s(m\_producer, 50, Phone->m\_producer);

strcpy\_s(m\_model, 50, Phone->m\_model);

m\_size = Phone->m\_size;

m\_price = Phone->m\_price;

}

Phone()

{

strcpy(m\_producer, "HUAWEI");

strcpy(m\_model, "EVA-01");

m\_size = 18.9;

m\_price = 19990;

}

~Phone()

{

delete[] m\_producer;

delete[] m\_model;

}

void Input(char\* producer, char\* model, double size, int price)

{

strcpy(m\_producer, producer);

strcpy(m\_model, model);

m\_size = size;

m\_price = price;

}

void Output() {

printf("Производитель: %s\n", m\_producer);

printf("Модель: %s\n", m\_model);

printf("Размер экрана: %f\n", m\_size);

printf("Цена: %d\n", m\_price);

printf("\n");

}

bool operator > (const Phone obj)

{

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

}

Phone& operator= (const Phone& obj)

{

strcpy(m\_producer, obj.m\_producer);

strcpy(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);

};

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

{

obj.Output();

return out;

}

template <typename T> bool funk1(T a, T b)

{

return a > b;

}

template <typename T> bool funk2(T a, T b)

{

return a <= b;

}

template <typename T> void cmp(T a, T b)

{

if (funk1(a, b) == true)

{

cout << "Левый операнд больше правого" << endl;

}

if (funk2(a, b) == true)

{

cout << "Левый операнд меньше или равен правому" << endl;

}

}

template <typename T> class Queue

{

private:

int size;

T\* queuePtr;

int last;

int first;

public:

Queue()

{

size = 5;

queuePtr = new T[size];

last = -1;

first = -1;

}

Queue(int a)

{

size = a;

queuePtr = new T[size];

last = -1;

first = -1;

}

void push(T data)

{

T data\_tmp = data;

if (last == size - 1)

{

cout << "Очередь заполнена" << endl;

}

else

{

last++;

queuePtr[last] = data\_tmp;

}

}

T pop()

{

if (last == -1)

{

cout << "Очередь пуста";

}

else

{

first++;

return queuePtr[last];

}

}

void printStack()

{

for (int i = last; i > first; i--)

{

cout << queuePtr[i] << endl;

}

}

~Queue()

{

delete[] queuePtr;

}

};

**Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "Phone.h";

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

char producer[] = "";

char model[] = "";

double size;

int price;

int szPhoneQ = 2;

Phone\* phone = new Phone[szPhoneQ];

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

{

cout << "Введите производителя" << endl;

cin >> producer;

cout << "Введите модель" << endl;

cin >> model;

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

cin >> size;

cout << "Введите цену" << endl;

cin >> price;

phone[i].Input(producer, model, size, price);

}

cout << "демонстрация выполнения параметризованных функций для объектов Phone" << endl;

cmp(phone[0], phone[1]);

cout << "демонстрация выполнения параметризованных функций для объектов char" << endl;

cmp('b', 'a');

cout << "демонстрация работы динамической структуры с типом Phone" << endl;

Queue <Phone> PhoneQ(szPhoneQ);

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

{

PhoneQ.push(phone[i]);

}

PhoneQ.printStack();

PhoneQ.pop();

cout << "Очередь после pop" << endl;

PhoneQ.printStack();

delete[] phone;

cout << "демонстрация работы динамической структуры с типом int" << endl;

int szIntQ = 2;

Queue <int> IntQ(szIntQ);

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

{

IntQ.push(i);

}

IntQ.printStack();

IntQ.pop();

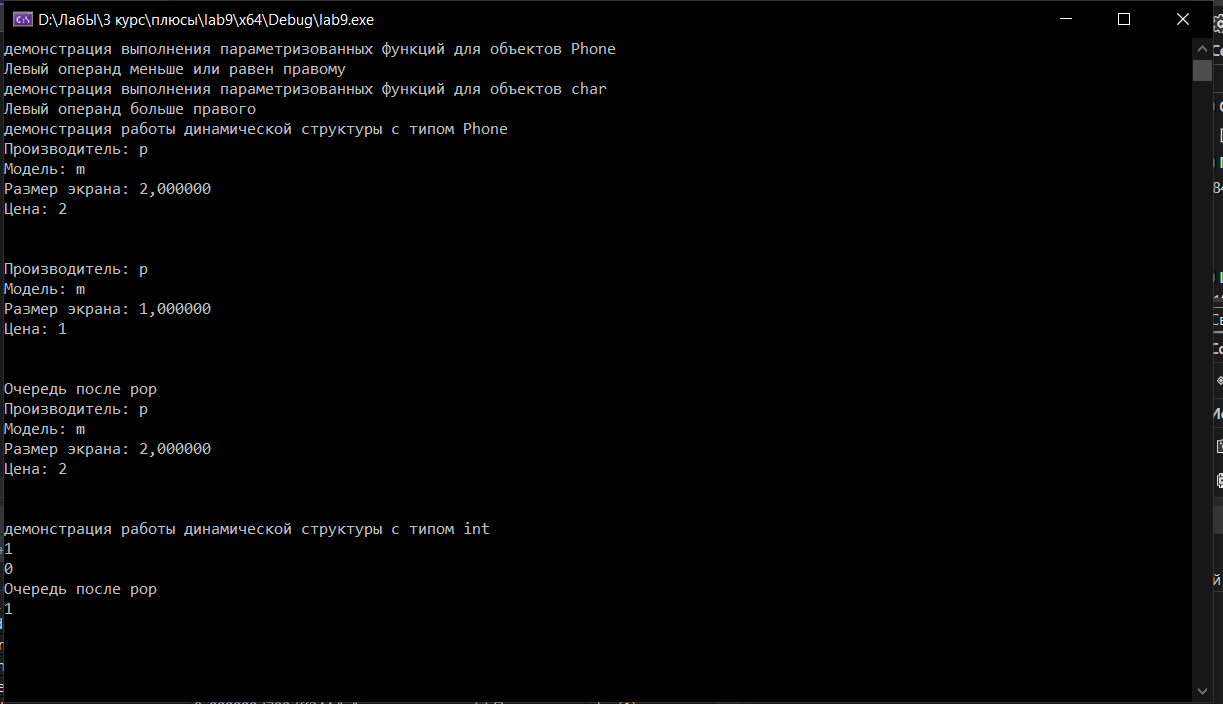
cout << "Очередь после pop" << endl;

IntQ.printStack();

return 0;

}

### Результаты работы программы



### Вывод

Мы изучили механизм параметризованных классов и функций предназначенных для реализации обобщенного программирования.