BAB I
SEKILAS TENTANG C

Tujuan :

1. Menjelaskan sejarah dan ruang lingkup pemakaian bahasa C
2. Menjelaskan kelebihan dan kekurangan bahasa C
3. Menjelaskan proses kompilasi dan linking program C
4. Menjelaskan struktur penulisan bahasa C dan menjelaskan komponen-komponen program dalam contoh aplikasi sederhana

1.1. Sejarah dan Ruang Lingkup C

Akar dari bahasa C adalah bahasa BCPL yang dikembangkan oleh Martin Richards pada tahun 1967. Bahasa ini memberikan ide kepada Ken Thompson yang kemudian mengembangkan bahasa yang disebut dengan B pada tahun 1970. Perkembangan selanjutnya dari bahasa B adalah bahasa C oleh Dennis Ritchie sekitar tahun 1970-an di Bell Telephone Laboratories Inc. (sekarang adalah AT&T Bell Laboratories). Bahasa C pertama kali digunakan pada komputer Digital Equipment Corporation PDP-11 yang menggunakan sistem operasi UNIX.

C adalah bahasa yang standar, artinya suatu program yang ditulis dengan versi bahasa C tertentu akan dapat dikompilasi dengan versi bahasa C yang lain dengan sedikit modifikasi. Standar bahasa C yang asli adalah standar dari UNIX. Sistem operasi, kompiler C dan seluruh program aplikasi UNIX yang esensial ditulis dalam bahasa C. Patokan dari standar UNIX ini diambilkan dari buku yang ditulis oleh Brian Kerninghan dan Dennis Ritchie berjudul *"The C Programming Language",* diterbitkan oleh Prentice-Hall tahun 1978. Deskripsi C dari Kerninghan dan Ritchie ini kemudian dikenal secara umum sebagai "K&R C".

Kepopuleran bahasa C membuat versi-versi dari bahasa ini banyak dibuat untuk komputer mikro. Untuk membuat versi-versi tersebut menjadi standar, ANSI *(American National Standards Institute)* membentuk suatu komite *(ANSI committee X3J11)* pada tahun 1983 yang kemudian menetapkan standar ANSI untuk bahasa C. Standar ANSI ini didasarkan kepada standar UNIX yang diperluas. Standar ANSI menetapkan sebanyak 32

buah kata-kata kunci *(keywords)* standar. Versi-versi bahasa C yang menyediakan paling tidak 32 kata-kata kunci ini dengan sintaks yang sesuai dengan yang ditentukan oleh standar, maka dapat dikatakan mengikuti standar ANSI. Buku ajar ini didasarkan pada bahasa C dari standar ANSI.

Pada saat ini C merupakan bahasa pemrograman yang sangat populer di dunia. Banyak pemrograman yang dibuat dengan bahasa C seperti assembler, interpreter, program paket, sistem operasi, editor, kompiler, program bantu, Word Star, Dbase, aplikasi untuk bisnis, matematika, dan game, bahkan ada pula yang menerapkannya untuk kecerdasan buatan.

Dalam beberapa literatur bahasa C digolongkan sebagai bahasa tingkat menengah. Penggolongan ke dalam bahasa tingkat menengah bukanlah berarti bahwa bahasa C lebih sulit dibandingkan dengan bahasa tingkat tinggi seperti PASCAL atau BASIC. Demikian juga bahasa C bukanlah bahasa yang berorientasi pada mesin seperti bahasa mesin dan assembly. Pada kenyataannya bahasa C mengkombinasikan elemen dalam bahasa tingkat tinggi dan bahasa tingkat rendah. Kemudahan dalam membuat program yang ditawarkan pada bahasa tingkat tinggi dan kecepatan eksekusi dari bahasa tingkat rendah merupakan tujuan diwujudkannya bahasa C.

1.2.Kelebihan dan Kelemahan C.

Beberapa kelebihan dari bahasa C:

- Bahasa C tersedia hampir di semua jenis komputer, baik mikro, mini maupun komputer besar (mainframe computer).
- Kode bahasa C bersifat portabel. Suatu aplikasi yang ditulis dengan bahasa C untuk suatu komputer tertentu dapat digunakan di komputer lain hanya dengan sedikit modifikasi.
- Berbagai struktur data dan pengendalian proses disediakan dalam C sehingga memungkinkan untuk membuat program yang terstruktur. Struktur bahasa yang baik, selain mudah dipelajari juga memudahkan dalam pembuatan program, pelacakan kesalahan program dan akan menghasilkan dokumentasi program yang baik.
- Dibandingkan dengan bahasa mesin atau assembly, C jauh lebih mudah dipahami dan pemrogram tidak perlu mengetahui mesin komputer secara detil. Dengan demikian tidak akan menyita waktu yang terlampau banyak dalam menyelesaikan suatu masalah

ke dalam bentuk program. Hal ini dikarenakan C merupakan bahasa yang berorientasi pada permasalahan, bukan berorientasi pada mesin.

