

Algoritma & Pemrograman #7

by Sufajar Butsianto, M. Kom



Pembahasan Soal TTS

Modular Programming

- ❑ Program pendek dan simple = mudah dihandle.
- ❑ Program besar, banyak dan kompleks = tidak mudah dihandle.
- ❑ Kesulitan:
 - sulit mencari dan mengingat variabel-variabel yang sudah dideklarasikan
 - sulit melakukan dokumentasi
 - sulit mencari kesalahan program
 - sulit melihat efisiensi algoritma
 - code program kadang ditulis berulang-ulang padahal mengerjakan suatu hal yang sama

Modular Programming (2)

- ❑ Merupakan paradigma pemrograman yang pertama kali diperkenalkan oleh Information & Systems Institute, Inc. pada the National Symposium on Modular Programming pada 1968.
- ❑ Salah satu tokoh modular programming adalah Larry Constantine
- ❑ Pemrograman Modular adalah suatu teknik pemrograman di mana program yang biasanya cukup besar dibagi-bagi menjadi beberapa bagian program yang lebih kecil
- ❑ Keuntungan:
 - Program lebih pendek
 - Mudah dibaca dan dimengerti
 - Mudah didokumentasi
 - Mengurangi kesalahan dan mudah mencari kesalahan
 - ❑ Kesalahan yang terjadi bersifat "lokal"

Modular programming pada C

- ❑ Bahasa C sangat mendukung modular programming
- ❑ Sejak awal bahasa C sudah membagi program-programnya menjadi modul-modul (bagian-bagian)
- ❑ Modul pada bahasa C dikenal dengan nama fungsi (function)
- ❑ Bahasa C terdiri dari fungsi-fungsi, baik yang langsung dideklarasikan dalam program ataupun dipisah di dalam header file.
- ❑ Fungsi yang selalu ada pada program C adalah fungsi **main**

Function

- ❑ Fungsi/function adalah suatu kumpulan instruksi/perintah/program yang dikelompokkan menjadi satu, letaknya terpisah dari program yang menggunakan fungsi tersebut, memiliki nama tertentu yang unik, dan digunakan untuk mengerjakan suatu tujuan tertentu.
- ❑ Dalam bahasa pemrograman lain fungsi dapat disebut sebagai subrutin (basic, VB) atau procedure (pascal, Delphi)

Keuntungan Fungsi

- ❑ Dapat melakukan pendekatan *top-down* dan *divide-and-conquer*:
 - Top-down: penelusuran program mudah
 - Divide-and-conquer: program besar dapat dipisah menjadi program-program kecil.
- ❑ Kode program menjadi lebih pendek, mudah dibaca, dan mudah dipahami
- ❑ Program dapat dikerjakan oleh beberapa orang sehingga program cepat selesai dengan koordinasi yang mudah.
- ❑ Mudah dalam mencari kesalahan-kesalahan karena alur logika jelas dan sederhana
- ❑ Kesalahan dapat dilokalisasi dalam suatu modul tertentu saja.
- ❑ Modifikasi program dapat dilakukan pada suatu modul tertentu saja tanpa mengganggu program keseluruhan

Keuntungan Fungsi (2)

- ❑ Fungsi – fungsi menjadikan program mempunyai struktur yang jelas.
 - Dengan memisahkan langkah – langkah detail ke satu atau lebih fungsi – fungsi, maka fungsi utama (**main**) akan menjadi lebih pendek, jelas dan mudah dimengerti.
- ❑ Fungsi –fungsi digunakan untuk menghindari penulisan program yang sama yang ditulis secara berulang – ulang. Langkah – langkah tersebut dapat dituliskan sekali saja secara terpisah dalam bentuk fungsi. Selanjutnya bagian program yang membutuhkan langkah – langkah ini tidak perlu selalu menuliskannya, tidak cukup memanggil fungsi tersebut.
- ❑ Mempermudah dokumentasi.
- ❑ *Reusability*: Suatu fungsi dapat digunakan kembali oleh program atau fungsi lain

Sifat-sifat fungsi

- Nilai **fan-in** tinggi, artinya semakin sering suatu modul dipanggil oleh pengguna semakin tinggi nilai fan-in
- Nilai **Fan-out** rendah, artinya semakin spesifik fungsi suatu modul akan semakin rendah nilai fan-out
- Memiliki **Self-contained** tinggi: artinya kemampuan untuk memenuhi kebutuhannya sendiri

