

II Pengenalan Bahasa C

II.1 Pendahuluan

- Bahasa C dikembangkan pertama kali pada laboratorium Bell (USA) sekitar tahun 1972 oleh Dennis Ritchie pada komputer DEC PDP-11 dengan sistem operasi UNIX.
- Beberapa versi C mulai dikembangkan oleh beberapa pakar untuk dijalankan pada sistem operasi selain UNIX, seperti PC-DOS dan MS-DOS.
- Untuk melaksanakan pembakuan (standarisasi) terhadap bahasa C, ANSI (*American National Standards Institute*) membentuk team untuk membuat bahasa C standard ANSI, yang dimulai tahun 1983. Standard ANSI inilah yang selanjutnya digunakan sebagai acuan dari berbagai versi C yang beredar dewasa ini.

II.2 Lingkup Pemakaian Bahasa C

- Bahasa C mempunyai kemampuan lebih dibanding dengan bahasa pemrograman yang lain. Bahasa C merupakan bahasa pemrograman yang bersifat portabel, yaitu suatu program yang dibuat dengan bahasa C pada suatu komputer akan dapat dijalankan pada komputer lain dengan sedikit (atau tanpa) perubahan yang berarti.
- Bahasa C merupakan bahasa yang biasa digunakan untuk keperluan pemrograman sistem, antara lain untuk membuat:
 - o assembler
 - o interpreter
 - o kompiler
 - o sistem operasi
 - o program bantu (*utility*)
 - o editor
 - o paket program aplikasi
- Beberapa program paket yang beredar seperti dBase dibuat dengan menggunakan bahasa C, bahkan sistem Operasi UNIX juga dibuat dengan menggunakan bahasa C.
- Bahasa C sesungguhnya merupakan bahasa pemrograman yang serbaguna yang pemakaiannya tidak terbatas untuk pemrograman sistem, namun juga dapat digunakan untuk aplikasi bisnis, matematis maupun *games*, bahkan untuk aplikasi kecerdasan buatan.
- Dalam beberapa literatur, bahasa C digolongkan sebagai bahasa aras menengah (*medium level language*).

- Penggolongan ini bukan berarti bahasa C kurang ampuh atau lebih sulit dibandingkan dengan bahasa aras tinggi (*high level language*, seperti Pascal, Basic, Fortran, dll), namun untuk menegaskan bahwa bahasa C bukanlah bahasa yang berorientasi pada mesin (yang merupakan ciri bahasa aras rendah (*low level language*), yaitu bahasa mesin dan assembly).
- Pada kenyataannya, C mengkombinasikan elemen dalam bahasa aras tinggi dan bahasa aras rendah, yaitu kemudahan dalam membuat program yang ditawarkan pada bahasa aras tinggi dan kecepatan eksekusi dari bahasa aras rendah.

II.3 Kelebihan dan Kelemahan C

Bahasa C mempunyai beberapa kelebihan dibanding dengan bahasa pemrograman yang lain, yaitu:

- C mempunyai operator yang lengkap untuk memanipulasi data.
- Berbagai struktur data dan pengendalian proses disediakan dalam C, sehingga memungkinkan dibuat program yang terstruktur, bahkan program yang berorientasi pada obyek (OOP = *object oriented programming*).
- Dibanding dengan bahasa mesin atau rakitan (assembly), C jauh lebih mudah dipahami dan pemrogram tidak perlu tahu detail mesin komputer yang digunakan sehingga tidak menyita waktu dalam menyelesaikan masalah ke dalam bentuk program. C merupakan bahasa yang berorientasi pada permasalahan (object), dan bukan berorientasi pada mesin.
- Kecepatan eksekusi C mendekati kecepatan eksekusi program yang dibuat dengan bahasa aras rendah, namun kemudahan dalam memprogram setara dengan bahasa aras tinggi.
- C memungkinkan memanipulasi data dalam bentuk bit maupun byte secara efisien. Disamping itu juga memungkinkan untuk melakukan manipulasi alamat dari suatu data yang dalam C dinamakan pointer.

