BAB VII
FUNCTION

IF

Tujuan

1. Menjelaskan tentang prinsip dasar fungsi
2. Menjelaskan tentang parameter formal dan parameter aktual

Dasar Teori

Fungsi adalah suatu bagian dari program yang dirancang untuk melaksanakan tugas tertentu dan letaknya dipisahkan dari program yang menggunakannya. Elemen utama dari program bahasa C berupa fungsi-fungsi, dalam hal ini program dari bagasa C dibentuk dari kumpulan fungsi pustaka (standar) dan fungsi yang dibuat sendiri oleh pemrogram. Fungsi banyak digunakan pada program C dengan tujuan :

a. Program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan. Dengan memisahkan langkah-langkah detail kesatu atau lebih fungsi-fungsi, maka fungsi utama (`main()`) menjadi lebih pendek, jelas dan mudah dimengerti.
b. Dapat mengurangi pengulangan (duplikasi) kode. Langkah-langkah program yang sama dan dipakai berulang-ulang diprogram dapat dituliskan sekali saja secara terpisah dalam bentuk fungsi-fungsi. Selanjutnya bagian program yang membutuhkan langkah-langkah ini tidak perlu selalu menuliskannya, tetapi cukup memanggil fungsi-fungsi tersebut.

Dasar Fungsi

Fungsi standar C yang mengemban tugas khusus contohnya adalah :

a. `printf` (), yaitu untuk menampilkan informasi atau data ke layar
b. `scanf` (), yaitu untuk membaca kode tombol yang diinputkan

Pada umumnya fungsi memerlukan nilai masukan atau parameter yang disebut sebagai **argumen**. Nilai masukan ini akan diolah oleh fungsi. Hasil akhir fungsi berupa sebuah nilai (disebut sebagai ***return value*** atau nilai keluaran fungsi). Oleh karena itu, fungsi sering digambarkan sebagai “black box” seperti ditunjukkan pada gambar dibawah ini :



Penggamabaran sebagai kotak gelap diantaranya menjelaskan bahwa bagian dalam fungsibersifat probadi bagi fungsi. Tidak ada suatu pernyataan diluar fungsi yang bisa mengakses bagian dalam fungsi, selain melalui parameter (atau varianel eksternal). Bentuk umum dari definisi sebuah fungsi adalah sebagai berikut :

```
tipe-keluaran-fungsi namafungsi (deklarasi argumen)
{
  tubuh fungsi
}
```

Keterangan :

- `tipe-keluaran-fungsi`, dapat berupa salah satu tipe data C, misalnya `char` atau `int`. Jika penentu tipe tidak disebutkan, maka dianggap bertipe int (secara default).
- `tubuh fungsi` berisi deklarasi variabel (jika ada) dan statemen-statemen yang akan melakukan tugas yang akan diberikan kepada fungsi yang bersangkutan. Tubuh fungsi ini ditulis didalam tanda kurung kurawal buka dan kurung kurawal tutup.

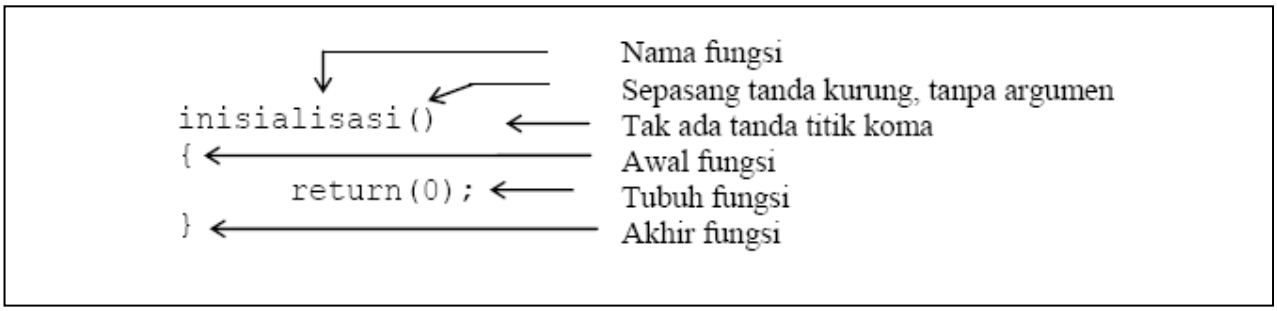
Sebuah fungsi yang sederhana bisa saja tidak mengandung parameter sama sekali dan tentu saja untuk keadaan ini deklarasi parameter juga tidak ada. Contoh :

```
int inisialisasi()
{
     return(0);
}

inisialisasi()
{
     return(0);
}
```

