TIF1101 – Dasar-Dasar Pemrograman

HO 10b - Fungsi
(Bagian Kedua)

Opim Salim Sitompul

Department of Information Technology
Universitas Sumatera Utara

Outline

Tiga Bentuk Implementasi Fungsi

- Di dalam sebuah program, fungsi dapat diimplementasikan dalam tiga bentuk:
  1. Pendeklarasian fungsi sebagai sebuah prototipe,
  2. Pemanggilan fungsi dari bagian program lain, dan
  3. Pendefinisian fungsi.
- Ketiga bentuk implementasi ini masing-masing memiliki cara penulisan dan kegunaan yang berbeda antara satu dengan lainnya.

Pendeklarasian Prototipe Fungsi

- Pendeklarasian fungsi sebagai sebuah prototipe bertujuan agar kompiler dapat mengenal ciri-ciri fungsi yang diberikan.
- Pendeklarasian terdiri dari jenis data yang dikembalikan, nama fungsi, daftar jenis data argumen yang dibutuhkan, diakhiri **;**.
- Bentuk umum deklarasi prototipe fungsi sebagai berikut: **jenis_data** *namafungsi (jenis_data, jenis_data, ...)*;

Pendeklarasian Prototipe Fungsi

- Jenis data di depan nama fungsi menentukan nilai yang akan dikembalikan, di antaranya: **int**, **float**, dan **char**.
- Nama fungsi adalah nama yang diberikan oleh pemrogram untuk fungsi yang dibuatnya.
- Panjang nama fungsi max 31 karakter seperti aturan nama file.
- Jenis data yang terdapat di dalam tanda kurung disebut argumen-argumen formal.

Pendeklarasian Prototipe Fungsi

- Dalam penulisan nama berkas terdapat kesepakatan umum untuk menggunakan huruf-huruf kecil.
- Jika nama fungsi terdiri dari 2 kata, maka nama fungsi tersebut dipisahkan oleh garis bawah atau digabungkan dengan kata kedua dimulai dengan huruf besar. Contoh:
  float pangkat_dua(**float**, **float**);
  char inputKar(**char**);
- Fungsi pangkat_dua() mengembalikan nilai float dan memiliki dua buah argumen berjenis **float** pula, sedangkan fungsi inputKar() mengembalikan nilai karakter dan memiliki sebuah argumen berjenis karakter.

Pemanggilan Fungsi

- Fungsi-fungsi yang telah didefinisikan prototipenya dapat dipanggil dari bagian-bagian fungsi yang lain, misalnya fungsi *main()*.
- Pada saat sebuah fungsi dipanggil, alur eksekusi program akan berpindah ke fungsi yang dipanggil (melaksanakan perintah yang terdapat pada definisi fungsi).
- Setelah selesai mengeksekusi fungsi, kendali program akan dikembalikan kepada fungsi yang memanggil, dan alur eksekusi program dilanjutkan pada pernyataan setelah pemanggilan fungsi tersebut.

Pemanggilan Fungsi

- Bentuk umum pemanggilan fungsi:
  $val = namafungsi([arg\_akt_1], [arg\_akt_2], ..., [arg\_akt_n])$;
- arg_akt_1, arg_akt_2, ..., arg_akt_n adalah argumen-argumen yang dikirim ke fungsi berupa argumen aktual.
- Apabila fungsi yang dipanggil tidak memiliki argumen, pemanggilan fungsi dilakukan hanya dengan memberikan sepasang tanda kurung, tanpa argumen aktual.

Contoh Pemanggilan Fungsi

kpkppb.c

```
/* Nama file: kpkppb.c
   Program untuk mencari KPK dan PPB */
#include <stdio.h>
void faktor(int);

int main()
{
        int bil;

        printf("Berikan sebuah bilangan bulat: ");
        scanf("%d", &bil);
        faktor(bil);

        return 0;
}
```

Pendefinisian Fungsi

- Mendefinisikan fungsi berarti menyusun perintah-perintah yang akan dilakukan fungsi itu sesuai dengan tugas yang akan dilakukannya.
- Selain memuat instruksi-instruksi yang akan dilaksanakan oleh komputer, definisi fungsi juga memuat pernyataan-pernyataan deklarasi.
- Bentuk umum definisi fungsi sebagai berikut:
  [**jenis_data**] *nama_fungsi* ([**jenis_data** *arg1*], [**jenis_data** *arg2*], ...)
  {
  [deklarasi variabel]

  [kode program]
  [pernyataan **return**]
  }

Pendefinisian Fungsi

- Baris pertama definisi fungsi (judul fungsi), hampir sama seperti pada prototipenya, kecuali tidak ada tanda **;** setelah tanda), dan adanya nama variabel pada daftar argumen fungsi tersebut.
- Argumen yang ada pada definisi fungsi disebut argumen formal dan bila tidak ada argumen yang diterima, argumen fungsi tersebut adalah **void**.
- Setelah judul fungsi, ada sepasang kurung {} yang menandai blok badan fungsi.

