VARIABEL,TIPE DATA DAN KONSTANTA

Overview

Tipe data, variabel, dan konstanta merupakan suatu kesatuan konsep yang paling mendasar didalam pemprograman komputer, karena tipe-tipe data sangat berpengaruh dalam penggunaan suatu variabel dan kostanta dapat membentuk berbagai macam ekspresi yang akan digunakan dalam program. Apabila seorang programmer salah dalam memilih tipe data untuk sebuah variabel atau konstanta maka hasil yang akan dicapai atau output tidak sesuai dengan harapan. Oleh karena itu pemilihan tipe data sangat penting dalam suatu program,

Tujuan

- ☑ Mengenal dan membedakan tipe-tipe data dasar
- ☑ Memahami penggunaan tipe-tipe data dasar dalam program
- ☑ Memahami penggunaan Variabel dan Konstanta dalam program dan penggunaannya dalam program

Untuk dapat menulis program yang dapat membantu menjalankan tugas-tugas kita, kita harus mengenal konsep dari **variabel**. Sebagai ilustrasi, ingat 2 buah angka, angka pertama adalah 5 dan angka kedua adalah 2. Selanjutnya tambahkan 1 pada angka pertama kemudian hasilnya dikurangi angka kedua (dimana hasil akhirnya adalah 4).

Seluruh proses ini dapat diekspresikan dalam C++ dengan serangkaian instruksi sbb :

```
a = 5;
b = 2;
a = a + 1;
result = a - b;
```

Jelas ini merupakan satu contoh yang sangat sederhana karena kita hanya menggunakan 2 nilai integer yang kecil, tetapi komputer dapat menyimpan jutaan angka dalam waktu yang bersamaan dan dapat melakukan operasi matematika yang rumit.

Karena itu, kita dapat mendefinisikan variable sebagai bagian dari memory untuk menyimpan nilai yang telah ditentukan. Setiap variable memerlukan **identifier** yang dapat membedakannya dari variable yang lain, sebagai contoh dari kode diatas identifier variabelnya adalah **a**, **b** dan **result**, tetapi kita dapat membuat nama untuk variabel selama masih merupakan identifier yang benar.

4.1 Identifiers

Identifier adalah untaian satu atau lebih huruf, angka, atau garis bawah (_). Panjang dari identifier, tidak terbatas, walaupun untuk beberapa kompiler hanya 32 karakter pertama saja yang dibaca sebagai identifier (sisanya diabaikan). Identifier harus selalu diawali dengan huruf atau garis bawah (_).

Ketentuan lainnya yang harus diperhatikan dalam menentukan identifier adalah tidak boleh menggunakan **key word** dari bahasa C++. Dibawah ini adalah **key** word dalam C++ :

Asm	auto	bool	break	case
Catch	char	class	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern
False	float	for	friend	goto
If	inline	int	long	mutable
namespace	new	operator	private	protected
Public	register	reinterpr et_cast	return	short
Signed	sizeof	static	static_cast	struct
Switch	template	this	throw	true
Try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile
wchar_t				

Sebagai tambahan, represetasi alternatif dari operator, tidak dapat digunakan sebagai identifier. Contoh :

and, and_eq, bitand, bitor, compl, not, not_eq, or, or_eq, xor, xor_eq

catatan: Bahasa C++ adalah bahasa yang "case sensitive", ini berarti identifier yang dituliskan dengan huruf kapital akan dianggap berbeda dengan identifier yang sama tetapi dituliskan dengan huruf kecil, sabagai contoh : variabel **RESULT** tidak sama dengan variable **result** ataupun variabel **Result**.

Tipe Data

Tipe data yang ada pada C++, berikut nilai kisaran yang dapat direpresentasikan :

Tabel 4.1. Tipe data pada bahasa C++

Tipe Data	Ukuran Memori	Jangkauan Nilai	Jumlah Digit
Char	1 Byte	-128 s.d 127	
Int	2 Byte	-32768 s.d 32767	
Short	2 Byte	-32768 s.d 32767	
Long	4 Byte	-2,147,435,648 s.d 2,147,435,647	
Float	4 Byte	3.4 x 10-38 s.d 3.4 x 10+38	5 – 7
Double	8 Byte	1.7 x 10-308 s.d 1.7 x 10+308	15 – 16
Long Double	10 Byte	3.4 x 10-4932 s.d 1.1 x 10+4932	19

4.2 Deklarasi variabel

Untuk menggunakan variabel pada C++, kita harus mendeklarasikan tipe data yang akan digunakan. Sintaks penulisan deklarasi variabel adalah dengan menuliskan tipe data yang akan digunakan diikuti dengan identifier yang benar. Variabel yang akan digunakan dalam program haruslah dideklarasikan terlebih dahulu. Pengertian deklarasi disini berarti mengenalkan sebuah pengenal ke program dan menentukan jenis data yang disimpam didalamnya.