Pada fungsi diatas :

- tipe keluaran fungsi tidak disebutkan, berarti keluaran fungsi bertipe int.
- Inisialisasi adalah nama fungsi
- Tanda () sesudah nama fungsi menyatakan bahwa fungsi tak memiliki parameter
- Tanda { dan } adalah awal dan akhir fungsi
- `return(0)` merupakan sebuah pernyataan dalam tubuh fungsi.



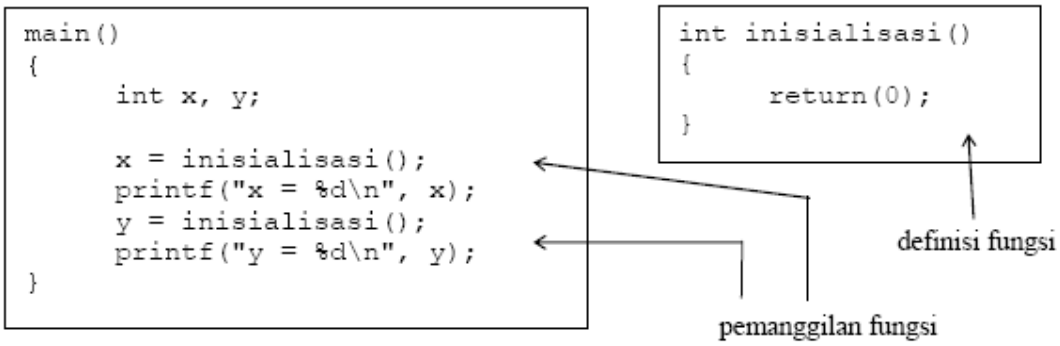
Memberikan Nilai Keluaran Fungsi

Suatu fungsi dibuat untuk maksud menyelesaikan tugas tertentu. Suatu fungsi dapat hanya melakukan suatu tugas saja tanpa memberikan suatu hasil keluiaran atau melakukan suatu tugas dan kemudian memberikan hasil keluaran. Fungsi yang hanya melakukan suatu tugas saja tanpa memberikan hasil keluaran misalnya adalah fungsi untuk menampilkan hasil dilayar.

Dalam tubuh fungsi, pernyataan yang digunakan untuk memberikan nilai keluaran fungsi berupa `return`. Sebagai contoh, pada fungsi inisialisasi() diatas terdapat pernyataan

```
return (0);
```

merupakan pernyataan untuk memberikan nilai keluaran fungsi berupa nol.



Hasil Eksekusi :

x = 0

y = 0

Program diatas sekaligus menjelaskan bahwa suatu fungsi cukup didefinisikan satu kali tetatpi bisa digunakan beberapa kali. Pada keadaan semacam ini, seandainya tubuh fungsi mengandung banyak pernyataan, maka pemakaian fungsi dapat menghindari duplikasi kode dan tentu saja menghemat penulisan program maupun kode dalam memori.

Misalnya pada saat pernyataan

```
x = inisialisasi ();
```

dijalankan, mula-mula eksekusi akan diarahkan ke fungsi inisialisasi (), selanjutnya suatu nilai keluaran (hasil fungsi) akhir fungsi diberikan ke x. Proses yang serupa, dilakukan untuk pernyataan

```
y = inisialisasi ();
```

Bagi suatu fungsi, jika suatu pernyataan return dieksekusi, maka eksekusi terhadap fungsi akan berakhir dan nilai pada parameter return akan menjadi keluaran fungsi. Untuk fungsi yang tidak memiliki pernyataan return, tanda } pada bagian akhir fungsi akan menyatakan akhir eksekusi fungsi.

