SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Algoritmi i strukture podataka

Vježba 1.

Methods and Criteria

Ivan Banovac, 220

Datum izvođenja vježbe:
03.11.2025

Zadatak 1. – Passing by reference

```
using System;
    class Program
    {
        public static void Swap(ref int a, ref int b)
        {
            int temp = a;
            a = b;
            b = temp;
        }
       static void Main(string[] args)
        {
            int a = 1;
            int b = 2;

            Console.WriteLine($"a={a}");
            Console.WriteLine($"b={b}");

            Swap(ref a, ref b);

            Console.WriteLine("a="+a);
            Console.WriteLine("b="+b);

            int[] array = { 2, 3, 4, 1, 8, 6, 5, 7 };

            foreach (int i in array)
            {
                Console.Write(i);
            }
            Console.WriteLine();

            Swap(ref array[2], ref array[3]);

            foreach (int i in array)
            {
                Console.Write(i);
            }
            Console.WriteLine();
            Console.ReadLine();
        }
    }
```

Ovaj zadatak koristi ref ključnu riječ za zamjenu vrijednosti varijabli i elemenata niza.

Definiramo metodu `Swap(ref a, ref b)` koja radi zamjenu 2 elementa prethodno prosljeđena preko argumenta (koristeći referencu). Prvo radimo zamjenu na 2 cijela broja a zatim radimo zamjenu 3 i 4 člana niza. Naposljetku ispisujemo originalni niz te niz u kojem smo napravili swap.

Rezultat:

```
a=1
b=2
a=2
b=1
23418657
23481657
```

Zadatak 2. – Sorting via IComparable

Zadatak implementira IComparable sučelje u klasi Student za sortiranje po ocjeni, koristeći prilagođeni Bubble sort.

Kako bi usporedio dva objekta, IComparable definira samo jednu metodu CompareTo koja definira kriterij usporedbe. CompareTo prima objekt i vraća cijeli broj koji može biti:

Negativan broj – ako je prvi objekt manji od drugog,

Nula – ako su objekti jednaki,

Pozitivan broj – ako je prvi objekt veći od drugog.

Klasa Student:

```
using System;

namespace Comparables
{
    class Student : IComparable
    {
        private string name;
        private double grade;

        public Student(string name, double grade)
        {
            this.name = name;
            this.grade = grade;
        }
        public override string ToString()
        {
            return "{name}: {grade}";
        }

        public int CompareTo(object obj)
        {
            Student other = (Student)obj;
            if (this.grade < other.grade) return -1;
            if (this.grade > other.grade) return 1;
            return 0;
        }
    }
}
```

Klasa Student koristi IComparable interface, pomoću konstruktora postavljamo početne vrijednosti.

CompareTo: Uspoređuje trenutnog studenta (this) s drugim objektom. Vraća negativan broj ako je manji, nulu ako su jednaki, ili pozitivan broj ako je veći. U kodu smo postavili da uspoređuje grade

Bubble.cs:

```
using System;

namespace Comparables
{
    class Bubble
    {
        public static void Sort(IComparable[] array)
        {
            for (int i = 0; i < array.Length; i++)
            {
                for (int j = i + 1; j < array.Length; j++)
                {
                    if (array[j].CompareTo(array[i]) > 0)
                    {
                        IComparable temp = array[j];
                        array[j] = array[i];
                        array[i] = temp;
                    }
                }
            }
        }
    }
}
```

Klasa Bubble: Sadrži algoritam Bubble sort-a. Svaki pojedini element se uspoređuje sa ostalim elementima u nizu unutar petlje. Umjesto operatora >, koristi array[j].CompareTo(array[i]) da odluči treba li zamijeniti elemente.