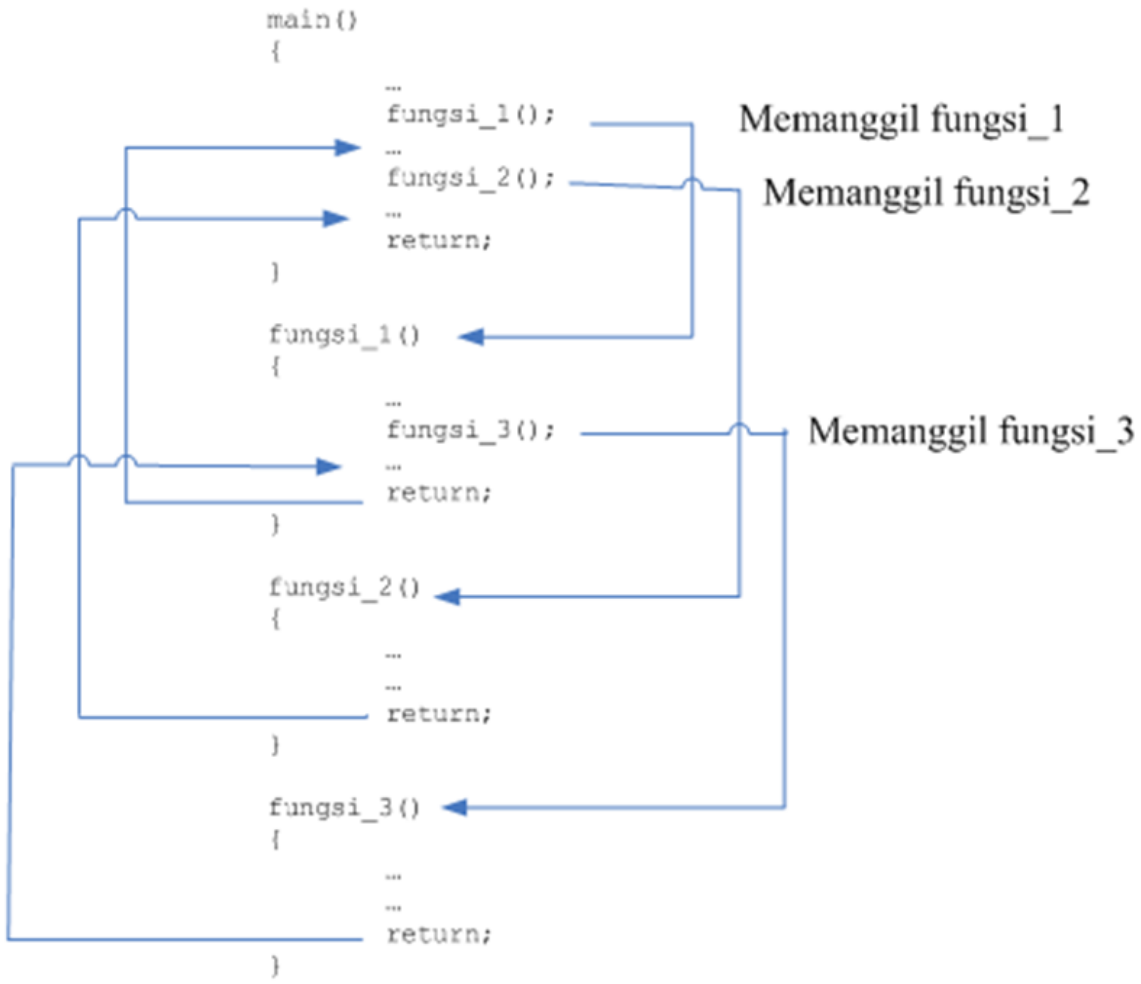
Pendefinisian Fungsi

- Tanda kurung siku [] menyatakan bahwa bagian-bagian tersebut bersifat opsional.
- Bagian pertama badan fungsi adalah deklarasi variabel-variabel lokal yang dibutuhkan oleh fungsi.
- Setelah itu diikuti oleh kode-kode program berupa pernyataan (tunggal atau majemuk), keduanya disusun dalam bentuk blok-blok kode program yang dibatasi oleh {} yang disebut badan fungsi.