Kategori fungsi dalam C

□ **Standard Library Function**

- Yaitu fungsi-fungsi yang telah disediakan oleh C dalam file-file header atau librarynya.
- Misalnya: `clrscr()`, `printf()`, `getch()`
- Untuk function ini kita harus mendeklarasikan terlebih dahulu library yang akan digunakan, yaitu dengan menggunakan preproesor direktif.
 - Misalnya: `#include <conio.h>`

□ **Programmer-Defined Function**

- Adalah function yang dibuat oleh programmer sendiri.
- Function ini memiliki nama tertentu yang unik dalam program, letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalam suatu library buatan programmer itu sendiri yang kemudian juga di-include-kan jika ingin menggunakannya.

Perancangan Fungsi

Dalam membuat fungsi, perlu diperhatikan:

- ❑ Data yang diperlukan sebagai inputan
- ❑ Informasi apa yang harus diberikan oleh fungsi yang dibuat ke pemanggilnya
- ❑ Algoritma apa yang harus digunakan untuk mengolah data menjadi informasi

Contoh fungsi

Contoh:

```
int GetMax(int nFirst, int nLast)
{
    int nReturn;
    if (nFirst>nLast)
        nReturn=nFirst;
    else
        nReturn=nLast;
    return nReturn;
}
```

Contoh fungsi

```
#include<stdio.h>
int HITUNG(int A, int B);
```

```
void main()
{
  int A,B,T;
  A=5; B=2;T=0;
  T = HITUNG(A,B);
  printf("\n %d", T);
}
```

```
int HITUNG(int A, int B)
{  int T;
  A = A * 2;
  B = B * 2;
  T= A+B;
  return(T);
}
```

Bagian ini yang disebut : main program atau program induk atau disebut juga Fungsi Induk atau Function main Oleh Bahasa C diberi nama `main()`

Dalam program induk ada instruksi yang memanggil atau menCALL Function lain, baik fungsi yang kita buat sendiri, maupun fungsi pustaka yang disediakan oleh C/C++

Bagian ini memuat fungsi. Fungsi ini fungsi yang kita buat sendiri
Fungsi ini mempunyai :
Nama : HITUNG
Tipe : int

Dalam contoh ini Fungsi HITUNG ditulis dibawah atau sesudah fungsi main Sebuah Fungsi dapat juga ditulis diatas atau sebelum Fungsi main()

Contoh fungsi

Contoh-02.

```
#include<stdio.h>
void main()
{
    printf("Jakarta");
}
```

Program ini
tidak menggunakan Funtion lain
selain main function

Tercetak: Jakarta

Struktur Fungsi

- Deklarasi function (function prototype/declaration)

Terdiri dari:

- Judul fungsi
- Tipe data yang akan dikembalikan/void
- Tidak ada kode implementasi function tersebut

Bentuk umum:

tipe_data|void nama_fungsi([arguman 1, argument 2,....]);

Deklarasi fungsi (2)

- ❑ Deklarasi fungsi diakhiri dengan titik koma
- ❑ Tipe_data dapat berupa segala tipe data yang dikenal C ataupun tipe data buatan, namun tipe data dapat juga tidak ada dan digantikan dengan void yang berarti fungsi tersebut tidak mengembalikan nilai apapun
- ❑ Nama fungsi adalah nama yang unik
- ❑ Argumen dapat ada atau tidak (opsional) yang digunakan untuk menerima argumen/parameter. Antar argumen-argumen dipisahkan dengan menggunakan tanda koma.

Deklarasi Fungsi (3)

- ❑ Suatu fungsi perlu dideklarasikan sebelum digunakan.
- ❑ Untuk alasan dokumentasi program yang baik, sebaiknya semua fungsi yang digunakan dideklarasikan terlebih dahulu
- ❑ Deklarasi fungsi ditulis sebelum fungsi tersebut digunakan

Struktur Fungsi

- Tubuh Function/Definisi Function (Function Definition)

Terdiri dari:

- function prototype yang disertai dengan kode implementasi dari function tersebut, yang berisikan statemen/instruksi yang akan melakukan tugas sesuai dengan tujuan dibuatnya fungsi tersebut.