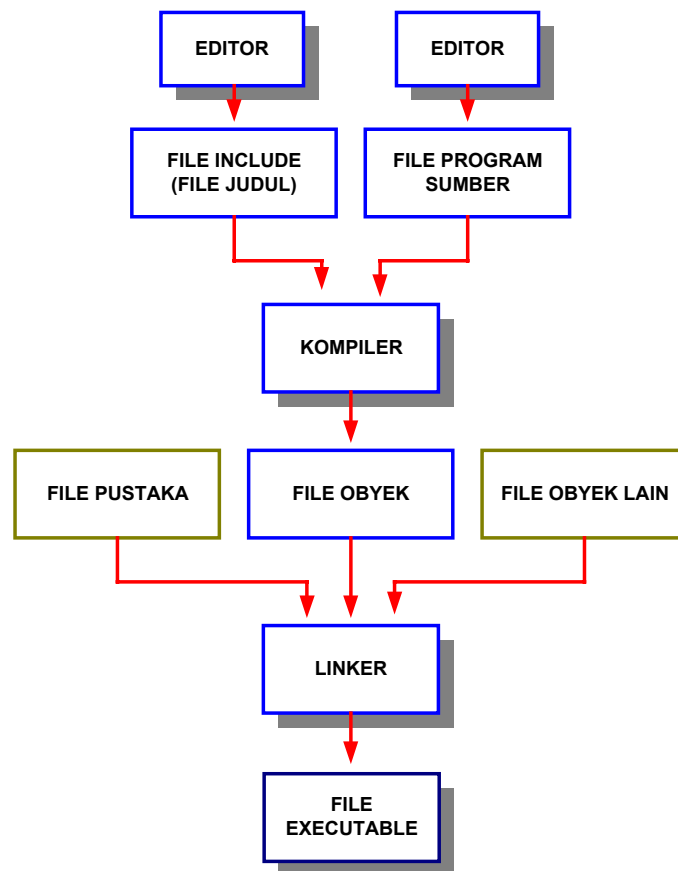
Adapun kelemahan atau lebih tepatnya kesulitan bahasa pemrograman C terutama yang dirasakan oleh pemrogram pemula diantaranya adalah:

- Banyaknya operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai, yang jika belum familiah akan menimbulkan masalah.
- Para pemrogram C tingkat pemula umumnya belum pernah mengenal pointer dan tidak terbiasa menggunakannya, padahal kemampuan bahasa C justru terletak pada pointer.

Namun umumnya kesulitan di atas hanya bersifat sementara saja.

II.4 Interpreter dan Kompiler

- Agar suatu program dalam bahasa pemrograman dapat dimengerti dan dieksekusi oleh komputer, maka program yang ditulis harus diterjemahkan terlebih dulu ke dalam kode mesin. Adapun penterjemah yang digunakan dapat berupa *interpreter* maupun *kompiler*.
- **Interpreter** adalah suatu jenis penterjemah yang menterjemahkan per baris instruksi untuk setiap saat.
- Keuntungan pemakaian interpreter adalah penyusunan program relatif lebih cepat dan bisa langsung diuji meskipun masih ada beberapa kesalahan secara kaidah dalam program.
- Kelemahannya adalah kecepatannya menjadi lambat, sebab sebelum suatu instruksi dijalankan selalu diterjemahkan terlebih dahulu. Disamping itu saat program dieksekusi, interpreter juga harus berada dalam memori.
- Kelemahan yang lain berkaitan dengan kerahasiaan program. Program yang memakai interpreter, program sumbernya (*source program*) tidak dapat dirahasiakan.
- **Kompiler** merupakan jenis penterjemah yang cara kerjanya menterjemahkan seluruh instruksi dalam program sekaligus.
- Proses kompilasi cukup dilakukan sekali saja, selanjutnya hasil penterjemahan (setelah melalui tahap-tahap yang lain) dapat dijalankan secara langsung tanpa bergantung pada program sumber maupun kompilernya.
- Proses ekeekusi dapat berjalan lebih cepat, sebab tidak ada lagi proses penterjemahan.
- Program sumber juga dapat dirahasiakan, sebab yang dieksekusi adalah adalah file program yang sudah dalam bentuk kode mesin.
- Kelemahannya proses pembuatan dan pengujiannya relatif lebih lama, sebab harus melalui tahap penterjemahan (*compiling*) dan proses *linking*.
- Program hanya akan berhasil dikompilasi jika sudah tidak mengandung kesalahan secara kaidah (*syntax error*) sama sekali.



