**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інженерії програмного забезпечення**

**ЗВІТ**

з лабораторної роботи № 4

**«Використання ключових слів const та readonly.**

**Передача параметрів у методи.»**

**Варіант 5**

Дисципліна «Об'єктно-орієнтоване програмування»

Спеціальність: **Інженерія програмного забезпечення**

121-ЛР.04-208.22120802

***Студент\_\_Д. М.  Дзундза***

*(підпис)*

*\_\_09.27.2021* \_\_

*(дата)*

***Викладач \_\_\_\_\_\_\_\_С. Ю. Боровльова***

*(підпис)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

Миколаїв – 2021

**Лабораторна робота №4**

|  |  |
| --- | --- |
| Тема роботи: | Використання ключових слів const та readonly. Передача параметрів у методи |

|  |
| --- |
| Мета роботи: |

Ознайомитись з використанням : модифікаторів для параметрів методів. Навчитись використовувати параметри за посиланням та за значенням. Використати статичні змінні та конструктори. **Завдання:**

1. На базі класів із попередньої лабораторної роботи створіть додаток для демонстрації в коді способи передачі параметрів у метод

* За значенням (value та reference)
* Використання ключових слів ref та out в тому числі і out в с#7.
* Застосування ключового слова params
* Передача іменованих аргументів .
* Використання необов‘язкових аргументів
* Створити метод, який може приймає два або більше параметрів

1. На базі класів із попередньої лабораторної роботи створіть додаток для демонстрації в коді:

* Навести приклад використання 3 статичних методів класу Array
* Використання ключових слів const та readonly
* Використання статичних змінних, методів та статичного конструктора

1. \*Додаткове завдання:

* Класс містить два масиви: одновимірний та двовимірний. Реалізувати один індексатор, який може працювати з одновимірним або двовимірним масивом в залежності від ситуації.(0.5 балів)
* Розібрати теоретично різницю між const та readonly.(0.1 бал)
* Розібрати роботу статичних класів. (0.1 бал)

**Лістинг коду:**

using System;

namespace Lection2

{

public static class TemperatureConverter

{

public static double CelsiusToFahrenheit(string temperatureCelsius)

{

double celsius = Double.Parse(temperatureCelsius);

double fahrenheit = (celsius \* 9 / 5) + 32;

return fahrenheit;

}

public static double FahrenheitToCelsius(string temperatureFahrenheit)

{

double fahrenheit = Double.Parse(temperatureFahrenheit);

double celsius = (fahrenheit - 32) \* 5 / 9;

return celsius;

}

}

class Worker

{

int paidDays;

double salary;

const int maxWorkingDays = 24;

public static string dzen = "I have no dzen.";

readonly int minWorkingDays;

int[] firstHalf = new int[15];

int[] secondHalf = new int[15];

static Worker() => dzen = "Beautiful is better than ugly.";

public Worker(int \_paidDays, double \_salary, int \_minWorkingDays = 3)

{

paidDays = \_paidDays; // 22

salary = \_salary;

minWorkingDays = \_minWorkingDays;

firstHalf = new int[(int)(\_paidDays/2)];

secondHalf = new int[(int)(\_paidDays+1/2)];

}

public void Hours(int x)

{

Console.WriteLine("x input = " + x);

x = 16;

Console.WriteLine("x out doesn't change = " + x);

}

public void Hours(ref int x)

{

Console.WriteLine("x input = " + x);

x = 8;

Console.WriteLine("x out = " + x);

}

public void Weekend(out int x)

{

x = (int)(30 - paidDays);

}

public int Salary(params int[] salary\_per\_day)

{

int sum = 0;

for (int i=0; i<salary\_per\_day.Length; i++)

{

sum += salary\_per\_day[i];

}

return sum;

}

public double CalculateTaxes(int daysWorked, double tax = 0.3, double fee = 0.1)

{

// tax - налог на ЗП, fee - Сборы на что-то там

if (daysWorked < minWorkingDays)

{

return 0;

