**MODUL 4**

**FUNGSI (1)**

Dengan mempergunakan fungsi, maka struktur program akan terlihat lebih ramping. Fungsi merupakan sebuah blok instruksi yang dieksekusi dan dipanggil dari bagian lain tubuh program. Format penulisannya adalah sebagai berikut:

***tipe nama(argumen1, argumen2,…) pernyataan;***

Dimana:

***tipe*** berisi tipe data yang akan dikembalikan oleh fungsi

***nama*** merupakan pengenal untuk memanggil fungsi

***argumen*** (dapat dideklarasikan sesuai dengan kebutuhan).

Tiap-tiap argumen terdiri dari tipe-tipe data yang diikuti oleh pengenalnya. Sama seperti

mendeklarasikan variable baru (contoh, int x).

***pernyataan*** merupakan bagian tubuh fungsi.

Dapat berupa pernyataan tunggal ataupun pernyataan majemuk.

Contoh penggunaan fungsi:

//contoh fungsi

#include <iostream.h>

#include <conio.h>

int penjumlahan(int a, int b)

{ int r;

r=a+b;

return (r); }

int main() {

int z;

z=penjumlahan(5,3);

cout<<"Hasil penjumlahan = " << z;

return 0;

}

Hasil eksekusinya adalah :

Hasil penjumlahan = 8

Ketika program dieksekusi, yang dijalankan pertama kali adalah fingsi main(). Terlihat jelas bahwa dalam main() terdapat variable z dengan tipe data integer. Setelah itu, fungsi **penjumlahan** dipanggil. Maka akan terdapat proses pertukaran data sebagai berikut:

**int penjumlahan(int a, int b)**

**t t**

**z = penjumlahan ( 5 , 3 );**

maka setelah terjadi pengisian nilai, variable a akan terisi dengan nilai 5 dan

variable b akan terisi dengan nilai 3. Fungsi **penjumlahan** mendeklarasikan sebuah variable baru lagi (int r;) dan kemudian menjumlahkan nilai r=a+b; dengan hasil akhir variable r = 8. Karena masing-masing variable a dan b sudah terisi dengan nilai 5 dan 3.

Kode :

return (r);

merupakan pengakhir fungsi **penjumlahan** dan memberikan hasil akhir nilai r kepada fungsi yang memanggilnya (dalam hal ini fungsi **main()**). Proses pengembalian nilai dapat digambarkan sebagai berikut:

**int penjumlahan(int a, int b)**

**I**8

**z = penjumlahan ( 5 , 3 );**

maka ketika perintah cout<<"Hasil penjumlahan = " << z; dijalankan, hasilnya adalah 8.

**Scope (Batasan) Variabel**

Variabel yang dideklarasikan di dalam tubuh fungsi, hanya dapat diakses oleh fungsi itu. Dan tidak dapat dipergunakan di luar fungsi.

Contoh:

Pada program sebelumnya, variable a dan b atau r tidak dapat digunakan dalam fungsi main(), sebab variable tersebut merupakan **variable lokal** dalam fungsi penjumlahan. Demikian juga halnya dengan variable z. Tidak dapat dipergunakan dalam fungsi penjumlahan karena merupakan variable lokal dalam fungsi main(). Selain variable lokal, terdapat pula **variable global** yang dapat diakses dari mana saja. Dari dalam maupun luar tubuh fungsi. Untuk mendeklarasikan variable global, harus dituliskan di luar fungsi atau blok instruksi.

Contoh lain fungsi:

#include<iostream.h>

#include<conio.h>

Int kurang(int a, int D)

{

int r; r=a-b;

} return (r);

int main()

int x= 5, y=3, z;

z=kurang(7,2);

cout<<"Pertama : " << z<<endl;

cout<<"Kedua : " << kurang(7,2)<<endl;

cout<<"Ketiga : " << kurang (x,y) <<endl;

z=4+kurang(x,y);

cout<<"Keempat : "<<z<<endl;

} return 0;

Hasilnya adalah

Pertama : 5

Kedua : 5

Ketiga : 2

Keempat : 6

Dalam program di atas, terdapat sebuah fungsi yang dinamakan **kurang**. Fungsi

ini mengerjakan tugas untuk mengurangkan nilai dua buah variable dan kemudian

mengembalikan hasilnya.

