Arrays

Introduction to Computer Programming

Maciej Piernik

Review o Lecture 3

- Built-in types and literals
- Enumerations
- Controlling the loops

Outline

- Arrays
  - Definition
  - Looping over elements
  - Copying

1-dimensional arrays

- `type[] name;`
- Initialization

```
int[] t = new int[5];

t[0] = 23;
t[4] = 50;

Console.WriteLine(t[3]);
```

Multidimensional arrays

- `type[,] name;`
- Initialization

```
int[,] t = new int[5, 10];

t[2, 8] = 23;

int[, , ,] t = new int[5, 10, 10, 23];

t[3, 8, 9, 17] = 23;
```

Nested arrays

- `type[][] name;`
- Initialization

```
int[][] t = new int[2][];
t[0] = new int[10];
t[1] = new int[23];


int[][,] t = new int[2][,];
t[0] = new int[5, 10];
t[1] = new int[3, 3];
```

Arrays

- Looping over elements in array using index

```
int[] t = new int[10];

for (int i = 0; i < t.Length; i++)
{
    t[i] = i * i;
    Console.WriteLine(t[i]);
}
```

`foreach` loop

- Looping over elements in array using `foreach`

```
int[,] t = new int[2,3];

foreach (int element in t)
{
    Console.WriteLine(element);
}
```

Read-only!

Value and reference types

Value type

```
int a = 5;
int b = a;

a--;
b++;

Console.WriteLine(a);




Console.WriteLine(b);
```

```
4
6
```

Reference type

```
int[] a = { 1, 1, 1 };
int[] b = a;

a[2] = 70;
b[1] = 23;

foreach (int e in a)
{
    Console.Write(e + " ");
}
Console.WriteLine();

foreach (int e in b)
{
    Console.Write(e + " ");
}
```

```
1 23 70
1 23 70
```

Copying arrays

```
int[] source = { 1, 2, 3, 4 };
int[] target = new int[4];

//Version 1
source.CopyTo(target, 0);

//Version 2
Array.Copy(source, target, source.Length);
```

Summary

- Creating arrays
- 1- and multidimensional arrays
- Looping through arrays
- Value vs reference types
- Copying arrays