FUNGSI

Sebuah fungsi berisi sejumlah pernyataan yang dikemas dalam sebuah nama. Nama ini selanjutnya dapat dipanggil beberapa kali di beberapa tempat dalam program.

Tujuan pembuatan fungsi adalah :
1. Memudahkan dalam mengembangkan program.
   Hal ini merupakan kunci dalam pembuatan program yang terstruktur dimana program dibagi menjadi beberapa modul yang kecil
2. Menghemat ukuran program.
   Manfaat ini terasakan kalau ada beberapa deretan instruksi yang sama digunakan pada beberapa tempat di dalam program.

Bentuk umum :

```
tipe nama_fungsi (deklarasi parameter)
{
  pernyataan;
  pernyataan;
}
```

- tipe
  Tipe nilai yang dihasilkan oleh fungsi. Jika tidak dinyatakan, hasil fungsi dianggap bilangan bulat (int)

- deklarasi parameter
  Daftar tipe dan nama variabel yang akan menerima nilai pada saat fungsi tersebut dipanggil. Setiap parameter dipisahkan oleh tanda koma. Jika fungsi tidak mempunyai parameter daftar ini akan kosong. Jadi hanya tanda kurung saja.

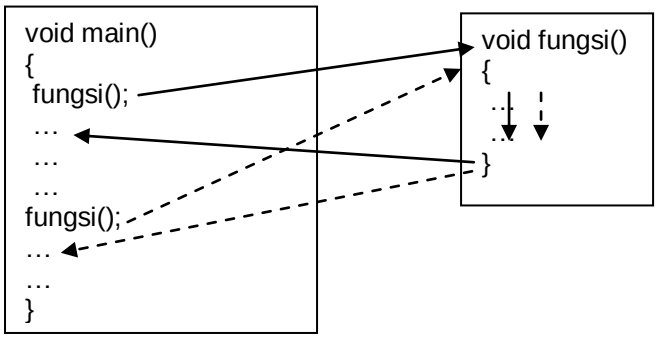
  Deklarasi parameter agak berbeda dengan deklarasi variabel. Pada deklarasi variabel, Anda dapat menyatakan sebuah tipe untuk beberapa variabel yang tipenya sama. Contoh : int a,b,c;

  Tetapi pada deklarasi parameter Anda harus menyatakan setiap tipe dari parameter.
  Bentuk umum :

  f (tipe nama_var1, tipe nama_var2, …, tipe nama_varn);

  Contoh : f (int a, int b, int c);

Bagan pemanggilan fungsi :



Ada dua cara untuk kembali ke program pemanggil :

1. Pada saat pernyataan terakhir dari function dijumpai (dijumpai tanda akhir fungsi "*}*")
   Contoh :

```
#include<iostream.h>
#include<conio.h>
void garis();
void main()
{
 clrscr();
 garis();
 cout <<"BIODATA" <<endl;
 garis();
 cout <<"NPM  : 10494570" <<endl;
 cout <<"Nama : Yudi Irawan Chandra" <<endl;
 garis();
}
void garis()
{
 cout <<"--------------------------" <<endl;
}
```

Prototipe fungsi garis()

   Hasil :

```
--------------------------
BIODATA
--------------------------
NPM  : 10494570
Nama : Yudi Irawan Chandra
--------------------------
```

   Pada contoh di atas, fungsi garis() digunakan untuk menampilkan karakter garis. Fungsi ini dipanggil tiga kali pada fungsi main(). Sebuah fungsi tidak dapat dipanggil kecuali sudah dideklarasikan. Manfaat dari pototipe fungsi adalah untuk menjamin tipe argumen yang dilewatkan pada pemanggilan fungsi benar-benar sesuai. Fungsi garis() tidak memiliki argumen dan nilai baliknya tidak ada (void).

2. Dengan menggunakan pernyataan return. Pernyataan return juga dapat dipakai tanpa menghasilkan sebuah nilai.
   Pernyataan return menyatakan dua hal :
   1. Return mengakhiri jalannya function dan kembali ke program pemanggil
   2. Menghasilkan sebuah nilai

   Contoh :

```
#include<iostream.h>
#include<conio.h>
void hai();
void main()
{
 clrscr();
 hai();
}
void hai()
{
 cout <<"Hai.. Apa kabar..?" <<endl;
 return;
```

```
 cout <<"Baik-baik saja kannn..?" <<endl;
}
```

