BAB V
FUNGSI

Program komputer yang dibuat untuk menjawab permasalahan umumnya berukuran sangat besar. Pengalaman telah menujukkan bahwa cara terbaik untuk mengembangkan dan menangani program besar adalah menyusunnya dari potongan-potongan program yang berukuran kecil-kecil (atau disebut **modul**) yang lebih mudah untuk ditangani dibanding dengan program yang terdiri dari banyak sekali baris. Teknik ini disebut *conquer*.

Modul program di dalam C disebut **fungsi** (*function*). Program C ditulis dengan mengombinasikan fungsi baru yang ditulis oleh pemrogram (*disebut programmer-defined function*) dengan fungsi yang telah tersedia di dalam pustakanya (disebut *standard library*). Sebagai contoh, fungsi printf, scanf, pow, dan lain-lain adalah fungsi baku yang telah tersedia dan siap untuk digunakan.

DEFINISI FUNGSI

Fungsi menuntun pemrogram untuk modularitas program. Semua variabel yang dideklarasikan di dalam fungsi bersifat lokal. Setiap fungsi memiliki nol atau lebih parameter, yang semuanya juga bersifat variabel lokal. Bentuk dari definisi suatu fungsi adalah:

```
tipe-kembali  nama-fungsi (daftar-parameter)
{
    deklarasi;
    pernyataan;
}
```

'nama-fungsi' adalah identifikasi yang valid yang diberikan oleh pemrogram, sedangkan 'tipe-kembali' adalah tipe data yang dikembalikan oleh hasil fungsi tersebut. Kata **void** digunakan sebagai tipe-kembali untuk menunjukkan bahwa fungsi tidak mengembalikan suatu nilai apapun. Suatu fungsi yang tidak menuliskan tipe-kembali dianggap oleh kompilator mempunyai tipe-kembali **int**.

Daftar parameter merupakan suatu daftar deklarasi dari parameter yang diterima oleh fungsi pada saat dipanggil, yang masing-masing dipisahkan oleh tanda koma (,). Jika fungsi tidak menerima suatu nilai apapun, maka daftar-parameter diisi **void**.

Deklarasi dan pernyataan di dalam fungsi ditulis di antara tanda kurung kurawal { … } yang semuanya disebut dengan tubuh fungsi (*function body*). Di dalam tubuh fungsi terdapat pernyataan yang menunjukkan nilai yang dihasilkan oleh fungsi, yaitu:

```
return ekspresi;
```


Tabel berikut menyajikan daftar sebagian fungsi baku matematika yang sudah tersedia di dalam pustaka C dan siap untuk digunakan.

Fungsi	Uraian	Contoh
sqrt(x)	akar kuadrat dari x ($\sqrt{x}$)	sqrt(9.0) = 3
exp(x)	fungsi eksponen e^x	exp(1.0) = 2.718282
log(x)	logaritme alami dari x (dasar e)	log(2.718282) = 1.0
log10(x)	logaritme dari x (dasar 10)	log10(100.0) = 2.0
fabs(x)	nilai mutlak dari x	fabs(-10.0) = 10.0
ceil(x)	pembulatan ke bilangan bulat terkecil yang tidak lebih kecil dari x	ceil(9.2) = 10.0 ceil(-9.8) = -9.0
floor(x)	pembulatan ke bilangan bulat terbesar yang tidak lebih besar dari x	floor(9.2) = 9.0 floor(-9.8) = -10.0
pow(x,y)	x^y	pow(2,7) = 128.0
fmod(x,y)	pembagian x/y sebagai tipe float	fmod(13.657,2.333)=1.992
sin(x)	sinus dari x (dalam radian)	sin(0.0) = 0.0
cos(x)	cos dari x (dalam radian)	cos(0.0) = 1.0
tan(x)	tangent dari x (dalam radian)	tan(0.0) = 0.0

Salah satu hal yang penting dalam kaitannya dengan fungsi dalam program C adalah prototipe fungsi, yang terdiri dari nama-fungsi, tipe-kembali, dan daftar-parameter. Prototipe fungsi ini dituliskan untuk mengatur struktur program agar implementasi dari suatu fungsi dapat dicantumkan di bagian bawah dari fungsi **main**, sehingga hanya prototipenya yang dituliskan sebelumnya. Perhatikan contoh program pada halaman 9 Bab III sebelumnya.

FILE HEADER

Setiap pustaka baku mempunyai file *header* yang mengandung prototipe fungsi untuk semua fungsi yang ada di dalam pustaka dan definisi dari tipe-tipe data maupun konstanta yang diperlukan oleh fungsi tersebut. Berikut dicantumkan daftar file *header* yang dapat di-*include* ke dalam program.

<assert.h> berisi makro dan informasi untuk diagnosis yang membantu pemeriksaan kesalahan proses dalam program.

<ctype.h> mengandung prototipe fungsi untuk menguji properti dari karakter, dan dapat digunakan untuk mengubah karakter huruf kecil ke besar.

<errno.h> mendefinisikan makro untuk melaporkan kondisi kesalahan.

<float.h> berisi batasan di dalam sistem bilangan bertipe float.

<limits.h> berisi batasan sistem integral.

<locale.h> berisi prototipe fungsi dan informasi lainnya yang dapat memodifikasi sistem data pada saat program diproses.

<math.h> berisi prototipe fungsi untuk pustaka matematika.

<setjmp.h> berisi prototipe fungsi yang menuntun urutan pemanggilan fungsi.

<signal.h> berisi prototipe fungsi dan makro untuk menangani kondisi yang bermacam-macam saat program diproses.