Pendefinisian Fungsi

- Alur eksekusi program akan berpindah ke fungsi yang dipanggil untuk melaksanakan perintah yang terdapat pada definisi fungsi.
- Setelah selesai mengeksekusi fungsi, kendali program akan dikembalikan kepada fungsi yang memanggil.

Pendefinisian Fungsi



Gambar 1: Alur Eksekusi Fungsi

Pendefinisian Fungsi

▸ kpkppb.c

```
void faktor(int bil)
{
        int f, g;

        f = 2;
        while (bil % f != 0)
              f++;
        g = bil / f;
        printf("KPK = %d, PPB = %d\n", f, g);
}
```

Pendefinisian Fungsi

- Di bagian badan fungsi boleh terdapat pernyataan **return** yang bertujuan untuk mengembalikan nilai ke fungsi yang memanggil.
- Apabila fungsi tersebut tidak mengembalikan nilai, maka menurut standar ANSI C, jenisnya adalah **void** dan pada badan fungsi tidak boleh terdapat **return**.
- Jadi, di dalam sebuah fungsi pernyataan **return** bisa tidak ada, hanya satu buah (di akhir badan fungsi), atau dapat lebih dari satu (di bagian-bagian badan fungsi tertentu).

Pendefinisian Fungsi

- Segera setelah ditemukannya pernyataan return, kendali akan dikembalikan ke bagian yang memanggil.
- Oleh karena itu, setelah pernyataan ini tidak boleh ada pernyataan-pernyataan lain.
  - Kode program yang dituliskan setelahnya tidak akan terjangkau
  - Kompiler akan memberi pesan peringatan bahwa kode program tersebut tidak mempunyai pengaruh.

Pengiriman Argumen

- Komunikasi yang terjadi antara satu fungsi dengan fungsi lain dilakukan dengan cara saling bertukar data.
- Jika sebuah fungsi memanggil fungsi yang lain untuk melakukan tugas tertentu, fungsi yang dipanggil itu dapat diberikan semua data yang dibutuhkannya untuk melaksanakan tugasnya.
- Setelah melaksanakan tugas tersebut, fungsi yang dipanggil dapat mengembalikan sebagian data kepada fungsi pemanggil.

Pengiriman Argumen

- Pada sebuah program, pengiriman data dari satu fungsi ke fungsi lain, dapat dilakukan dengan dua cara:
  1. mengirimkan salinan data, disebut juga pengiriman dengan nilai (*pass by value*), dan
  2. mengirimkan alamat data, disebut juga pengiriman dengan alamat (*pass by reference* atau *passing by address*).

Pass by value

- Pengiriman dengan nilai dilakukan dengan cara mengirimkan salinan nilai variabel melalui argumen aktual pada fungsi pemanggil ke fungsi yang dipanggil.
- Fungsi yang dipanggil kemudian menampung data ini ke dalam variabel-variabel yang terdapat pada daftar argumen formal.

Pass by value

- Cara pemanggilan ini dapat memberikan banyak keuntungan, di antaranya:
  - Data yang terdapat pada fungsi pemanggil tidak akan terganggu oleh adanya perubahan-perubahan yang dilakukan pada fungsi yang dipanggil.
  - Semua perubahan data yang terjadi pada fungsi yang dipanggil hanya bersifat lokal di dalam fungsi tersebut.

Pass by reference

- Pemanggilan dengan alamat dilakukan dengan cara mengirimkan alamat-alamat variabel melalui argumen aktualnya ke fungsi yang dipanggil.
- Fungsi yang dipanggil kemudian menampung alamat-alamat ini pada variabel-variabel pointer yang terdapat dalam daftar argumen formal.
- Semua perubahan data yang dilakukan pada fungsi yang dipanggil akan otomatis mengubah data pada fungsi yang memanggil (disebut efek samping).

