# VARIABEL, TIPE DATA DAN EKSPRESI
# Bab 2

## 2.1 IDENTIFIER

Indentifier adalah nama yang diberikan untuk nama objek, nama fungsi, nama variable, dll ( sifatnya 'case sensitive'). Identifier pada C++ terdiri dari :

1. huruf 'A' sampai 'Z'
2. huruf 'a' sampai 'z'
3. underscore ( _ )
4. bilangan antara '0' sampai '9'

Ketentuan dalam memberi nama identifier dalah C++ adalah :

1. karakter pertama harus huruf atau underscore
2. untuk compiler Borland, panjang maksimum 32 karakter
3. identifier harus tidak sama dengan keyword yang ada di C++

contoh identifier :

- Yang benar : nilai, Nilai_nama, No8
- Yang salah : 1Buah, nomor-data, if

## 2.2 TIPE DATA DI C++

Tipe data diklasifikasikan berdasarkan bagaimana keadaan data disimpan dalam memori, dan jenis operasi yang dapat dilakukan.

### 2.2.1 CHAR

Adalah sembarang huruf, angka, tanda baca tunggal. Ada 2 (dua) macam char, yaitu :

1. signed
   mendeklarasikan char bertanda, digunakan untuk nilai negative. Rentang nilai mulai -128 sampai 127
2. unsigned
   mendeklarasikan char tidak bertanda, untuk nilai positif. Rentang nilai mulai 0 sampai 255

contoh deklarasi char :

**char** letter = ‘A’ ;
**unsigned cha**r number = 245 ;
**signed char** value = -71 ;

## 2.2.2 SHORT, INT, LONG

Digunakan untuk menyatakan bilangan bulat. Seperti pada char, perubah tipe signed dan unsigned dapat ditambahkan.
Rentang nilai short int mulai -32.768 sampai 32.767
Rentang nilai long / int mulai -2.147.483.648 sampai 2.147.483.647

Contoh deklarasi int :

**Int** nilai, total ; *atau*
**Int** nilai = 90 ;

## 2.2.3 FLOAT, DOUBLE

Menyatakan bilangan pecahan/real, maupun eksponensial. Dalam keadaan default, bilang floting point dianggap bertipe double.

Rentang nilai float mulai 3,4 E $^{-38}$ sampai 3,4 E $^{+38}$
Rentang nilai double mulai 1,7 E $^{-308}$ sampai 1,7 E $^{+308}$

## 2.2.4 ENUMERATION / ENUM

Adalah serangkaian symbol berurutan yang menspesifikasikan konstanta bertipe integer. Dalam C++ tidak terdapat tipe Bolean, sehingga untuk merepresentasikan TRUE dengan nilai integer bukan nol ( 1, 2, dst ), sedangkan FALSE dengan nilai nol ( 0 )

Contoh deklarasi enum :

**Enum** BOOLEAN { False, True } ; *atau*
**Enum** BOOLEAN { Benar = 3, Salah = 0 } ;

## 2.2.5 VOID

Menyatakan tipe kosong untuk :

1. mendeklarasikan fungsi yang tidak mengembalikan nilai apapun.
2. mendeklarasikan fungsi yang tidak menerima parameter apapun.
3. bila diawali dengan operator *, menyatakan penunjuk terhadap sembarang tipe data.

Contoh deklarasi void :

**Void** cctrputs (char*, int ) ;  *atau*
Main (**void**) ;  *atau*
**Void*** action ;
Int ivalue = 100 ;
Action = &ivalue ;

## 2.2.6 PENUNJUK / POINTER

Adalah variable yang berisi nilai alamat suatu lokasi memori tertentu. Deklarasi penunjuk dilakukan dengan menspesifikasikan *, sebelum nama varabel / konstanta.

## 2.2.7 PENUNJUK / POINTER

Adalah sekelompok data bertipe sama yang menduduki lokasi memori yang berurutan. Jumlah elemen array dinyatakan dengan cara mengapit jumlah yang di maksud dengan tanda ‘ [ … ] ‘

**Bentuk umum :** tipe namaArray [ jumlahelemen ] ;

Untuk menyatakan array berdimensi lebih dari 1 (satu), tambahkan tanda ‘[ … ]’ sebanyak dimensi yang diinginkan.

