Выполнили:

студенты группы 19ВВ2

Гусев Д.О.

Кубасов И.М.

Приняли:

Синев М.П.

Дорошенко И.Н.

Пенза 2021

ОТЧЕТ

по лабораторной работе №3

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 №2. 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 **string**.
* Параметр ссылки на исходный класс **Derived3**.

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 **string**. 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 **string**. 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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab3oop

{

interface Interface

{

void Input();

}

abstract class Abstruct

{

public abstract void Output();

};

class Phone : Abstruct, Interface

{

private string m\_producer;

public string Producer

{

get

{

return this.m\_producer;

}

set

{

this.m\_producer = value;

}

}

private string m\_model;

public string Model

{

get

{

return this.m\_model;

}

set

{

this.m\_model = value;

}

}

private double m\_size;

protected double Size

{

get

{

return this.m\_size;

}

set

{

this.m\_size = value;

}

}

private int m\_price;

public int Price

{

get

{

return this.m\_price;

}

set

{

this.m\_price = value;

}

}

public Phone()

{

Producer = "";

Model = "";

Size = -1;

Price = -1;

}

public Phone(string producer = "Samsung", string model = "A50", double size = 22.2, int price = 19992)

{

Producer = producer;

Model = model;

Size = size;

Price = price;

}

public void Input()

{

Console.WriteLine("Введите производителя: ");

Producer = Console.ReadLine();

Console.WriteLine("Введите модель: ");

Model = Console.ReadLine();

Console.WriteLine("Введите размер экрана: ");

Size = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Введите цену: ");

Price = Convert.ToInt32(Console.ReadLine());

}

public override void Output()

{

if (Size == -1)

{

Console.WriteLine("Объект не инициализирован!");

}

else

{

Console.WriteLine("Производитель: " + Producer + "\n" + "Модель: " + Model + "\n" + "Размер экрана: " + Size + "\n" + "Цена: " + Price);

}

}

~Phone()

{

Producer = "";

Model = "";

Size = -1;

Price = -1;

}

};

class Derived1 : Phone

{

private string m\_country;

public string Country

{

get

{

return this.m\_country;

}

set

{

m\_country = value;

}

}

public Derived1()

{

Country = "";

}

public Derived1(string country = "Япония", string producer = "Samsung", string model = "A50", double size = 22.2, int price = 19992)

{

Producer = producer;

Model = model;

Size = size;

Price = price;

Country = country;

}

public void Input()

{

Console.WriteLine("Введите страну: ");

Country = Console.ReadLine();

base.Input();

}

public override void Output()

{

base.Output();

if (Size != -1)

{

Console.WriteLine("Страна: " + Country);

}

}

~Derived1()

{

Country = "";

}

};

class Derived2 : Phone

{

private int m\_year;

public int Year

{

get

{

return this.m\_year;

}

set

{

m\_year = value;

}

}

public Derived2()

{

Year = -1;

}

public Derived2(int year = 2018, string producer = "Samsung", string model = "A50", double size = 22.2, int price = 19992)

{

Producer = producer;

Model = model;

Size = size;

Price = price;

Year = year;

}

public void Input()

{

Console.WriteLine("Введите год: ");

Year = Convert.ToInt32(Console.ReadLine());

base.Input();

}

public override void Output()

{

base.Output();

if (Size != -1)

{

Console.WriteLine("Год: " + Year);

}

}

~Derived2()

{

Year = -1;

}

};

class Derived3 : Derived2

{

private string m\_city;

public string City

{

get

{

return this.m\_city;

}

set

{

m\_city = value;

}

}

public Derived3()

{

City = "";

}

public Derived3(string city = "Токио", int year = 2020, string producer = "Samsung", string model = "A50", double size = 22.2, int price = 19992)

{

Producer = producer;

Model = model;

Size = size;

Price = price;

Year = year;

City = city;

}

public void Input()

{

Console.WriteLine("Введите город: ");

City = Console.ReadLine();

base.Input();

}

public override void Output()

{

base.Output();

if (Size != -1)

{

Console.WriteLine("Город: " + City);

}

}

public static Derived3 operator &(Derived3 obj, int a)

{

string temp\_a = Convert.ToString(a);

int genre\_len = obj.City.Length - 5;

if (genre\_len > 0)

{

obj.City = obj.City.Remove(genre\_len);

obj.City += temp\_a;