Bentuk pendefinisian variabel :

tipe daftar_variabel

Perhatikan contoh cara mendeklarasikan variabel dalam bahasa C++

```
Int a;
Float b;
Char kata;
```

Jika akan menggunakan tipe data yang sama untuk beberapa identifier maka dapata dituliskan dengan menggunakan tanda koma, contoh

```
int a, b, c;
float d, e, f;
```

4.2.1 Menentukan Tipe Variabel

Jika variabel hendak dipakai untuk menyimpan data bilangan bulat saja, maka pilihannya adalah tipe bilangan bulat (seperti **int** , **long**). Jika variabel hendak dipakai untuk data bilangan pecahan, maka variabel harus didefinisikan bertipe bilangan pecahan (seperti **float**).

4.2.2 Memberikan Nilai ke Variabel

Bentuk pernyataan yang digunakan untuk memberikan nilai ke variabel yang telah dideklarasikanatau didefinisikan :

variabel = nilai

Pernyataan diatas sering disebut sebagai pernyataan penugasan.

4.2.3 Insialisai Variabel

Adakalanya dalam penulisan program, variabel langsung diberi nilai setelah didefinisikan. Sebagai contoh :

```
int jumlah;
jumlah = 10;
```

Dua pernyataan seperti diatas sebenarnya dapat disingkat melalui pendefinisian yang disertai penugasan nilai, sebagi berikut :

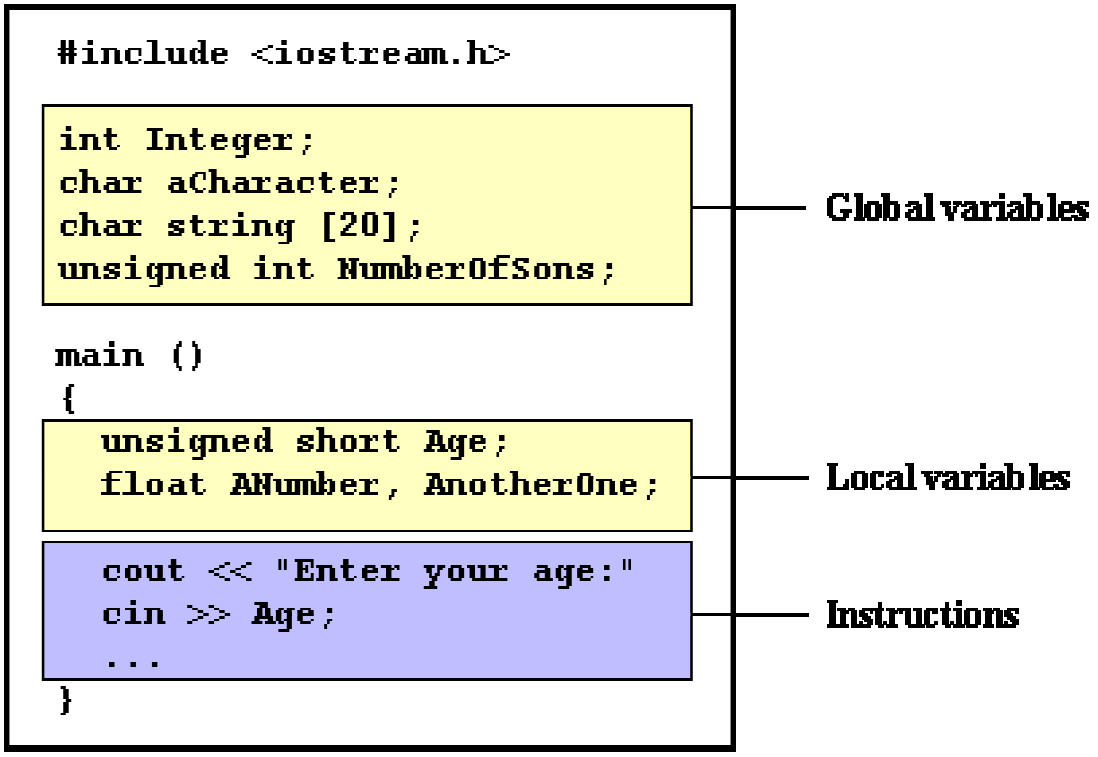
```
int jumlah = 10;
```

Contoh 2	Hasil

`// operating with variables`
`#include <iostream.h>`
 	
`int main ()`
`{`
`// declaring variables:`
`int a, b;`
`int result;`
 	
`// process:`
`a = 5;`
`b = 2;`
`a = a + 1;`
`result = a - b;`
`cout<<result;`
`return 0;`
`}`
	4

4.3 Lingkup Variabel

Pada C++, kita dapat mendeklarasikan variable dibagian mana saja dari program, bahkan diantara 2 kalimat perintah.