Contoh deklarasi array 2 dimensi :

Int matrix [2][3] ;

## 2.2.8 STRING

Deretan karakter yang diakhiri dengan sebuah karakter kosong. String ditulis dengan mengapit string dengan tanda petik dua ( “ …….” )

Contoh deklarasi string :

Char text [ ] = “ C++ “ ;
Puts (text) ;

## 2.2.9 STRUCT, UNION

Digunakan untuk mendeklarasikan sekelompok data yang memiliki tipe yang berlainan. **Struct** : elemennya ada dilokasi memori yang berbeda, dan **union** : elemennya ada dilokasi memori yang sama.

**Bentuk umum :**

```
Struct tipestruktur
{
        Tipeanggota1 namaAnggota1 ;
        Tipeanggota2 namaAnggota2 ;
        ……..
}
namaStruktur ;
```

## 2.3 DATA OBJEK

Data objek adalah bagian dari memori yang digunakan untuk menampung nilai dari variable. **Variable** umumnya digunakan untuk data objek yang nilainya dapat diubah selama pemrosesan berlangsung.

Contoh deklarasi variable :

Int **nilai** ;  atau  int **nilai** = 80 ;

Dalam C++ pendeklarasian termasuk statemen, sehingga pendeklarasian dapat diletakkan pada sembarang tempat dalam program.

**Konstanta** data objek adalah : variable yang nilainya tidak dapat diubah selama pemrosesan berlangsung.

Contoh deklarasi konstanta :

**Const** double pi = 3.14 ;

## 2.4 SCOPE IDENTIFIER

Ruang lingkup / scope adalah bagian mana dari program, identifier tersebut dapat diakses. Scope dari suatu identifier dimulai dari pendeklarasian sampai dengan akhir dari suatu blok. Scope identifier ada 2, yaitu :

1. local identifier
   dideklarasikan di dalam blok ‘ { … } ‘
2. global identifier
   dideklarasikan di luar dari blok

Scope relolution operator ( :: ) dapat digunakan untuk mengakses variable global secara langsung.

Contoh variable global :

```
Int x ;
F ( )
{
        Int x ;
        :: x = 4 ;      // pemberian nilai untuk variable global x
}
```

## 2.5 OPERATOR DAN EKSPRESI

Ekspresi adalah rangkaian dari operator, operand, dan punctuator ( ; )
contoh :

```
3 + 4 * 1 ;
```

### 2.5.1 OPERATOR ARITMATIKA

Terdiri dari :

- penjumlahan ( + )
- pengurangan ( - )
- sisa bagi / hanya  untuk tipe data integer ( % )
- perkalian ( * )
- pembagian ( / )

Jika operator bagi ( / ) diterapkan pada tipe integer, akan menghasilkan bilangan integer dengan decimal yang dihilangkan.

### 2.5.2 ASSIGNMENT OPERATOR ( = )

Berfungsi untuk memberi nilai pada variable.

Table kombinasi assignment :

| NO. | PENYINGKATAN | ARTI |
|---|---|---|
| 1 | X += Y | X = X + Y |
| 2 | X -= Y | X = X – Y |
| 3 | X *= Y | X = X * Y |
| 4 | X /= Y | X = X / Y |
| 5 | X %= Y | X = X % Y |

## 2.5.3 INCREMENT DAN DECREMENT OPERATOR

**Increment** adalah penambahan suatu variable dengan nilai 1, dan **decrement** adalah pengurangan suatu variabel dengan nilai 1.