}

double premium = (daysWorked > maxWorkingDays) ? 300 : 700;

double salaryWithoutTaxes = salary \* ((31 - daysWorked) - paidDays);

double taxes = salaryWithoutTaxes \* tax;

double fees = salaryWithoutTaxes \* fee;

double salaryAfterTaxes = salaryWithoutTaxes - taxes - fees;

Console.WriteLine($"Salary withour taxes = {salaryWithoutTaxes}; " +

$"Taxes = {taxes}; " +

$"Fees = {fees}; " +

$"Salary after taxes = {salaryAfterTaxes}.");

return salaryAfterTaxes;

}

public static string Print(params string[] args)

{

string result = dzen+"\n";

foreach (var i in args)

result += i + "\n";

return result;

}

public int this[int n]{

get {

if (n > firstHalf.Length-1 && n < firstHalf.Length-1 + secondHalf.Length-1) {

return secondHalf[n - firstHalf.Length];

} else if (n < firstHalf.Length-1 && n >= 0) {

return firstHalf[n];

}

return 0;

}

set {

if (n > firstHalf.Length && n < firstHalf.Length + secondHalf.Length-1)

{

secondHalf[n - firstHalf.Length] = value;

}

else if (n < firstHalf.Length && n >= 0)

{

firstHalf[n] = value;

}

}

}

}

class Program

{

static void Main(string[] args)

{

var rand = new Random();

int max\_rand\_value = 10;

Console.WriteLine(Worker.dzen);

Worker worker = new Worker(22, 1500);

int firstDaySalary = 100;

int secondDaySalary = 1000;

int thirdDaySalary = 2500;

int fourthDaySalary = 750;

int fifthDaySalary = 904;

double salary = worker.Salary(firstDaySalary, secondDaySalary, thirdDaySalary, fourthDaySalary, fifthDaySalary);

Console.WriteLine($"Salary for last 5 days: = {salary}.");

int howMuchHoursDoesWorkerWorks = 0;

worker.Hours(howMuchHoursDoesWorkerWorks);

Console.WriteLine($"howMuchHoursDoesWorkerWorks out value : {howMuchHoursDoesWorkerWorks}.");

worker.Hours(ref howMuchHoursDoesWorkerWorks);

Console.WriteLine($"Worker works {howMuchHoursDoesWorkerWorks} hours.");

double toPay = worker.CalculateTaxes(5, fee: 0.2, tax: 0.4);

Console.WriteLine($"Worker needs to be paid: {toPay} bucks");

double toPay2 = worker.CalculateTaxes(5, fee: 0.05);

Console.WriteLine($"Worker needs to be paid 2: {toPay2} bucks");

int weekDays;

worker.Weekend(out weekDays);

Console.WriteLine($"Worker have {weekDays} resting days in month.");

worker.Weekend(out var weekDays2);

Console.WriteLine($"2.0 Worker have {weekDays2} resting days in month.");

int[] firstArray = { 5, 2, 6, 7, 1, 4, 5, 9, 2 };

int[] secondArray = new int[3];

Array.Copy(firstArray, secondArray, 3);

Console.WriteLine("Copied array: ");

foreach (int value in secondArray)

Console.Write($" {value},");

Console.WriteLine("\b \b");

Console.WriteLine("Sorted array");

Array.Sort(firstArray);

foreach (int value in firstArray)

Console.Write($" {value},");

Console.WriteLine("\b \b");

Array.Clear(firstArray, 0, 3);

Console.WriteLine($"First array elements after clear: ");

foreach (int value in firstArray)

Console.Write($" {value},");

Console.WriteLine("\b \b");

Console.WriteLine(Worker.Print("Explicit is better than implicit",

"Simple is better than complex",

"Complex is better than complicated"));

for (int i = 0; i < 30; i++)

worker[i] = rand.Next(-max\_rand\_value, max\_rand\_value);

for (int i = 0; i < 30; i++)

Console.Write($" i{i}: { worker[i]},");

Console.WriteLine("\b \b");

Console.WriteLine($"Temp: {TemperatureConverter.FahrenheitToCelsius("35")} ");

Console.ReadKey();