Contoh Pass by value

kirim1.c

```
/* Nama file: kirim1.c
   Pengiriman argumen dengan nilai */
#include <stdio.h>

int main()
{
        int x = 5, y = 4;

        printf("Fungsi main():\n");
        printf("Sebelum memanggil fungsi1()\n");
        printf("x = %d\ty = %d\n", x, y);
        y += fungsi1(x);
        printf("Fungsi main():\n");
        printf("Setelah memanggil fungsi1()\n");
        printf("x = %d\ty = %d\n", x, y);
```

Contoh Pass by value

kirim1.c

```
int fungsi1(int x)
{
        int y;

        printf("Fungsi1():\n");
        printf("x yang diterima = %d\n", x);
        y = ++x;
        printf("x berubah menjadi = %d\n", x);
        printf("y yang dikembalikan = %d\n", y);

        return y;
}
```

Contoh Pass by reference

kirim2.c

```
/* Nama file: kirim2.c
    Pengiriman argumen dengan alamat */
#include <stdio.h>

int main()
{
        int x = 5, y = 4;

        printf("Fungsi main():\n");
        printf("Sebelum memanggil fungsi1()\n");
        printf("x = %d\ty = %d\n", x, y);
        y += fungsi1(&x);
        printf("Fungsi main():\n");
        printf("Setelah memanggil fungsi1()\n");
        printf("x = %d\ty = %d\n", x, y);
```

Contoh Pass by reference

▸ kirim2.c

```
int fungsi1(int *x)
{
        int y;

        printf("Fungsi1():\n");
        printf("x yang diterima = %d\n", *x);
        y = ++*x;
        printf("x berubah menjadi = %d\n");
        printf("y yang dikembalikan = %d\n", *x, y)
        return y;
}
```

Mengembalikan Nilai dari Fungsi

- Jenis data yang dikembalikan dari sebuah fungsi, tergantung dari jenis data yang dinyatakan pada judul atau prototipe fungsi.
- Jika jenis data pada judul fungsi tidak disebutkan, maka secara *default* jenis datanya adalah integer.
  - Apabila kita tidak menginginkan adanya nilai yang dikembalikan dari sebuah fungsi, maka jenis data fungsi tersebut hendaklah dinyatakan sebagai **void**.
- Perhatikan prototipe-prototipe fungsi berikut:
  float pangkatBilangan(**float**, **float**);
  int cetakBilangan(**int**);
  char cariHuruf(**void**);
  void tampilkanHasil(**char**, **int**);

Mengembalikan Nilai dari Fungsi

- Jenis data yang dikembalikan dari fungsi pada contoh pertama, kedua, dan ketiga berturut-turut adalah **float**, **int**, dan **char**.
  - Fungsi-fungsi ini akan mengembalikan nilai sesuai dengan jenis data tersebut.
- Fungsi pada contoh keempat tidak mengembalikan nilai (**void**).
  - Upaya untuk mengembalikan nilai dari fungsi ini akan memberikan pesan **error**.

Fungsi *main*() Berargumen

- Argumen formal fungsi *main*() sering dibiarkan kosong, padahal argumen formal yang terdapat pada fungsi main() ini merupakan salah satu keunggulan program C.
- Dengan argumen ini, kita dapat menjalankan program dengan memberikan nilai dari luar program itu.
- Argumen dari luar program ini disebut *command-line argument*.

Fungsi *main*() Berargumen

- Bentuk umum pendefinisian fungsi *main*() berargumen adalah:

```
int main(int argc, char **argv)
{
    [deklarasi lokal]
    [kode program yang dapat dieksekusi]
    return 0;
}
```

