

Modul Tutorial C# 2

DASAR PEMROGRAMAN C# : STATEMENT DAN OPERATOR

ENRICO BUDIANTO

MICROSOFT INNOVATION CENTER – UNIVERSITAS INDONESIA

JUNI 2010

## PENDAHULUAN

Setelah sebelumnya kita sudah berhasil membuat sebuah class library yang berbasis C#, sekarang saatnya bagi kita untuk masuk lebih dalam dan melihat dasar pemrograman berbasis C#. Kita akan melihat berbagai fitur yang disediakan oleh C#, dan akan sedikit diberikan perbandingan yang serupa di dalam bahasa JAVA, untuk mempermudah pemahaman.

Pada bagian ini akan diberikan penjelasan dalam bahasa C# mengenai :

- Statement dan Operator
- Variabel dan Data Types
- Arrays
- Class dan Structs
- Setter dan Getter
- Enumeration
- Delegates

Untuk mempermudah mencoba, kita akan mencoba membuat sebuah program yang command-line based. Caranya adalah pada tampilan muka Visual Studio anda, pilih menu File -> New -> Project. Setelah itu, pada installed templates, pilih Other Languages -> Visual C#, kemudian pilih Console Application. Tentukan nama project dan lokasi yang diinginkan.

## STATEMENT DAN OPERATOR

Statement adalah sebuah single line of code yang dibatasi oleh sebuah tanda "titik koma" atau semicolon. Secara umum, terdapat banyak kesamaan antara statement dalam JAVA dan C#, ini lah yang membuat pengguna C dan JAVA seharusnya tidak terlalu mengalami kesulitan untuk mempelajari C#. Terdapat beberapa kategori dalam statement, antara lain :

### a. Declaration Statements

Declaration statements adalah statement yang mendeklarasikan cara mengisi suatu nilai dari variable ataupun konstanta. Untuk mengisi suatu variable, cukup diassign dengan nilai variable lain, diassign sebuah nilai langsung, ataupun tidak perlu diassign dengan apapun. Sementara itu, untuk mengisi suatu nilai konstanta harus diikuti dengan keyword **const** sebelum tipe data, dan harus langsung dilakukan assignment.

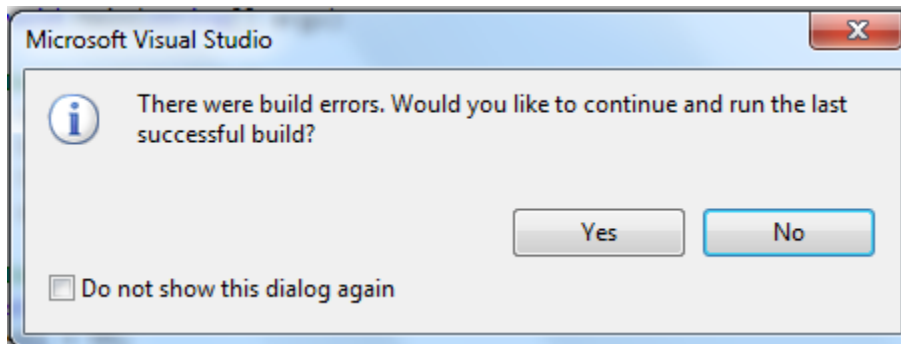
Berikut adalah contoh declaration statement untuk variable dan konstanta :

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace Belajar1
7 {
8     class Program
9     {
10         static void Main(string[] args)
11         {
12             // Deklarasi Variabel
13             int a;
14             int b = 10;
15             int temp = 30;
16             int assigned = temp;
17
18             // Deklarasi Konstanta
19             const int const1 = 4;
20
21             Console.WriteLine("Halo");
22             Console.ReadLine();
23         }
24     }
25 }
```

Bagaimana bila kita mencoba melakukan hal ini ?

