МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

Кафедра АПЕПС

Звіт до лабораторної роботи №5-7

Об’єктно-орієнтоване програмування-2

Варіант 9

Виконав:

студент 2 курсу,

групи ТВ-51, ТЕФ

Іванів А.П.

Київ — 2017

**Код програми:**

**Date1.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

[Serializable]

class Date1

{

int year, week;

public Date1() : this(2000, 1) { }

public Date1(int year, int week)

{

this.year = year;

this.week = week;

}

public int Year

{

get { return year; }

set

{

if (value < 0) year = 0; else year = value;

}

}

public int Week

{

get { return week; }

set

{

if (value < 1 || value > 53) week = 1; else week = value;

}

}

}

}

**Data2.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

[Serializable]

class Date2

{

int year, month, day;

public Date2() : this(2000, 1, 1) { }

public Date2(int year, int month, int day)

{

this.year = year;

this.month = month;

this.day = day;

}

public int Year

{

get { return year; }

set

{

if (value < 0) year = 0; else year = value;

}

}

public int Month

{

get { return month; }

set

{

if (value < 1 || value > 12) month = 1; else month = value;

}

}

public int Day

{

get { return day; }

set

{

if (value < 1 || value > 31) day = 1; else day = value;

}

}

}

}

**FlowerType.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

[Serializable]

class FlowerType

{

string name, color;

public FlowerType() : this("rose", "red") { }

public FlowerType(string name, string color)

{

this.name = name;

this.color = color;

}

public string Name

{

get { return name; }

set { name = value; }

}

public string Color

{

get { return color; }

set { color = value; }

}

}

}

**FlowerGoods.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

[Serializable]

class FlowerGoods

{

private FlowerType flowerType;

SalesType salesType;

public FlowerType FlowerType

{

get { return flowerType; }

set { flowerType = value; }

}

public SalesType SalesType

{

get { return salesType; }

set { salesType = value; }

}

public FlowerGoods() : this(new FlowerType(), new SalesType()) { }

public FlowerGoods(FlowerType flowerType, SalesType salesType)

{

this.FlowerType = flowerType;

this.SalesType = salesType;

}

//public override string ToString()

//{

// return FlowerType.Name + " - " + FlowerType.Color + " " + SalesType;

//}

}

}

**SalesType.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

enum SalesType

{

Pot,

Bouquet,

Apiece

}

}

**SupplyOfGoods.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

[Serializable]

class SupplyOfGoods

{

private Date2 dateOfSupply;

private Consignment consignment;

public Date2 DateOfSupply

{

get { return dateOfSupply; }

set

{

if (value.Year > 1900)

dateOfSupply = value;

}

}

public Consignment Consignment

{

get { return consignment; }

set { consignment = value; }

}

public SupplyOfGoods() : this(2001, 6, 12, new Consignment()) { }

public SupplyOfGoods(int y, int m, int d, Consignment consignment)

{

this.DateOfSupply = new Date2(y, m, d);

this.Consignment = consignment;

}

public override string ToString()

{

return Consignment.ToString() + " " + DateOfSupply.Day + "." + DateOfSupply.Month + "." + DateOfSupply.Year;

}

}

}

**Consigment.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

[Serializable]

class Consignment

{

private FlowerGoods flowerGoods;

private int count;

public FlowerGoods FlowerGoods

{

get { return flowerGoods; }

set { flowerGoods = value; }

}

public int Count

{

get { return count; }

set

{

if (count >= 0)

count = value;

else

count = 0;

}

}

public Consignment() : this(new FlowerGoods(), 0) { }

public Consignment(FlowerGoods flowerGoods, int count)

{

this.FlowerGoods = flowerGoods;

this.Count = count;

}

public override string ToString()

{

return FlowerGoods.FlowerType.Name + " - " + FlowerGoods.FlowerType.Color + ", in: " + FlowerGoods.SalesType.ToString() + ", count: " + Count;

}

}

}

**SalesOfGoods.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

[Serializable]

class SalesOfGoods

{

private Date2 dateOfSales;

private Consignment consignment;

public Date2 DateOfSales

{

get { return dateOfSales; }

set

{

if (value.Year > 1900)

dateOfSales = value;

}

}

public Consignment Consignment

{

get { return consignment; }

set { consignment = value; }

}

public SalesOfGoods() : this(2001, 6, 12, new Consignment()) { }

public SalesOfGoods(int y, int m, int d, Consignment consignment)

