**PRAKTIKUM 1**

**Pemrograman Berorientasi Objek**

1. **PENGANTAR C++**

C++ diciptakan oleh Bjarne Stroustrup di laboratorium Bell pada awal tahun 1980-an, sebagai pengembangan dari bahasa C dan Simula. Saat ini, C++ merupakan salah satu bahasa yang paling populer untuk pengembangan software berbasis OOP.

Kompiler untuk C++ telah banyak beredar di pasaran. Software developer yang paling diminati adalah Borland Inc. dan Microsoft Corp.

Produk dari Borland untuk kompiler C++ adalah Turbo C++, Borland C++, Borland C++ Builder. Sedangkan dari Microsoft adalah Ms. Visual C++. Walaupun banyak kompiler yang tersedia, namun pada intinya bahasa pemrograman yang dipakai adalah C++.

**22** Sebelum mulai melakukan kode program, sebaiknya diingat bahwa C++ bersifat *“****case sensitive****”,* yang artinya huruf besar dan huruf kecil dibedakan.

1. **STRUKTUR BAHASA C++**

Cara terbaik untuk balajar bahasa pemrograman adalah dengan langsung mempraktikannya. Cobalah contoh program berikut :

*// program pertama*

#include <iostream.h>

int main ()

{

cout << "Selamat Datang di PBO";

return 0;

}

Program di atas, misalnya dapat disimpan dengan nama *coba.cpp*. Cara untuk menyimpan dan mengkompile program berbeda-beda, tergantung kompiler yang dipakai.

Ketika di-*run*, maka di layar akan muncul sebuah tulisan “Selamat Belajar PBO”. Contoh di atas, adalah sebuah contoh program sederhana menggunakan C++. Namun, penggalan program tersebut telah menyertakan sintak-sintak dasar bahasa C++. Sintak dasar tersebut, akan kita bahas satu per satu:

***// program pertama***

merupakan sebuah baris komentar. Semua baris, yang ditandai dengan dua buah tanda slash (//), akan dianggap sebagai baris komentar dan tidak akan berpengaruh pada hasil. Biasanya, baris komentar dipakai oleh programmer untuk memberikan penjelasan tentang program. Baris komentar dalam C++, selain ditandai dengan (//) juga dapat ditandai dengan (/\*….\*/) Perbedaan mendasar dari keduanya adalah :

// baris komentar /\*

blok komentar \*/

**#include <iostream.h>**

pernyataan yang diawali dengan tanda (#) merupakan pernyataan untuk menyertakan preprocessor. Pernyataan ini bukan untuk dieksekusi. #include <iostream.h> berarti memerintahkan kompiler untuk menyertakan file header iostream.h. Dalam file header ini, terdapat beberapa fungsi standar yang dipakai dalam proses input dan output. Seperti misalnya perintah *cout* yang dipakai dalam program utama.

**int main ()**

baris ini menandai dimulainya kompiler akan mengeksekusi program. Atau dengan kata lain, pernyataan **main** sebagai penanda program utama. Adalah suatu keharusan, dimana sebuah program yang ditulis dalam bahasa C++ memiliki sebuah **main**. **main** diikuti oleh sebuah tanda kurung () karena main merupakan sebuah fungsi. Dalam bahasa C++ sebuah fungsi harus diikuti dengan tanda (), yang nantinya dapat berisi argumen. Dan sintak formalnya, sebuah fungsi dimulai dengan tanda {}, seperti dalam contoh program.

**cout << "Selamat Datang di PBO";**

perintah ini merupakan hal yang akan dieksekusi oleh compiler danmerupakan perintah yang akan dikerjakan. cout termasuk dalam file iostream.cout merupakan perintah untuk menampilkan ke layer.Perlu diingat, bahwa setiap pernyataan dalam C++ harus diakhiri dengan tandasemicolon (;) untuk memisahkan antara pernyataan satu dengan pernyataanlainnya.

**return 0;**

pernyataan return akan menyebabkan fungsi main() menghentikan programdan mengembalikan nilai kepada main. Dalam hal ini, yang dikembalikan adalahnilai 0. Mengenai pengembalian nilai, akan dijelaskan nanti mengenai **Fungsi** dalam C++. Coba tambahkan sebaris pernyataan lagi, sehingga program contoh di atas akan menjadi seperti berikut:

*// latihan kedua*

#include <iostream.h>

int main ()

{

cout <<"Selamat Datang di PBO”;

cout <<"di Esa Unggul";

return 0;

}

Maka perintah cout yang kedua akan menampilkan sebuah kalimat lagi di layar, dengan tulisan “di kampusku”.