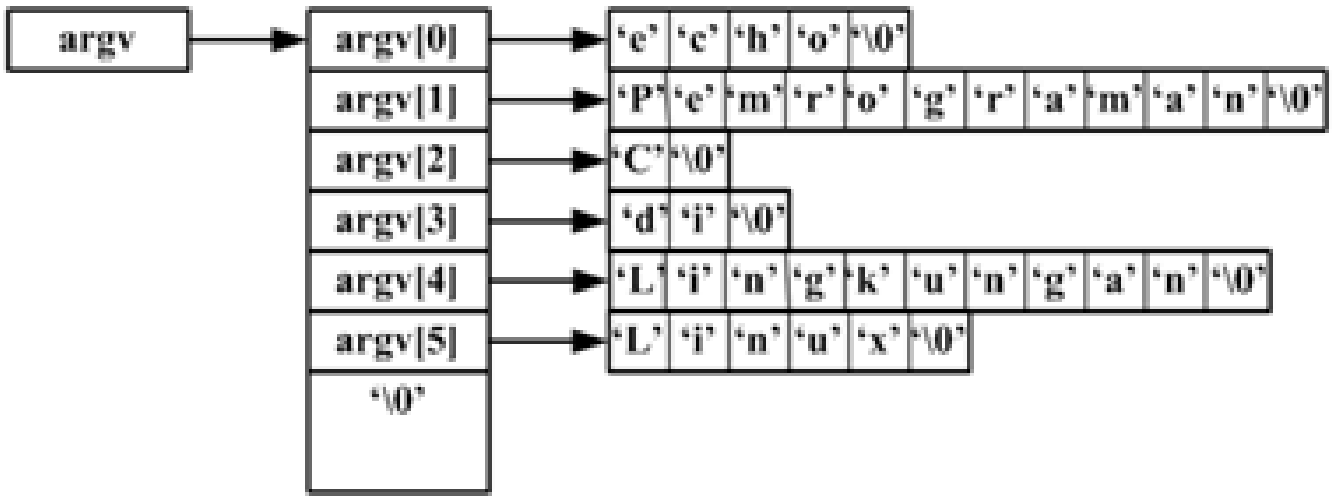
- *nama_file* beserta argumen-argumen yang diberikan disimpan dalam sebuah array *argv*.
- Vektor argumen *argv* adalah sebuah pointer yang menunjuk ke sebuah array *argv[i]*, i = 0, 1, 2, ... yang berisi pointer ke informasi tentang argumen yang diberikan.
- Kompiler akan menghitung jumlah argumen pada vektor argumen dan menyimpannya dalam variabel *argc*.

Fungsi *main*() Berargumen

- Untuk mengeksekusi program dengan fungsi main() berargumen, pengguna harus mengirim argumen ke fungsi tersebut dari luar program setelah nama file program.
- Bentuk umum pemanggilannya adalah:

  nama_file [arg1], [arg2], ...
- Jika main() memiliki lebih dari satu argumen, elemen array yang pertama (argv[0]) menunjuk ke nama program, elemen array kedua (argv[1]) menunjuk ke argumen yang pertama, dan seterusnya.

Fungsi *main*() Berargumen



Gambar 2: Vektor argumen

Fungsi *main*() Berargumen

▸ echo.c

```
/* Nama file: echo.c
   Ilustrasi fungsi main() berargumen */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
        int i;

        if (argc < 2)
                goto error;
        for(i=1; i<argc; i++)
                printf("%s ", argv[i]);
        printf("\n");
        return 0;
```

Fungsi *main*() Berargumen

- Dalam pemberian argumen ke fungsi main() ada beberapa hal yang perlu diperhatikan:
  1. Variabel *argc* akan menghitung banyaknya argumen yang dikirim mulai dari 0, 1, dst, dimana argumen ke-0 adalah nama file eksekusi yang diberikan.
  2. Argumen akan diperlakukan sebagai *string*, meskipun argumen yang diberikan itu adalah berupa angka.
     - Untuk memperlakukan argumen tersebut kembali sebagai angka, didalam program dapat digunakan fungsi-fungsi konversi teks ke angka.

Fungsi *main*() Berargumen

primakah.c

```
/* Nama file: primakah.c
    Menentukan apakah suatu bilangan prima atau buk
#include <stdio.h>
#include <stdlib.h>
int cekPrima(long);

int main(int argc, char **argv)
{
        int habis;
        long x;

        if (argc < 2)
                printf("Penggunaan: %s %s\n", *argv
```

Fungsi *main*() Berargumen

primakah.c

```
    else
        {
                x = atoi(argv[1]);
                if(x <= 1)
                        printf("Bilangan␣prima␣terk
                else
                {
                        habis = cekPrima(x);
                        if(!habis)
                                printf("%ld␣bilanga
                        else
                                printf("%ld␣bukan␣b
                }
        }
```

Fungsi *main*() Berargumen

primakah.c

```
int cekPrima(long x)
{
        int i, habis = 0;

        for(i=2; i < x; i++)
        {
                if((x % i) == 0)
                {
                        habis = 1;
                        break;
                }
        }
        return habis;
}
```