{

this.DateOfSales = new Date2(y, m, d);

this.Consignment = consignment;

}

public override string ToString()

{

return Consignment.ToString() + " " + DateOfSales.Day + "." + DateOfSales.Month + "." + DateOfSales.Year;

}

}

}

**WeekBalance.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

[Serializable]

class WeekBalance

{

private Date1 date;

List<SupplyOfGoods> supplyOfGoods;

List<SalesOfGoods> salesOfGoods;

public double sallary

{

get

{

double total = 0;

for (int i = 0; i < salesOfGoods.Count; i++)

total += 3.5 \* salesOfGoods[i].Consignment.Count;

return total;

}

}

public Date1 Date

{

get { return date; }

set { date = value; }

}

public WeekBalance() : this(new Date1(), new List<SupplyOfGoods>(), new List<SalesOfGoods>())

{ }

public WeekBalance(Date1 date, List<SupplyOfGoods> supplyOfGoods, List<SalesOfGoods> salesOfGoods)

{

Date = date;

this.supplyOfGoods = supplyOfGoods;

this.salesOfGoods = salesOfGoods;

}

public SalesOfGoods this[int index]

{

get

{

if (index >= 0 && index < salesOfGoods.Count)

return salesOfGoods[index];

else

throw new ArgumentException("Wrong index");

}

set

{

if (index >= 0 && index < salesOfGoods.Count)

salesOfGoods[index] = value;

else

throw new ArgumentException("Wrong index");

}

}

public string getInfo()

{

return supplyOfGoods[0].Consignment.FlowerGoods.FlowerType.Name;

}

public double getDouble()

{

return sallary;

}

public int getSupplyCount()

{

return supplyOfGoods.Count;

}

public int getSallaryCount()

{

return salesOfGoods.Count;

}

public int getSupplyProductsNumber()

{

int k = 0;

for (int i = 0; i < supplyOfGoods.Count; i++)

{

k += supplyOfGoods[i].Consignment.Count;

}

return k;

}

public int getSallaryProductsNumber()

{

int k = 0;

for (int i = 0; i < salesOfGoods.Count; i++)

{

k += salesOfGoods[i].Consignment.Count;

}

return k;

}

public static implicit operator string(WeekBalance t)

{

string tmp = "Week sallary " + t.sallary+ "\n";

for (int i = 0; i < t.getSupplyCount(); i++)

{

tmp += t[i].DateOfSales.Day + "." + t[i].DateOfSales.Month + "." + t[i].DateOfSales.Year + " | ";

tmp += t[i].Consignment.ToString();

if (i + 1 < t.getSupplyCount())

tmp += ", ";

else

tmp += ".";

tmp += "\n";

}

return tmp;

}

}

}

**Program.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_7

{

class Program

{

static void Main(string[] args)

{

List<SupplyOfGoods> supplyOfGoodsList = new List<SupplyOfGoods>();

supplyOfGoodsList.Add(new SupplyOfGoods(2009, 1, 8, new Consignment(new FlowerGoods(new FlowerType("rose", "white"), SalesType.Apiece), 150)));

supplyOfGoodsList.Add(new SupplyOfGoods(2009, 1, 9, new Consignment(new FlowerGoods(new FlowerType("fialka", "blue"), SalesType.Bouquet), 90)));

supplyOfGoodsList.Add(new SupplyOfGoods(2009, 1, 10, new Consignment(new FlowerGoods(new FlowerType("romashka", "white"), SalesType.Apiece), 120)));

supplyOfGoodsList.Add(new SupplyOfGoods(2009, 1, 11, new Consignment(new FlowerGoods(new FlowerType("rose", "green"), SalesType.Pot), 100)));

supplyOfGoodsList.Add(new SupplyOfGoods(2009, 1, 12, new Consignment(new FlowerGoods(new FlowerType("fialka", "red"), SalesType.Pot), 105)));

supplyOfGoodsList.Add(new SupplyOfGoods(2009, 1, 13, new Consignment(new FlowerGoods(new FlowerType(), SalesType.Bouquet), 95)));