<stdarg.h> mendefinisikan makro untuk menangani daftar argumen fungsi yang nilai dan tipenya tidak diketahui.

<stddef.h> berisi definisi umum dari tipe yang digunakan C untuk membentuk perhitungan tertentu.

<stdio.h> berisi prototipe fungsi untuk fungsi pustaka baku input/output.

<stdlib.h> berisi prorotipe fungsi untuk konversi nilai ke text atau sebaliknya, alokasi memori, bilangan acak, dan utilitas lainnya.

<string.h> berisi prorotipe fungsi untuk pemrosesan string.

<time.h> berisi prototipe fungsi dan tipe untuk memanipulasi data waktu (jam dan tanggal).

PEMANGGILAN FUNGSI : BY VALUE / BY REFERENCE

Ada dua cara pemrosesan fungsi dalam berbagai bahasa pemrograman, yaitu pemanggilan nilai (*call by value*) dan pemanggilan referensi (*call by reference*), yang keduanya mempunyai karakteristik berbeda. Metode pemanggilan nilai terjadi jika yang dikirimkan ke fungsi oleh pemanggilnya adalah berupa nilai, dan tidak berpengaruh pada variabel yang tertulis di dalam argumen pemanggilnya. Sedangkan metode pemanggilan referensi terjadi jika yang dikirimkan ke fungsi berupa referensi (alamat) dan diterima oleh fungsi sehingga perubahan terhadap variabel penerima ini akan mempengaruhi variabel argumen yang memanggilnya. Perhatikan contoh cuplikan program berikut:

```
…..
…..
void swap(int x, int y) {
     int t;
     t = x; x = y; y = t;
}
….
….
a = 5; b = 10;
swap(a,b);
```


Fungsi swap pada contoh tersebut bermaksud untuk menukar nilai dari dua variabel, sehingga dengan pemanggilan swab(a,b) diharapkan nilai a menjadi 10, dan b menjadi 5. Namun karena program C mempunyai metode pemanggilan nilai, maka setelah proses tersebut, nilai a dan b akan tetap, yaitu masing-masing 5 dan 10.

Agar perubahan x dan y berakibat pada variabel a dan b, maka metode pemanggilan fungsi tersebut haruslah pemanggilan referensi. Dalam C, proses seperti ini dapat dilakukan dengan menambahkan operator alamat di dalam daftar parameter fungsinya dan operator referensi (tak langsung) di dalam argumen pemanggilnya seperti contoh berikut:

```
…..
…..
void swap(int *x, int *y) {
     int t;
     t = *x; *x = *y; *y = t;
}
….
….
a = 5; b = 10;
swap(&a,&b);
….
```

Untuk kondisi ini, setelah proses pemanggilan fungsi swap, maka nilai a dan b masing-masing saling bertukar menjadi 10 dan 5.

FUNGSI REKURSIF

Fungsi yang telah dibahas sebelumnya dipanggil dari bagian lain di luar tubuh fungsi yang bersangkutan. Fungsi rekursif adalah suatu fungsi yang memanggil dirinya sendiri, artinya fungsi tersebut dipanggil di dalam tubuh fungsi itu sendiri.

Fungsi faktorial, yang menghitung nilai faktorial dari suatu bilangan bulat positif, merupakan pokok bahasan yang memudahkan pemahaman fungsi rekursif. Berikut adalah fungsi faktorial yang diselesaikan dengan cara biasa:

```
int faktorial(int n) {
     int counter, hasil = 1;
     for (counter = n ; counter >= 1 ; counter--)
          hasil *= counter;
     return hasil;
}
```

Fungsi tersebut menunjukkan bahwa nilai faktorial dihitung menggunakan looping sehingga melakukan proses sebagai berikut:

```
hasil = 1;
hasil = hasil * n;       →  artinya hasil = n;
hasil = hasil * (n-1)  →  artinya hasil = n x (n-1);
```

Demikian seterusnya sampai n bernilai 1, atau jika dituliskan sekaligus menjadi:

faktorial = n! = n x (n-1) x (n-2) …… x 1;

Fungsi ini dapat dituliskan dalam bentuk:

faktorial (n) = n! = n x (n-1) !;

yang menunjukkan sifat rekursif dari suatu fungsi, yaitu (n-1)!. Oleh karena itu, fungsi faktorial yang telah ditulis dalam program C sebelumnya, dapat ditulis kembali dalam bentuk rekursif sebagai berikut:

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

Contoh lain adalah menghitung jumlah dari suatu deret fibonacci, dimana deret tersebut didefinisikan sebagai:

```
fibonacci(0) = 0
fibonacci(1) = 1
fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)
```

sehingga fungsi dalam program C dapat dibuat sebagai berikut:

```
long fibonacci(long n) {
    if (n == 0 || n == 1)
        return n;
        else
        return fibonacci(n-1) + fibonacci(n-2);
}
```

LATIHAN 5

1. Ubahlah fungsi fibonacci di atas menjadi fungsi yang bersifat tidak rekursif.
2. Buat fungsi untuk menentukan apakah suatu bilangan bulat bersifat ganjil atau genap. Jika genap maka fungsi menghasilkan nilai 1, dan 0 untuk selainnya.
3. Buat fungsi bukan rekursif untuk menentukan apakah suatu bilangan bulat bersifat prima. Jika prima maka fungsi menghasilkan nilai 1, dan 0 untuk selainnya.
4. Buatlah fungsi pada soal nomor 3 menjadi rekursif.
5. Buatlah fungsi menjumlahkan bilangan 1,2,3, ….., n secara rekursif.