Bentuk Umum Definisi Fungsi

```
tipe_data/void nama_fungsi([arguman 1, argument 2,...]) //funciton prototype
{
    //bagian ini merupakan tubuh fungsi.
    [variabel_lokal;]

    [Statement_1;]
    [Statement_2;]
    ...
    [Statement_3;]
    [return (variabel)];
}
```

Definisi Fungsi (2)

- ❑ Tubuh fungsi dapat berisi segala perintah yang dikenal oleh C, pada dasarnya tubuh fungsi sama dengan membuat program seperti biasa.
- ❑ Return bersifat opsional, adalah keyword pengembalian nilai dari fungsi ke luar fungsi, return wajib jika fungsi tersebut mengembalikan nilai berupa tipe data tertentu, sedangkan return tidak wajib jika fungsi tersebut bersifat void.

Contoh Deklarasi dan Definisi Fungsi

Contoh-03.

```
#include<stdio.h>
void CETAK();
void main()
{
    CETAK();
}
```

```
void CETAK()
{
    printf("Jakarta");
}
```

Tercetak: Jakarta

Fungsi CETAK di-**DEKLARASI** -kan lebih dulu, sebelum fungsi **main()**. Perhatikan pakai tanda : ';' (titik koma) Kalau tidak pakai tanda ';' dianggap men-**DEFINISI**-kan fungsi

Instruksi meng**CALL** Fungsi **CETAK**

Tulisan ini disebut : Men **DEFINISIKAN** Fungsi

Fungsi yang dibuat sendiri
Nama : **CETAK**
Tipe : void (artinya tanpa tipe)

Dalam fungsi ini ada instruksi untuk mencetak perkataan "Jakarta"

Contoh Deklarasi dan Definisi Fungsi (2)

```
#include <stdio.h>
double Absolut(double X);    /*deklarsi fungsi Absolut */

void main()
{
    float Nilai;
    Nilai = -123,45;
    printf("%7,2f nilai mutlaknya adalah %7,2f\n",Nilai,
    Absolut(Nilai));
}

/*----- Fungsi untuk memutlakkan nilai negatip- -----*/

double Absolut(double X)    /* definisi fungsi */
{
    if(X<0) X = -X;
    return(X);
}
```

Jika program ini dijalankan, akan didapatkan hasil :

-123,45 nilai mutlaknya adalah 123,45

Contoh Definisi Fungsi

```
#include <stdio.h>
/*----- Fungsi untuk memutlakan nilai negatif -----*/
double Absolut(double X)    /* definisi fungsi */
{
    if(X<0) X= -X;
    return(X);
}

void main( )
{
    float Nilai;

    Nilai = -123,45;
    printf("%7,2f nilai mutlaknya adalah %7,2f\n",Absolut(Nilai));
}
```

Kapan menggunakan Deklarasi dan Definisi Fungsi?

- Karena prinsip kerja program C sekuensial, maka
 - Jika bagian dari program yang menggunakan fungsi diletakkan sebelum definisi dari fungsi, maka deklarasi dari fungsi diperlukan.
 - Akan tetapi jika bagian dari program yang menggunakan fungsi terletak setelah definisi dari fungsi, maka deklarasi dari fungsi dapat tidak dituliskan.

Jenis fungsi dalam C

- Fungsi yang tidak mengembalikan nilai (void)
- Fungsi yang mengembalikan nilai (non-void)

Fungsi Void

- ❑ Fungsi yang void sering disebut juga prosedur
- ❑ Disebut void karena fungsi tersebut tidak mengembalikan suatu nilai keluaran yang didapat dari hasil proses fungsi tersebut.
- ❑ Ciri: tidak adanya keyword return.
- ❑ Ciri: tidak adanya tipe data di dalam deklarasi fungsi.
- ❑ Ciri: menggunakan keyword void.
- ❑ Tidak dapat langsung ditampilkan hasilnya
- ❑ Tidak memiliki nilai kembalian fungsi
- ❑ Contoh?