Dibawah ini diberikan contoh sebuah fungsi yang mengandung dua buah pernyataan return. Fungsi digunakan untuk memperoleh nilai minimum diantara 2 buah nilai yang menjadi parameternya.

```
int minimum(int x, int y)
      {
            if (x < y)
            return(x);
      else
            return(y);
      }
```

Pada fungsi diatas terdapat dua buah parameter berupa x dan y. Oleh karena itu fungsi juga mengandung bagian untuk mendeklarasikan parameter, yang menyatakan x dan y bertipe int. Adapun penentuan nilai keluaran fungsi dilakukan pada tubuh fungsi, berupa pernyataan

```
if (x < y)
      return(x);
else
      return(y);
```

yang menyatakan :

- jika x < y maka nilai keluaran fungsi adalah sebesar nilai x
- untuk keadaan lainnya (x >=y) maka keluaran fungsi adalah sebesar y

Fungsi Dengan Keluaran Bukan Integer

Untuk fungsi yang mempunyai keluaran bertipe bukan integer, maka fungsi haruslah didefinisikan dengan diawali tipe keluaran fungsinya (ditulis didepan nama fungsi). Sebagai contoh untuk menghasilkan nilai terkecil diantara dua buah nilai real, maka definisinya berupa :

```
float minimum(float x, float y)
{
     if (x < y)
     return(x);
     else
     return(y);
}
```

Perhatikan, didepan nama minimum diberikan tipe keluaran fungsi berupa float. Seluruh parameter sendiri juga didefinisikan dengan tipe float.

Khusus untuk fungsi yang dirancang tanpa memberikan nilai keluaran (melainkan hanya menjalankan suatu tugas khusus) biasa didefinisikan dengan diawali kata kunci `void` (didepan nama fungsi).

Prototipe Fungsi

Prototipe fungsi digunakan untuk menjelaskan kepada kompiler mengenai :

- Tipe keluaran fungsi
- Jumlah parameter
- Tipe dari masing-masing parameter

Bagi kompiler, informasi dalam prototipe akan dipakai untuk memeriksa keabsahan (validitas) parameter dalam pemanggilan fungsi. Salah satu keuntungannya adalah, kompiler akan melakukan konversi seandainya antara tipe parameter dalam fungsi dan parameter saat pemanggilan fungsi tidak sama, atau akan menunjukkan kesalahan bila jumlah parameter dalam definisi dan saar oemanggilan berbeda.

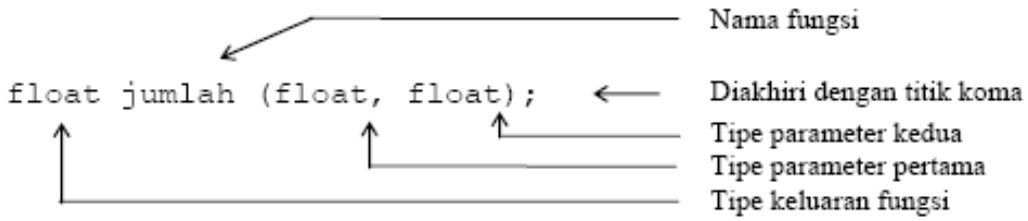
Contoh prototipe fungsi :

```
float jumlah (float x, float y);
```

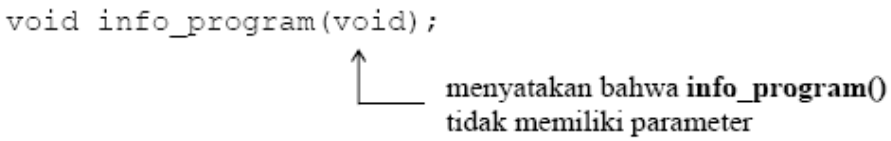
atau

```
float jumlah (float, float);
```

Penjelasannya adalah sebagai berikut :



