TIF1101 – Dasar-Dasar Pemrograman

HO 09 - Fungsi
(Bagian Pertama)

Opim Salim Sitompul

Department of Information Technology
Universitas Sumatera Utara

Outline

1. Pendahuluan
2. Kategori Fungsi
3. Prototipe Fungsi
4. Contoh Prototipe Fungsi
5. Lebih Lanjut dengan Pengarah Propengolahan
   - Penyisipan berkas
   - Subsitusi Makro
   - Pengarah kendali kompiler

Pendahuluan

- Fungsi adalah sekumpulan pernyataan yang berdiri sendiri dan dibuat utk melakukan tugas-tugas tertentu.
- Tujuan: agar tidak terjadi pengulangan penulisan kode program setiap kali suatu tugas yang sama akan dikerjakan.
- Fungsi berperan utk membuat program bersifat modular, selain menghemat kode program (*source code*).
- Fungsi juga berguna untuk menyembunyikan rincian program.
- Fungsi yang dirancang dengan baik dapat digunakan tanpa harus mengetahui bagaimana fungsi itu dibuat.

Pendahuluan

- Hal-hal yang perlu diperhatikan dalam merancang sebuah fungsi:
  1. Jenis data dari nilai yang dikembalikan oleh fungsi.
  2. Nama fungsi yang harus diawali oleh huruf atau garis bawah, dapat diikuti oleh kombinasi antara huruf, angka dan garis bawah.
  3. Daftar argumen yang dituliskan dalam tanda kurung biasa setelah nama fungsi.
  4. Badan fungsi yang memuat:
     - Deklarasi variabel,
     - Deklarasi prototipe fungsi, dan
     - Instruksi-instruksi yang akan dilaksanakan.
- Harus diawali dengan tanda { dan diakhiri dengan }. Sepasang { } ini menunjukkan satu blok pernyataan yang terdiri dari satu atau lebih pernyataan.

Pendahuluan

faktor.c

```
/* Nama file: faktor.c
   Program untuk menghitung faktorial bilangan bula
#include <stdio.h>
int main()
{
        long fakt(int); /* deklarasi prototipe fungs
        int bil;

        printf("Berikan sebuah bilangan bulat: ");
        scanf("%d", &bil);
        printf("%d faktorial = %ld.\n", bil, fakt(bi
        return 0;
}
```

Pendahuluan

faktor.c

```
long fakt(int n)   /* memanggil fungsi */
{
        int i;
        long f=1;

        for(i=2;i<=n;++i)
                f *= i;

        return f;
}
```

Kategori Fungsi

- Sebuah program C tersusun dari fungsi-fungsi.
  - Sebagian disediakan oleh kompiler C, berupa fungsi-fungsi standard.
  - Sebagian lainnya dibuat sendiri oleh pemrogram untuk melaksanakan suatu tugas tertentu.
- Setiap fungsi standar memiliki prototipe yang termuat di berkas judul berekstensi *.h*
- Prototipe fungsi-fungsi buatan pemrogram juga dapat disimpan di berkas judul berekstensi *.h*

Kategori Fungsi

- Berkas judul harus disertakan bila fungsi-fungsi tersebut digunakan.
- Penyertaannya dilakukan dengan menggunakan pengarah prapengolahan: #include.
  - Mis: #include <stdio.h>
- Untuk fungsi buatan pemrograman:
  - Mis: #include “myfunction.h”

Kategori Fungsi

- Di antara fungsi-fungsi buatan pemrogram yang paling penting adalah fungsi *main()*.
- Bersifat khusus, karena merupakan fungsi yang pertama sekali dicari oleh kompiler pada saat program dieksekusi.
- *main()* adalah sebuah kata cadangan yang tidak boleh diubah dengan nama lain, dan harus dituliskan dengan huruf kecil.

Prototipe Fungsi

- Model dari sebuah fungsi untuk mendeklarasikan ciri-ciri fungsi, yakni:
  1. Jenis data dari nilai yang dikembalikan,
  2. Nama fungsi,
  3. Daftar argumen yang diperlukan, dan
  4. Jenis data masing-masing argumen.
- Kompiler dapat melakukan pemeriksaan terhadap ciri-ciri tersebut untuk memastikan bahwa fungsi tersebut bekerja sebagaimana mestinya.

Prototipe Fungsi

- Dalam program C, letak fungsi menentukan keterlihatannya dari fungsi-fungsi yang lain.
  - Mis: letak fungsi *main()*, boleh pada bagian pertama program, sebelum fungsi-fungsi lain, atau boleh juga pada bagian akhir program, setelah fungsi-fungsi yang lain.
- Letak fungsi-fungsi ini secara umum, juga berkaitan dengan pendeklarasian prototipenya.