Gambar 2-1 Proses Kompilasi-Linking Program C

II.5 Fungsi Penyusun Program C

- Program Bahasa C pada hakekatnya tersusun atas sejumlah blok fungsi.
- Sebuah program minimal mengandung sebuah fungsi yaitu fungsi utama (main()).
- Setiap fungsi terdiri dari satu atau beberapa pernyataan yang secara keseluruhan dimaksudkan untuk melaksanakan tugas khusus.
- Bagian pernyataan fungsi (sering disebut tubuh fungsi) diawali dengan tanda kurung kurawal buka ({) dan diakhiri dengan tanda kurung kurawal tutup (}).
- Namun dalam kenyataannya suatu fungsi bisa saja tidak mengandung pernyataan sama sekali, seperti yang diperlihatkan dalam contoh berikut.

```

Main ( )
{
}
  
```

- Walaupun fungsi tidak mempunyai pernyataan, namun kurung kurawal harus tetap ada, karena mengisyaratkan awal dan akhir definisi fungsi.
- Secara umum suatu fungsi mempunyai bentuk sebagai berikut:

```

Nama-fungsi (daftar parameter)
Deklarasi parameter;
{
    tubuh fungsi
}

```

II.5.1 Fungsi main ()

- pada program C, **main ()** merupakan fungsi yang istimewa, karena fungsi **main ()** harus selalu ada dalam program, sebab fungsi inilah yang menjadi titik awal dan titik akhir eksekusi program.
- Tanda { di awal fungsi meyakinkan awal tubuh fungsi dan sekaligus awal program, sedangkan tanda } di akhir fungsi merupakan akhir tubuh fungsi dan sekaligus akhir eksekusi program.
- Jika program terdiri lebih dari satu fungsi, fungsi **main ()** biasa ditempatkan pada posisi yang paling atas dalam pendefinisian fungsi, untuk memudahkan pencarian program utama.

II.5.2 Fungsi printf()

- Fungsi **printf()** merupakan fungsi yang umum digunakan untuk menampilkan suatu keluaran program pada layar penampil (monitor).
- Untuk menampilkan tulisan
 - o Selamat Datang

Maka pernyataan yang diperlukan berupa:

Printf("Selamat Datang");

- Pernyataan di atas berupa pemanggilan fungsi **printf()** dengan argumen/parameter berupa string "Selamat Datang".
- Dalam C suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik ganda (").
- Pernyataan dalam C selalu diakhiri dengan tanda titik koma (;), yang dipakai sebagai **pemberhentian pernyataan** dan bukanlah sebagai pemisah antara dua pernyataan.
- Contoh prog-2 berikut adalah contoh program yang agak lengkap:

```

#include <stdio.h>

main ()
{
    printf(" Selamat datang di program pra-pasca");
}

```

- Jika program di atas dieksekusi, akan menghasilkan:

C>Prog-2

Selamat datang di program pra-pasca.

