PRAKTIKUM 11
FUNCTION

11.1 TUJUAN PRAKTIKUM

Tujuan Umum

Mahasiswa dapat memahami konsep modular menggunakan function pada bahasa pemrograman C

Tujuan Khusus

Mahasiswa dapat :

1. Memahami Konsep Fungsi dan Prosedur
2. Memahami Struktur Umum Fungsi dan Prosedur
3. Memahami Deklarasi Fungsi dan Prosedur

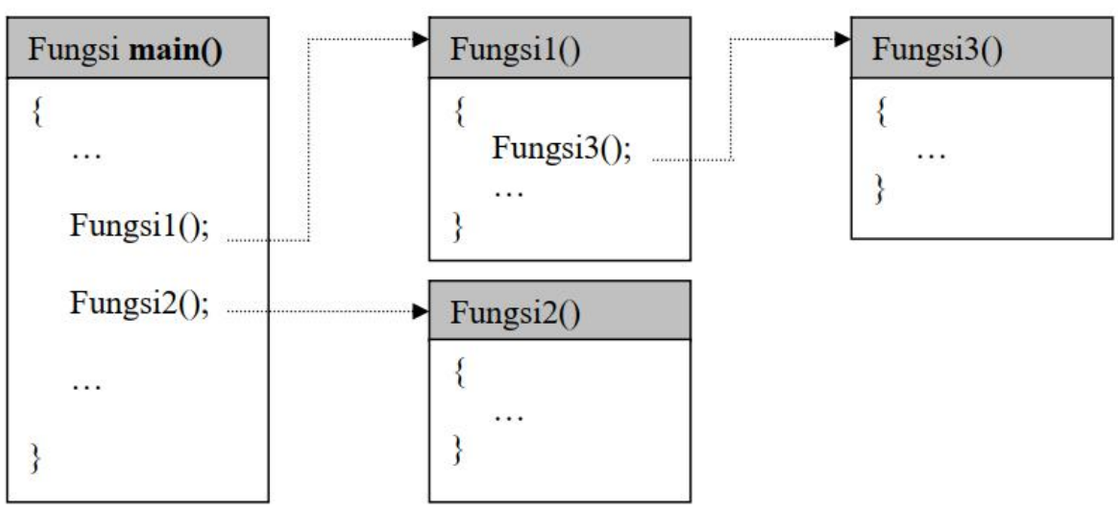
11.2 TEORI SINGKAT

Dalam bahasa C, sebuah program terdiri atas fungsi-fungsi, baik yang didefinisikan secara langsung di dalam program maupun yang disimpan di dalam file lain (misalnya file header). Satu fungsi yang pasti terdapat dalam program yang ditulis menggunakan bahasa C adalah fungsi main(). Fungsi tersebut merupakan fungsi utama dan merupakan fungsi yang akan dieksekusi pertama kali.

Menurut definisinya, fungsi adalah suatu blok program yang digunakan untuk melakukan proses-proses tertentu. Sebuah fungsi dibutuhkan untuk menjadikan program yang akan kita buat menjadi lebih modular dan mudah untuk dipahami alurnya. Dengan adanya fungsi, maka kita dapat mengurangi duplikasi kode program sehingga performa dari program yang kita buat pun akan meningkat.

Dalam bahasa C, fungsi terbagi menjadi dua macam, yaitu fungsi yang mengembalikan nilai (return value) dan fungsi yang tidak mengembalikan nilai. Fungsi yang tidak mengembalikan nilai tersebut dinamakan dengan void function.

Sebelum melangkah lebih jauh ke dalam pembentukan fungsi, di sini akan diterangkan bagaimana kompilator C membaca fungsi-fungsi yang didefiniskan di dalam program secara berurutan sesuai dengan waktu pemanggilannya. Untuk itu, perhatikanlah gambar ilustrasi berikut.



Mula-mula fungsi main() akan dieksekusi oleh kompilator. Oleh karena fungsi main() tersebut memanggil Fungsi1(), maka kompilator akan mengeksekusi Fungsi1(). Dalam Fungsi1() juga terdapat pemanggilan Fungsi3(), maka kompilator akan mengeksekusi Fungsi3() dan kembali lagi mengeksekusi baris selanjutnya yang terdapat pada Fungsi1() (jika ada). Setelah Fungsi1() selesai dieksekusi, kompilator akan kembali mengeksekusi baris berikutnya pada fungsi main(), yaitu dengan mengeksekusi kode-kode yang terdapat pada Fungsi2().

Setelah selesai, maka kompilator akan melanjutkan pengeksekusian kode pada baris-baris selanjutnya dalam fungsi main(). Apabila ternyata dalam fungsi main() tersebut kembali terdapat pemanggilan fungsi lain, maka kompilator akan meloncat ke fungsi lain tersebut, begitu seterusnya sampai semua baris kode dalam fungsi main() selesai dieksekusi.