Gambar 4.1. Ruang lingkup bahasa C++

variabel Global dapat digunakan untuk setiap bagian dari program, maupun fungsi, walaupun dideklarasikan diakhir program. Lingkup dari **variable local** terbatas. Hanya berlaku dimana variable tersebut dideklarasikan. Jika dideklarasikan diawal fungsi (seperti dalam **main**) maka lingkup dari variable tersebut adalah untuk seluruh fungsi **main**. Seperti contoh diatas, jika terdapat fungsi lain yang ditambahkan pada main(), maka variable local yang dideklarasikan dalam **main** tidak dapat digunakan pada fungsi lainnya dan sebaliknya.

Pada C++, lingkup variable local ditandai dengan blok dimana variable tersebut dideklarasikan (blok tersebut adalah sekumpulan instruksi dalam kurung kurawal {}). Jika dideklarasikan dalam fungsi tersebut, maka akan berlaku sebagai variable dalam fungsi tersebut, jika dideklarasikan dalam sebuah perulangan, maka hanya berlaku dalam perulangan tersebut, dan seterusnya.

4.4 Deklarasi Konstanta

Konstanta adalah ekspresi dengan nilai yang tetap. Terbagi dalam Nilai Integer, Nilai Floating-Point, Karakter and String.

4.4.1 Nilai Integer

Merupakan nilai konstanta numerik yang meng-identifikasikan nilai integer decimal. Karena merupakan nilai numeric, maka tidak memerlukan tanda kutip (") maupun karakter khusus lainnya. Contoh:

```
1776
707
-273
```

C++ memungkinkan kita untuk mempergunakan nilai oktal (base 8) dan heksadesimal (base 16). Jika menggunakan octal maka harus diawali dengan karakter **0** (karakter nol), dan untuk heksadesimal diawali dengan karakter **0x** (nol, x). Contoh :

```
75          // decimal
0113        // octal
0x4b        // hexadecimal
```

Dari contoh diatas, seluruhnya merepresentasikan nilai yang sama : 75.

4.4.2 Nilai Floating Point

Merepresentasikan nilai desimal dan/atau eksponen, termasuk titik desimal dan karakter **e** (Yang merepresentasikan "dikali 10 pangkat n" , dimana n merupakan nilai integer) atau keduanya. Contoh:

```
3.14159     // 3.14159
6.02e23     // 6.02 x 10^23
1.6e-19     // 1.6 x 10^-19
3.0         // 3.0
```

4.5 Karakter dan String

Merupakan konstanta non-numerik, Contoh:

'z'
'p'
"Helloworld"
"How do you do?"

Untuk karakter tunggal dituliskan diantara kutip tunggal (') dan untuk untaian beberapa karakter, dituliskan diantara kutip ganda (").

Kita dapat mendefinisikan sendiri nama untuk konstanta yang akan kita pergunakan, dengan menggunakan preprocessor directive **#define**.

Bentuk pendefinisian konstanta :

#define *identifier value*

Atau

Const tipe *nama_variabel = nilai;*

Contoh :

```
#define PI 3.14159265
#define NEWLINE '\n'
#define WIDTH 100
```

Setelah didefinisikan seperti diatas, maka kita dapat menggunakannya pada seluruh program yang kita buat, contoh :

```
circle = 2 * PI * r;
cout << NEWLINE;
```

Pada dasarnya, yang dilakukan oleh kompiler ketika membaca **#define** adalah menggantikan literal yang ada (dalam contoh, **PI**, **NEWLINE** atau **WIDTH**) dengan nilai yang telah ditetapkan (**3.14159265**, **'\n'** dan **100**). #define bukan merupakan instruksi, oleh sebab itu tidak diakhiri dengan tanda semicolon (;).

Dengan prefix **const** kita dapat mendeklarasikan konstanta dengan tipe yang spesifik seperti yang kita inginkan. contoh:

```
const int width = 100;
const char tab = '\t';
const zip = 12440;
```

Jika tipe data tidak disebutkan, maka kompiler akan meng-asumsikan sebagai **int**.

Contoh 3	Hasil

//*---------- *
//* contoh 3 : inisialisasi variabel dengan *
//* nilai konstan *
//*---------- *
#include <iostream.h>
#include <conio.h>
int main()
{
int jumlah = 10; // inisialisasi
float harga_per_unit = 17.5; // inisialisasi
clrscr();
cout << "Isi Jumlah = "
<< jumlah << '\n';
cout << "Isi harga per per unit = "
<< harga_per_unit << '\n';
}
	Isi Jumlah = 10
	Isi harga per unit = 17.5

Contoh 4	Hasil

//*---------- *
//* Contoh 4 : Contoh Inisialisasi variabel dengan *
//* suatu ungkapan *
//*---------- *
#include <iostream.h>
#include <conio.h>
void main()
{
float kphi = 2 * 3.14; //inisialisasi dengan ungkapan
 	
clrscr();
cout << "Isi duaphi = " << kphi << '\n';
}
	Isi duaphi = 6.28