



Visual Programming Control Statement: Looping

Indah Permatasari, M.Kom.

(4th meeting)

Outline

- Looping control
 - For
 - While
 - Do...while
 - Nested
 - Break
 - Continue
- Learning by coding
- Recall your programming basics!



Looping (Pengulangan)

Looping

- Suatu bagian yang bertugas melakukan kegiatan mengulang suatu proses sesuai dengan yang diinginkan
- Bagian penting:
 - Inisialisasi → kondisi awal
 - Proses → isi dari semua proses yang perlu dilakukan pengulangan
 - Iterasi → merupakan kondisi pertambahan agar pengulangan dapat berjalan
 - Terminasi/Kondisi Perulangan → kondisi berhenti

For

- Struktur control repetitive yang memungkinkan untuk menjalankan proses dengan jumlah pengulangan tertentu
- Biasanya digunakan untuk perulangan yang **sudah jelas perlu dilakukan berapa kali**, dengan kata lain jumlah perulangan yang diperlukan sudah diketahui oleh pembuat program
- Sintaks:

```
For (inisialisasi; kondisi; penambahan)
{
    Statemen
}
```

```

using System;

namespace Looping
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int a = 10; a < 15; a++)
            {
                Console.WriteLine("Nilai a adalah {0}", a);
            }
            Console.ReadLine();
        }
    }
}

/**
 * Output:
 * Nilai a adalah 10
 * Nilai a adalah 11
 * Nilai a adalah 12
 * Nilai a adalah 13
 * Nilai a adalah 14
 */

```

While

- Struktur control yang memungkinkan mengulangi suatu proses dengan jumlah pengulangan tertentu
- Sintaks

```
While (kondisi)
{
    statemen
}
```

```

using System;

namespace Looping
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 10;
            while(a<15)
            {
                Console.WriteLine("Nilai a adalah {0}", a);
                a++;
            }
            Console.ReadLine();
        }
    }
}

/**
 * Output:
 * Nilai a adalah 10
 * Nilai a adalah 11
 * Nilai a adalah 12
 * Nilai a adalah 13
 * Nilai a adalah 14
 */

```


do...while

- Proses eksekusi sama dengan while, namun pasti akan dieksekusi minimal satu kali
- Sintaks:

```
do
{
    Statemen;
}
dhile (kondisi)
```

```

using System;
namespace Looping
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 10;
            do
            {
                Console.WriteLine("Nilai a adalah {0}", a);
                a = a + 1;
            }
            while (a < 15);
            Console.ReadLine();
        }
    }
}

/**
 * Output:
 * Nilai a adalah 10
 * Nilai a adalah 11
 * Nilai a adalah 12
 * Nilai a adalah 13
 * Nilai a adalah 14
 */

```

Looping Nested

- Pengulangan yang bersarang, dimaksudkan untuk menggunakan suatu pengulangan di dalam pengulangan

- Sintaks:

- For

- ```
for(inisialisasi; kondisi; pertambahan)
{for (inisialisasi; kondisi; pertambahan){}}
```

- While

- ```
while(kondisi)
{while(kondisi){}}
```

- Do...while

- ```
do{statemen;
do{statemen;}while(kondisi);}
while(kondisi);
```

# For (nested)

- Sama seperti penjelasan nested pada *selection*, *nested* digunakan untuk membuat pengulangan di dalam pengulangan

```
using System;
namespace Looping
{
 class Program
 {
 static void Main(string[] args)
 {
 for(int a = 0; a < 5; a++)
 {
 Console.WriteLine("Nilai a adalah {0}",
a);
 for (int b = 4; b <= a; b++)
 {
 Console.WriteLine("Nilai b adalah
{0}", b);
 }
 }
 Console.ReadLine();
 }
 }
}
/**
 * Output:
 * Nilai a adalah 0
 * Nilai a adalah 1
 * Nilai a adalah 2
 * Nilai a adalah 3
 * Nilai a adalah 4
 * Nilai b adalah 4
 */
```

