BAB 6. FUNGSI

Suatu fungsi adalah suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya dipisahkan dari bagian program yang menggunakannya.

Tujuan penggunaan fungsi :

1. Program menjadi terstruktur
2. Dapat mengurangi pengulangan kode program.

Fungsi dapat diimplementasikan dalam tiga bentuk :

1. Pendeklarasian fungsi sebagai prototype fungsi.
2. Pendefinisian fungsi.
3. pemanggilan fungsi dari program lain.

1. Prototipe Fungsi

Prototipe fungsi merupakan model dari sebuah fungsi yang berguna untuk mendeklarasikan cirri-ciri fungsi, meliputi :

1. nama fungsi
2. tipe nilai balik fungsi.
3. jumlah dan tipe argument.

Bentuk umum : **jenis_data** nama fungsi (**jenis_data, jenis_data**,…);
Prototype fungsi tidak perlu dituliskan bila suatu fungsi terletak di atas fungsi yang memanggilnya.

Contoh :
```
long kuadrat(long);
int maks(int a, int b);
void garis( );
long nilai(int,int);
```

2. Pendefinisian Fungsi

Mendefinisikan fungsi adalah menyusun perintah-perintah yang akan dilakukan fungsi itu sesuai dengan tugas yang akan dilakukannya.

Bentuk umum :

```
[jenis data] nama_fungsi ([jenis_data arg1], jenis data ag2],…)
{
    [deklarasi variable];
    [kode program];
    [pernyataan return];
 }
```

3. Pemanggilan Fungsi

Fungsi-fungsi yang telah didefinisikan dapat dipanggil dari bagian-bagian fungsi lain. Pada saat sebuah fungsi dipanggil, alur eksekusi program akan berpindah ke fungsi yang dipanggil itu. Setelah selesai mengeksekusi fungsi, kendali program akan

dikembalikan kepada fungsi yang memanggil, dan alur eksekusi program dilanjtukan pada pernyataan setelah pemanggilan fungsi tersebut.

Bentuk umum :

```
Nama_fungsi( [jenis_data arg1], [jenis data ag2],…);
Nama fungsi( );
```

Contoh fungsi tanpa nilai balik :

```
#include <stdio.h>
void main( )
{       void factor( );              /*prototype fungsi*/
        int angka;
        printf("berikan sebuah bilangan bulat: ");
        scanf("%d", &angka);
        factor(angka);               /*pemanggilan fungsi*/
}
void factor(int angka)   /*fungsi*/
{       int f,g;
        f = 2;
        while (angka %f != 0)       /*selama angka mod f tdk sama dengan 0*/
        f++;
        g = angka/f; printf("KPK = %d, PBB = %d\n",f ,g);
        }
```

output : berikan sebuah bilangan bulat: 27
KPK = 3, PBB = 9

Pengiriman Argumen

Pengiriman data dari satu fungsi ke fungsi lain, dapat dilakukan dengan dua cara:

1. mengirimkan salinan data, disebut juga pengiriman dengan nilai (passing by value), dan
2. mengirimkan alamat data, disebut juga pengiriman dengan alamat (passing by reference atau passing by address).

Contoh :

/*1. pengiriman argument dengan nilai dan fungsi dengan nilai balik*/

```
#include<stdio.h>
void main ( )
{
        int fungsi(int x);          /*prototipe fungsi*/
        int x = 5, y =4;
        printf("Dalam fungsi main( ): x = %d, y = %d\n", x, y);
        y += fungsi(x);             /*memanggil fungsi &mengirim nilai x */
        printf("Dalam fungsi main( ): x = %d, y = %d\n", x, y);
}
```

```
int fungsi(int x)              /*pendefinisian fungsi*/
{
        int y;
        y = ++x; printf("Dalam fungsi : x = %d, y = %d\n", x,y);
        return y;              // kembali ke prog utama&mengirim nilai y ke y+ = fungsi(x)
}
```

output : Dalam fungsi main( ) : x = ,y =
Dalam fungsi : x = , y =
Dalam fungsi main( ) : x = ,y =

return merupakan perintah yang berfungsi untuk mengembalikan nilai dari fungsi.