- C memungkinkan memanipulasi data dalam bentuk bit maupun byte. Di samping itu juga memungkinkan untuk memanipulasi alamat dari suatu data atau pointer.

Adapun kelemahan bahasa C yang dirasakan oleh para pemula bahasa C:

- Banyaknya operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai. Kalau tidak dikuasai sudah tentu akan menimbulkan masalah.
- Para pemrogram C tingkat pemula umumnya belum pernah mengenal pointer dan tidak terbiasa menggunakannya. Padahal keampuhan C justru terletak pada pointer.

Kesulitan yang diuraikan di depan akan bersifat sementara saja. Kalau para pemula C mau mempelajarinya, sebenarnya tak ada yang dikatakan sulit sekali mengenai C. Mereka yang sudah terbiasa justru menyatakan bahwa bekerja dengan C sangat menyenangkan. Pepatah mengatakan "Di mana ada kemauan di situ ada jalan" dan "Jika tak kenal maka tak sayang".

1.3.Proses Kompilasi dan Linking Program C

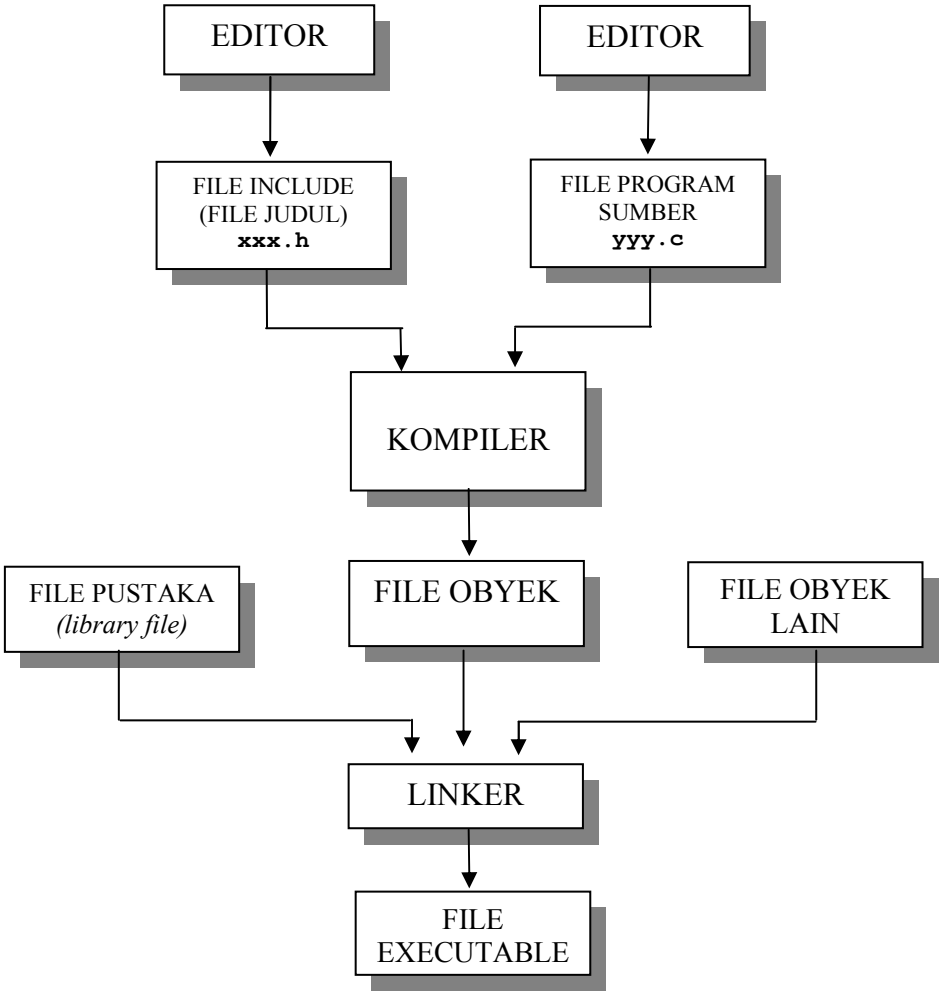
Agar suatu program dalam bahasa pemrograman dapat dimengerti oleh komputer, program haruslah diterjemahkan dahulu ke dalam kode mesin. Adapun penerjemah yang digunakan bisa berupa interpreter atau kompiler.

