C# i platforma .NET

Laboratorium nr 9
26.11.2024

Andrzej Janik

Informatyka w Sterowaniu i Zarządzaniu

1. Ćwiczenie 1 – Indeksy i zakresy

Kod:

```
static void Task1()
{
    string[] slowa = new string[]
    {
        "Już północ", // 0 ^10
        "cień",       // 1 ^9
        "ponury",     // 2 ^8
        "pół",        // 3 ^7
        "świata",     // 4 ^6
        "okrywa",     // 5 ^5
        "A jeszcze",  // 6 ^4
        "serce",      // 7 ^3
        "zmysłom",    // 8 ^2
        "spoczynku nie daje" // 9 ^1
                             // 10(słowa.Length) ^0
    };

    // 1
    Console.WriteLine($"{slowa[^1]}");

    // 2
    string[] sonet1 = slowa[2..6];
    foreach (var slowo in sonet1)
    {
        Console.Write($"< {slowo} >");
    }
    Console.WriteLine();
```

```
// 3
string[] sonet2 = slowa[^3..^0];
foreach (var slowo in sonet2)
{
    Console.Write($"{slowo} ");
}
Console.WriteLine();

// 4
// All elements of the array
string[] sonet3 = slowa[..];
Console.WriteLine("Sonet3:");
foreach (var slowo in sonet3)
{
    Console.Write($"{slowo} ");
}
Console.WriteLine();

// First 5 elements
string[] sonet4 = slowa[..5];
Console.WriteLine("Sonet4:");
foreach (var slowo in sonet4)
{
    Console.Write($"{slowo} ");
}
Console.WriteLine();
```

```
    // Elements from index 7 to the end
    string[] sonet5 = slowa[7..];
    Console.WriteLine("Sonet5:");
    foreach (var slowo in sonet5)
    {
        Console.Write($"{slowo} ");
    }
    Console.WriteLine();

    // 5
    Index stri = ^5;
    Console.WriteLine(slowa[stri]);

    // 6
    Range fraza = 2..7;
    string[] tekst = slowa[fraza];
    foreach (var slowo in tekst)
    {
        Console.Write($" {slowo} ");
    }
    Console.WriteLine();
}
```

W programie zdefiniowano następującą tablicę 10 stringów:

```
string[] slowa = new string[]
{
    "Już północ", // 0 ^10
    "cień",       // 1 ^9
    "ponury",     // 2 ^8
    "pół",        // 3 ^7
    "świata",     // 4 ^6
    "okrywa",     // 5 ^5
    "A jeszcze",  // 6 ^4
    "serce",      // 7 ^3
    "zmysłom",    // 8 ^2
    "spoczynku nie daje" // 9 ^1
                         // 10(słowa.Length) ^0
};
```

Wyniki wywołania następujących komend:

a)

```
1. Console.WriteLine($"{slowa[^1]}");
```

Wynik:

```
spoczynku nie daje
```

b)

```
2.      string[] sonet1 = slowa[2..6];
        foreach (var slowo in sonet1)
                Console.Write($"< { slowo} >");
        Console.WriteLine();
```

Wynik:

```
< ponury >< pół >< świata >< okrywa >
```

c)

```
3. string[] sonet2 = words[^3..^0];
   foreach (var word in sonet2)
        Console.Write($"{slowo}");
   Console.WriteLine();
```

Wynik:

```
serce zmysłom spoczynku nie daje
```

d)

```
4. Podobnie sprawdz zawartość tablic

   string[] sonet3 = slowa[..];
   string[] sonet4 = slowa[..5];
   string[] sonet5 = slowa[7..];
```

Wynik:

```
Sonet3:
Już północ cień ponury pół świata okrywa A jeszcze serce zmysłom spoczynku nie daje
Sonet4:
Już północ cień ponury pół świata
Sonet5:
serce zmysłom spoczynku nie daje
```

e)

```
5. Zdefinuj zmienną

   Index stri = ^5;
        i wypisz:
   Console.WriteLine(slowa[stri]);
```

Wynik:

```
okrywa
```

f)

6. Zdefinuj zmienną

```
Range fraza = 2..7;
```

i wypisz

```
string[] tekst = slowa[fraza];
foreach (var slowo  in tekst)
    Console.Write($" {slowo} ");
      Console.WriteLine();
```

Wynik:

```
 ponury  pół  świata  okrywa  A jeszcze
```

2. Ćwiczenie 2 – Event

Napisz program, który w pętli wczytuje znaki z klawiatury i informuje, że naciśnięto literę lub cyfrę. Jeśli naciśnięto cyfrę to wywołuje event **OnDigit** wypisujący komunikat: „Naciśnięto cyfrę!”, jeśli naciśnięto literę to wywołuje event **OnCharacter**, wypisujący „Naciśnięto literę!”. Naciśnięcie innego znaku niż litera lub cyfra kończy pętlę i cały program.