# While (nested)

```
/**
 * Output:
 * Nilai a adalah 10
 * Nilai a adalah 11
 * Nilai a adalah 12
 * Nilai b adalah 12
 * Nilai a adalah 13
 * Nilai b adalah 13
 * Nilai a adalah 14
 * Nilai b adalah 14
 */
```

```
using System;

namespace Looping
{
 class Program
 {
 static void Main(string[] args)
 {
 int a = 10;
 int b = 12;
 while (a < 15)
 {
 Console.WriteLine("Nilai a adalah
{0}", a);
 while (b == a)
 {
 Console.WriteLine("Nilai b
adalah {0}", b);
 b++;
 }
 a++;
 }
 Console.ReadLine();
 }
 }
}
```

do...while (Nested)

Create your code(?)

# Break

- Sama seperti fungsi *break* pada switch, *break* digunakan untuk menghentikan eksekusi
- Sintaks:  
`break;`

```
using System;
namespace Looping
{
 class Program
 {
 static void Main(string[] args)
 {
 int a = 10;
 while(a < 20)
 {
 Console.WriteLine("Nilai a adalah
{0}", a);
 a++;
 if (a > 15)
 {
 break;
 }
 }
 Console.ReadLine();
 }
 }
}
/**
 * Output:
 * Nilai a adalah 10
 * Nilai a adalah 11
 * Nilai a adalah 12
 * Nilai a adalah 13
 * Nilai a adalah 14
 * Nilai a adalah 15
 */
```

# Continue

- Digunakan untuk melompat pada iterasi selanjutnya pada pengulangan
  - for, kata kunci menyebabkan aliran proses melompat langsung pada bagian pertambahan
  - while, aliran proses melompat langsung pada bagian kondisi
- Syntax:
  - `continue;`



```

using System;
namespace Looping
{
 class Program
 {
 static void Main(string[] args)
 {
 int a = 10;

 do
 {
 if(a == 12)
 {
 a += 1;
 Console.WriteLine("Langsung lanjut saja!");
 continue;
 }
 Console.WriteLine("Nilai a adalah {0}", a);
 a++;
 }
 while (a < 15) ;

 Console.ReadLine();
 }
 }
}

```

```

/**
 * Output:
 * Nilai a adalah 10
 * Nilai a adalah 11
 * Langsung lanjut saja!
 * Nilai a adalah 13
 * Nilai a adalah 14
 */

```



# Learning by coding

Explain the following code and output

```
using System;
namespace Looping
{
 class Program
 {
 static void Main(string[] args)
 {
 for (int a=0; a<=20; a++)
 {
 if(a % 2 == 0)
 {
 Console.WriteLine("{0}", a);
 }
 }

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

Explain the following code and output

```
using System;
namespace Looping
{
 class Program
 {
 static void Main(string[] args)
 {
 Console.Write("Masukkan tinggi: ");
 int tinggi = Convert.ToInt32(Console.ReadLine());
 for(int i=1; i<=tinggi; i++)
 {
 for(int j=1; j<=i; j++)
 {
 Console.Write(" *");
 }
 Console.WriteLine();
 }
 Console.ReadLine();
 }
 }
}
```



# Recall your programming basics!

# General Structure of a C# Program

- *using* → digunakan untuk menyertakan namespaces ke dalam program
- Class → data structure that may contain data members (constants and fields), function members (methods, properties, events, indexers, operators, instance constructors, destructors and static constructors), and nested types.
- Struct → Structs are similar to classes in that they represent data structures that can contain data members and function members. However, unlike classes, structs are value types and do not require heap allocation
- Interface → digunakan untuk sekelompok fungsi terkait yang harus diterapkan oleh non-abstract class atau struct
- Enum → mendeklarasikan sekumpulan konstanta
- Namespace → kumpulan class yang saling berhubungan dan memiliki kegunaan untuk menjalankan program yang dibuat

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/basic-concepts>