Tabelnya :

| INCREMENT | DECREMENT |
|---|---|
| ++X ; | - -X ; |
| X += 1 ; | X -= 1 ; |
| X = X + 1 | X = X – 1 ; |

Operator increment dan decrement dapat diletakkan pada awal atau akhir variable, seperti dibawah ini :

**++X** , nilai variable X dinaikkan dahulu sebelum diproses
**X++** , nilai variable X diproses dahulu sebelum dinaikkan

Contoh program increment :

```
# include <iostream.h>
Main ( )
{
Int X = 5 ;
Cout << " Nilai X = " << X << ' \n ' ;
Cout << " Nilai X++ = " << X++ << ' \n ' ;
Cout << " X = " << X << ' \n ' :
X = 5 ;
Cout << " Nilai X = " << X << ' \n ' ;
Cout << " Nilai ++X = " << ++X << ' \n ' ;
Cout << " X = " << X << ' \n ' ;
Return 0 ;
}
```

Outputnya :

```
Nilai X = 5
Nilai X++ = 5
X = 6
Nilai X = 5
Nilai ++X = 6
X = 6
```

## 2.5.4 EQUALITY, RELATIONAL, LOGIKA OPERATOR

Equality digunakan untuk menentukan apakah 2 buah variable memiliki nilai yang sama atau tidak.

**= =** , sama dengan, contoh : 5 **= =** 5
**! =** , tidak sama dengan , contoh : 5 **! =** 4

Relational operator digunakan untuk menentukan apakah suatu variable memiliki nilai lebih besar atau sama dengan lebih besar, lebih kecil atau lebih kecil sama dengan.

Operatornya : **<** , **< =** , **>** , **> =**

Logika operator adalah : (penulisan dibawah ini berdasarkan prioritas operator yang akan diproses terlebih dahulu )

1. **!** ( not )
2. **&&** ( and )
3. **||** ( or )

## 2.6 EKSPRESI CONDITIONAL

**Bentuk umumnya :**

Ekspresi C **?** ekspresi T **:** ekspresi S ;

Keterangan :
- ekspresi C = kondisi yang akan diproses lebih dahulu
- ekspresi T = jika kondisi ekspresi C nilainya TRUE, akan dijalankan
- ekspresi S = jika kondisi ekspresi C nilainya FALSE, akan dijalankan

contoh program :

```
# include <iostream.h>
Main ( )
{
Double nilai ;
Cout << " Masukkan suatu nilai = ' ;
Cin >> nilai ;
Nilai = (nilai < 0) ? –nilai : nilai ;
Cout << "nilai absolutnya =" << nilai ;
Return 0 ;
}
```

## 2.7 FORMAT OUTPUT PADA BILANGAN REAL

Beberapa format yang dapat dilakukan :

1. derajat ketelitian, dengan fungsi : **precision**.
2. lebar output dapat diubah dengan fungsi : **width**.
3. format bilangan real diubah dengan fungsi : **setf** diikuti argument : **ios :: scientific** atau **ios :: fixed**
4. aligment (rata kiri/kanan) dengan fungsi : **setf** atau **unsetf** diikuti argument : **ios :: left** atau **ios :: right**
5. karakter pengisi dengan fungsi : **fill** diikuti argument karakter
6. tampilan tanda '+' diubah dengan fungsi : **setf** atau **unsetf** diikuti argument : **ios :: showpos**
7. tampilan tanda '.' (titik) bila ada angka dibelakang koma diubah dengan fungsi : **setf** atau **unsetf** diikuti argument **ios :: showpoint**

contoh program :

```
# include <iostream.h>
# pragma hdrstop
Void main ( )
{
Double y = 1234.56789 ;
Cout << "menuliskan bilangan real dengan presisi berbeda' ;
Cout << " \n Presisi 3 = " ;
Cout.precision (3) ;
Cout.width (15) ;
Cout << y ;
Cout << " \n Presisi 7 = " ;
Cout.precision (7) ;
Cout.width (15) ;
Cout << y ;
Cout << " \n \n Menggunakan notasi scientific / fixed " ;
Cout.setf (ios :: scientific | ios :: showpos) ;
Cout << " \n Scientific = " << y ;
Cout.setf (ios :: fixed) ;
Cout << " \n Fixed = ' << y ;
Cout.unsetf (ios :: scientific | ios :: fixed | ios :: showpos) ;
Cout << " \n \n Menggunakan shompoint " ;
Double z = 123 ;
Cout.setf (ios :: showpoint) ;
Cout << " \n Showpoint aktif = " << z ;
Cout.unsetf (ios :: showpoint) ;
Cout << "\n Showpoint non aktif : " << z ;
}
```