}

else

{

obj.City = temp\_a;

}

return obj;

}

public static Derived3 operator &(Derived3 obj, string str)

{

string temp\_str = str;

int City\_len = obj.City.Length - 5;

if (City\_len > 0)

{

obj.City = obj.City.Remove(City\_len);

obj.City += temp\_str;

}

else

{

obj.City = temp\_str;

}

return obj;

}

public static Derived3 operator &(Derived3 obj, Derived3 obj2)

{

string temp\_str = obj2.City;

int City\_len = obj.City.Length - 5;

if (City\_len > 0)

{

obj.City = obj.City.Remove(City\_len);

obj.City += temp\_str;

}

else

{

obj.City = temp\_str;

}

return obj;

}

public static bool operator <(Derived3 obj, int a)

{

string temp\_a = Convert.ToString(a);

int temp\_a\_len = temp\_a.Length;

string firstfive;

if ((temp\_a\_len - 5) > 0)

{

firstfive = temp\_a.Substring(0, 5);

}

else

{

firstfive = temp\_a;

}

string temp\_City = obj.City;

int temp\_City\_len = temp\_City.Length;

string lastfive\_ptr;

if ((temp\_City\_len - 5) > 0)

{

lastfive\_ptr = temp\_City.Substring(temp\_City.Length - 5);

}

else

{

lastfive\_ptr = temp\_City;

}

if (lastfive\_ptr.CompareTo(firstfive) < 0)

{

return true;

}

else

{

return false;

}

}

public static bool operator >(Derived3 obj, int a)

{

string temp\_a = Convert.ToString(a);

int temp\_a\_len = temp\_a.Length;

string firstfive;

if ((temp\_a\_len - 5) > 0)

{

firstfive = temp\_a.Substring(0, 5);

}

else

{

firstfive = temp\_a;

}

string temp\_City = obj.City;

int temp\_City\_len = temp\_City.Length;

string lastfive\_ptr;

if ((temp\_City\_len - 5) > 0)

{

lastfive\_ptr = temp\_City.Substring(temp\_City.Length - 5);

}

else

{

lastfive\_ptr = temp\_City;

}

if (lastfive\_ptr.CompareTo(firstfive) >= 0)

{

return true;

}

else

{

return false;

}

}

public static bool operator <(Derived3 obj, string a)

{

string temp\_a = a;

int temp\_a\_len = temp\_a.Length;

string firstfive;

if ((temp\_a\_len - 5) > 0)

{

firstfive = temp\_a.Substring(0, 5);

}

else

{

firstfive = temp\_a;

}

string temp\_City = obj.City;

int temp\_City\_len = temp\_City.Length;

string lastfive\_ptr;

if ((temp\_City\_len - 5) > 0)

{

lastfive\_ptr = temp\_City.Substring(temp\_City.Length - 5);

}

else

{

lastfive\_ptr = temp\_City;

}

if (lastfive\_ptr.CompareTo(firstfive) < 0)

{

return true;

}

else

{

return false;

}

}

public static bool operator >(Derived3 obj, string a)

{

string temp\_a = a;

int temp\_a\_len = temp\_a.Length;

string firstfive;

if ((temp\_a\_len - 5) > 0)

{

firstfive = temp\_a.Substring(0, 5);

}

else

{

firstfive = temp\_a;

}

string temp\_City = obj.City;

int temp\_City\_len = temp\_City.Length;

string lastfive\_ptr;

if ((temp\_City\_len - 5) > 0)

{

lastfive\_ptr = temp\_City.Remove(temp\_City\_len - 5);

}

else

{

lastfive\_ptr = temp\_City;

}

if (lastfive\_ptr.CompareTo(firstfive) >= 0)

{

return true;

}

else

{

return false;

}

}

public static bool operator <(Derived3 obj, Derived3 a)

{

string temp\_a = a.City;

int temp\_a\_len = temp\_a.Length;

string firstfive;

if ((temp\_a\_len - 5) > 0)

{

firstfive = temp\_a.Substring(0, 5);

}

else

{

firstfive = temp\_a;

}

string temp\_City = obj.City;

int temp\_City\_len = temp\_City.Length;

string lastfive\_ptr;

if ((temp\_City\_len - 5) > 0)

{

lastfive\_ptr = temp\_City.Substring(temp\_City.Length - 5);