Untuk fungsi yang tidak memiliki argumen (contoh program void.c), maka deklarasinya adalah :

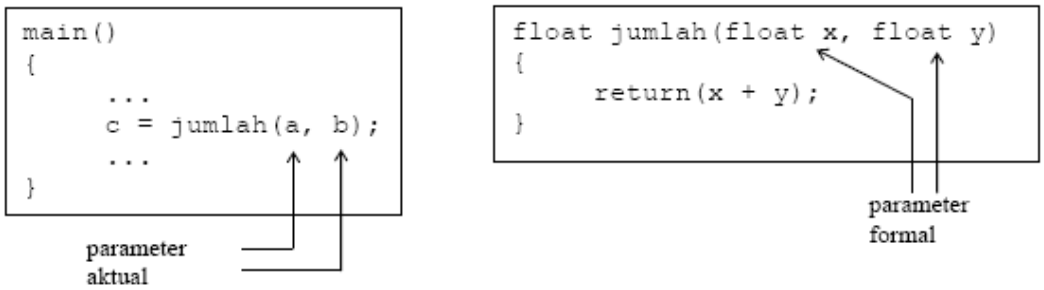


Catatan :

- Untuk fungsi-fungsi pustaka, prototipe dari fungsi-fungsi berada di file-file judulnya (`header file`). Misalnya fungsi pustaka `printf()` dan `scanf()` prototipenya berada pada file dengan nama `stdio.h`
- Untuk fungsi pustaka pencantuman pada prototipe fungsi dapat dilakukan dengan menggunakan `#include`

Parameter Formal dan Parameter Aktual

Parameter formal adalah variabel yang ada pada daftar parameter dalam definisi fungsi. Pada coontoh program diatas misalnya, maka dalam fungsi `jumlah()` variabel x dan y dinamakan sebagai parameter formal. Adapun parameter aktual adalah parameter (tidak selalu berupa variabel) yang dipakai dalam pemanggilan fungsi.

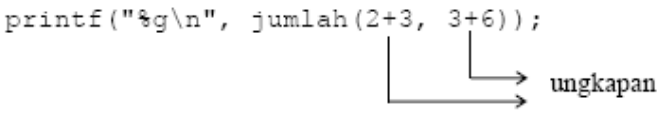


Pada pernyataan :

x = jumlah (a,b)

y = jumlah (20.1, 45.6)

a dan b merupakan parameter aktual dari fungsi `jumlah()` dalam hal ini parameter berupa variabel. Demikian juga 20.1 dan 45.6 adalah parameter aktual, dalam hal ini berupa konstanta. Bahkan bisa juga parameter aktual berupa ungkapan yang melibatkan operator, misalnya :



Pemanggilan dengan Nilai dan Pemanggilan dengan Referensi

Pemanggilan dengan nilai (*call by value*) merupakan cara yang dipakai untuk seluruh fungsi buatan yang telah dibahas pada praktikum sebelumnya. Pada menanggilan dengan nilai, nilai dari parameter aktual akan disalin ke parameter formal. Dengan cara ini nilai parameter aktual tidak bisa dirubah sekalipun nilai parameter formal berubah. Untuk lebih jelasnya lihat pada fungsi `tukar()` pada contoh berikut ini.

```
/* File program : tukar1.c
Untuk melihat pengaruh pemanggilan nilai pada fungsi untuk penukaran
dua data */
#include <stdio.h>
void tukar (int, int);
main()
{
int a = 88, b = 77;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
tukar(a,b);
printf("\nNilai setelah pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
}

void tukar(int x, int y)
{
int z;
z = x;
x = y;
y = z;
printf("\nNilai di akhir fungsi tukar()\n");
printf("x = %d y = %d\n", x, y);
}
```

```
Nilai sebelum pemanggilan fungsi
a = 88 b = 77

Nilai di akhir fungsi tukar()
x = 77 y = 88

Nilai setelah pemanggilan fungsi
a = 88 b = 77
```