Hasil :

```
Hai.. Apa kabar..?
```

DEFINISI FUNGSI

Setiap fungsi yang dipanggil di dalam program harus didefinisikan. Letaknya bisa dimana saja. Khusus untuk fungsi yang disediakan sistem, definisinya sebenarnya ada dalam pustaka, yang akan digabungkan dengan progam sewaktu proses *linking.*

Contoh :

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void garis();
long kuadrat(long x);
void main()
{
 clrscr();
 cout <<"Nilai X" <<setw(10) <<"Kuadrat" <<endl;
 garis();
 for(long bil=1; bil <=10; bil++)
  cout <<bil <<setw(13) <<kuadrat(bil) <<endl;
 garis();
}

void garis()
{
 cout <<"------------------" <<endl;
}

long kuadrat(long x)
{
 return(x*x);
}
```

Hasil :

```
Nilai X   Kuadrat
------------------
1             1
2             4
3             9
4            16
5            25
6            36
7            49
8            64
9            81
10          100
------------------
```

Pernyataan return didalam fungsi digunakan untuk memberikan nilai balik fungsi.

LINGKUP VARIABEL

Lingkup variabel menentukan keberadaan suatu variabel tertentu dalam fungsi,
Jenis-jenis lingkup variabel yaitu :

1. Variabel Otomatis
   Yaitu variabel yang didefinisikan di dalam suatu fungsi, berlaku sebagai variabel lokal bagi fungsi, artinya variabel tersebut hanya dikenal di dalam fungsi tempat variabel didefinisikan. Pendeklarasian variabel dapat ditulis dengan awalan *auto.*
   Contoh :

```
#include<iostream.h>
#include<conio.h>
void alpha();
void main()
{
 int x=22;
 double y=44.5;
 clrscr();
 cout <<"Nilai pada fungsi main  : ";
 cout <<"X= " <<x <<" dan Y= " <<y <<endl;
 alpha();
 cout <<"Nilai pada fungsi main  : ";
 cout <<"X= " <<x <<" dan Y= " <<y <<endl;
}
void alpha()
{
 auto int x=11;
 auto double y=22.5;
 cout <<"Nilai pada fungsi alpha : ";
 cout <<"X= " <<x <<" dan Y= " <<y <<endl;
}
```

   Hasil :

```
Nilai pada fungsi main  : X= 22 dan Y= 44.5
Nilai pada fungsi alpha : X= 11 dan Y= 22.5
Nilai pada fungsi main  : X= 22 dan Y= 44.5
```

2. Variabel Eksternal
   Adalah variabel yang didefinisikan di luar fungsi manapun. Variabel ini dikenal juga sebagai variabel *global*, karena variabel ini dikenal di semua fungsi. Pendeklarasian variabel dapat ditulis dengan awalan *extern.*
   Contoh :

```
#include<iostream.h>
#include<conio.h>
int x=1;
void tambah();
void main()
{
 clrscr();
 cout <<"Nilai awal X = " <<x <<endl;
 tambah();
 tambah();
 tambah();
 cout <<"Nilai X setelah fungsi = " <<x <<endl;
}
```

```
void tambah()
{
 extern x;
 x++;
}
```

Hasil :

```
Nilai awal X = 1
Nilai X setelah fungsi = 4
```

3. Variabel Statis

Baik variabel eksternal maupun otomatis dapat berkedudukan sebagai variabel statis. Variabel statis ditulis dengan awalan *static.* Suatu variabel statis mempunyai sifat :

a. Jika variabel lokal berdiri sebagai variabel statis, maka :
   - Variabel tetap hanya dapat diakses pada fungsi yang mendefinisikannya
   - Variabel tidak hilang saat eksekusi fungsi berakhir
   - Inisialisasi akan dilakukan sekali saja, jika tidak ada maka variabel diisi dengan nol

b. Jika variabel ekternal dijadikan variabel statis maka variabel ini dapat diakses oleh semua file yang didefinisikan.

Contoh :

```
#include<iostream.h>
#include<conio.h>
void tambah();
void main()
{
 int x=100;
 clrscr();
 cout <<"Nilai awal X di fungsi main = " <<x <<endl;
 tambah();
 tambah();
 tambah();
 cout <<"Nilai X di fungsi main = " <<x <<endl;
}
void tambah()
{
 static int x=44;
 x++;
 cout <<"Nilai X di fungsi tambah = " <<x <<endl;
}
```

Hasil :

```
Nilai awal X di fungsi main = 100
Nilai X di fungsi tambah = 45
Nilai X di fungsi tambah = 46
Nilai X di fungsi tambah = 47
Nilai X di fungsi main = 100
```

Jika perintah *static* pada fungsi tambah dihapus, maka akan menghasilkan :

