Выполнили:

студенты группы 19ВВ2

Гусев Д.О.

Кубасов И.М.

Приняли:

Синев М.П.

Дорошенко И.Н.

Пенза 2021

ОТЧЕТ

по лабораторной работе №5

по курсу «Технологии программирования»

на тему «Операторы»

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Цель работы

Изучение механизмов перегрузки операторов. Получение навыков реализации перегруженных операторов.

### Лабораторное задание

Лабораторное задание выполняется на базе лабораторной работы №4. В производный класс **Derived3** добавляется реализация операторов в соответствии с вариантом задания.

Задание включает реализацию 2 операторов:

* Оператор, который возвращает логическое значение (истина или ложь).
* Оператор, который выполняет изменение данных класса.

Каждый оператор должен быть реализован для 3 типов параметров:

* Параметр целочисленного типа **int**.
* Параметр строкового типа **char\*** (или **wchar\_t\***).
* Параметр ссылки на исходный класс **Derived3&** (или **const** **Derived3&**).

Каждый оператор должен быть реализован и внутри и вне описания класса. Т.е. как минимум одна реализация (для одного типа параметров) должна быть внутри описания класса и как минимум одна реализация должна быть вне описания класса.

В основной функции программы должен присутствовать вызов каждого оператора для всех типов параметров.

Оператор, который возвращает логическое значение:

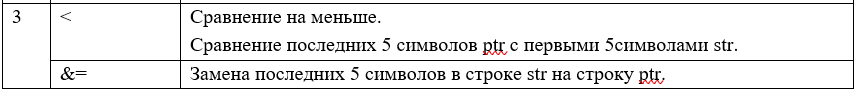
Логическая операция указана в варианте задания.

* Реализация с параметром целочисленного типа **int**. Логическая операция выполняется между целочисленным параметром оператора и длиной строки любой переменной класса строкового типа (**str**).
* Реализация с параметром строкового типа **char\***. Логическая операция выполняется между строковым параметром оператора (**ptr**) и любой переменной класса строкового типа (**str**).
* Реализация с параметром ссылки на исходный класс **Derived3&**. Логическая операция выполняется между одноименными переменными двух классов строкового типа.

Оператор, который выполняет изменение данных класса:

Выполняемое действие указано в варианте задания. В результате выполнения действия изменяется только переменная в классе-приемнике. Данные, переданные в параметрах оператора изменяться не должны.

* Реализация с параметром целочисленного типа **int**. Числовой параметр оператора печатается в строку заданного формата (строка **ptr**). Действие выполняется между сформированной строкой (**ptr**) и любой переменной класса строкового типа (**str**).
* Реализация с параметром строкового типа **char\***. Действие выполняется между строковым параметром оператора (**ptr**) и любой переменной класса строкового типа (**str**).
* Реализация с параметром ссылки на исходный класс **Derived3&**. Действие выполняется между одноименными переменными двух классов строкового типа.



**Листинг**

**Phone.h**

#pragma once

#include <iostream>

class Phone

{

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");

}

protected:

double m\_size;

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

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

int m\_price;

};

class derived1 : public Phone {

protected:

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

public:

derived1() {

strcpy\_s(m\_country, 50, "\0");

}

derived1(const char\* country, char\* producer, char\* model, double size, int price) {

strcpy(m\_country, country);

}

derived1(derived1& d) {

strcpy(m\_country, d.m\_country);

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

strcpy(m\_model, d.m\_model);

m\_size = d.m\_size;

m\_price = d.m\_price;

}

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

{

strcpy(m\_country, country);

Phone::Input(producer, model, size, price);

}

void Output() {

printf("Страна: %s\n", m\_country);

Phone::Output();

}

~derived1() {

delete[] m\_country;

}

};

class derived2 : public Phone {

protected:

int m\_year;

public:

derived2() {

m\_year = 0;

