

# Pertemuan 1

## *1. Pengenalan Bahasa C++*

### **Objektif :**

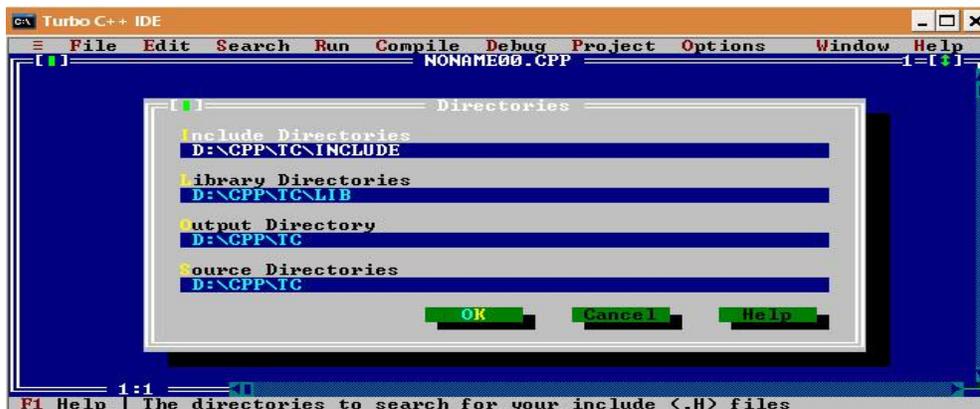
1. Mengerti konsep dasar penggunaan C++
2. Dapat memahami Variabel, Tipe Data, Deklarasi, Ekspresi dan Operator dalam C++
3. Dapat memahami Input / Output pada C++
4. Dapat menerapkan struktur program C++ dan membuat program sederhana menggunakan C++

## P1.1 Pengenalan Editor Turbo C++ IDE

Untuk membuat sebuah program C++ , suatu hal yang dibutuhkan adalah teks editor dan compiler , dengan Borland Turbo C++ yang berfungsi sebagai teks editor sekaligus compiler memudahkan kita untuk membuat sebuah program terutama C++, disamping penggunaan Turbo C++ bisa juga menjadi compiler untuk bahasa C. Berikut Tampilan interface IDE dari turbo C++ versi 3.0



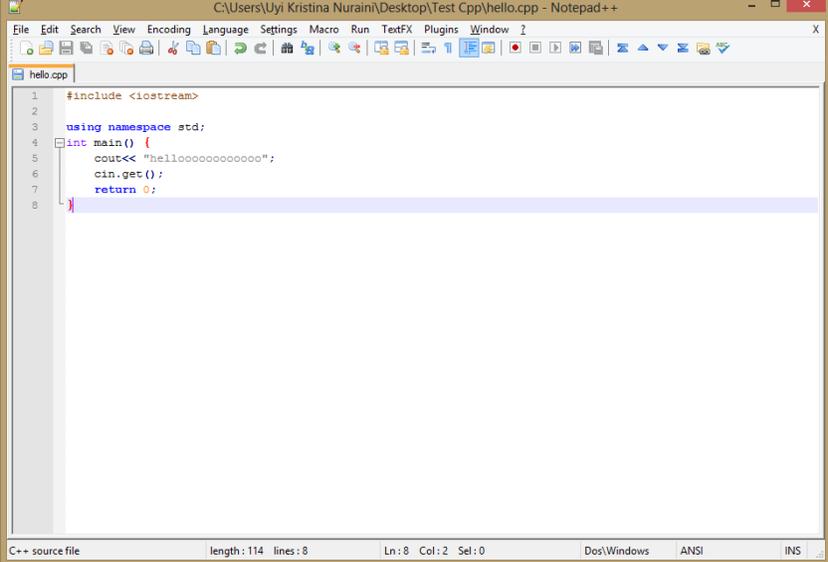
Sebelum menulis suatu program pada Turbo C++ IDE , ada baiknya untuk mensetting dan mencek menu options > directories , sesuaikan letak file pendukung include dan library.



Struktur Bahasa C++

Turbo C++ IDE digunakan untuk pengguna sistem operasi Windows, sedangkan pengguna sistem operasi Linux dapat menggunakan MinGW dan notepad++ sebagai editornya.