Prototipe Fungsi

- Prototipe suatu fungsi harus dideklarasikan, apabila fungsi yang dipanggil berada setelah fungsi yang memanggil.
- Apabila berada setelah fungsi yang dipanggil, prototipe fungsi tersebut boleh diberikan boleh tidak, karena fungsi yang memanggil telah mengetahui keberadaannya.
  - Jika ditinjau dari kegunaan prototipe, bentuk penulisan fungsi dengan prototipe adalah lebih baik.

Contoh Prototipe Fungsi

fproto1.c

```
/* Nama file : proto1.c
   Perlunya deklarasi prototipe */
#include <stdio.h>
int main()
{
   int *f1(int *);  /* deklarasi prototipe fungsi *
   int x = 11, z = x, *y;

   y = f1(&z);              /* pemanggilan fungsi */
   printf("*y = %d\n", *y);

   return 0;
}
```

Contoh Prototipe Fungsi

fproto1.c

```
int *f1(int *z)        /* definisi fungsi */
{
    (*z)++;
    return z;
}
```

Contoh Prototipe Fungsi

▸ fproto2.c

```
/* Nama file : proto2.
   Tanpa deklarasi prototipe fungsi */

#include <stdio.h>

int f2(int *z)
{
  int x = 3;

  *z += x;
}
```

Contoh Prototipe Fungsi

fproto2.c

```
int main()
{
   int x = 7, *z = &x;

   f2(z);
   printf("*z = %d\n", *z);

   return 0;
}
```

Contoh Prototipe Fungsi

- Pada **proto2.c** fungsi *main()* terletak di akhir program, sehingga fungsi *f2()* dapat langsung dipanggil *main()* tanpa terlebih dulu mendeklarasikannya.
- Cara ini **tidak dianjurkan**, karena ciri-ciri fungsi *f2()* tersebut tidak diberitahukan secara eksplisit kepada kompiler.

Lebih Lanjut dengan Pengarah Propengolahan

- Fasilitas prapengolahan bertujuan untuk menyiapkan berkas program sebelum diolah oleh kompiler.
- Beberapa di antaranya adalah:
  - Penyisipan berkas,
  - Substitusi makro, dan
  - Pengarah kendali kompiler.

Penyisipan berkas

- Pelayanan penyisipan berkas dilakukan dengan menggunakan pengarah #include, yang bentuk umumnya adalah:
  #include <namafile> atau
  #include “namafile”
- Isi berkas yang dinyatakan oleh namafile tersebut akan disisipkan ke dalam program.

Penyisipan berkas

- Jika namafile diapit oleh pembatas < >, pencarian akan dimulai dari direktori **include**.
- Umumnya berkas judul yang disisipkan dengan cara ini adalah berkas-berkas judul standar yang telah disediakan terlebih dahulu.
- Apabila namafile yang akan disisipkan itu diapit oleh pembatas " " (tanda kutip ganda), maka pencarian berkas akan dimulai dari direktori di mana berkas program itu berada.

Penyisipan berkas

- Jika tidak ditemukan di sana, pencarian dilanjutkan dengan mengikuti aturan implementasi yang ditentukan dlm pencarian berkas.
- Aturan implementasi yang lazim digunakan adalah pada direktori **include** (*default*).
- Umumnya berkas judul yang disisipkan dengan cara ini adalah berkas-berkas judul buatan pemrogram sendiri sesuai dengan kebutuhan yang diperlukan pada program yang disusunnya.

Subsitusi Makro

- Bentuk umumnya perintah pendefinisian makro adalah: **#define** *pengenal teks-pengganti*
- Bentuk tersebut adalah bentuk pensubstitusian makro yang paling sederhana, yaitu hanya untuk melakukan pensubstitusian.
- Setiap pengenal yang dijumpai di dalam berkas program akan diganti dengan teks-pengganti.
- Jika definisi teks pengganti terlalu panjang, dapat ditulis dalam beberapa baris dengan menggunakan karakter backslash (\) pada teks yang akan disambung.

Subsitusi Makro

- Sebagai ilustrasi perhatikan perintah-perintah berikut ini:
  #define PI 3.14159
  #define FOREVER for(; ;)
  #define Begin {
  #define End }
  #define Sembarang_Tombol printf("Tekan sembarang tombol...\n");
- Perhatikan bahwa definisi pengenal bersifat sensitif terhadap huruf yang digunakan, jika pengenal yang digunakan ditulis dengan huruf besar, penggunaannya di dalam program harus dengan huruf besar.
- Selain bentuk pensubstitusian sederhana seperti di atas, pendefinisian makro juga dapat digunakan dengan cara memberikan argumen.