Fungsi non-void

- ❑ Fungsi non-void disebut juga function
- ❑ Disebut non-void karena mengembalikan nilai kembalian yang berasal dari keluaran hasil proses function tersebut
- ❑ Ciri: ada keyword return
- ❑ Ciri: ada tipe data yang mengawali deklarasi fungsi
- ❑ Ciri: tidak ada keyword void
- ❑ Memiliki nilai kembalian
- ❑ Dapat dianalogikan sebagai suatu variabel yang memiliki tipe data tertentu sehingga dapat langsung ditampilkan hasilnya.
- ❑ Contoh?

Contoh fungsi void dan non-void

- Void:

```
void tampilkan_jml(int a,int b){
    int jml;
    jml = a + b;
    printf("%d",jml);
}
```
- Non-void:

```
int jumlah(int a,int b){
    int jml;
    jml = a + b;
    return jml;
}
```

Keyword void

- Keyword void juga digunakan jika suatu function tidak mengandung suatu parameter apapun.

```
void print_error(void) {  
    printf("Error : unexpected error occurred!");  
}
```

Contoh fungsi Faktorial

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

int faktorial (int N);    /*prototype fungsi faktorial*/

void main()
{
    int N;
    int fak;
    printf ("berapa faktorial ? ");scanf ("%d",&N);
    fak = faktorial(N);
    printf ("%d faktorial = %d\n",N,fak);
    getch();
}