II.6 Praprosesor `#include`

- pada contoh program sebelumnya terdapat baris yang berisi `#include <stdio.h>`
- `#include` merupakan salah satu jenis pengarah praprosesor yang digunakan untuk memberitahu kompiler agar dalam proses linking membaca file yang dinamakan file judul (*header file*), yaitu file yang diantaranya berisi deklarasi fungsi dan definisi konstanta.
- Bahasa C menyediakan beberapa file judul yang ditandai dengan ekstensi `.h`.
- Misal pada Program di atas, `#include <stdio.h>` menyatakan pada kompiler agar membca file bernama `stdio.h` saat melakukan kompilasi.
- Bentuk umum `#include`

```
#include <namafile>
atau
#include "namafile"
```

- Bentuk pertama (`#include <namafile>`) mengisyaratkan bahwa pencarian file dilakukan pada direktori khusus (direktori file include), yang merupakan *default* direktori file-file judul yang disediakan oleh bahasa pemrograman.
- Bentuk kedua (`#include "namafile"`) menyatakan bahwa pencerian file dilakukan pertama kali pada direktori aktif tempat program sumber, dan seandainya tidak ditemukan pencarian akan dilanjutkan pada direktori lainnya sesuai dengan perintah pada sistem operasi (yaitu PATH).
- Kebanyakan program melibatkan file `stdio.h`, yaitu file-judul I/O standard yang disediakan dalam C, yang diperlukan untuk program-program yang menggunakan pustaka fungsi I/O standard seperti `printf()`.

II.7 Lebih Lanjut dengan Fungsi `printf()`

- Fungsi `printf()` mempunyai kegunaan yang luas dalam C, dipakai untuk menampilkan string ataupun berbagai jenis data lainnya.
- Dengan menggunakan fungsi ini, tampilan dapat diatur (diformat) dengan mudah.

- Pada program 2 di atas, fungsi *main()* hanya mengandung satu pernyataan yaitu:

```
printf("Selamat datang di program pra-pasca");
```

- Pernyataan di atas dapat ditulis menjadi dua pernyataan sebagai berikut, yang akan menghasilkan keluaran yang sama.

```
printf("Selamat datang");  
printf(" di program pra-pasca");
```

↑
1 spasi

- Sekarang, bagaimana jika diinginkan keluaran berupa:

```
Selamat datang  
Di program pra-pasca
```

- Penyelesaiannya dapat diperoleh dengan menyertakan karakter yang dinamakan sebagai karakter baris baru (pindah baris), yaitu berupa :
`\n` di bagian akhir string pertama ("*Selamat datang*"). Seperti yang diperlihatkan pada contoh program 2-3 berikut.

```
#include <stdio.h>  
  
main()  
{  
    printf("Selamat datang\n");  
    printf("di program pra-pasca");  
}
```

contoh hasil eksekusinya adalah:

```
C>Prog2-3  
Selamat datang  
Di program pra-pasca
```

Program 2-3

- Tanda `\` pada string yang dilewatkan sebagai argumen *printf()* mempunyai makna yang khusus, yaitu digunakan untuk menyatakan karakter khusus seperti karakter baris-baru atau karakter *backslash* (miring kiri).
- Jadi karakter `\n` sebenarnya menyatakan sebuah karakter.

- Contoh lain karakter yang ditulis dengan diawali tanda \ adalah :
 - \” menyatakan karakter petik-ganda
 - \\ menyatakan karakter backslash
 - \t menyatakan karakter tab
- Untuk memperoleh keluaran berupa
Selamat datang di program “pra-pasca”
 Penulisan program yang diperlukan adalah:
 Printf(“Selamat datang di program \”pra-pasca\””);
- Program berikut ini memberikan gambaran pemakaian \t (tab) untuk mengatur agar format gambar keluaran menjorok ke kanan.

```
#include <stdio.h>

main()
{
    printf(“\t*****\n”);
    printf(“\t*      *\n”);
    printf(“\t*****\n”);
}
```

Contoh eksekusi:

```
C>Prog2-4
*****
*      *
*****
```

Program 2-4

- Bentuk umum format *printf()* adalah sebagai berikut:
Printf(“string kontrol”, daftar argumen);
- Dengan string kontrol dapat berupa satu atau sejumlah karakter yang akan ditampilkan ataupun berupa penentu format yang akan mengatur penampilan dari argumen yang terletak pada daftar argumen.
- Daftar penentu format dalam C diantaranya adalah seperti yang diperlihatkan pada tabel 2.1 berikut.