```
using System;

namespace YourNamespace
{
 class YourClass
 {
 }

 struct YourStruct
 {
 }

 interface IYourInterface
 {
 }

 delegate int YourDelegate();

 enum YourEnum
 {
 }

 namespace YourNestedNamespace
 {
 struct YourStruct
 {
 }
 }

 class Program
 {
 static void Main(string[] args)
 {
 //Your program starts here...
 }
 }
}
```

# Console

```
C#

using System;

class Hello
{
 static void Main() {
 Console.WriteLine("Hello, World");
 }
}
```

- C# source files typically have the file extension .cs. Assuming that the "Hello, World" program is stored in the file hello.cs, the program can be compiled with the Microsoft C# compiler using the command line
- which produces an executable assembly named hello.exe. The output produced by this application when it is run
- The output of the program is produced by the **WriteLine** method of the **Console** class in the **System namespace**. This class is provided by the .NET Framework class libraries, which, by default, are automatically referenced by the Microsoft C# compiler.

```
Console

csc hello.cs

Console

Hello, World
```

# Operator

## Aritmatika

| Operator | Deskripsi   | Contoh   |
|----------|-------------|----------|
| +        | Penjumlahan | $A + B$  |
| -        | Pengurangan | $A - B$  |
| *        | Perkalian   | $A * B$  |
| /        | Pembagian   | $A / B$  |
| %        | Modulus     | $A \% B$ |
| ++       | Peningkatan | $A++$    |
| --       | Penurunan   | $B--$    |

## Rasional

| Operator | Deskripsi                                 | Contoh   |
|----------|-------------------------------------------|----------|
| ==       | Kedua operan sama                         | $A == B$ |
| !=       | Kedua operan tidak sama                   | $A != B$ |
| >        | Nilai operan lebih besar                  | $A > B$  |
| <        | Nilai operan lebih kecil                  | $A < B$  |
| >=       | Nilai operan lebih besa atau sama dengan  | $A >= B$ |
| <=       | Nilai operan lebih kecil atau sama dengan | $A <= B$ |



# Operator

## Logika

| Operator | Deskripsi       | Contoh |
|----------|-----------------|--------|
| &&       | AND / Konjungsi | A && B |
|          | OR / Disjungsi  | A    B |
| !        | NOT / Negasi    | !A     |

## Assignment

| Operator | Deskripsi                                               | Contoh              |
|----------|---------------------------------------------------------|---------------------|
| =        | Memasukkan nilai operan kanan ke kiri                   | A = B+C             |
| +=       | Menjumlahkan kedua operan dan dimasukkan ke operan kiri | A += B →<br>A = A+B |
| -=       | Mengurangkan kedua operan dan dimasukkan ke operan kiri | A -= B →<br>A = A-B |
| *=       | Mengalikan kedua operan dan dimasukkan ke operan kiri   | A *= B →<br>A = A*B |
| /=       | Membagi kedua operan dan dimasukkan ke operan kiri      | A /= B →<br>A = A/B |
| %=       | Me-modulus kedua operan dan dimasukkan ke operan kiri   | A %= B →<br>A = A%B |

# Konversi Tipe Data

```
int i = int.MaxValue;
// System.Int32.MaxValue constant

string s = i.ToString();
// System.Int32.ToString() instance method

string t = 123.ToString();
// System.Int32.ToString() instance method
```

| Reserved word | Aliased type   |
|---------------|----------------|
| sbyte         | System.SByte   |
| byte          | System.Byte    |
| short         | System.Int16   |
| ushort        | System.UInt16  |
| int           | System.Int32   |
| uint          | System.UInt32  |
| long          | System.Int64   |
| ulong         | System.UInt64  |
| char          | System.Char    |
| float         | System.Single  |
| double        | System.Double  |
| bool          | System.Boolean |
| decimal       | System.Decimal |

# Terima kasih.