Tampak bahwa sekeluarnya dari pemanggilan fungsi `tukar ()`, variabel a dan b (yang dilewatkan ke fungsi tukar() tidak berubah, walaupun pada fungsi `tukar()` telah terjadi penukaran antara parameter x dan y. Mengapa hal ini bisa terjadi ? sebab x hanyalah salinan dari a dan y adalah salinan dari b. Pada saat pemanggilan fungsi, maka

- x bernilai 88 (nilai a)
- y bernilai 77 (nilai b)

Pemanggilan dengan referensi (call by reference) merupakan upaya untuk melewatkan alamat dari suatu variabel kedalam fungsi. Cara ini dapat dipakai untuk mengubah isi suatu variabel diluar fungsi dengan pelaksanaan pengubahan dilakukan didalam fungsi. Sebagai contoh perhatikan program tukar2.c yang merupakan modifikasi dari tukar1.c. Perubahan yang pertama terletak dalam definisi fungsi, yang kini berupa

```
void tukar(int *px, int *py)
{

      int z;
      z = *px;
      *px = *py;
      *py = z;
      printf("\nNilai di akhir fungsi tukar()\n");
      printf("x = %d y = %d\n", *px, *py);
}
```

Adapun perubahan dalam parameter aktualnya menjadi :

```
tukar(&a,&b); //alamat a dan alamat b
```

Dalam deklarasi parameter

```
int  *px, int *py
```

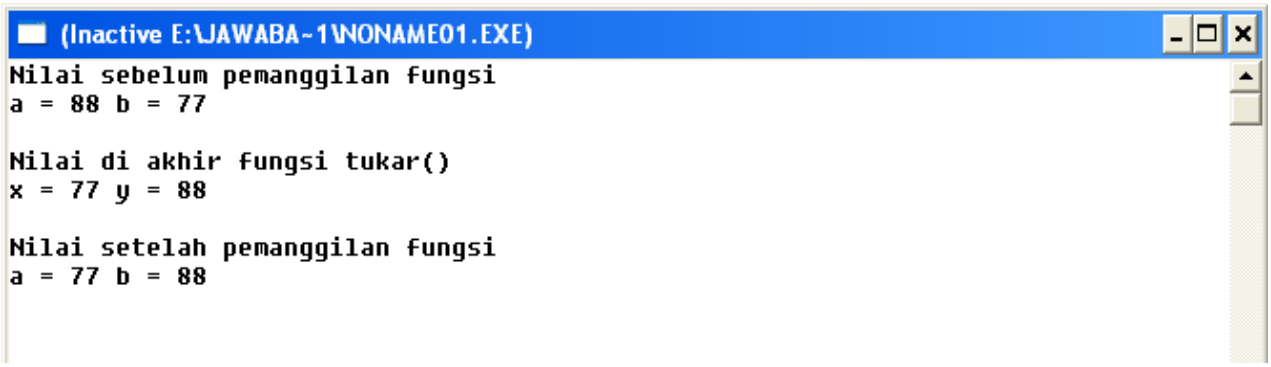
menyatakan bahwa px dan py adalah suatu variabel pointer. Yang dimaksudkan sebagai variabel pointer adalah suatu variabel yang menunjuk ke variabel lain. Lebih jelasnya variabel pointer berisi alamat dari variabel lain.

Adapun pada pemanggilan fungsi, `&a` dan `&b` masing-masing berarti “alamat a” dan “alamat b”. Dengan pemanggilan seperti ini, hubungan antara variabel pointer `px` dan `py` dengan variabel a dan b adalah seperti ditunjukkan dibawah ini. Dalam hal ini, `px` dikatakan menunjuk variabel a dan `py` menunjuk variabel b.

```
/* File program : tukar2.c
Untuk melihat pengaruh pemanggilan nilai pada fungsi untuk penukaran
dua data */
#include <stdio.h>
void tukar (int *px, int *py); //prototype fungsi
main()
{
   int a= 88,b = 77;
   printf("Nilai sebelum pemanggilan fungsi\n");
   printf("a = %d b = %d\n", a, b);
   tukar(&a,&b); //alamat a dan alamat b
   printf("\nNilai setelah pemanggilan fungsi\n");
```

```
    printf("a = %d b = %d\n", a, b);
}

void tukar(int *px, int *py)
{
    int z;
    z = *px;
    *px = *py;
    *py = z;
    printf("\nNilai di akhir fungsi tukar()\n");
    printf("x = %d y = %d\n", *px, *py);
}
```