Tabel 2.1 Format string kontrol.

Format	Fungsi untuk menampilkan
%d	bilangan bulat (integer)
%ld	long integer
%u	unsigned integer
%x	hexadesimal integer
%f	float (bilangan pecahan)
%lf	double float
%e	float tipe exponen menggunakan e
%c	karakter
%s	string

Contoh pemakaiannya diperlihatkan dalam program 2-5 berikut.

```
#include <stdio.h>

main( )
{
    printf("Nama siswa : %s\n, "Amir");
    printf("No. Mhs.    : %d\n", 12547);
    printf("Nilai : %f Predikat : %c\n", 75.6, 'B');
}
```

Contoh hasil eksekusi:

```
C>Prog2-5
Nama Siswa : Amir
No. Mhs.    : 12547
Nilai : 75.600000 Predikat: B
```

Program 2-5

```

printf("Nama siswa : %s\n", "Amir");

printf("No. Mhs.    : %d\n", 12547);

printf("Nilai : %f Predikat : %c\n", 75.6, 'B');

```

Gambar 2.2 Penggunaan Penentu Format pada printf ().

II.8 Gaya Penulisan Program

- Bahasa C memberikan keleluasaan dalam menulis bentuk program.
- Program 2-3, misalnya dapat ditulis kembali tanpa mengubah hasil eksekusi program menjadi:

```

#include <stdio.h>

main () {
    printf("Selamat datang\n");
    printf("di program pra-pasca\n");
}

```

Program 2-6

Atau

```

#include <stdio.h>

main ()
{
    printf("Selamat datang\n");
    printf("di program pra-pasca\n");
}

```

Program 2-7

Bahkan penulisan seperti berikut, dengan menghilangkan sejumlah spasi tetap diperkenankan:

```
#include <stdio.h>
main () { printf("Selamat datang\n");
printf("di program pra-pasca\n"); }
```

Program 2-8

- Pemrogram tinggal memilih bentuk yang disukai.
- Sebagai pedoman, penulisan atau pemilihan bentuk hendaknya memberikan kemudahan bagi yang ingin membaca program, terutama untuk program yang berukuran besar.
- Jika dilihat pada program 2-8, penulisan yang sangat mampat dapat menyebabkan program sukar dibaca/dipahami.

II.9 Komentar dalam Program

- Untuk keperluan dokumentasi dan pemeliharaan program dengan maksud agar program mudah dipahami di saat yang lain, biasanya pada penulisan program disertakan suatu komentar atau keterangan mengenai program.
- Komentar atau keterangan ini dapat diletakkan pada awal suatu program atau fungsi, atau bahkan di akhir suatu baris instruksi, jika diperlukan.
- Dalam C, suatu komentar ditulis dengan diawali tanda `/*` dan diakhiri dengan tanda `*/`.
- Contoh komentar adalah sebagai berikut:

`/* Ini hanya sebagai komentar*/`

- Contoh yang mengandung keterangan lebih dari satu baris:

```

      awal keterangan
    ↙
/* ----- */
* Program Invers Matriks *
* Dibuat oleh : Rudy Hartanto *
* Tanggal    : 2 Juni 2002 *
* Direvisi   : 12 Juni 2002 *
* ----- */
      ↘
    akhir keterangan
```

Quisioner

1. Buatlah sebuah program untuk menampilkan tulisan sebagai berikut:

```
Selamat datang
Di program
“Magister Teknik Elektro”
```

2. Apa hasil dari program berikut:

```
#include <stdio.h>
main ()
{
    printf(“%d kelas\nper kelas %d”, 5, 10);
}
```

3. Tunjukkan letak kesalahan program berikut dan betulkanlah:

```
#include <stdio.h>
main ()
{
    printf(“%d kelas\nper kelas %f”, 5, 10)
    printf(“%f Jumlah siswa total\n”)
}
```