Bentuk Umum: return [(ekspresi)]

Contoh : return; return 0;
return (x*x);
return (x > y ? x : y)

Lingkup Variabel

- Variabel internal (lokal) adalah variabel yang dideklarasikan dalam fungsi. Hanya dikenal oleh fungsi tempat variabel dideklarasikan.
- Variabel eksternal (global) merupakan variabel yang dideklarasikan di luar fungsi. Dapat diakses oleh semua fungsi.
- Variabel Statis dapat berupa variabel internal (lokal) maupun variabel eksternal (global). Variabel statis tidak akan hilang sekeluarnya dari fungsi (nilai pada variabel akan tetap).
- Variabel register adalah variabel yang nilainya disimpan dalam register dan bukan dalam memori RAM.

Fungsi Rekursi

Fungsi C dapat dipakai secara rekursi maksudnya adalah suatu fungsi dapat memanggil dirinya sendiri.

Syarat-syarat :

1. Harus ada kasus penghentian.
2. Tiap-tiap sturuktur pemanggilan harus lebih sederhana daripada srtruktur pemanggilan sebelumnnya.

Proses rekursi

1. Semua variable otomatis yang terdapat di dalam fungsi rekursif akan memiliki lokasi memory sendiri pada setiap pemanggilannya.
2. Pernyataan-pernyataan yang terdapat sebelum rekursif akan dieksekusi dengan urutan yang sama.
3. Pernyatan-pernyataan yang terdapat setelah pemanggilan rekursif akan dikerjakan dengan urutan yang terbalik terhadap pemanggilan.
4. Kode program untuk fungsi rekursif tidak duplikat, walaupun fungsi tersebut dipanggil beberapa kali.

Contoh 1:

```
#include <stdio.h>
void main( )
{
void hitung(int) ;
hitung(1) ;                /*pemanggilan fungsi dan mengirim nilai 1 */     }

void hitung (int n)
{     printf("%d angka, ",n);
      if (n<5)
            hitung(n+1);  // fungsi rekursi dengan menaikkan nilai n selama n < dari 5
      printf("%d angka ,",n);   // pemanggilan rekursif dikerjakan dengan urutan yang
                                    terbalik terhadap pemanggilan, yaitu n dari 5 sampai 1
}
```

output : 1 angka, 2 angka, 3 angka, 4 angka, 5 angka, 5 angka, 4 angka, 3 angka, 2 angka, 1 angka

Contoh2 :

```
#include <stdio.h>
#include<conio.h>
void main( )
{     int m,n;
      long pangkat (int, int);          /*prototipe fungsi*/
      printf("input bilangan dan eksponen : ");
      scanf("%d %d, &m, &n);
      printf("%d pangkat  %d = %ld\n", m, n, pangkat(m,n) ; getch( );

long pangkat(int m,int n)              /*fungsi*/
{
      static long p = 1;               /*p berupa variable statis*/
      if (n = = 0) return 1;           /*syarat penghentian*/
      else
      p = m* pangkat(m, n-1);          /*pemanggilan rekursif*/
      return p;   }
```

output : input bilangan dan eksponen : 3 4
3 pangkat 4 = 81

Jenis-jenis rekusif :

1. Rekursi akhir (tail recusive), apabila pemanggilan dilakukan pada bagian akhir fungsi.
2. Rekursi tunggal (singly rekursi), apabila tiap-tiap pemanggilan paling banyak membentuk satu pemanggilan lagi.
3. rekursi ganda (doubly recursive), apabila tiap-tiap pemanggilan membentuk lebih dari satu pemanggilan.

Contoh variable local, global dan static:

```
#include <stdio.h>
int A = 10;     // A variable global
void ingat();  //deklarasi fungsi atau prototype fungsi
void main( )
{       int m = 10;  //m variable lokal
        printf("main() : m = %d dan A = %d\n", m,A);
        ingat();  ingat();  ingat();  }

void ingat()
{   static int m = 77 ;  m++ ;
        printf("ingat() : m = %d dan A = %d" , m,A);
}
```

output :

```
main() : m = 10 dan A = 10
ingat() : m = 78 dan A = 10
ingat() : m = 79 dan A = 10
```