1. **TIPE DATA**

Terdapat 5 tipe data bawaan dari bahasa C, yaitu : **void**, **integer**, **float**, **double**, dan **char**.

|  |  |
| --- | --- |
| **Type** | **Keterangan** |
| void | diartikan sebagai tanpa tipe data dan tanpa pengembalian nilai |
| int | bilangan bulat (integer) |
| float | bilangan pecahan (floating point) |
| double | pecahan dengan jangkauan data yang lebih luas |
| char | Karakter |

Sedangkan C++ sendiri menambahkan dua buah tipe data lagi, yakni : **bool** dan **wchar\_t**.

|  |  |
| --- | --- |
| **Type** | **Keterangan** |
| bool | isi bilangan Boolean (True dan False) |
| wchar | wide character |

Dengan jangkauan nya adalah sebagai berikut

|  |  |  |
| --- | --- | --- |
| **Type** | **Ukuran (bits)** | **Range** |
| unsigned char | 8 | 0 s/d 255 |
| char | 8 | -128 s/d 127 |
| short int | 16 | -32,768 s/d 32,767 |
| unsigned int | 32 | 0 s/d 4,294,967,295 |
| int | 32 | -2,147,483,648 s/d 2,147,483,647 |
| unsigned long | 32 | 0 s/d 4,294,697,295 |
| long | 32 | -2,147,483,648 s/d 2,147,483,647 |
| float | 32 | 3.4 e-38 s/d 1.7 E +38 |
| double | 64 | 1.7 E-308 s/d 3.4 E + 308 |
| long double | 80 | 3.4 E-4932 s/d 1.1 E + 4932 |

1. **VARIABEL**

Berbeda dengan pendeklarasian variabel di bahasa pemrograman lain, dalam C++ sebelum mendeklarasikan variabel, hal pertama yang harus dideklarasikan adalah tipe data yang akan digunakan untuk menampung data. Format penulisannya adalah :

Tipe\_data pengenal = nilai ;

Sebagai contoh :

int a; float

nomor;

atau dapat juga pemberian nilai awal untuk variable dilakukan pada saat deklarasi, contoh :

int a=10;

char s=’a’;

Jika hendak mendeklarasikan beberapa variabel sekaligus dengan tipe data yang sama, dapat dilakukan dengan 2 cara, yaitu :

int a;

int b;

int c;

atau dapat disederhanakan dengan deklarasi :

int a,b,c;

Perhatikan contoh berikut:

*// bekerja dengan variabel*

#include <iostream.h>

int main ()

{

*// inisialisasi variabel :*

int a, b;

int hasil;

*// proses :*

a = 5;

b = 2;

a = a + 1;

hasil = a - b;

*// cetak hasilnya :*

cout << hasil;

*// menghentikan program :*

return 0;

}

1. **KONSTANTA**

Konstanta mirip dengan variable, namun memiliki nilai tetap. Konstanta dapat berupa nilai Integer, Float, Karakter dan String. Pendeklarasian konstanta dapat dilakukan dengan 2 cara :

* menggunakan (**#define**)

deklarasi konstanta dengan cara ini, lebih gampang dilakukan karena akan menyertakan **#define** sebagai preprocessor directive. Dan sintaknya diletakkan bersama – sama dengan pernyataan #include (di atas main()). Format penulisannya adalah :

**#define** pengenal nilai

Contoh penggunaan :

#define phi 3.14159265

#define Newline ‘\n’

#define lebar 100

pendeklarasian dengan #define tanpa diperlukan adanya tanda = untuk memasukkan nilai ke dalam pengenal dan juga tanpa diakhiri dengan tanda semicolon(;).

* menggunakan (**const**)

Sedangkan dengan kata kunci const, pendeklarasian konstanta mirip dengan deklarasi variable yang ditambah kata depan const. Contoh :

const int lebar = 100;

const char tab = ‘\t’;

const zip = 1212;

Untuk contoh terakhir, deklarasi variable zip yang tanpa tipe data, maka compiler akan secara otomatis memasukkannya ke dalam tipe int.