List<SalesOfGoods> salesOfGoodsList = new List<SalesOfGoods>();

salesOfGoodsList.Add(new SalesOfGoods(2009, 1, 9, new Consignment(new FlowerGoods(new FlowerType("rose", "white"), SalesType.Apiece), 60)));

salesOfGoodsList.Add(new SalesOfGoods(2009, 1, 10, new Consignment(new FlowerGoods(new FlowerType("fialka", "blue"), SalesType.Bouquet), 85)));

salesOfGoodsList.Add(new SalesOfGoods(2009, 1, 11, new Consignment(new FlowerGoods(new FlowerType("romashka", "white"), SalesType.Apiece), 110)));

salesOfGoodsList.Add(new SalesOfGoods(2009, 1, 12, new Consignment(new FlowerGoods(new FlowerType("rose", "green"), SalesType.Pot), 99)));

salesOfGoodsList.Add(new SalesOfGoods(2009, 1, 13, new Consignment(new FlowerGoods(new FlowerType("fialka", "red"), SalesType.Pot), 90)));

salesOfGoodsList.Add(new SalesOfGoods(2009, 1, 14, new Consignment(new FlowerGoods(new FlowerType(), SalesType.Bouquet), 95)));

WeekBalance weekBalance = new WeekBalance(new Date1(2009, 2), supplyOfGoodsList, salesOfGoodsList);

for (int i = 0; i < weekBalance.getSupplyCount(); i++)

Console.WriteLine(weekBalance[i]);

Console.WriteLine("Week sallary - {0}", weekBalance.sallary);

Console.ReadKey(true);

Console.Write("Enter size: ");

int n = Convert.ToInt32(Console.ReadLine());

DataBase ar = new DataBase(n);

Observer ob1 = new Observer("One");

Observer ob2 = new Observer("Two");

ar.Register(ob1.proc);

ar.AddToArray(4, weekBalance);

Console.Write("\n\n");

for (int i = 0; i < 10; i++)

{

List<SupplyOfGoods> supplyOfGoodsList2 = new List<SupplyOfGoods>();

supplyOfGoodsList2.Add(new SupplyOfGoods(2010, 1 + i, 8, new Consignment(new FlowerGoods(new FlowerType("rose", "white"), SalesType.Apiece), 150 + i \* 3)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2010, 1 + i, 9, new Consignment(new FlowerGoods(new FlowerType("fialka", "blue"), SalesType.Bouquet), 90 + i \* 2)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2010, 1 + i, 10, new Consignment(new FlowerGoods(new FlowerType("romashka", "white"), SalesType.Apiece), 120 + i)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2010, 1 + i, 11, new Consignment(new FlowerGoods(new FlowerType("rose", "green"), SalesType.Pot), 100 + i \* 3)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2010, 1 + i, 12, new Consignment(new FlowerGoods(new FlowerType("fialka", "red"), SalesType.Pot), 105 + i)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2010, 1 + i, 13, new Consignment(new FlowerGoods(new FlowerType(), SalesType.Bouquet), 95 + i \* 2)));

List<SalesOfGoods> salesOfGoodsList2 = new List<SalesOfGoods>();

salesOfGoodsList2.Add(new SalesOfGoods(2010, 1 + i, 9, new Consignment(new FlowerGoods(new FlowerType("rose", "white"), SalesType.Apiece), 60 + i \* 3)));

salesOfGoodsList2.Add(new SalesOfGoods(2010, 1 + i, 10, new Consignment(new FlowerGoods(new FlowerType("fialka", "blue"), SalesType.Bouquet), 85 + i \* 2)));

salesOfGoodsList2.Add(new SalesOfGoods(2010, 1 + i, 11, new Consignment(new FlowerGoods(new FlowerType("romashka", "white"), SalesType.Apiece), 110 - i)));

salesOfGoodsList2.Add(new SalesOfGoods(2010, 1 + i, 12, new Consignment(new FlowerGoods(new FlowerType("rose", "green"), SalesType.Pot), 99 + i \* 3)));

salesOfGoodsList2.Add(new SalesOfGoods(2010, 1 + i, 13, new Consignment(new FlowerGoods(new FlowerType("fialka", "red"), SalesType.Pot), 90 - i)));

salesOfGoodsList2.Add(new SalesOfGoods(2010, 1 + i, 14, new Consignment(new FlowerGoods(new FlowerType(), SalesType.Bouquet), 95 + i \* 2)));

var t = new WeekBalance(new Date1(), supplyOfGoodsList2, salesOfGoodsList2);

ar.AddToArray(i, t);

Console.WriteLine(i + ":" + t);

}

Console.WriteLine();

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

{

Console.WriteLine(i + ":");

