Пенза 2020

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Выполнил:

студенты группы 17ВВ1

Кокин Денис

Беленков Никита

Принял:

Дорошенко И.Н.

**ОТЧЁТ**

по лабораторной работе №3

по курсу «Технологии программирования»

на тему «Операторы»

**Цель работы:** Изучение механизмов перегрузки операторов. Получение навыков реализации перегруженных операторов.

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:

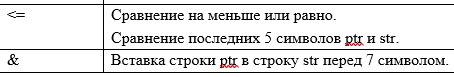
Логическая операция указана в варианте задания.

* 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 программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Confrontation

{

class Township : Locality

{

public int SchoolCount;

public override void output()

{

base.output();

Console.WriteLine("A count of schools: " + this.SchoolCount);

}

public void input()

{

base.input();

Console.Write("Enter a count of schools: ");

SchoolCount = Int32.Parse(Console.ReadLine());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Confrontation

{

abstract class OutputAbstract

{

public abstract void output();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Confrontation

{

interface InputInterface

{

void input();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Confrontation

{

class City : Locality

{

public bool isRegionCenter;

public override void output()

{

base.output();

Console.WriteLine("This is a region center: " + this.isRegionCenter);

}

public void input()

{

base.input();

Console.Write("Is this a region center?: ");

isRegionCenter = System.Convert.ToBoolean(Console.ReadLine());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Confrontation

{

class Locality : OutputAbstract, InputInterface

{

public String Name;

public String Region;

public Locality()

{

Population = 0;

Budget = 0;

Code = 0;

}

public Locality(int Population, int Budget, int Code, String Name = "Unnamed", String Region = "Unnamed")

{

this.Population = Population;

this.Budget = Budget;

this.Code = Code;

this.Name = Name;

this.Region = Region;

}

public Locality(String Name, String Region, int Population = 0, int Budget = 0, int Code = 0)

{

this.Population = Population;

this.Budget = Budget;

this.Code = Code;

this.Name = Name;

this.Region = Region;

}

public void ClassInfo()

{

Console.WriteLine("Full region name: " + this.Name + " in " + this.Region);

this.AbleBodied();

this.BudgetToUSD();

}

~Locality()

{

Name = null;

Region = null;

Population = 0;

Budget = 0;

Code = 0;

}

public void input()

{

Console.Write("Input a name of the region: ");

this.Name = Console.ReadLine();

Console.Write("Input a regional center of the region: ");

this.Region = Console.ReadLine();

Console.Write("Input a population of the region: ");

this.Population = Int32.Parse(Console.ReadLine());

Console.Write("Input a budget of the region: ");

this.Budget = Int32.Parse(Console.ReadLine());

Console.Write("Input a code of the region: ");

this.Code = Int32.Parse(Console.ReadLine());

}

public override void output()

{

Console.WriteLine("A name of the region: " + this.Name);

Console.WriteLine("A regional center of the region: " + this.Region);

Console.WriteLine("A population of the region: " + this.Population);

Console.WriteLine("A budget of the region: " + this.Budget);

Console.WriteLine("A code of the region: " + this.Code);

}

protected int Population;

protected int Code;

protected int Budget;

protected void AbleBodied()

{

Console.WriteLine("Able-bodied population: " + (this.Population / 2));

}

private void BudgetToUSD()

{

Console.WriteLine("Budget of locality: " + (this.Budget / 70) + " USD");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Confrontation

{

class Village : Township

{

public int ShopCount;

public override void output()

{

base.output();

Console.WriteLine("A count of shops: " + this.ShopCount);

}

public void input()

{

base.input();

Console.Write("Enter a count of shops: ");

ShopCount = Int32.Parse(Console.ReadLine());

}

public static bool operator <=(Village pSampleVillage, int pParam)

{

if ((pSampleVillage.Name.Length <= pParam))

return true;

else

return false;

}

public static bool operator >=(Village pSampleVillage, int pParam)