Program.cs:

```
using System;

namespace Comparables
{
    class Program
    {
        static void Main(string[] args)
        {
            Student[] students = {
                new Student("Ivo", 4.1),
                new Student("Ana", 4.9),
                new Student("Iva", 4.3),
                new Student("Bob", 4.5),
                new Student("Joe", 4.7)
            };

            foreach (var s in students) Console.WriteLine(s);
            Console.WriteLine("----------------");

            Bubble.Sort(students);

            foreach (var s in students) Console.WriteLine(s);
            Console.ReadLine();
        }
    }
}
```

Rezultat:

```
Ivo: 4,1
Ana: 4,9
Iva: 4,3
Bob: 4,5
Joe: 4,7
---------
Ana: 4,9
Joe: 4,7
Bob: 4,5
Iva: 4,3
Ivo: 4,1
```

Zadatak 3. – Sorting via IComparer

U ovom zadatku implementirati će se interface IComparer za kreiranje više kriterija za uspoređivanje objekata koji se prilikom izvođenja mogu prebacivati.

Student.cs:

```
namespace Comparers
{
    class Student
    {
        public string name;
        public double grade;

        public Student(string name, double grade)
        {
            this.name = name;
            this.grade = grade;
        }

        public override string ToString()
        {
            return $"{name}: {grade}";
        }
    }
}
```

U klasi Student deklarirani su field-ovi name i grade, te konstruktor Student i metoda ToString().

Bubble.cs:

```
using System.Collections;

namespace Comparers
{
    class Bubble
    {
        public static void Sort(object[] array, IComparer comparer)
        {
            for (int i = 0; i < array.Length; i++)
            {
                for (int j = i + 1; j < array.Length; j++)
                {
                    if (comparer.Compare(array[j], array[i]) < 0)
                    {
                        object temp = array[j];
                        array[j] = array[i];
                        array[i] = temp;
                    }
                }
            }
        }
    }
}
```

StudentComparer.cs:

```
using System.Collections;

namespace Comparers
{
    enum StudentComparerType { Name, Grade }

    class StudentComparer : IComparer
    {
        private StudentComparerType criterion;

        public StudentComparer(StudentComparerType criterion)
        {
            this.criterion = criterion;
        }

        public int Compare(object x, object y)
        {
            Student s1 = (Student)x;
            Student s2 = (Student)y;

            if (criterion == StudentComparerType.Name)
            {
                return string.Compare(s1.name, s2.name);
            }
            else
            {
                if (s1.grade < s2.grade) return -1;
                if (s1.grade > s2.grade) return 1;
                return 0;
            }
        }
    }
}
```

Klasa StudentComparer: Implementira Icomparer interface. U konstruktoru prima željeni kriterij. Metoda Compare(x, y) provjerava taj kriterij: ako je "Name", koristi string.Compare, a ako je "Grade", uspoređuje brojeve. Metoda Sort sada prima dva parametra: niz i objekt comparer. Unutar petlje, odluku o zamjeni donosi comparer.Compare().

Program.cs:

```
using System;

namespace Comparers
{
    class Program
    {
        static void Main(string[] args)
        {
            Student[] students = {
                new Student("Ivo", 4.1),
                new Student("Ana", 4.9),
                new Student("Iva", 4.3),
                new Student("Bob", 4.5),
                new Student("Joe", 4.7)
            };

            foreach (var s in students) Console.WriteLine(s);
            Console.WriteLine("----------------");

            StudentComparer comparerName = new
StudentComparer(StudentComparerType.Name);
            Bubble.Sort(students, comparerName);
            foreach (var s in students) Console.WriteLine(s);
            Console.WriteLine("----------------");

            StudentComparer comparerGrade = new
StudentComparer(StudentComparerType.Grade);
            Bubble.Sort(students, comparerGrade);
            foreach (var s in students) Console.WriteLine(s);

            Console.ReadLine();
        }
    }
}
```

Rezultat:

```
Ivo: 4,1
Ana: 4,9
Iva: 4,3
Bob: 4,5
Joe: 4,7
---------
Ana: 4,9
Bob: 4,5
Iva: 4,3
Ivo: 4,1
Joe: 4,7
---------
Ivo: 4,1
Iva: 4,3
Bob: 4,5
Joe: 4,7
Ana: 4,9
```