}

else

{

lastfive\_ptr = temp\_City;

}

if (lastfive\_ptr.CompareTo(firstfive) < 0)

{

return true;

}

else

{

return false;

}

}

public static bool operator >(Derived3 obj, Derived3 a)

{

string temp\_a = a.City;

int temp\_a\_len = temp\_a.Length;

string firstfive;

if ((temp\_a\_len - 5) > 0)

{

firstfive = temp\_a.Substring(0, 5);

}

else

{

firstfive = temp\_a;

}

string temp\_City = obj.City;

int temp\_City\_len = temp\_City.Length;

string lastfive\_ptr;

if ((temp\_City\_len - 5) > 0)

{

lastfive\_ptr = temp\_City.Remove(temp\_City\_len - 5);

}

else

{

lastfive\_ptr = temp\_City;

}

if (lastfive\_ptr.CompareTo(firstfive) >= 0)

{

return true;

}

else

{

return false;

}

}

~Derived3()

{

City = "";

}

};

internal class Program

{

static void f1(Derived3 a, Derived3 b, int i, string str)

{

Console.WriteLine("\n");

Console.WriteLine("\n");

bool P = a < b;

if (P == true)

{

Console.WriteLine("TRUE");

}

else

{

Console.WriteLine("FALSE");

}

Console.WriteLine("\n");

P = a < b;

if (P == true)

{

Console.WriteLine("TRUE");

}

else

{

Console.WriteLine("FALSE");

}

Console.WriteLine("\n");

P = a < b;

if (P == true)

{

Console.WriteLine("TRUE");

}

else

{

Console.WriteLine("FALSE");

}

}

static void f2(Derived3 a, Derived3 b, int i, string str)

{

a &= i;

a.Output();

Console.WriteLine("\n");

a &= str;

a.Output();

Console.WriteLine("\n");

a &= b;

a.Output();

}

static int Main(string[] args)

{

int i = 123;

string str = "MyString";

Derived3 a = new Derived3();

a.Input();

Derived3 b = new Derived3();

b.Input();

f1(a, b, i, str);

f2(a, b, i, str);

return 0;

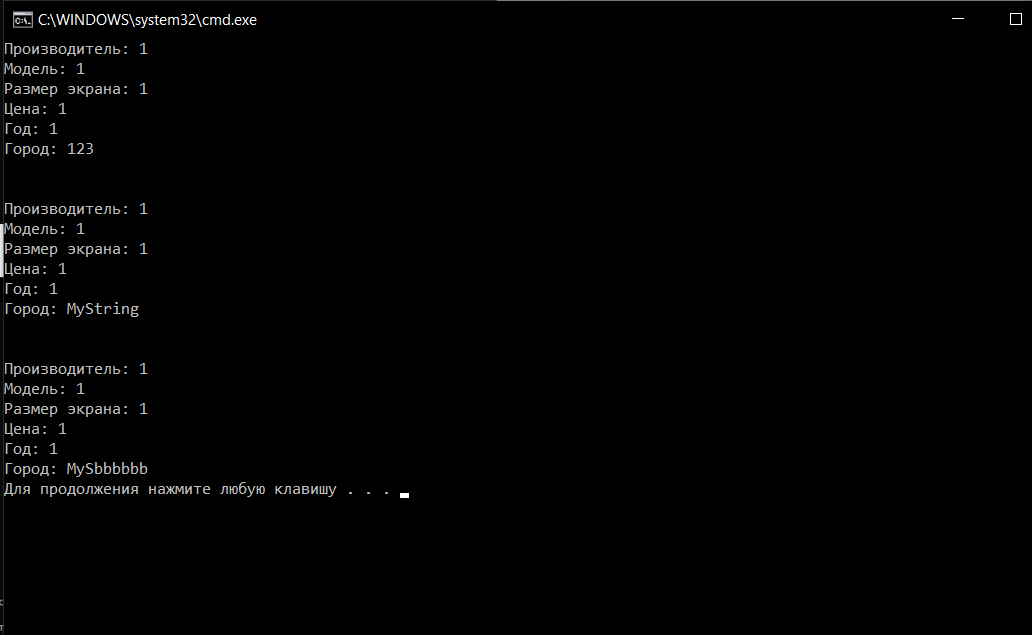
}

}

}

### Результаты работы программы





### Вывод:

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