11.3 PELAKSANAAN PRAKTIKUM

11.3.1 Fungsi Tanpa Nilai Balik

Pada umumnya fungsi tanpa nilai balik (return value) ini digunakan untuk melakukan proses-proses yang tidak menghasilkan nilai, seperti melakukan pengulangan, proses pengesetan nilai ataupun yang lainnya. Dalam bahasa C, fungsi semacam ini tipe kembaliannya akan diisi dengan nilai void. Berikut ini format pendefinisian fungsi tanpa nilai balik.

```
void nama_fungsi(parameter1, parameter2,…) {
  Statemen_yang_akan_dieksekusi;
  ...
}
```

Praktikum 11.1 Fungsi tanpa nilai balik

```
1  #include <stdio.h>
2  /* Mendefinisikan sebuah fungsi dengan nama Tulis10Kali */
3  void Tulis10Kali(void)
4  {
5      int j;
6      for (j=0; j<10; j++) {
7          printf("Saya sedang belajar bahasa C\n");
8      }
9  }
10
11 main()
12 {
13     Tulis10Kali(); /* Memanggil fungsi Tulis10Kali() */
14 }
15
```

11.3.2 Fungsi Dengan Nilai Balik

Berbeda dengan fungsi di atas yang hanya mengandung proses tanpa adanya nilai kembalian, di sini kita akan membahas mengenai fungsi yang digunakan untuk melakukan proses-proses yang berhubungan dengan nilai. Adapun cara pendefinisiannya adalah dengan menuliskan tipe data dari

nilai yang akan dikembalikan di depan nama fungsi, berikut ini bentuk umum dari pendefinisian fungsi dengan nilai balik di dalam bahasa C. Berikut ini format pendefinisian fungsi dengan nilai balik.

```
tipe_data nama_fungsi(parameter1, parameter2,...) {
  Statemen_yang_akan_dieksekusi;
  ...
  return nilai_balik;
}
```

Praktikum 11.2 Fungsi dengan nilai balik

```c
#include <stdio.h>
int HitungLuasBujurSangkar(int sisi) {
    int L;
    L = sisi * sisi;
    return L; /* mengembalikan nilai yang didapat dari hasil proses */
}
main() {
    int S, Luas;
    S = 10;
    Luas = HitungLuasBujurSangkar(S);
    printf("Luas bujur sangkar dengan sisi %d adalah %d", S, Luas);
}
```

11.3.3 Fungsi dengan parameter

Parameter adalah suatu variabel yang berfungsi untuk menampung nilai yang akan dikirimkan ke dalam fungsi. Dengan adanya parameter, sebuah fungsi dapat bersifat dinamis. Parameter itu sendiri terbagi menjadi dua macam, yaitu parameter formal dan parameter aktual. Parameter formal adalah parameter yang terdapat pada pendefinisian fungsi, sedangkan parameter aktual adalah parameter yang terdapat pada saat pemanggilan fungsi. Untuk lebih memahaminya, perhatikan contoh pendefinisian fungsi di bawah ini.

Praktikum 11.3 Fungsi dengan parameter

```c
#include <stdio.h>
#define PI 3.14159

double HitungKelilingLingkaran(int radius) {
    double K;
    K = 2 * PI * radius;
    return K;
}

main() {
    int R;
    printf("Masukkan nilai jari-jari lingkaran : ");
    scanf("%d", &R);
    double Keliling = HitungKelilingLingkaran(R);
    printf("Keliling lingkaran dengan jari-jari %d : %f", R, Keliling);
}
```

* Manakah yang disebut sebagai parameter formal dan parameter aktual?

11.3.4 Melewatkan parameter berdasarkan nilai (Pass By Value)

Terdapat dua buah cara untuk melewatkan parameter ke dalam sebuah fungsi, yaitu dengan cara melewatkan berdasarkan nilainya (pass by value) dan berdasarkan alamatnya (pass by reference). Namun pada bagian ini hanya akan dibahas mengenai pelewatan parameter berdasarkan nilai saja, sedangkan untuk pelewatan parameter berdasarkan alamat akan kita bahas pada sub bab berikutnya.

Pada pelewatan parameter berdasarkan nilai, terjadi proses penyalinan (copy) nilai dari parameter formal ke parameter aktual. Hal ini akan meyebabkan nilai variabel yang dihasilkan oleh proses di dalam fungsi tidak akan berpengaruh terhadap nilai variabel yang terdapat di luar fungsi. Untuk lebih memahaminya, perhatikan contoh program berikut dimana di dalamnya terdapat sebuah parameter yang dilewatkan dengan cara pass by value.