}

derived2(int year, char\* producer, char\* model, double size, int price) {

m\_year = year;

}

derived2(derived2& d) {

m\_year = d.m\_year;

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

strcpy(m\_model, d.m\_model);

m\_size = d.m\_size;

m\_price = d.m\_price;

}

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

{

m\_year = year;

Phone::Input(producer, model, size, price);

}

void Output() {

printf("Год: %d\n", m\_year);

Phone::Output();

}

~derived2() {

}

};

class derived3 : public derived1 {

public:

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

derived3() {

strcpy\_s(m\_city, 50, "\0");

}

derived3(char\* city, char\* country, char\* producer, char\* model, double size, int price) {

strcpy(m\_city, city);

}

derived3(derived3& d) {

strcpy(m\_city, d.m\_city);

strcpy(m\_country, d.m\_country);

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

strcpy(m\_model, d.m\_model);

m\_size = d.m\_size;

m\_price = d.m\_price;

}

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

{

strcpy(m\_city, city);

derived1::Input(country, producer, model, size, price);

}

void Output() {

printf("Город: %s\n", m\_city);

derived1::Output();

}

derived3& operator &= (const int a)

{

char\* atmp = new char[255];

\_itoa(a, atmp, 10);

int len\_m = strlen(m\_city);

int len\_mtmp = len\_m - 5;

if (len\_mtmp > 0)

{

m\_city[len\_mtmp] = { '\0' };

strcat(m\_city, atmp);

}

else

{

strcpy(m\_city, atmp);

}

delete[] atmp;

return \*this;

}

derived3& operator &= (const char\* a)

{

int len\_m = strlen(m\_city);

int len\_mtmp = len\_m - 5;

if (len\_mtmp > 0)

{

m\_city[len\_mtmp] = { '\0' };

strcat(m\_city, a);

}

else

{

strcpy(m\_city, a);

}

return \*this;

}

bool operator < (const char\* a)

{

char\* atmp = new char[255];

strcpy(atmp, a);

int len\_a = strlen(atmp);

int len\_atmp = len\_a - 5;

if (len\_atmp > 0)

{

atmp[5] = { '\0' };

}

int len\_m = strlen(m\_city);

int len\_mtmp = len\_m - 5;

const char\* lastfive\_m;

if (len\_mtmp > 0)

{

lastfive\_m = &m\_city[len\_m - 5];

}

else

{

lastfive\_m = m\_city;

}

if (strcmp(lastfive\_m, atmp) == 1)

{

delete[] atmp;

return false;

}

else

{

delete[] atmp;

return true;

}

}

bool operator < (const int a)

{

char\* atmp = new char[255];

\_itoa(a, atmp, 10);

int len\_a = strlen(atmp);

int len\_atmp = len\_a - 5;

if (len\_atmp > 0)

{

atmp[5] = { '\0' };

}

int len\_m = strlen(m\_city);

int len\_mtmp = len\_m - 5;

const char\* lastfive\_m;

if (len\_mtmp > 0)

{

lastfive\_m = &m\_city[len\_m - 5];

}

else

{

lastfive\_m = m\_city;

}

if (strcmp(lastfive\_m, atmp) == 1)

{

delete[] atmp;

return false;

}

else

{

delete[] atmp;

return true;

}

}

};

bool operator < (const derived3& obj1, const derived3& obj2)

{

char\* atmp = new char[255];

strcpy(atmp, obj2.m\_city);

int len\_a = strlen(atmp);

int len\_atmp = len\_a - 5;

if (len\_atmp > 0)

{

atmp[5] = { '\0' };

}

int len\_m = strlen(obj1.m\_city);

int len\_mtmp = len\_m - 5;

const char\* lastfive\_m;

if (len\_mtmp > 0)

{

lastfive\_m = &obj1.m\_city[len\_m - 5];

}

else

{

lastfive\_m = obj1.m\_city;

}

if (strcmp(lastfive\_m, atmp) == 1)