Berikut Tampilan interface Notepad++



```
1 #include <iostream>
2
3 using namespace std;
4 int main() {
5     cout<< "hellooooooooooooo";
6     cin.get();
7     return 0;
8 }
```

Kemudian untuk meng-*compile* dan *run* program dapat dilakukan dengan cara :

1. Buka Command Prompt (cmd)
2. Masuk ke dalam folder tempat Anda menyimpan file yang akan dijalankan
3. *Compile* dengan mengetikkan : **g++ [nama file diakhiri .cpp] -o [nama objek]**
4. Jalankan (*run*) dengan **mengetikkan nama objek**

Berikut Tampilan *compile* dan *run* pada cmd :



```
C:\Windows\system32\cmd.exe - hello
D:\Praktikan>g++ hello.cpp -o hello
D:\Praktikan>hello
hellooooooooooooo_
```

**Cara *Compile* dan *Run* pada MinGW**  
***Compile* : g++ [nama file diakhiri .cpp] -o [nama objek]**  
***Run* : [nama objek]**

Struktur program dapat dituliskan dalam bentuk yang lain agar lebih mudah dibaca, contoh :

```
int main ()
{
    cout << " Hello World ";
    return 0;
}
```

Atau dapat juga dituliskan :

```
int main () { cout << " Hello World "; return 0; }
```

Dalam satu baris dan memiliki arti yang sama dengan program-program sebelumnya. pada C++ pembatas antar instruksi ditandai dengan semicolon (;) pada setiap akhir instruksi.

### **Komentar**

Komentar adalah bagian dari program yang diabaikan oleh kompiler. Tidak melaksanakan aksi apapun. Mereka berguna untuk memungkinkan para programmer untuk memasukan catatan atau deskripsi tambahan mengenai program tersebut. C++ memiliki dua cara untuk menuliskan komentar :

```
//    Komentar baris
/*    Komentar Blok    */
```

Komentar baris, akan mengabaikan apapun mulai dari tanda (//) sampai akhir dari baris yang sama. Dan juga akan mengabaikan apapun yang berada diantara tanda /\* dan \*/.

### **Variabel, tipe data, konstanta**

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 :

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

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.

### **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++. Di bawah ini adalah **key word** dalam C++ :

<b>asm</b>	<b>auto</b>	<b>bool</b>	<b>break</b>	<b>case</b>
<b>catch</b>	<b>char</b>	<b>class</b>	<b>const</b>	<b>const_cast</b>
<b>continue</b>	<b>default</b>	<b>delete</b>	<b>do</b>	<b>double</b>
<b>dynamic_cast</b>	<b>else</b>	<b>enum</b>	<b>explicit</b>	<b>extern</b>
<b>false</b>	<b>float</b>	<b>for</b>	<b>friend</b>	<b>goto</b>
<b>if</b>	<b>inline</b>	<b>int</b>	<b>long</b>	<b>mutable</b>
<b>namespace</b>	<b>new</b>	<b>operator</b>	<b>private</b>	<b>protected</b>
<b>public</b>	<b>register</b>	<b>reinterpret_cast</b>	<b>return</b>	<b>short</b>
<b>signed</b>	<b>sizeof</b>	<b>static</b>	<b>static_cast</b>	<b>struct</b>
<b>switch</b>	<b>template</b>	<b>this</b>	<b>throw</b>	<b>true</b>
<b>try</b>	<b>typedef</b>	<b>typeid</b>	<b>typename</b>	<b>union</b>
<b>unsigned</b>	<b>using</b>	<b>virtual</b>	<b>void</b>	<b>volatile</b>
<b>wchar_t</b>				

Sebagai tambahan, repretasi 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**.

## [Type Data](#)

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

### DATA TYPES

Name	Bytes*	Description	Range*
<b>char</b>	1	character or integer 8 bits length.	<b>signed:</b> -128 to 127 <b>unsigned:</b> 0 to 255
<b>short</b>	2	integer 16 bits length.	<b>signed:</b> -32768 to 32767