```
Nilai awal X di fungsi main = 100
Nilai X di fungsi tambah = 45
Nilai X di fungsi tambah = 45
Nilai X di fungsi tambah = 45
Nilai X di fungsi main = 100
```

OPERATOR RESOLUSI LINGKUP

Operator ini digunakan untuk mengakses variabel yang didefinisikan di luar suatu fungsi dengan nama yang sama. Operator ini menggunakan dua buah tanda titik dua (::).

Contoh :

```
#include<iostream.h>
#include<conio.h>
int x=44;
void main()
{
 double x;
 clrscr();
 x=123.456;
 cout <<"Nilai x = "<<x <<" dan ::x = " <<::x <<endl;
 ::x=75;
 cout <<"Nilai x = "<<x <<" dan ::x = " <<::x <<endl;
 {
  char x='Y';
  ::x=11;
  cout <<"Nilai x = "<<x <<" dan ::x = " <<::x <<endl;
 }
}
```

Hasil :

```
Nilai x = 123.456 dan ::x = 44
Nilai x = 123.456 dan ::x = 75
Nilai x = Y dan ::x = 11
```

REFERENSI

Digunakan untuk memberikan nama alias dari variabel.

Bentuk deklarasi : *int &ref = nama_variabel;*

Contoh:

```
#include<iostream.h>
#include<conio.h>
void main()
{
 int i;
 int &r = i;
 clrscr();
 i=22;
 cout <<"Nilai i = "<<i <<" dan r = " <<r <<endl;

 i=44;
 cout <<"Nilai i = "<<i <<" dan r = " <<r <<endl;
}
```

Hasil :

```
Nilai i = 22 dan r = 22
Nilai i = 44 dan r = 44
```

Tampak bahwa pengubahan nilai terhadap i maupun r akan memberikan efek yang sama.

FUNGSI REKURSI

Suatu fungsi dapat memanggil fungsi yang merupakan dirinya sendiri. Rekursi jarang dipakai, diantaranya disebabkan :

- Membuat fungsi sulit untuk dipahami
- Hanya cocok untuk persoalan tertentu saja
- Memerlukan *stack* dengan ukuran yang lebih besar.

Contoh :

```
#include<iostream.h>
#include<conio.h>
long faktorial(int m);
void main()
{
 int x;
 clrscr();
 cout <<"Mencari Nilai Faktorial" <<endl;
 cout <<"Masukkan sebuah bilangan bulat positif : "; cin >>x;
 cout <<"Nilai faktorial dari " <<x <<" adalah " <<faktorial(x);
}

long faktorial(int m)
{
 if (m==0)
        return(1);
 else
        return(m*faktorial(m-1));
}
```

Hasil :

```
Mencari Nilai Faktorial
Masukkan sebuah bilangan bulat positif : 4
Nilai faktorial dari 4 adalah 24
```

Buat program dengan fungsi dan input untuk mencari nilai terbesar dari dua bilangan.

Jawab :

```
#include<iostream.h>
#include<conio.h>
int proses (int a, int b);
void main()
{
 int x, y;
 clrscr();
 cout <<"Mencari Nilai Terbesar" <<endl;
 cout <<"Masukkan nilai x : "; cin >>x;
 cout <<"Masukkan nilai y : "; cin >>y;
 cout <<"Nilai terbesar adalah : " << proses(x,y);
}
```

```
int proses (int a, int b)
{
 int hasil;
 hasil = (a > b) ?  a : b;
 return hasil;
}
```

Hasil :

```
Mencari Nilai Terbesar
Masukkan nilai x : 22
Masukkan nilai y : 44
Nilai terbesar adalah : 44
```

TUGAS

1. Buat program untuk menghitung gaji harian PT. XYZ dengan ketentuan :
   - Gaji pokok karyawan Rp. 2000/jam
   - Bila karyawan bekerja lebih dari 7 jam/hari maka kelebihannya dihitung lembur yang besarnya 1.5 dari gaji pokok
   - Untuk karyawan yang bekerja 8 jam/hari atau lebih akan mendapat tambahan uang makan sebesar Rp. 3500
   - Karyawan yang bekerja 9 jam/hari atau lebih akan mendapat uang transport lembur sebesar Rp. 4000

   Program ini akan terdiri dari 5 buah fungsi : main(), pokok(), lembur(), makan() dan jasa()
   Input : NIP, Nama, Jumlah jam kerja
   Output : NIP, Nama, Gaji pokok, Lembur, Uang makan, Transport lembur