Interpreter adalah suatu jenis penerjemah yang menerjemahkan baris per baris intsruksi untuk setiap saat. Keuntungan pemakaian interpreter, penyusunan program relatif lebih cepat dan bisa langsung diuji sekalipun masih ada beberapa kesalahan secara kaidah dalam program. Sedangkan kelemahannya, kecepatannya menjadi lambat sebab sebelum suatu instruksi dijalankan selalu harus diterjemahkan terlebih dahulu. Selain itu, saat program dieksekusi, interpreter juga harus berada dalam memori. Jadi memori selalu digunakan baik untuk program maupun interpreter. Di samping itu, program sumber *(source program)* yaitu program aslinya tidak dapat dirahasiakan (orang lain selalu bisa melihatnya).

Kebanyakan versi C yang beredar di pasaran menggunakan penerjemah berupa kompiler. Kompiler merupakan jenis penerjemah yang lain, dengan cara kerjanya yaitu menerjemahkan seluruh instruksi dalam program sekaligus. Proses pengkompilasian ini

cukup dilakukan sekali saja. Selanjutnya hasil penerjemahan (setelah melalui tahapan yang lain) bisa dijalankan secara langsung, tanpa tergantung lagi oleh program sumber maupun kompilernya. Keuntungannya, proses eksekusi dapat berjalan dengan cepat, sebab tak ada lagi proses penerjemahan. Di samping itu, program sumber bisa dirahasiakan, sebab yang dieksekusi adalah program yang sudah dalam bentuk kode mesin. Sedangkan kelemahannya, proses pembuatan dan pengujian membutuhkan waktu relatif lebih lama, sebab ada waktu untuk mengkompilasi (menerjemahkan) dan ada pula waktu melakukan proses *linking*. Perlu pula diketahui, program akan berhasil dikompilasi hanya jika program tak mengandung kesalahan secara kaidah sama sekali.

Proses dari bentuk program sumber C (*source program*, yaitu program yang ditulis dalam bahasa C) hingga menjadi program yang *executable* (dapat dieksekusi secara langsung) ditunjukkan pada Gambar 1.1 di bawah ini.



Gambar 1.1 Proses Kompilasi-*Linking* dari program C

Keterangan Gambar :

- Pertama-tama program C ditulis dengan menggunakan editor. Program ini disimpan dalam file yang disebut file program sumber (dengan ciri utama memiliki ekstensi **.c**).
- File *include* (umumnya memiliki ekstensi **.h,** misalnya **stdio.h**, atau biasa disebut dengan file judul *(header file)*) berisi kode yang akan dilibatkan dalam program C (Pada program tertentu bisa saja tidak melibatkan file *include*).
- Berikutnya, kode dalam file program sumber maupun kode pada file *include* akan dikompilasi oleh kompiler menjadi kode obyek. Kode obyek ini disimpan pada file yang biasanya berekstensi **.obj**, atau **.o** (bergantung kepada lingkungan/*environment* sistem operasi yang dipakai). Kode obyek berbentuk kode mesin, oleh karena itu tidak dapat dibaca oleh pemrogram. Akan tetapi kode ini sendiri juga belum bisa dipahami komputer.
- Supaya bisa dimengerti oleh komputer, maka kode obyek bersama-sama dengan kode obyek yang lain (kalau ada) dan isi file pustaka (*library file*, yaitu file yang berisi rutin untuk melaksanakan tugas tertentu. File ini disediakan oleh pembuat kompiler, biasanya memiliki ekstensi **.lib**) perlu dikaitkan *(linking*) dengan menggunakan *linker*, membentuk sebuah program yang *executable* (program yang dapat dijalankan/dieksekusi secara langsung dalam lingkungan sistem operasi).
- Program hasil *linker* ini disimpan dalam file yang disebut file *executable*, yang biasanya berekstensi **.exe**.

1.4. Struktur Penulisan Program C

Untuk dapat memahami bagaimana suatu program ditulis, maka struktur dari program harus dimengerti terlebih dahulu. Tiap bahasa komputer mempunyai struktur program yang berbeda. Struktur program memberikan gambaran secara luas, bagaimana bentuk program secara umum.