List<SupplyOfGoods> supplyOfGoodsList2 = new List<SupplyOfGoods>();

supplyOfGoodsList2.Add(new SupplyOfGoods(2011, 1, 8, new Consignment(new FlowerGoods(new FlowerType("rose", "white"), SalesType.Apiece), 70 + i \* 3)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2011, 1, 9, new Consignment(new FlowerGoods(new FlowerType("fialka", "blue"), SalesType.Bouquet), 30 + i \* 3)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2011, 1, 10, new Consignment(new FlowerGoods(new FlowerType("romashka", "white"), SalesType.Apiece), 12 + i \* 3)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2011, 1, 11, new Consignment(new FlowerGoods(new FlowerType("rose", "green"), SalesType.Pot), 30 + i \* 3)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2011, 1, 12, new Consignment(new FlowerGoods(new FlowerType("fialka", "red"), SalesType.Pot), 20 + i \* 3)));

supplyOfGoodsList2.Add(new SupplyOfGoods(2011, 1, 13, new Consignment(new FlowerGoods(new FlowerType(), SalesType.Bouquet), 15 + i \* 4)));

List<SalesOfGoods> salesOfGoodsList2 = new List<SalesOfGoods>();

salesOfGoodsList2.Add(new SalesOfGoods(2011, 1, 9, new Consignment(new FlowerGoods(new FlowerType("rose", "white"), SalesType.Apiece), 60 + i \* 2)));

salesOfGoodsList2.Add(new SalesOfGoods(2011, 1, 10, new Consignment(new FlowerGoods(new FlowerType("fialka", "blue"), SalesType.Bouquet), 25 + i \* 2)));

salesOfGoodsList2.Add(new SalesOfGoods(2011, 1, 11, new Consignment(new FlowerGoods(new FlowerType("romashka", "white"), SalesType.Apiece), 11 + i \* 2)));

salesOfGoodsList2.Add(new SalesOfGoods(2011, 1, 12, new Consignment(new FlowerGoods(new FlowerType("rose", "green"), SalesType.Pot), 2 + i)));

salesOfGoodsList2.Add(new SalesOfGoods(2011, 1, 13, new Consignment(new FlowerGoods(new FlowerType("fialka", "red"), SalesType.Pot), 5 + i)));

salesOfGoodsList2.Add(new SalesOfGoods(2011, 1, 14, new Consignment(new FlowerGoods(new FlowerType(), SalesType.Bouquet), 5 + i)));

var t = new WeekBalance(new Date1(), supplyOfGoodsList2, salesOfGoodsList2);

ar.Insert(i, t);

Console.WriteLine(t);

}

//ar.SearchByStringInArrayList("rose");

//ar.SortByDouble();

Calculate.Table(Calculate.Function1, 0, 0.1, 3);

Calculate.Table(Calculate.Function2, 0, 0.2, 3);

Calculate.Histo(Calculate.Function1, 0, 0.1, 3);

Calculate.Histo(Calculate.Function2, 0, 0.2, 3);

BinaryFormatter formatter = new BinaryFormatter();

FileStream fs = new FileStream("D:/result/lab7.dat", FileMode.Create);

formatter.Serialize(fs, ar);

fs.Close();

FileInfo file = new FileInfo("D:/result/lab7.txt");

StreamWriter writer = file.CreateText();

writer.Write(ar);

writer.Close();

Console.ReadKey(true);

}

}

}

**Теоретичні питання:**

1. **Що таке перевизначення або заміщення методу при успадкуванні? Наведіть приклад.**

Це перевизначення функції, яка описана в батьківському классі.

class Base

{

public virtual void Foo(int i)

{

Console.WriteLine("Base.Foo(int)");

}

}

class Derived : Base

{

public override void Foo(int i)

{

Console.WriteLine("Derived.Foo(int)");

}

public void Foo(object o)

{

Console.WriteLine("Derived.Foo(object)");

}

}

**2) Чи можуть інтерфейси містити константи, поля, конструктори, чи якісь статичні елементи?**

interface IPrint {

void Print(); void PrintPreview();

}

class Point : IPrint {

protected int xPos; protected int yPos;

public Point(int x, int y)

{

xPos = x; yPos = y;

}

void IPrint.Print() {

Console.WriteLine("Печать точки ({0}, {1})", this.X, this.Y);

}

void IPrint.PrintPreview() {

Console.WriteLine(

"Просмотр перед печатью точки ({0}, {1})", this.X, this.Y);