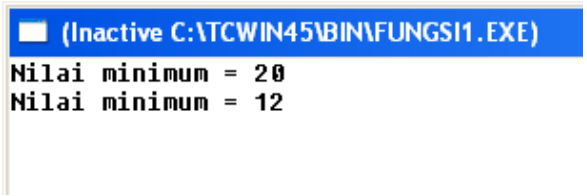
Kumpulan Contoh

```
//contoh fungsi dengan nilai akhir
#include <stdio.h>
#include <conio.h>

int minimum(int x,int y);

main ()
{
      int hasil;
      int a=20;
      int b=44;
      hasil=minimum(a,b);
      printf("Nilai minimum = %d\n",hasil);
      printf("Nilai minimum = %d\n",minimum(12,15));
}

int minimum(int x,int y)
{
      if(x<y)
            return (x);
      else
            return (y);
}
```



//contoh fungsi bertipe bukan integer (misal :float)

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

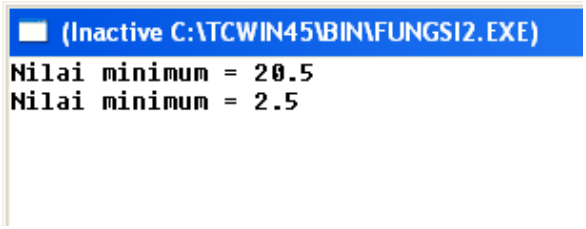
float minimum (float x, float y);

main()
{
      float hasil;
      float a=20.5, b=44.5;

      hasil=minimum(a,b);
      printf("Nilai minimum = %g\n",hasil);
      printf("Nilai minimum = %g\n",minimum(3.5,2.5));

}

float minimum(float x, float y)
{
      if (x<y)
            return (x);
      else
            return (y);
}
```



//contoh fungsi bertipe void (tanpa nilai keluaran)

```
#include <stdio.h>

void sekilas_info();

main()
{
      printf("Panggil sekilas info.. \n");
      sekilas_info();
}

void sekilas_info()
{
      printf("Ini program percobaan\n");
      printf("Membuat fungsi tanpa nilai keluaran\n");
}
```

//contoh pemakaian prototipe fungsi

```
#include <stdio.h>

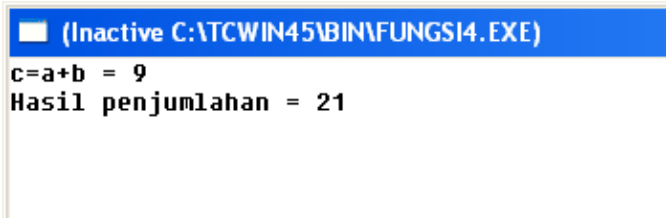
float jumlah (float,float); //prototipe fungsi

main()
{
      int a=6;
      int b=3;
      float c;

      c=jumlah(a,b);
```

```
      printf("c=a+b = %g\n",c);
      printf("Hasil penjumlahan = %g\n",jumlah (20.1, 0.9));
}

float jumlah (float x,float y)
{
      return(x+y);
}
```



/*contoh program menggunakan parameter formal dan aktual
menentukan apakah bilangan yang diinputkan ganjil atau genap */

```
#include <stdio.h>
int ganjil(int);

main()
{
      int a, hasil;
      printf("Masukan sembarang bilangan bulat : ");
      scanf("%d", &a);
      hasil=ganjil(a);

      if(hasil==0)
            printf("Bilangan tersebut genap\n");
      else if(hasil==1)
            printf("Bilangan tersebut ganjil\n");
}

int ganjil (int x)
{
      if(x%2 == 0)
            return (0);
      else
            return (1);
}
```