			<b>unsigned:</b> 0 to 65535
<b>long</b>	4	integer 32 bits length.	<b>signed:</b> -2147483648 to 2147483647 <b>unsigned:</b> 0 to 4294967295
<b>int</b>	*	Integer. Its length traditionally depends on the length of the system's <b>Word type</b> , thus in MSDOS it is 16 bits long, whereas in 32 bit systems (like Windows 9x/2000/NT and systems that work under protected mode in x86 systems) it is 32 bits long (4 bytes).	See <b>short, long</b>
<b>float</b>	4	floating point number.	3.4e + / - 38 (7 digits)
<b>double</b>	8	double precision floating point number.	1.7e + / - 308 (15 digits)
<b>long double</b>	10	long double precision floating point number.	1.2e + / - 4932 (19 digits)
<b>bool</b>	1	Boolean value. It can take one of two values: <b>true</b> or <b>false</b> NOTE: this is a type recently added by the ANSI-C++ standard. Not all compilers support it. Consult section <a href="#">bool type</a> for compatibility information.	<b>true</b> or <b>false</b>
<b>wchar_t</b>	2	Wide character. It is designed as a type to store international characters of a two-byte character set. NOTE: this is a type recently added by the ANSI-C++ standard. Not all compilers support it.	wide characters

### 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, contoh :

```
int a;
float mynumber;
```

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

```
int a, b, c;
```

Tipe data integer (**char**, **short**, **long** dan **int**) dapat berupa signed atau unsigned tergantung dari kisaran nilai yang akan direpresentasikan. Dilakukan dengan menyertakan keyword **signed** atau **unsigned** sebelum tipe data, contoh :

```
unsigned short NumberOfSons;
signed int MyAccountBalance;
```

Jika tidak dituliskan, maka akan dianggap sebagai **signed**.

## Inisialisasi Variabel

Ketika mendeklarasikan variabel local, kita dapat memberikan nilai tertentu. Sintaks penulisan sbb :

```
type identifier = initial_value ;
```

Misalkan kita akan mendeklarasikan variabel **int** dengan nama **a** yang bernilai **0**, maka dapat dituliskan :

```
int a = 0;
```

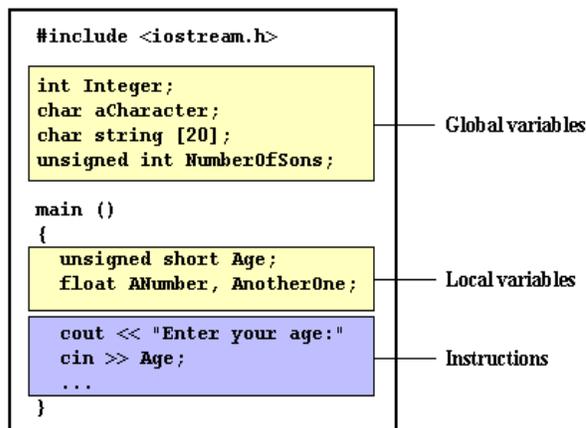
Atau dengan cara lainnya, yaitu menyertakan nilai yang akan diberikan dalam tanda **()**:

```
type identifier (initial_value) ;
```

Contoh :

```
int a (0);
```

## Lingkup Variabel



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

**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.

## Konstanta : Literals.

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

## 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.

## 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 1023
1.6e-19 // 1.6 x 10-19
3.0     // 3.0
```

## Karakter dan String

Merupakan konstanta non-numerik, Contoh :

```
'z'
'p'
"Hello world"
"How do you do?"
```

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

Konstanta karakter dan string memiliki beberapa hal khusus, seperti **escape codes**.

<b>\n</b>	newline
<b>\r</b>	carriage return
<b>\t</b>	tabulation
<b>\v</b>	vertical tabulation
<b>\b</b>	backspace
<b>\f</b>	page feed

<code>\a</code>	alert (beep)
<code>\'</code>	single quotes (')
<code>\"</code>	double quotes (")
<code>\?</code>	question (?)
<code>\\</code>	inverted slash (\)

Contoh:

```
'\n'  
'\t'  
"Left \t Right"  
"one\ntwo\nthree"
```

Sebagai tambahan, kita dapat menuliskan karakter apapun dengan menuliskan yang diikuti dengan kode ASCII, mengekspresikan sebagai octal (contoh, `\23` atau `\40`) maupun heksadesimal (contoh, `\x20` atau `\x4A`).

### Konstanta Define (**#define**)

Kita dapat mendefinisikan sendiri nama untuk konstanta yang akan kita gunakan, dengan menggunakan preprocessor directive **#define**. Dengan format :

```
#define identifier value
```

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 (;).

### Deklarasi Konstanta (**const**)

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**.

## Operator

Operator-operator yang disediakan C++ berupa *keyword* atau karakter khusus. Operator-operator ini cukup penting untuk diketahui karena merupakan salah satu dasar bahasa C++.

### **Assignment (=).**

Operator *assignment* digunakan untuk memberikan nilai ke suatu variable.