```
// Deklarasi Konstanta
const int const1 = 4;
const1 = 55;
```

Ternyata akan muncul sebuah error. Kita harus mengingat bahwa setelah sebuah konstanta diinisialisasi, maka nilai tersebut tidak akan bisa diubah. Pada bahasa pemrograman JAVA, **const** mempunyai kemiripan dengan **final static**.



#### b. Selection Statements

Selection statement terdiri dari beberapa keyword yang sudah kita kenal, yaitu if, else, switch, case. Tidak berbeda jauh dengan kebanyakan bahasa lain, cara penulisannya dari setiap selection statements tersebut adalah sebagai berikut :

- If-Else Statement

Berikut adalah sample code yang menunjukkan cara membuat if-else statement dari kedua kondisi tersebut :

```
if (temp > 30)
{
    Console.WriteLine("Temp lebih besar dari 30");
}
else {
    Console.WriteLine("Temp lebih kecil atau sama dengan 30");
}
```

- Switch-Case Statement

Berikut adalah sample code yang menunjukkan cara membuat if-else statement dari kedua kondisi tersebut :

```
char kataku = 'a';
switch(kataku){
    case 'a':
        Console.WriteLine("kataku adalah a");
        break;
    case 'b':
        Console.WriteLine("kataku adalah b");
        break;
    case 'c':
        Console.WriteLine("kataku adalah c");
        break;
}
```

Kita juga dapat membuat sebuah segment code yang dapat handle lebih dari satu case. Contoh penerapannya adalah sebagai berikut :

```
char kataku = 'a';
switch(kataku){
    case 'a':
    case 'b':
        Console.WriteLine("kataku adalah a atau b");
        break;
    case 'c':
        Console.WriteLine("kataku adalah c");
        break;
}
```

### c. Iteration Statements

Iteration statement terdiri dari beberapa kondisi yang ditandai dengan keyword-keyword, yaitu do, for, foreach, in, while. Seperti halnya iterasi pada bahasa JAVA, iterasi di C# memungkinkan kita untuk melakukan looping terhadap suatu collection, ataupun melakukan suatu hal yang dapat dilakukan secara berulang. Di bawah ini adalah salah satu contoh cara penulisannya :

- Do-while Statement

```
temp = 3;
int awal = 0;

do
{
    Console.WriteLine("Looping ke-" + awal);
    awal++;
}
while (awal <= temp);
```

- For Statement

```
temp = 3;

for (int awal = 0; awal < temp; awal++) {
    Console.WriteLine("Langkah ke-"+awal);
}
```

- Foreach-in Statement

C# menyediakan fasilitas foreach yang memungkinkan kita untuk melakukan penelusuran atas item dalam sekumpulan / koleksi data, oleh karena itu statement ini banyak digunakan dalam melakukan penelusuran pada collection.

```

string[] stringArray = { "one", "two", "three" };
foreach (string element in stringArray)
{
    // This code loops three times, with the element variable set to
    // "one", then "two", and then "three".
    Console.WriteLine(element + " ");
}

```

Dengan menggunakan foreach ini, kita dapat melakukan penelusuran pada array, tidak hanya untuk tipe variable primitive seperti int, ataupun bool. Dengan tipe variable reference pun hal ini dapat dilakukan.

- While Statement

```

temp = 2;
int awal = 0;
while (awal < temp) {
    Console.WriteLine("Nilai adalah "+awal);
    awal++;
}

```

d. Exception Handling Statements

Exception handling statement memungkinkan kita untuk memulihkan diri dari exception yang terjadi pada saat runtime secara "anggun". Terdapat beberapa keyword yang berhubungan dengan exception handling ini, yaitu throw, try-catch, try-finally, dan try-catch-finally. Setiap fungsi dari exception handling ini mempunyai kesamaan dengan exception handling pada JAVA. Kondisi **try** dijalankan untuk mengecek suatu kondisi apakah terjadi exception atau tidak, kemudian **catch** digunakan untuk melakukan recovery dari keadaan exception pada saat runtime tersebut, dan **finally** akan dijalankan baik pada saat terjadi exception, ataupun tidak terjadi exception.

Berikutnya, cobalah untuk menjalankan potongan code di bawah ini :

```

object o2 = null;
int i2;
try
{
    i2 = (int)o2;    // Error
}
catch (Exception err)
{
    Console.WriteLine(err);
    i2 = 10;
}
finally {
    Console.WriteLine("Happy Ending :)");
}

```