}

}

**3) Яким у списку параметрів функції має бути параметр з модифікатор params? Наведіть приклад.**

Цей модифікатор дозволяє передавати у вигляді одного логічного параметра змінну кількість аргументів. У кожному методі може бути присутнім тільки один модифікатор params і він повинен обов'язково оголошеним останнім у списку параметрів. В реальності необхідність у використанні модифікатора params виникає не особливо часто, однак він застосовується в багатьох методах всередині бібліотек базових класів

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void MaxArr(ref int i, params int[] arr)

{

// Изначально нужно обязательно выполнить проверку

// на непустоту массива

if (arr.Length == 0)

{

Console.WriteLine("Пустой массив!");

i=0;

return;

}

else

{

if (arr.Length == 1)

{

i = arr[0];

return;

}

}

i = arr[0];

// Ищем максимум

for (int j = 1; j < arr.Length; j++)

if (arr[j] > i)

i = arr[j];

}

static void Main()

{

int result = 0;

int[] arr1 = new int[8];

int[] arr2 = new int[5];

Random ran = new Random();

// Инициализируем оба массива случайными числами

for (int i = 0; i < 8; i++)

arr1[i] = ran.Next(1, 20);

for (int i = 0; i < 5; i++)

arr2[i] = ran.Next(100, 200);

Console.WriteLine("Массив arr1: \n");

foreach (int i in arr1)

Console.Write("{0}\t",i);

MaxArr(ref result, arr1);

Console.WriteLine("Максимум: {0}",result);

Console.WriteLine("\nМассив arr2: \n");

foreach (int i in arr2)

Console.Write("{0}\t", i);

MaxArr(ref result, arr2);

Console.WriteLine("Максимум: {0}", result);

Console.ReadLine();

}

}

}

**4) Чи використовуються процедури доступу get та set при визначенні індексатора? Наведіть приклад.**

Так, використовуються.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class MyArr

{

int[] arr;

public int Length;

public MyArr(int Size)

{

arr = new int[Size];

Length = Size;

}

// Создаем простейший индексатор

public int this[int index]

{

set

{

arr[index] = value;

}

get

{

return arr[index];

}

}

}

class Program

{

static void Main()

{

MyArr arr1 = new MyArr(Size: 5);

Random ran = new Random();

// Инициализируем каждый индекс экземпляра класса arr1

for (int i = 0; i < arr1.Length; i++)

{

arr1[i] = ran.Next(1,100);

Console.Write("{0}\t", arr1[i]);

}

Console.ReadLine();

}

}

}

**5) Для яких цілей застосовуються делегати? Наведіть приклад.**

Опис делегата задає сигнатуру методів, які можуть бути викликані з його допомогою:

[attributes] [specificators] delegate type delegate\_name ([параметри])

Атрибути дозволяють додавати до класу додаткову інформацію про клас.

Приклад опису делегата:

public delegate void D (int i);

Тут описаний тип делегата, який може зберігати посилання на методи, що повертають значення типу void і що приймають один параметр цілого типу.

Найголовніше, що делегат може служити для виклику будь-якого методу з відповідною сигнатурою і повертається типом. Більш того, що викликається метод може бути методом примірника, пов'язаних з окремим об'єктом, або ж статичним методом, пов'язаних з конкретним класом. Значення має лише одне: що повертається тип і сигнатура методу повинні бути узгоджені з тими, які вказані в оголошенні делегата.

**6) Що означає модифікатор параметра функції out на мові C#? Наведіть приклад.**

Модифікатор параметра out служить тільки для передачі значення за межі методу. Тому змінної, використовуваної в якості параметра out, не потрібно (та й марно) привласнювати якесь значення. Більш того, в методі параметр out вважається неініціалізованих, тобто передбачається, що у нього відсутня початкове значення. Це означає, що значення має бути присвоєно даному параметру в методі до його завершення. Отже, після виклику методу параметр out буде містити деякий значення.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

// Метод возвращающий целую и дробную части

// числа, квадрат и корень числа

static int TrNumber(double d, out double dr, out double sqr, out double sqrt)

{

int i = (int)d;

dr = d - i;

sqr = d \* d;

sqrt = Math.Sqrt(d);

return i;

}

static void Main()

{

int i;

double myDr, mySqr, mySqrt, myD = 12.987;

i = TrNumber(myD, out myDr, out mySqr, out mySqrt);

Console.WriteLine("Исходное число: {0}\nЦелая часть числа: {1}\nДробная часть числа: {2}\nКвадрат числа: {3}\nКвадратный корень числа: {4}",myD,i,myDr,mySqr,mySqrt);

Console.ReadLine();

}

}

}