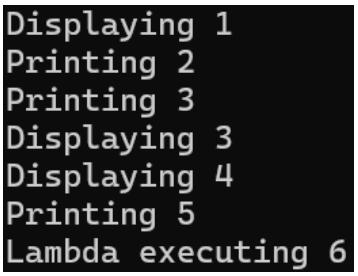
Zadatak 4. – Using Delegates

U ovom zadatku koristit će se delegati za pozivanje metoda. Delegat je objekt koji sadrži pointer na funkciju

```
using System;

namespace Delegates
{
    delegate void Invoker(int i);

    class Program
    {
        static void Display(int i)
        {
            Console.WriteLine("Displaying"+i);
        }

        static void Print(int i)
        {
            Console.WriteLine("Printing" +i);
        }

        static void Method(Invoker invoker, int value)
        {
            invoker(value);
        }

        static void Main(string[] args)
        {
            Invoker invoker = new Invoker(Display);
            invoker(1);

            invoker = Print;
            invoker(2);

            invoker += Display;
            invoker(3);

            Method(Display, 4);
            Method(Print, 5);

            invoker = (x) => Console.WriteLine("Lambda executing" +x);
            invoker(6);

            Console.ReadLine();
        }
    }
}
```

U Main programu kreira se delegat Invoker koji prima cijeli broj i tipa je void - svaka metoda koja vraća void i prima jedan int može se spremiti u ovaj delegat. Delegat pokazuje na metodu Displaying 1 a zatim se pozivom Printing 2 mijenja referenca. Na kraju postavlja se delegat pomoću anonimne metode i lambda izraza (argumenti 6 i 7). Umjesto definiranja zasebne imenovane metode, koristite lambda izraz.

Rezultat:



Zadatak 5. – Sorting via Delegates

U ovom zadatku koristit će se delegati za implementiranje kriterija za sortiranje

Student.cs:

```
using System;

namespace Comparisons
{
    class Student
    {
        private string name;
        private double grade;

        public Student(string name, double grade)
        {
            this.name = name;
            this.grade = grade;
        }

        public override string ToString()
        {
            return $"{name}: {grade}";
        }

        public static bool CompareName(object a, object b)
        {
            Student s1 = (Student)a;
            Student s2 = (Student)b;
            return string.Compare(s1.name, s2.name) > 0;
        }

        public static bool CompareGrade(object a, object b)
        {
            Student s1 = (Student)a;
            Student s2 = (Student)b;
            return s1.grade < s2.grade;
        }
    }
}
```

Unutar klase Student napisali smo metode CompareName i CompareGrade koje vraćaju bool (true/false) ovisno o tome treba li zamijeniti elemente.

Bubble.cs:

```
namespace Comparisons
{
    delegate bool Comparison(object a, object b);

    class Bubble
    {
        public static void Sort(object[] array, Comparison comparison)
        {
            for (int i = 0; i < array.Length; i++)
            {
                for (int j = i + 1; j < array.Length; j++)
                {
                    if (comparison(array[i], array[j]))
                    {
                        object temp = array[j];
                        array[j] = array[i];
                        array[i] = temp;
                    }
                }
            }
        }
    }
}
```

Delegat Comparison koji je definiran u klasi Bubble, predstavlja bilo koju metodu koja prima dva objekta i vraća bool. Preko delegata su prosljeđene metode CompareName i CompareGrade.

Program.cs:

```
using System;

namespace Comparisons
{
    class Program
    {
        static void Main(string[] args)
        {
            Student[] students = {
                new Student("Ivo", 4.1),
                new Student("Ana", 4.9),
                new Student("Iva", 4.3),
                new Student("Bob", 4.5),
                new Student("Joe", 4.7)
            };

            foreach (var s in students) Console.WriteLine(s);
            Console.WriteLine("----------------");

            Bubble.Sort(students, Student.CompareName);
            foreach (var s in students) Console.WriteLine(s);
            Console.WriteLine("----------------");

            Bubble.Sort(students, Student.CompareGrade);
            foreach (var s in students) Console.WriteLine(s);

            Console.ReadLine();
        }
    }
}
```

Rezulat:

```
Ivo: 4,1
Ana: 4,9
Iva: 4,3
Bob: 4,5
Joe: 4,7
---------
Ana: 4,9
Bob: 4,5
Iva: 4,3
Ivo: 4,1
Joe: 4,7
---------
Ana: 4,9
Joe: 4,7
Bob: 4,5
Iva: 4,3
Ivo: 4,1
```