Sedangkan di dalam fungsi main(), terdapat beberapa kali pemanggilan terhadap fungsi kurang. Disana terdapat dengan jelas perbedaan cara pengaksesan dan pengaruh terhadap hasilnya. Untuk lebih memperjelas, beberapa sintak akan dijelaskan secara rinci sebagai berikut:

z=kurang(7,2);

cout<<"Pertama : " << z<<endl;

Jika diganti dengan hasil (sesuai dengan pemanggilan fungsinya), maka akan didapatkan baris:

Z =5;

cout<<"Pertama : " << z<<endl;

sama seperti baris:

cout<<"Kedua : " << kurang(7,2)<<endl;

jika sesuai dengan pemanggilan fungsinya, maka akan didapatkan:

cout<<"Kedua : " << kurang(7,2)<<endl;

sedangkan untuk:

cout<<"Ketiga : " << kurang (x,y) <<endl;

maka isi dari variabel x=5 dan y=3, sehingga dapat diartikan sebagai baris:

cout<<"Ketiga : " << kurang (5,3) <<endl;

demikian juga halnya dengan baris:

z=4+kurang(x,y);

dapat diartikan dengan baris:

z=4+kurang(5,3);

**Pendeklarasian fungsi tanpa tipe (menggunakan *void)***

Kadang-kadang terdapat fungsi yang tanpa memerlukan adanya pengembalian nilai. Misalkan, sebuah fungsi yang hanya bertugas mencetak kalimat ke layar monitor dan tanpa memerlukan adanya pertukaran parameter. Dalam kondisi seperti ini, maka dipergunakan kata kunci ***void****.*

Contoh program:

#include<iostream.h>

#include<conio.h> void

Ucapan(void) {

cout<<"Selamat Belajar C++";

} int main()

{

Ucapan();

return 0;

}

Yang harus diperhatikan adalah, pemanggilan fungsi hams disertai dengan tanda (). Seperti contoh di atas, fungsi Ucapan walaupun didekarasikan tanpa tipe data dan tanpa argumen, dipanggil dalam fungsi main dengan Ucapan().

**FUNGSI (2)**

Cara pelewatan argumen adalah :

1. Pemanggilan dengan nilai *(arguments passed by value)*
2. Pemanggilan dengan acuan *(arguments passed by reference)*

Sampai saat ini, fungsi-fungsi yang telah dibuat adalah menggunakan pemanggilan argumen berdasarkan nilai. Contoh :

int x= 5, y=3, z;

z=kurang(x,y);

Pada penggalan di atas, terjadi pemanggilan terhadap fungsi kurang() dengan x dan y masing-masing bernilai 5 dan 3.

int kuranq(int a, int b)

**t t**

z = kurang( x, y);

Dengan pemanggilan tersebut, maka pengisian nilai terhadap variabel a dan b dilakukan oleh variabel x dan y yaitu a=5 dan b=3. Ketika terjadi pengisian nilai seperti ini, maka nilai x dan y tidak akan mengalami perubahan apapun. Namun, kadangkala diinginkan sebuah pertukaran nilai yang mempengaruhi nilai variabel pemberinya. Untuk melakukannya, diperlukan sebuah fungsi dengan pertukaran nilai secara acuan (*argumen passed by value*), seperti contoh berikut:

#include<iostream.h>

#include<conio.h>

void kali (int& a, int& b, int& c) {

a \*= 2;

b \*= 2;

c \*= 2; }

int main() {

int x=l, y=3, z= 7;

ka **1** i (x,y,z);

cout<< “x= ”<<x<<”, y= “<<y<<”, Z= “<<Z;

return 0;

}

Hasilnya adalah : **x=2, y=6, z=14**

Yang berbeda adalah cara pertukaran argumen menggunakan tanda ampersand (&), yang artinya fungsi melayani pengisian berdasarkan acuan (reference).

void kali (int& a, int& b, int& c)

**lx ly lz**

kali( x, y, z );

dengan pengisian nilai seperti ini, maka apabila terjadi perubahan nilai pada variable a, b dan c maka akan mempengaruhi nilai variable x, y dan z.