{

delete[] atmp;

return false;

}

else

{

delete[] atmp;

return true;

}

}

derived3& operator &= (derived3& obj1, derived3& obj2)

{

int len\_m = strlen(obj1.m\_city);

int len\_mtmp = len\_m - 5;

if (len\_mtmp > 0)

{

obj1.m\_city[len\_mtmp] = { '\0' };

strcat(obj1.m\_city, obj2.m\_city);

}

else

{

strcpy(obj1.m\_city, obj2.m\_city);

}

return obj1;

}

**Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <conio.h>

#include <Windows.h>

#include <locale.h>

#include <iostream>

#include "phone.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

char country[] = "1";

char city[] = "CityName";

int year = 1;

char producer[] = "1";

char model[] = "1";

double size = 1;

int price = 1;

int a = 123456789;

char b[11] = { 'a','b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', '\0' };

char\* p = b;

derived3 A;

A.Input(city, country, producer, model, size, price);

A.Output();

derived3 B;

strcpy(city, "CityNor");

B.Input(city, country, producer, model, size, price);

B.Output();

if ((A < a) == true)

{

cout << "Число больше 5ти символов строки в поле m\_city" << endl << "Число: " << a << endl;

cout << A.m\_city << endl;

}

else

{

cout << "Число меньше 5ти символов строки в поле m\_city" << endl << "Число: " << a << endl;

}

//////////////////////////////////////////////////////////////////////////////////////////////////////

if ((A < p) == true)

{

cout << "Строка больше 5ти символов строки в поле m\_city" << endl << "Строка: " << b << endl;

}

else

{

cout << "Строка меньше 5ти символов строки в поле m\_city" << endl << "Строка: " << b << endl;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////

if ((A < B) == true)

{

cout << "m\_city второго объекта больше 5ти символов строки в поле m\_city первого объекта" << endl;

}

else

{

cout << "m\_city второго объекта меньше 5ти символов строки в поле m\_city первого объекта" << endl;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////

A &= a;

cout << A.m\_city << endl;

////////////////////////////////////////////////////////////////////////////////////////////////////////

A &= b;

cout << A.m\_city << endl;

////////////////////////////////////////////////////////////////////////////////////////////////////////

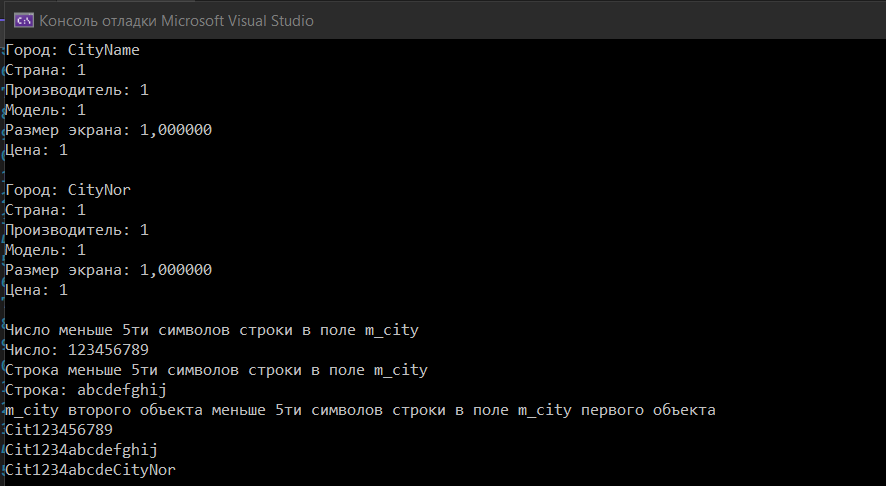
A &= B;

cout << A.m\_city << endl;

return 0;

}

**Результаты работы программы**



Вывод: Мы изучили механизмы перегрузки операторов. Получили навыки реализации перегруженных операторов.