1. **OPERATOR**

Dalam C++, terdapat berbagai macam operator yang dapat dimanfaatkan dalam aplikasi.

* **Operator Assign** (=)

Operator (=), akan memberikan nilai ke dalam suatu variable.

**^** *rvalue*

*lvalue* < a=5

artinya memberikan nilai 5 ke dalam variable a. Sebelah kiri tanda =

dalam pernyataan di atas, dikenal dengan ***lvalue*** (left value) dan di sebelah

kanan tanda = dikenal dengan ***rvalue*** (right value). lvalue harus selalu berupa

variable, sedangkan rvalue dapat berupa variable, nilai, konstanta, hasil operasi

ataupun kombinasinya.

* **Operator Aritmatika** ( +, , \*, /, %)

|  |  |
| --- | --- |
| **Operator** | **Keterangan** |
| + | Penjumlahan |
| - | Pengurangan |
| \* | Perkalian |
| / | Pembagian |
| % | Modulus |

Untuk operator %, sama dengan modulus, yaitu untuk mengetahui sisa hasil

bagi. Misalnya a = 11 % 3, maka variable a akan terisi nilai 2 karena sisa

hasil bagi 11 dan 3 adalah 2.

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

Dalam C++, operasi aritmatika dapat disederhanakan penulisannya dengan format penulisan operator majemuk. Misalnya :

a += 5 sama artinya dengan menuliskan a = a+5 a

\*= 5 sama artinya dengan menuliskan a = a\*5

a /= 5 sama artinya dengan menuliskan a = a/5

a %= 5 sama artinya dengan menuliskan a = a % 5

a = 5

* **Operator Penaikan dan Penurunan** (++ dan )

Operator penaikan (++) akan menaikkan atau menambahkan 1 nilai variable. Sedangkan operator (--) akan menurunkan atau mengurangi 1 nilai variable. Misalnya :

a++;

a+= 1;

a=a+l;

untuk ketiga pernyataan tersebut, memiliki arti yang sama yaitu menaikkan lnilai variable 1. Karakteristik dari operator ini adalah dapat dipakai di awal (++a) atau di akhir (--a) variable. Untuk penggunaan biasa, mungkin tidak akan ditemui perbedaan hasil dari cara penulisannya. Namun untuk beberapa operasi nantinya harus diperhatikan cara peletakan operator ini, karena akan berpengaruh terhadap hasil.

Contoh 1 : Contoh 2:

B=3; B=3;

A=++B; A=B++;

// A= 4, B=4 //hasil A=3, B=4

Dari contohl, nilai B dinaikkan sebelum dikopi ke variable A. Sedangkan

pada contoh2, nilai B dikopi terlebih dahulu ke variable A baru kemudian

dinaikkan.

* **Operator Relasional** (==, ! =, >, <, >=, <=)

Yang dihasilkan dari operator ini bukan berupa sebuah nilai, namun berupa bilangan bool yaitu benar atau salah.

|  |  |
| --- | --- |
| **Operator** | **Keterangan** |
| == | Sama dengan |
| != | Tidak sama dengan |
| > | Lebih besar dari |
| < | Kurang dari |
| >= | Lebih besar dari atau sama dengan |
| <= | Kurang dari atau sama dengan |

Contoh :

(7==5) hasilnya adalah **false**

(5>4) hasilnya adalah **true**

(5<5) hasilnya adalah **false**

* **Operator Logika** ( !, &&, || )

Operator logika juga digunakan untuk memberikan nilai atau kondisi **true** dan **false**. Biasanya operator logika dipakai untuk membandingkan duakondisi. Misalnya:

((5==5) && (3>6)) mengembalikan nilai **false,** karena (true && false) untuk logika NOT (!), contohnya !(5==5) akan mengembalikan nilai **false**, karena !(true).

* **Operator Kondisional** ( ? )

Format penulisan operator kondisional adalah :

kondisi ? hasil1 : hasil2

Jika kondisi benar maka yang dijalankan adalah hasil1 dan jika kondisi salah, maka akan dijalankan hasil2.

Contoh :

7==5 ? 4 : 3 hasilnya adalah **3**, karena 7 tidak sama dengan 5

5>3 ? a : b hasilnya adalah a, karena **5** lebih besar dari 3