```
a = 5;
```

Memberikan nilai integer **5** ke variabel **a**. Sisi kiri dari operator disebut *lvalue* (left value) dan sisi kanan disebut *rvalue* (right value). *lvalue* harus selalu berupa variabel dan sisi kanan dapat berupa konstanta, variabel, hasil dari suatu operasi atau kombinasi dari semuanya.

Contoh:

```
int a, b; // a:? b:?
a = 10; // a:10 b:?
b = 4; // a:10 b:4
a = b; // a:4 b:4
b = 7; // a:4 b:7
```

Hasil dari contoh diatas, **a** bernilai **4** dan **b** bernilai **7**.

Contoh:

```
a = 2 + (b = 5);
```

equivalen dengan :

```
b = 5;
a = 2 + b;
```

### **Arithmetic operators ( +, -, \*, /, % )**

```
+ addition
- subtraction
* multiplication
/ division
% module
```

## Compound assignation operators

(+=, -=, \*=, /=, %=, >>=, <<=, &=, ^=, |=)

contoh :

```
value += increase; equivalen dengan value = value +
increase;
a -= 5; equivalen dengan a = a - 5;
a /= b; equivalen dengan a = a / b;
price *= units + 1; equivalen dengan price = price *
(units + 1);
```

## Increment / Increase / Penambahan (++) dan Decrement / Decrease / Pengurangan(--).

Contoh:

```
a++;
a+=1;
a=a+1;
```

Contoh diatas adalah equivalen secara fungsional. Nilai a dikurangi 1.

Operator Increase dan Decrease dapat digunakan sebagai *prefix* atau *suffix*. Dengan kata lain dapat dituliskan sebelum identifier variabel (**++a**) atau sesudahnya (**a++**). operator increase yang digunakan sebagai *prefix* (**++a**), Perbedaannya terlihat pada tabel dibawah ini :

<u>Example 1</u>	<u>Example 2</u>
<pre>B=3; A=++B; // A is 4, B is 4</pre>	<pre>B=3; A=B++; // A is 3, B is 4</pre>

Pada contoh 1, **B** ditambahkan sebelum nilainya diberikan ke **A**. Sedangkan contoh 2, Nilai **B** diberikan terlebih dahulu ke **A** dan **B** ditambahkan kemudian.

## **Relational operators** (==, !=, >, <, >=, <=)

Untuk mengevaluasi antara 2 ekspresi, dapat digunakan operator Relasional. Hasil dari operator ini adalah nilai **bool** yaitu hanya berupa **true** atau **false**, atau dapat juga dalam nilai **int**, **0** untuk merepresentasikan "**false**" dan **1** untuk merepresentasikan "**true**". Operator-operator relasional pada C++ :