{

if ((pSampleVillage.Name.Length >= pParam))

return true;

else

return false;

}

public static bool operator <=(Village pSampleVillage, string pParam)

{

if (pSampleVillage.Name.Length < pParam.Length)

return true;

if (pSampleVillage.Name.Length >= 5 && pParam.Length >= 5)

{

if (pSampleVillage.Name.Substring(

pSampleVillage.Name.Length - 5).Equals(

pParam.Substring(pParam.Length - 5)

))

return true;

else

return false;

}

return false;

}

public static bool operator >=(Village pSampleVillage, string pParam)

{

if (pSampleVillage.Name.Length > pParam.Length)

return true;

if (pSampleVillage.Name.Length >= 5 && pParam.Length >= 5)

{

if (pSampleVillage.Name.Substring(

pSampleVillage.Name.Length - 5).Equals(

pParam.Substring(pParam.Length - 5)

))

return true;

else

return false;

}

return false;

}

public static bool operator >=(Village pSampleVillage, Village pParam)

{

if (pSampleVillage.Name.Length > pParam.Name.Length)

return true;

if (pSampleVillage.Name.Length >= 5 && pParam.Name.Length >= 5)

{

if (pSampleVillage.Name.Substring(

pSampleVillage.Name.Length - 5).Equals(

pParam.Name.Substring(pParam.Name.Length - 5)

))

return true;

else

return false;

}

return false;

}

public static bool operator <=(Village pSampleVillage, Village pParam)

{

if (pSampleVillage.Name.Length < pParam.Name.Length)

return true;

if (pSampleVillage.Name.Length >= 5 && pParam.Name.Length >= 5)

{

if (pSampleVillage.Name.Substring(

pSampleVillage.Name.Length - 5).Equals(

pParam.Name.Substring(pParam.Name.Length - 5)

))

return true;

else

return false;

}

return false;

}

static public Village operator &(Village pSampleVillage, int pParam){

pSampleVillage.Name = pSampleVillage.Name.Substring(0, 6) + pParam + pSampleVillage.Name.Substring(6);

return pSampleVillage;

}

static public Village operator &(Village pSampleVillage, string pParam)

{

pSampleVillage.Name = pSampleVillage.Name.Substring(0, 6) + pParam +pSampleVillage.Name.Substring(6);

return pSampleVillage;

}

static public Village operator &(Village pSampleVillage, Village pParam)

{

pSampleVillage.Name = pSampleVillage.Name.Substring(0, 6) + pParam.Name + pSampleVillage.Name.Substring(6);

return pSampleVillage;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Confrontation {

class Program{

static void Main(string[] args){

Village SampleVillage = new Village();

SampleVillage.input();

Village SecondVillageSample = new Village();

SecondVillageSample.input();

Console.WriteLine(SampleVillage <= 5);

Console.WriteLine(SampleVillage <= "sdzxc");

Console.WriteLine(SampleVillage <= SecondVillageSample);

Console.WriteLine((SampleVillage & 5).Name);

Console.WriteLine((SampleVillage & "Test").Name);

Console.WriteLine((SampleVillage & SecondVillageSample).Name);

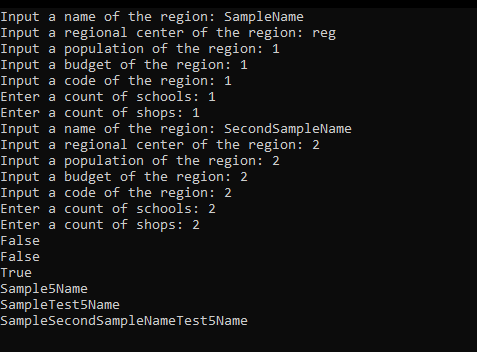
Console.Read();

}

}

}

**Результаты работы программы**



**Вывод:** студенты изучили механизмы сохранения и восстановления объектов данных. Получили навыки разработки самовосстанавливающихся структур данных.