Praktikum 11.4 Pass by Value

```c
#include <stdio.h>

void TambahSatu(int X) {
    X++;
    /* Menampilkan nilai yang terdapat di dalam fungsi */
    printf("Nilai di dalam fungsi : %d\n", X);
}

main() {
    int Bilangan;
    printf("Masukkan sebuah bilangan bulat : ");
    scanf("%d", &Bilangan);
    /* Menampilkan nilai awal */
    printf("\nNilai awal : %d\n", Bilangan);
    /* Memanggil fungsi TambahSatu */
    TambahSatu(Bilangan);
    /* Menampilkan nilai akhir */
    printf("Nilai akhir : %d\n", Bilangan);
}
```

11.3.5 Melewatkan parameter berdasarkan alamat(Pass By Reference)

Di sini, parameter yang dilewatkan ke dalam fungsi bukanlah berupa nilai, melainkan suatu alamat memori. Pada saat kita melewatkan parameter berdasarkan alamat, terjadi proses referensial antara variabel yang terdapat pada parameter formal dengan variabel yang terdapat parameter aktual. Hal tersebut menyebabkan kedua variabel tersebut akan berada pada satu alamat di memori yang sama sehingga apabila terdapat perubahan nilai terhadap salah satu dari variabel tersebut, maka nilai variabel satunya juga akan ikut berubah. Untuk melakukan hal itu, parameter fungsi tersebut harus kita jadikan sebagai pointer (untuk informasi lebih detil mengenai pointer, lihat bab 7 - Pointer).

Agar dapat lebih memahami materi ini, di sini kita akan menuliskan kembali kasus di atas ke dalam sebuah program. Namun, sekarang kita akan melakukannya dengan menggunakan cara pass by reference. Adapun sintak programnya adalah sebagai berikut.

Praktikum 12.5 Pass by Reference

```c
#include <stdio.h>

void TambahSatu(int *X) {
    (*X)++;
    /* Menampilkan nilai yang terdapat di dalam fungsi */
    printf("Nilai di dalam fungsi : %d\n", *X);
}

main() {
    int Bilangan;
    printf("Masukkan sebuah bilangan bulat : ");
    scanf("%d", &Bilangan);
    /* Menampilkan nilai awal */
    printf("\nNilai awal : %d\n", Bilangan);
    /* Memanggil fungsi TambahSatu dengan mengirimkan
    alamat variabel Bilangan */
    TambahSatu(&Bilangan);
    printf("Nilai akhir : %d\n", Bilangan);
}
```

11.3.6 Membuat prototype fungsi

Secara default, pendefinisian fungsi-fungsi di dalam program akan dilakukan sebelum fungsi main(). Namun bahasa C telah mengizinkan kita untuk dapat mendefinisikan fungsi-fungsi lain tersebut setelah melakukan penulisan fungsi main() asalkan kita menuliskan prototipe dari fungsi-fungsi tersebut sebelum fungsi main().

Praktikum 11.6 Membuat prototype fungsi (konversi suhu celcius ke fahrenheit)

```c
#include <stdio.h>
/* Membuat prototipe dari fungsi CelciusKeFahrenheit() */
float CelciusKeFahrenheit(float suhu);

main() {
    float C, F;
    printf("Masukkan suhu yang akan dikonversi : ");
    scanf("%f", &C);

    /* Memanggil fungsi CelciusKeFahrenheit()
    dan menampungnya ke variabel F*/
    F = CelciusKeFahrenheit(C);

    /* Menampilkan nilai hasil konversi */
    printf("%.2f C = %.2f F", C, F);
}

/* Implementasi fungsi CelciusKeFahrenheit */
float CelciusKeFahrenheit(float suhu) {
    float hasil;
    hasil = ((9 * suhu) / 5) + 32;
    return hasil;
}
```

11.3.7 Rekursi

Rekursi adalah proses pemanggilan fungsi oleh dirinya sendiri secara berulang. Istilah ‘rekursi’ sebenarnya berasal dari bahasa Latin ‘recursus’, yang berarti ‘menjalankan ke belakang’. Rekursi digunakan untuk penyederhanaan algoritma dari suatu proses sehingga program yang dihasilkan menjadi lebih efesien. Pada bagian ini kita akan mempelajarinya langsung melalui contoh-contoh program.

Praktikum 11.7 Fungsi rekursi (Menentukan bilangan faktorial)

```c
#include <stdio.h>
/* Mendefinisikan fungsi untuk menghitung nilai faktorial */
int Faktorial(int N) {
    if (N == 0) {
        return 1;
    }else {
        return N * Faktorial(N-1);
    }
}

main() {
    int bilangan;
    printf("Masukkan bilangan yang akan dihitung : ");
    scanf("%d", &bilangan);
    printf("%d! = %d", bilangan, Faktorial(bilangan));
}
```

Latihan

Lanjutkan contoh pada praktikum 11.2, tambahkan fungsi-fungsi untuk menghitung Luas dan keliling bangun datar lainnya yaitu:

a) Lingkaran
b) Segitiga
c) Trapesium
d) Jajar genjang
e) Belah ketupat