}

}

}

**Результат роботи програми:**

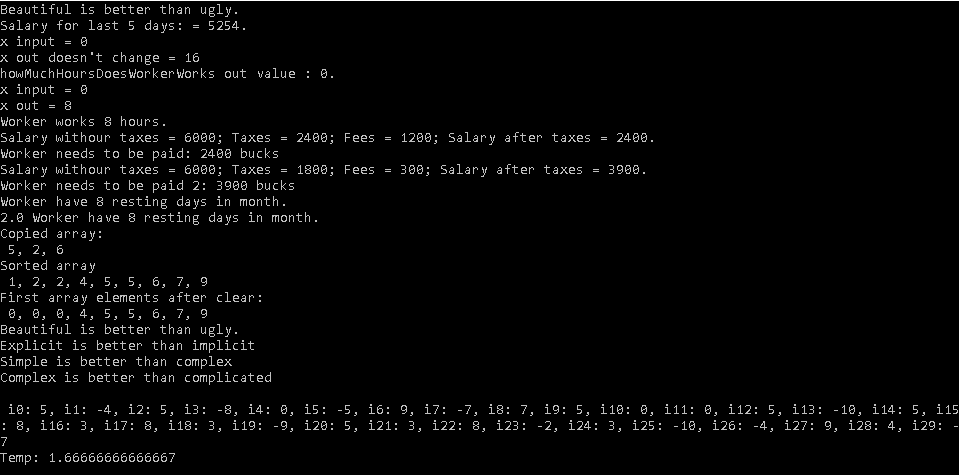


Рисунок 1 – Результат роботи програми

**Питання до захисту:**

1. Типи передачі параметрів.

За значенням або за посиланням.

1. Чим відрізняється передача параметрів з модифікатором ref від передачі з використанням out.

При використанні out обов’язково потрібно змінити значення цієї змінної. При використанні ref ми можемо не змінювати значення змінної.

1. Новий синтаксис використання out.

Worker.Weekend(out var new\_variable)

Console.Write(new\_variable );

1. Використання модифікатора params.

Модифікатор params використовується коли ми хочемо передати змінну кількість параметрів одного типу (**бо generics не підтримується у С#**).

1. Використання параметрів, які мають значення за замовчуванням

public int sum(int a, b=3, c=5){

return a+b+c;

}

1. Використання іменованих аргументів.

sum(1, c:4, b:8);

1. Методи класу Array.

BinarySearch, Clear, Copy, Exists(), Find(), FindAll(), IndexOf(), Resize(), Reverse(), Sort()…

1. Різниця в використанні const та readonly.

Значення полів з модифікатором const при компіляції підставляються в тих місцях коду, де до них відбувається звернення. Це означає, що в разі, коли таке поле оголошено в класі однієї збірки, а використовується класами інший збірки, для того щоб змінене значення було застосовано в залежною збірці, потрібно перекомпіляція залежною збірки. Задати значення можна тільки в момент оголошення. Загалом то const-поля це не зовсім поля.

Поля з модифікатором readonly є повноцінними полями, з тією різницею, що змінювати їх значення дозволено тільки в конструкторі. Відповідно static readonly - в статичному конструкторі (код ініціалізації полів при оголошенні переноситься в конструктор компілятором). Таким чином, значення readonly-полів можуть, наприклад обчислюватися на основі параметрів переданих в конструктор, що неможливо для const-полів.

У більшості випадків рекомендується використовувати саме readonly-поля замість const-полів коли ваш проект включає більше одного модуля.

1. Статичні змінні та \*статичний конструктор.

Статичні змінні існують в одному екземплярі і розділяються всіма об’єктами класу. Звертатись до змінних можна коли жодного об’єкту класу не існує. Звертатись до члену класу необхідно з використанням імені класу, а не об’єкту. Статичним членам класу не передається посилання на об’єкт класу this.

Статичні члени класу та конструктор викликається до створення екземплярів класу та до звернення до статичних членів класу.

Статичний конструктор може бути тільки один. Не має модифікатору доступу і не приймає параметрів. Виконується тільки один раз, незалежно від кількості створених об’єктів.

**Висновок:**

Протягом процесу виконання лабораторної роботи було створено 1 консольний застосунок що використовує передачу параметрів за посиланням та значенням. Було показано різницю в їх використанні. Було використано статичні методи та статичні змінні.