```
== Equal
!= Different
```

- > Greater than
- < Less than
- >= Greater or equal than
- <= Less or equal than

Contoh:

`(7 == 5)` would return **false**.

`(3 != 2)` would return **true**.

`(5 < 5)` would return **false**.

Contoh, misalkan `a=2`, `b=3` dan `c=6` :

`(a == 5)` would return **false**.

`(a*b >= c)` would return **true** since  $(2*3 \geq 6)$  is it.

`(b+4 > a*c)` would return **false** since  $(3+4 > 2*6)$  is it.

`((b=2) == a)` would return **true**.

**Logic operators ( !, &&, || ).**

Operator **!** equivalen dengan operasi boolean NOT, hanya mempunyai 1 operand, berguna untuk membalikkan nilai dari operand yang bersangkutan. Contoh:

`!(5 == 5)` returns **false** because the expression at its right  $(5 == 5)$  would be **true**.

`!(6 <= 4)` returns **true** because  $(6 \leq 4)$  would be **false**.

`!true` returns **false**.

`!false` returns **true**.

operator Logika **&&** dan **||** digunakan untuk mengevaluasi 2 ekspresi dan menghasilkan 1 nilai akhir. mempunyai arti sama dengan operator logika Boolean *AND* dan *OR*.  
Contoh :

First Operand a	Second Operand b	result a && b	result a    b
true	true	<b>true</b>	<b>true</b>
true	false	<b>false</b>	<b>true</b>
false	true	<b>false</b>	<b>true</b>
false	false	<b>false</b>	<b>false</b>

Contoh:

`((5 == 5) && (3 > 6))` returns **false** (`true && false`).

`((5 == 5) || (3 > 6))` returns **true** (`true || false`).

**Conditional operator ( ? ).**

operator kondisional mengevaluasi ekspresi dan memberikan hasil tergantung dari hasil evaluasi (*true* atau *false*). Sintaks :

$$condition \ ? \ result1 \ : \ result2$$

Jika kondisi **true** maka akan menghasilkan *result1*, jika tidak akan menghasilkan *result2*.

$7==5 \ ? \ 4 \ : \ 3$  returns 3 since 7 is not equal to 5.  
 $7==5+2 \ ? \ 4 \ : \ 3$  returns 4 since 7 is equal to 5+2.  
 $5>3 \ ? \ a \ : \ b$  returns a, since 5 is greater than 3.  
 $a>b \ ? \ a \ : \ b$  returns the greater one, a or b.

**Bitwise Operators ( &, |, ^, ~, <<, >> ).**

Operator Bitwise memodifikasi variabel menurut bit yang merepresentasikan nilai yang disimpan, atau dengan kata lain dalam representasi binary.

op	asm	Description
&	<b>AND</b>	Logical AND
	<b>OR</b>	Logical OR
^	<b>XOR</b>	Logical exclusive OR
~	<b>NOT</b>	Complement to one (bit inversion)
<<	<b>SHL</b>	Shift Left
>>	<b>SHR</b>	Shift Right

**Explicit type casting operators**

Type casting operators memungkinkan untuk mengkonversikan tipe data yang sudah diberikan ke tipe data yang lain. Ada beberapa cara yang dapat dilakukan dalam C++, yang paling populer yaitu tipe baru dituliskan dalam tanda kurung ( ) contoh :

```
int i;
float f = 3.14;
i = (int) f;
```

Contoh diatas, mengkonversikan nilai **3.14** menjadi nilai integer (**3**). Type casting operator yang digunakan (**int**). Cara lainnya :

```
i = int ( f );
```

**sizeof()**

Operator ini menerima 1 parameter, dapat berupa type variabel atau variabel itu sendiri dan mengembalikan ukurannya type atau object tersebut dalam bytes :

```
a = sizeof (char);
```

Contoh diatas akan memberikan nilai 1 ke **a** karena **char** adalah tipe data dengan panjang 1 byte. Nilai yang diberikan oleh **sizeof** bersifat konstsn constant.

**Prioritas pada operator**

Contoh:

```
a = 5 + 7 % 2
```

Jawaban dari contoh diatas adalah 6. Dibawah ini adalah prioritas operator dari tinggi ke rendah :

Priority	Operator	Description	Associativity
1	::	scope	Left
2	() [ ] -> . sizeof		Left
3	++ --	increment/decrement	Right
	~	Complement to one (bitwise)	
	!	unary NOT	
	& *	Reference and Dereference (pointers)	
	(type)	Type casting	
	+ -	Unary less sign	
4	* / %	arithmetical operations	Left
5	+ -	arithmetical operations	Left
6	<< >>	bit shifting (bitwise)	Left
7	< <= > >=	Relational operators	Left
8	== !=	Relational operators	Left
9	& ^	Bitwise operators	Left
10	&&	Logic operators	Left
11	?:	Conditional	Right
12	= += -= *= /= %= >>= <<= &= ^=  =	Assignment	Right
13	,	Comma, Separator	Left

## Komunikasi melalui console

*Console* merupakan interface dasar pada computers, biasanya berupa keyboard dan monitor. Keyboard merupakan alat *input* standar dan monitor adalah alat *output* standar. Dalam library *iostream* C++ , standard operasi *input* dan *output* untuk pemrograman didukung oleh 2 data streams: **cin** untuk input dan **cout** untuk output. Juga, **cerr** dan **clog** sebagai tambahan untuk output streams yang di desain khusus untuk menampilkan *error messages*. Dapat diarahkan langsung ke standard output maupun ke log file.

Biasanya **cout** (standard output stream) ditujukan untuk monitor dan **cin** (standard input stream) ditujukan untuk keyboard. Dengan menggunakan dua streams ini, maka kita dapat berinteraksi dengan user dengan menampilkan messages pada monitor dan menerima input dari keyboard.

### Output (cout)

Penggunaan **cout** stream dhubungkan dengan operator overloaded << (Sepasang tanda "*less than*"). Contoh :