Kod:

```
public static event Action OnDigit;
public static event Action OnCharacter;

1 odwołanie | 0 zmian | 0 autorów, 0 zmian
static void Task2()
{
    OnDigit -= DigitPressed;
    OnCharacter -= CharacterPressed;

    OnDigit += DigitPressed;
    OnCharacter += CharacterPressed;

    Console.WriteLine("Press a key. Press any non-alphanumeric key to exit.");
    while (true)
    {
        var key = Console.ReadKey(intercept: true);
        if (char.IsDigit(key.KeyChar))
        {
            OnDigit?.Invoke();
        }
        else if (char.IsLetter(key.KeyChar))
        {
            OnCharacter?.Invoke();
        }
        else
        {
            Console.WriteLine("Non-alphanumeric character pressed.");
            break;
        }
```

```
    }
}

Odwołania: 2 | 0 zmian | 0 autorów, 0 zmian
static void DigitPressed()
{
    Console.WriteLine("Naciśnięto cyfrę!");
}

Odwołania: 2 | 0 zmian | 0 autorów, 0 zmian
static void CharacterPressed()
{
    Console.WriteLine("Naciśnięto literę!");
}
```

Działanie:

```
Press a key. Press any non-alphanumeric key to exit.
Naciśnięto cyfrę!
Naciśnięto cyfrę!
Naciśnięto literę!
Naciśnięto cyfrę!
Naciśnięto literę!
Naciśnięto literę!
Naciśnięto literę!
Naciśnięto literę!
Naciśnięto cyfrę!
Naciśnięto cyfrę!
Naciśnięto literę!
Naciśnięto cyfrę!
Naciśnięto literę!
Non-alphanumeric character pressed.
```

3. Ćwiczenie 3 – Oczyszczanie pamięci

Zaimplementuj dla dowolnie wybranej klasy interfejs ***IDispsible*** uwzględniając jednocześnie niedeterministyczny scenariusz usunięcia zalokowanej pamięci niezarządzanej, zgodnie z podanym poniżej wzorcem.

```
public class DisposeObject : IDisposable
  {
     protected bool disposed = false;    //zabezpieczenie, aby nie wywoływać wielokrotnie
     private IntPtr handle;              // niezarządzane zasoby
     private Component components;   // zarządzane zasoby

     ~DisposeObject()                    //niederministyczny
     {
        Dispose(false);
     }
     public void Dispose()               //derministyczny
     {
        Dispose(true);
        GC.SuppressFinalize(this);       //  zabezpieczenie przed powtórnym wywołaniem finalizatora
     }
    protected virtual void Dispose(bool disposing)
     {
        if (!disposed)                   //tylko jeden raz
        {
           if (disposing)
           {
              components.Dispose();      //zarządzane (kod użytkownika)
           }
           CloseHandle(handle);          //niezarządzane
        }
        disposed = true;
     }
  }
```

Uwaga:

Alokacja pamięci niezarządzanej:

`IntPtr ptr = Marshal.AllocHGlobal(copylen + 1);`

Usuniecie niezarządzanego fragmentu pamięci.

`Marshal.FreeHGlobal(ptr);`

Nie trzeba tworzyć żadnego obiektu typu Component. Wystarczy zaznaczyć miejsce, gdzie należy go wstawić.

Kod:

```
static void Task3()
{
    // Utworzenie obiektu DisposeObject
    DisposeObject disposableObject = new DisposeObject();

    // Zwolnienie zasobów
    disposableObject.Dispose();
    Console.WriteLine("Zwolniono zasoby obiektu DisposeObject.");
}
```

```
public class DisposeObject : IDisposable
{
    protected bool disposed = false; // zabezpieczenie, aby nie wywoływać wielokrotnie
    private IntPtr handle; // niezarządzane zasoby
    private Component components; // zarządzane zasoby

    1 odwołanie | 0 zmian | 0 autorów, 0 zmian
    public DisposeObject()
    {
        // Alokacja pamięci niezarządzanej
        handle = Marshal.AllocHGlobal(1024);
        Console.WriteLine("Alokowano pamięć niezarządzaną.");

        // Utworzenie zarządzanego obiektu
        components = new Component();
        Console.WriteLine("Utworzono zarządzany obiekt.");
    }

    Odwołania: 0 | 0 zmian | 0 autorów, 0 zmian
    ~DisposeObject() //niedeterministyczny
    {
        Dispose(false);
    }

    1 odwołanie | 0 zmian | 0 autorów, 0 zmian
    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this); // zabezpieczenie przed powtórnym wywołaniem finalizatora
    }

    protected virtual void Dispose(bool disposing)
    {
        if (disposed)
            return; // tylko jeden raz

        if (disposing)
        {
            // zwolnienie zarządzanych zasobów (kod użytkownika)
            components.Dispose();
            Console.WriteLine("Zwolniono zarządzane zasoby.");
        }

        // zwolnienie niezarządzanych zasobów
        if (handle != IntPtr.Zero)
        {
            Marshal.FreeHGlobal(handle);
            Console.WriteLine("Zwolniono pamięć niezarządzaną.");
        }

        disposed = true;
    }
}
```

Wynik:

```
Alokowano pamięć niezarządzaną.
Utworzono zarządzany obiekt.
Zwolniono zarządzane zasoby.
Zwolniono pamięć niezarządzaną.
Zwolniono zasoby obiektu DisposeObject.
```