**Nilai default dalam argument**

Ketika mendeklarasikan fungsi, maka tiap-tiap parameter yang dideklarasikan dapat diberikan sebuah nilai default. Nilai default ini akan dipergunakan bila dalam pemanggilan fungsi, tidak diberikan nilai kepada parameter. Contoh:

#include<iostream.h>

#include<conio.h>

void bagi (int a, int b=2) {

int r;

r=a/b;

return (r); }

int main()

{

cout<<bagi(12);

cout<<endl;

cout<<bagi(20,4);

return 0;

}

Hasilnya : **6**

**5**

Dalam program di atas, terdapat dua kali pemanggilan terhadap fungsi bagi(),

bagi (12)

hanya memberikan sebuah nilai kepada fungsi bagi(), sedangkan dalam pendeklarasian fungsinya, bagi() memerlukan 2 buah parameter. Maka variable b akan otomatis bemilai 2 sesuai dengan nilai defaultnya. Sedangkan pada pemanggilan yang kedua,

bagi(20,4)

Variabel a dan b masing-masing diberikan nilai 20 dan 4. Untuk nilai default, dalam hal ini akan diabaikan.

**Fungsi Rekursif**

Fungsi rekursif adalah suatu fungsi yang memanggil dirinya sendiri, artinya fungsi tersebut dipanggil di dalam tubuh fungsi itu sendiri. Fungsi rekursif sangat berguna bila diimplementasikan untuk pekerjaan pengurutan data, atau menghitung nilai factorial suatu bilangan. Misalnya:

#include<iostream.h>

#include<conio.h>

long factorial (long a)

{

if (a>l)

return (a\* factorial (a-1));

else

return (1);

}

Int main()

{

long l;

cout<<”tuliskan bilangan : ” ;

cin>>l;

cout<<”!”<<!<<” = “<<factorial(1);

} return 0;

Hasil :

**Tuliskan bilangan : 9**

**!9 = 362880**

**Prototype Fungsi**

Sampai saat ini, setiap dideklarasikan sebuah fungsi baru diletakkan di atas fungsi main(). Namun terdapat pula alternative lain dalam pendeklarasian fungsi baru, yaitu dideklarasikan di bawah fungsi main() dengan menggunakan *prototype* *fungsi*.

Bagi compiler, informasi dalam prototype akan dipakai untuk memeriksa validitas parameter dalam pemanggilan fungsi. Keuntungan pemakaian prototype yaitu compiler akan melakukan konversi

seandainya antara tipe parameter dalam definisi dan parameter saat pemanggilan fungsi tidak sama, atau akan menunjukkan kesalahan kalau jumlah parameter dalam definisi dan saat pemanggilan berbeda. Sintak prototype:

tipe nama (argumen1, argumen2,…);

sama seperti pendeklarasian judul fungsi, kecuali:

1. tidak memiliki baris pernyataan (tubuh fungsi) yang ditandai dengan { dan }.
2. diakhiri dengan tanda ;
3. dalam pendeklarasian argumennya, cukup hanya dengan mendeklarasikan tipe datanya saja. Walaupun sangat dianjurkan untuk menyertakan argumen secara lengkap.

Contoh:

#include<iostream.h>

#include<conio.h>

void bagi (int a, int b);

int Tiain ()

{

} cout<<bagi(20,4);

return 0;

void bagi (int a, int b)

{

int r; r=a/b;

return (r);

}

untuk pendeklarasian prototype fungsi dapat berbentuk seperti berikut:

void bagi (int a, int b);

atau

void bagi (int , int );