Program C pada hakekatnya tersusun atas sejumlah blok fungsi. Sebuah program minimal mengandung sebuah fungsi. Fungsi pertama yang harus ada dalam program C dan sudah ditentukan namanya adalah *main()*. Setiap fungsi terdiri atas 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 (}). Di antara kurung

kurawal itu dapat dituliskan statemen-statemen program C. Namun pada kenyataannya, suatu fungsi bisa saja tidak mengandung pernyataan sama sekali. Walaupun fungsi tidak memiliki pernyataan, kurung kurawal haruslah tetap ada. Sebab kurung kurawal mengisyaratkan awal dan akhir definisi fungsi. Berikut ini adalah struktur dari program C

```
main()
{
        statemen-statemen;          }   fungsi utama
}

fungsi_fungsi_lain()
{
        statemen-statemen;          }   fungsi-fungsi lain yang
}                                       ditulis oleh pemrogram
```

Bahasa C dikatakan sebagai bahasa pemrograman terstruktur karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagiannya *(subroutine)*. Fungsi-fungsi yang ada selain fungsi utama (*main()*) merupakan program-program bagian. Fungsi-fungsi ini dapat ditulis setelah fungsi utama atau diletakkan di file pustaka *(library)*. Jika fungsi-fungsi diletakkan di file pustaka dan akan dipakai di suatu program, maka nama file judulnya *(header file)* harus dilibatkan dalam program yang menggunakannya dengan *preprocessor directive* berupa *#include.*

1.5.Pengenalan Program C

1.5.1. Pengenalan Fungsi-Fungsi Dasar

a. Fungsi *main()*

Pada program C, *main()* merupakan fungsi yang istimewa. Fungsi *main()* harus ada pada program, sebab fungsi inilah yang menjadi titik awal dan titik akhir eksekusi program. Tanda { di awal fungsi menyatakan awal tubuh fungsi dan sekaligus awal eksekusi program, sedangkan tanda } di akhir fungsi merupakan akhir tubuh fungsi dan sekaligus adalah akhir eksekusi program. Jika program terdiri atas lebih dari satu fungsi, fungsi *main()* biasa ditempatkan pada posisi yang paling atas dalam pendefinisian fungsi. Hal ini hanya merupakan kebiasaan. Tujuannya untuk memudahkan pencarian terhadap program utama bagi pemrogram. Jadi bukanlah merupakan suatu keharusan.

b. Fungsi *printf()*

Fungsi *printf()* merupakan fungsi yang umum dipakai untuk menampilkan suatu keluaran pada layar peraga. Untuk menampilkan tulisan

```
Selamat belajar bahasa C
```

misalnya, pernyataan yang diperlukan berupa:

```
printf("Selamat belajar bahasa C");
```

Pernyataan di atas berupa pemanggilan fungsi *printf()* dengan argumen atau parameter berupa string. Dalam C suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik-ganda (**"**). Perlu juga diketahui pernyataan dalam C selalu diakhiri dengan tanda titik koma (**;**). Tanda titik koma dipakai sebagai tanda pemberhentian sebuah pernyataan dan bukanlah sebagai pemisah antara dua pernyataan.

Tanda \ pada string yang dilewatkan sebagai argumen *printf()* mempunyai makna yang khusus. Tanda ini bisa digunakan untuk menyatakan karakter khusus seperti karakter baris-baru ataupun karakter *backslash* (miring kiri). Jadi karakter seperti `\n` sebenarnya menyatakan sebuah karakter. Contoh karakter yang ditulis dengan diawali tanda \ adalah:

`\"`	menyatakan karakter petik-ganda
`\\`	menyatakan karakter backslash
`\t`	menyatakan karakter tab

Dalam bentuk yang lebih umum, format *printf()*

```
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. Mengenai penentu format di antaranya berupa:

`%d`	untuk menampilkan bilangan bulat (integer)
`%f`	untuk menampilkan bilangan titik-mengambang (pecahan)
`%c`	untuk menampilkan sebuah karakter
`%s`	untuk menampilkan sebuah string

Contoh:

```
#include <stdio.h>

main( )
{
  printf("No    : %d\n", 10);
  printf("Nama  : %s\n", "Ali");
  printf("Nilai : %f\n",80.5);
  printf("Huruf : %c\n",'A');
}
```

1.5.2. Pengenalan Praprosesor #include

#include merupakan salah satu jenis pengarah praprosesor *(preprocessor directive)*. Pengarah praprosesor ini dipakai untuk membaca file yang di antaranya berisi deklarasi fungsi dan definisi konstanta. Beberapa file judul disediakan dalam C. File-file ini mempunyai ciri yaitu namanya diakhiri dengan ekstensi **.h**. Misalnya pada program `#include <stdio.h>` menyatakan pada kompiler agar membaca file bernama *stdio.h* saat pelaksanaan kompilasi.

Bentuk umum #include:

```
#include "namafile"
```

Bentuk pertama (`#include` <namafile>) mengisyaratkan bahwa pencarian file dilakukan pada direktori khusus, yaitu direktori file *include*. Sedangkan bentuk kedua (`#include` "namafile") menyatakan bahwa pencarian file dilakukan pertama kali pada direktori aktif tempat program sumber dan seandainya tidak ditemukan pencarian akan dilanjutkan pada direktori lainnya yang sesuai dengan perintah pada sistem operasi.

Kebanyakan program melibatkan file **stdio.h** (file-judul I/O standard, yang disediakan dalam C). Program yang melibatkan file ini yaitu program yang menggunakan pustaka I/O (input-output) standar seperti *printf()*.

1.5.3. Komentar dalam Program

Untuk keperluan dokumentasi dengan maksud agar program mudah dipahami di suatu saat lain, biasanya pada program disertakan komentar atau keterangan mengenai program. Dalam C, suatu komentar ditulis dengan diawali dengan tanda `/*` dan diakhiri dengan tanda `*/`.

Contoh :

```
/*
  Tanda ini adalah komentar
  tidak masuk dalam eksekusi program
*/

#include <stdio.h>

main()
{
 printf("Coba\n");     //Ini adl program pertama
}
```

Kesimpulan :

- Akar dari bahasa C adalah bahasa BCPL yang dikembangkan oleh Martin Richards pada tahun 1967.
- Bahasa C pertama kali digunakan pada komputer *Digital Equipment Corporation* PDP-11 yang menggunakan sistem operasi UNIX.
- C adalah bahasa yang standar, artinya suatu program yang ditulis dengan versi bahasa C tertentu akan dapat dikompilasi dengan versi bahasa C yang lain dengan sedikit modifikasi. Standar bahasa C yang asli adalah standar dari UNIX.
- Interpreter adalah suatu jenis penerjemah yang menerjemahkan baris per baris intsruksi untuk setiap saat, sedangkan kompiler merupakan jenis penerjemah cara kerjanya adalah menerjemahkan seluruh instruksi dalam program sekaligus.
- Program C pada hakekatnya tersusun atas sejumlah blok fungsi.
- Fungsi *main()* merupakan fungsi istimewa yang harus ada pada program, sebab fungsi inilah yang menjadi titik awal dan titik akhir eksekusi program.
- Fungsi *printf()* merupakan fungsi yang umum dipakai untuk menampilkan suatu keluaran pada layar peraga.

- *#include* merupakan salah satu jenis pengarah praprosesor *(preprocessor directive)* yang dipakai untuk membaca file yang di antaranya berisi deklarasi fungsi dan definisi konstanta.
- Untuk keperluan dokumentasi, di dalam program disertakan komentar yang ditulis dengan diawali dengan tanda `/*` dan diakhiri dengan tanda `*/`.

Latihan :

Buatlah potongan program untuk soal-soal di bawah ini

1. Apakah keluaran dari program di bawah ini :

```
#include <stdio.h>
main()
{
      printf("The black dog was big. ");
      printf("The cow jumped over the moon.\n");
 }
```

2. Gunakan pernyataan *printf()* untuk menampilkan (di layar) nilai dari sebuah variabel (misalkan namanya = **sum**) yang bertipe integer.

3. Gunakan pernyataan *printf()* untuk menampilkan (di layar) string `"Welcome"` yang diikuti dengan sebuah perintah ganti baris.

4. Gunakan pernyataan *printf()* untuk menampilkan (di layar) sebuah karakter dari variabel yang bertipe karakter (misalkan namanya = **letter**).

5. Gunakan pernyataan *printf()* untuk menampilkan (di layar) nilai dari sebuah variabel float (misalkan namanya = **discount**).

6. Gunakan pernyataan *scanf()* untuk membaca masukan sebuah nilai desimal dari keyboard dan memasukkannya ke sebuah variabel integer (misalkan namanya = **sum**).

7. Gunakan pernyataan *scanf()* untuk membaca masukan nilai float dari keyboard dan memasukkannya ke sebuah variabel float (misalkan namanya = **discount_rate**).

8. Gunakan pernyataan *scanf()* untuk membaca masukan sebuah karakter dari keyboard dan memasukkannya ke sebuah variabel karakter (misalkan namanya = **opr**).