```
cout << "Output sentence"; // prints Output sentence on screen
cout << 120;               // prints number 120 on screen
cout << x;                 // prints the content of variable x on screen
```

Operator << dikenal sebagai *insertion operator*, dimana berfungsi untuk menginput data yang mengikutinya. Jika berupa string, maka harus diapit dengan kutip ganda ("), sehingga membedakannya dari variable. Contoh :

```
cout << "Hello"; // prints Hello on screen
cout << Hello;   // prints the content of Hello variable on screen
```

Operator *insertion* (<<) dapat digunakan lebih dari 1 kali dalam kalimat yang sama, Contoh :

```
cout << "Hello, " << "I am " << "a C++ sentence";
```

Contoh diatas akan menampilkan **Hello, I am a C++ sentence** pada layar monitor. Manfaat dari pengulangan penggunaan operator insertion (<<) adalah untuk menampilkan kombinasi dari satu variabel dan konstanta atau lebih, contoh :

```
cout << "Hello, I am " << age << " years old and my zipcode is " <<
zipcode;
```

Misalkan variable age = 24 dan variable zipcode = 90064 maka output yang dihasilkan :

```
Hello, I am 24 years old and my zipcode is 90064
```

Contoh:

```
cout << "First sentence.\n ";  
cout << "Second sentence.\nThird sentence.";
```

Output :

```
First sentence.  
Second sentence.  
Third sentence.
```

Selain dengan karakter new-line, dapat juga menggunakan manipulator **endl**, contoh :

```
cout << "First sentence." << endl;  
cout << "Second sentence." << endl;
```

Output :

```
First sentence.  
Second sentence.
```

### **Input (cin).**

Menangani standard input pada C++ dengan menambahkan overloaded operator *extraction* (>>) pada **cin** stream. Harus diikuti dengan variable yang akan menyimpan data. Contoh :

```
int age;  
cin >> age;
```

Contoh diatas mendeklarasikan variabel `age` dengan tipe `int` dan menunggu input dari `cin` (keyboard) untuk disimpan di variabel `age`.

**cin** akan memproses input dari keyboard sekali saja dan tombol ENTER harus ditekan.

Contoh:

```
// i/o example
#include <iostream.h>

int main ()
{
    int i;
    cout << "Please enter an integer value: ";
    cin >> i;
    cout << "The value you entered is " << i;
    cout << " and its double is " << i*2 << ".\n";
    return 0;
}
```

Output :

```
Please enter an integer value: 702
The value you entered is 702 and its double is 1404.
```

**cin** juga dapat digunakan untuk lebih dari satu input :

```
cin >> a >> b;
```

Equivalen dengan :

```
cin >> a;
cin >> b;
```

Dalam hal ini data yang di input harus 2, satu untuk variabel **a** dan lainnya untuk variabel **b** yang penulisannya dipisahkan dengan : spasi, tabular atau *newline*.

## P1.2 Contoh Kasus

- *my first program in C++ :*

```
// my first program in C++
#include <iostream.h>
int main ()
{
    cout << "Hello World!";
    return 0;
}
```

Hasil :

```
Hello World!
```

Sisi kiri merupakan *source code*, yang dapat diberi nama *hiworld.cpp* dan sisi kanan adalah hasilnya setelah di-kompilasi dan di-eksekusi.

Program diatas merupakan salah satu program paling sederhana dalam C++, tetapi dalam program tersebut mengandung komponen dasar yang selalu ada pada setiap pemrograman C++. Jika dilihat satu persatu :

```
// my first program in C++
```

Baris ini adalah komentar. semua baris yang diawali dengan dua garis miring (//) akan dianggap sebagai komentar dan tidak akan berpengaruh terhadap program. Dapat digunakan oleh programmer untuk menyertakan penjelasan singkat atau observasi yang terkait dengan program tersebut.

```
#include <iostream.h>
```

Kalimat yang diawali dengan tanda (#) adalah are *preprocessor directive*. Bukan merupakan baris kode yang dieksekusi, tetapi indikasi untuk kompilasi. Dalam kasus ini kalimat **#include <iostream.h>** memberitahukan preprocessor kompilasi untuk menyertakan header file standard **iostream**. File spesifik ini juga termasuk library deklarasi standard I/O pada C++ dan file ini disertakan karena fungsi-fungsinya akan digunakan nanti dalam program.