Subsitusi Makro

- Bentuk umum prapengolah berargumen ini adalah:
  #define pengenal(argumen1, argumen2, ...)
  ekspresi-pengganti
- Argumen-argumen yang terdapat pada definisi makro di atas disebut **argumen-argumen formal**, sedangkan di dalam program, argumen-argumen yang diberikan pada pemanggilan makro disebut **argumen-argumen aktual**.
- Argumen-argumen formal dituliskan di dalam tanda kurung dan antara pengenal dan tanda kurung tidak boleh terdapat spasi.

Subsitusi Makro

- Ekspresi pengganti dapat berupa sembarang ekspresi yang suku-sukunya terdiri dari argumen-argumen yang telah ditentukan.
- Pengenal yang didefinisikan pada prapengolah akan dipanggil dari badan fungsi dengan memberikan argumen-argumen aktual yang sesuai.
- Prapengolah akan mengganti argumen formal pada makro dengan argumen aktual yang diberikan itu.

Subsitusi Makro

- Dalam pensubstitusian makro, hendaklah diperhatikan benar urutan pengerjaan yang dilakukan.
- Untuk mengatasi kekeliruan dalam pengolahan, hendaklah setiap suku pada teks pengganti diapit oleh tanda kurung.
- Sebagai contoh perhatikan definisi-definisi berikut:
  #define KUADRAT(X) ((X) * (X))
  #define MAKS(a, b) ((a) > (b) ? (a) : (b))
  #define MIN(x, y) ((x) < (y) ? (x) : (y))
  #define MUTLAK(x) ((x) > 0 ? (x) : (-x))

Subsitusi Makro

- Pada pendefinisian KUADRAT(X) apabila pada teks penggantinya tidak diberikan tanda kurung secukupnya, maka akan dapat terjadi kesalahan.
- Perhatikan contoh berikut ini:
  #define KUADRAT(X) (X * X)
- Dalam badan fungsi, apabila makro di atas dipanggil seperti berikut ini:
  y = KUADRAT(i + j);
- maka pernyataan di atas akan diganti dengan:
  y = (i + j * i + j);
- yang tentu saja tidak seperti yang kita inginkan.

Contoh Prototipe Fungsi

▸ makro1.c

```
#include <stdio.h>
#define KUADRAT(X) (X * X)

int main()
{
        int i = 5, j = 6;

        printf("Kuadrat dari (%d+%d) = %d.\n", i, j

        return 0;
}
```

```
Output:
Kuadrat dari (5+6) = 41.
```

Contoh Prototipe Fungsi

▸ makro2.c

```
#include <stdio.h>
#define KUADRAT(X) ((X) * (X))

int main()
{
        int i = 5, j = 6;

        printf("Kuadrat dari (%d+%d) = %d.\n", i, j

        return 0;
}
```

```
Output:
Kuadrat dari (5+6) = 121.
```

Pengarah kendali kompiler

- Pengarah-pengarah prapengolahan yang termasuk dalam kelompok ini adalah: #if, #ifdef, #undef, #ifndef, #else, #elif, dan #endif.
- Pengarah-pengarah ini berperan untuk meangaktifkan dan menonaktifkan baris-baris program tertentu dan mengarahkan kompiler untuk melompati bagian-bagian kode program tertentu jika tidak diperlukan.
- Pengarah-pengarah bersyarat di atas berguna untuk mengendalikan prapengolahan sewaktu dilakukannya pengevaluasian.
- Misalnya, pendefinisian makro dapat dilakukan secara selektif, tergantung pada nilai kondisi yang dievaluasi.

Pengarah kendali kompiler

- Pengarah #if melakukan evaluasi terhadap sebuah ekspresi konstanta integer.
- Jika hasil evaluasinya memberikan nilai tidak nol, baris-baris yang mengikuti #if itu akan dilaksanakan sampai ditemukannya pengarah #elseif, #else, atau #endif.

Contoh Prototipe Fungsi

▸ ujikondisional.c

```
/* Namafile: ujikondisional.c
   Menguji penggunaan prototipe kondisional */
#include <stdio.h>
#define TEST 1
int main()
{
        int n;

        printf("Pengulangan while\n");
        n = 5;
        while(n < 7)
        {
```

Contoh Prototipe Fungsi

ujimakro.c

```
            #if TEST >= 1
            printf("n=%d\n", n);
            #endif
            n++;
            #if TEST >= 1
            printf("Sekarang n=%d\n", n);
            #endif
        }
        #if TEST == 0
        printf("Nilai n=%d\n", n);
        #endif

        return 0;
}
```