/*-----fungsi untuk menghitung nilai N faktorial -----*/
int faktorial(int N)    /*definisi fungsi*/
{
    int I;
    int F=1;
    if(N<=0)
    return(0);
    for(I=2;I<=N;I++) F *= I;
    return(F);
}
```



The main Function

- ❑ function **main()** dibutuhkan agar program C dapat dieksekusi!
- ❑ Tanpa function main, program C dapat dicompile tapi tidak dapat dieksekusi (harus dengan flag parameter `-c`, jika di UNIX)
- ❑ Pada saat program C dijalankan, maka compiler C pertama kali akan mencari function `main()` dan melaksanakan instruksi-instruksi yang ada di sana.
- ❑ Function main, sering dideklarasikan dalam 2 bentuk:
 - **int main()**
 - **void main()**

int main()

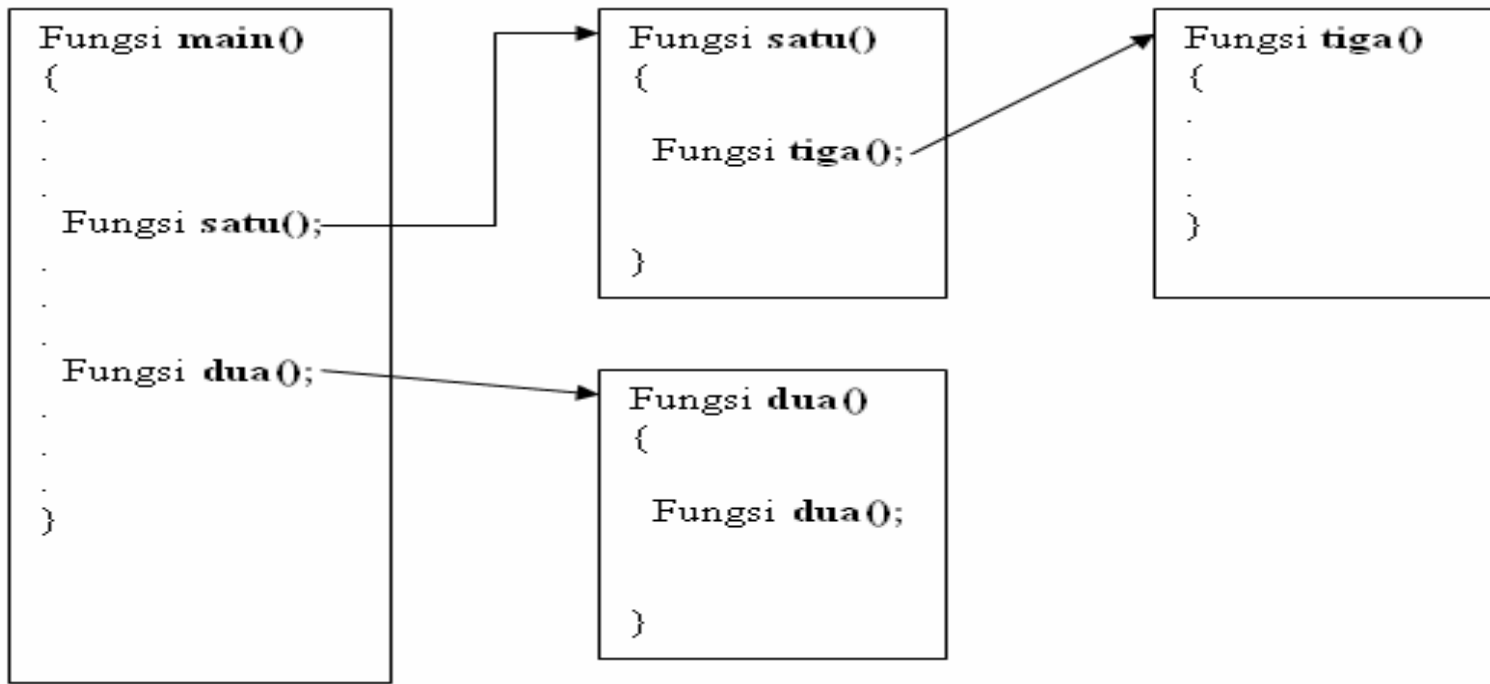
- ❑ Berarti di dalam function main tersebut harus terdapat keyword return di bagian akhir fungsi dan mengembalikan nilai bertipe data int,
- ❑ Mengapa hasil return harus bertipe int juga? karena tipe data yang mendahului fungsi main() diatas dideklarasikan int
- ❑ Tujuan nilai kembalian berupa integer adalah untuk mengetahui status eksekusi program.
 - jika "terminated successfully" (EXIT_SUCCESS) maka, akan dikembalikan status 0,
 - sedangkan jika "terminated unsuccessfully" (EXIT_FAILURE) akan dikembalikan nilai status tidak 0, biasanya bernilai 1
- ❑ Biasanya dipakai di lingkungan UNIX

void main()

- ❑ Berarti berupa function yang void sehingga tidak mengembalikan nilai status program sehingga nilai status program tidak bisa diketahui
- ❑ Biasanya dipakai pada program C di lingkungan Windows

Bentuk pemanggilan fungsi di C

- Pada dasarnya fungsi dapat memanggil fungsi lain, bahkan fungsi dapat memanggil dirinya sendiri (rekursif)



-
1. Which is not a proper prototype?
 - A. `int funct(char x, char y);`
 - B. `double funct(char x)`
 - C. `void funct();`
 - D. `char x();`
 2. What is the return type of the function with prototype: `"int func(char x, float v, double t);"`
 - A. `char`
 - B. `int`
 - C. `float`
 - D. `double`
 3. Which of the following is a valid function call (assuming the function exists)?
 - A. `funct;`
 - B. `funct x, y;`
 - C. `funct();`
 - D. `int funct();`
 4. Which of the following is a complete function?
 - A. `int funct();`
 - B. `int funct(int x) {return x=x+1;}`
 - C. `void funct(int) {printf("Hello");}`
 - D. `void funct(x) {printf("Hello"); }`

Latihan

- ❑ Buatlah fungsi untuk menampilkan tulisan “Algoritma dan Pemrograman” sebanyak 30 kali!
 - Ubahlah menjadi sebanyak n kali!
- ❑ Buatlah fungsi untuk menjumlahkan dua buah bilangan
 - Tambahkan: mengurangi, membagi, mengkali dua buah bilangan
- ❑ Buatlah fungsi untuk menentukan bilangan terkecil dari 3 buah bilangan yang diinputkan
- ❑ Buatlah fungsi untuk mengubah nilai ke huruf (A, B, C, D, dan E)
 - Buatlah fungsi untuk mengubah nilai huruf ke bobotnya

-
- Buatlah fungsi untuk mengubah bilangan pecahan ke bilangan bulat!
 - Buatlah fungsi untuk menjumlahkan deret: $1+3+5+7+\dots +n$
 - Buatlah fungsi untuk mengetahui kuadran suatu koordinat!
 - Buatlah fungsi untuk menyederhanakan b/c menjadi $A_{b/c}$