```
int main ()
```

Baris ini mencocokkan pada awal dari deklarasi fungsi **main**. fungsi **main** merupakan titik awal dimana seluruh program C++ akan mulai dieksekusi. Diletakkan diawal, ditengah atau diakhir program, isi dari fungsi main akan selalu dieksekusi pertama kali. Pada dasarnya, seluruh program C++ memiliki fungsi **main**.

**main** diikuti oleh sepasang tanda kurung ( ) karena merupakan fungsi. pada C++, semua fungsi diikuti oleh sepasang tanda kurung ( ) dimana, dapat berisi argumen didalamnya. Isi dari fungsi **main** selanjutnya akan mengikuti, berupa deklarasi formal dan dituliskan diantara kurung kurawal ({}), seperti dalam contoh.

```
cout << "Hello World";
```

Intruksi ini merupakan hal yang paling penting dalam program contoh. **cout** merupakan standard output stream dalam C++ (biasanya monitor). **cout** dideklarasikan dalam header file **iostream.h**, sehingga agar dapat digunakan maka file ini harus disertakan.

Perhatikan setiap kalimat diakhiri dengan tanda semicolon (;). Karakter ini menandakan akhir dari instruksi dan harus disertakan pada setiap akhir instruksi pada program C++ manapun.

```
return 0;
```

Intruksi **return** menyebabkan fungsi **main()** berakhir dan mengembalikan kode yang mengikuti instruksi tersebut, dalam kasus ini **0**. Ini merupakan cara yang paling sering digunakan untuk mengakhiri program.

Tidak semua baris pada program ini melakukan aksi. Ada baris yang hanya berisi

komentar (diawali //), baris yang berisi instruksi untuk preprocessor kompilasi (Yang diawali #), kemudian baris yang merupakan inisialisasi sebuah fungsi (dalam kasus ini, fungsi **main**) dan baris yang berisi instruksi (seperti, **cout <<**), baris yang terakhir ini disertakan dalam blok yang dibatasi oleh kurung kurawal (**{ }**) dari fungsi **main**.

- Contoh Kasus Sederhana Menggunakan Input :

```
#include <iostream.h>
#include <conio.h>

int main()
{
    char nama[20];
    int tahun, umur;
    clrscr();
    cout << "Masukan nama Anda ? ";
    cin >> nama;
    cout << "Masukan tahun lahir anda ? ";
    cin >> tahun;
    umur = 2007 - tahun;
    cout << "Halo " << nama << ", Anda sedang belajar C++ ya ?" << endl;
    cout << "Umur " << nama << " tahun 2007 adalah " << umur << " thn";

    getch();
    return 0;
}
```

endl => pindah baris, atau bisa dengan (“\n”).

Karena program kita menggunakan **INT** (lihat pada fungsi main), maka perlu adanya pengembalian nilai. Untuk itu kita gunakan **return 0** agar tidak terjadi kesalahpahaman antara kita dengan program.

### P. 1. 3 Latihan

1. Perhatikan program di bawah ini, carilah output dari program tersebut :

```
// operating with variables
#include <iostream.h>           //pada MinGW (linux) tanpa ekstensi .h
#include <conio.h>              //hanya digunakan pada windows
// using namespace std;       //hanya digunakan pada MinGW (Linux)
int main ()
{
    // declaring variables:
    int a, b;
    int result;

    // process:
    a = 5;
    b = 2;
    a = a + 1;
    result = a - b;
    cout<< result;
    getch();                   //membutuhkan conio.h jika di MinGW (linux)
                                diganti menjadi cin.get();

    return 0;
}
```

### P. 1.4 Daftar Pustaka

1. Ayuliana, modul pengenalan bahasa C++, Gunadarma Jakarta, February 2004
2. Hari, Konsep Dasar Objek Oriented Programming, FTI budiluhur Jakarta, 2003
3. r.hubbard, John , schaum's outline of theory and problems of programming with C++ second edition, mcgraw-hill, New York 2000
4. <http://www.cplusplus.com/>
5. <http://cs.binghamton.edu/~steflik/>
6. <http://en.wikipedia.org/wiki/c++>
7. <http://www.mingw.org/>