Praktikum 1 (2/2)
PENGENALAN BAHASA C

A. TUJUAN

1. Mengenal sintaks dan fungsi-fungsi dasar dalam bahasa C
2. Mampu membuat flowchart untuk algoritma untuk memecahkan suatu masalah sederhana, selanjutnya mengimplementasikannya dalam bahasa C

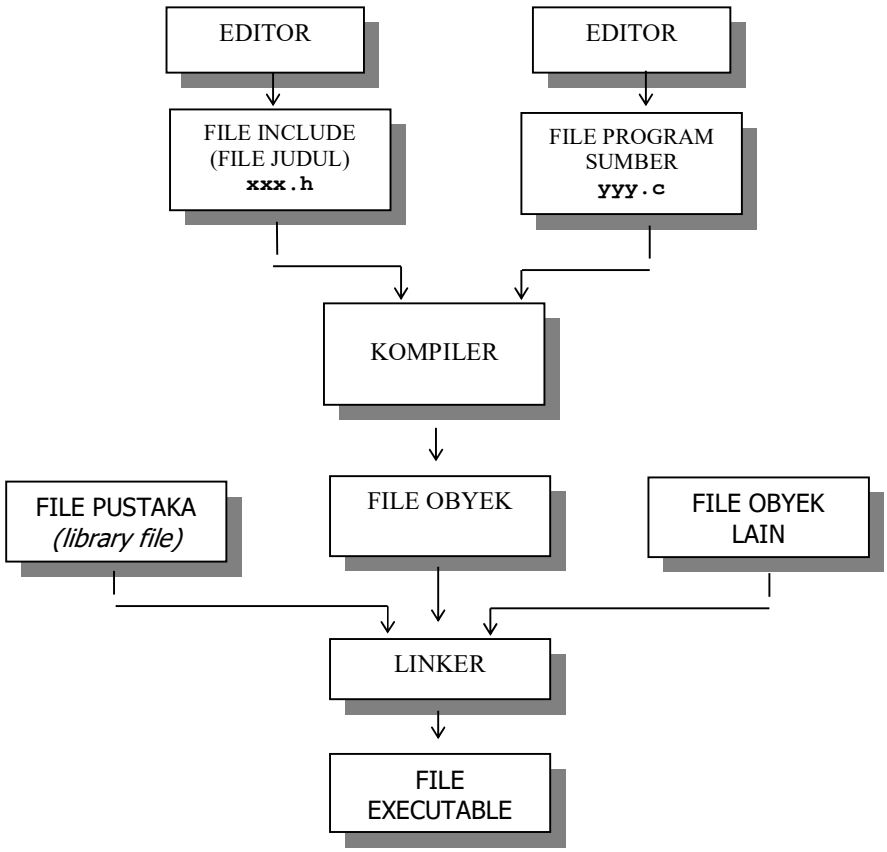
B. DASAR TEORI

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.

Standar bahasa C yang asli adalah standar dari UNIX. Sistem operasi, kompiler C dan seluruh program aplikasi UNIX yang esensial ditulis dalam bahasa 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.

Proses Kompilasi dan Linking Program C

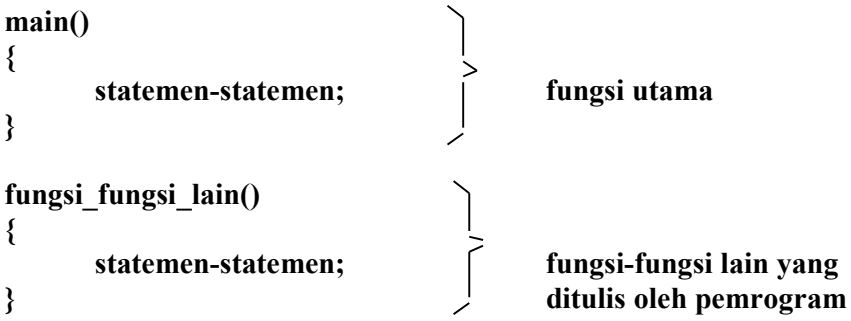
Proses dari bentuk *source program*, yaitu program yang ditulis dalam bahasa C hingga menjadi program yang *executable* ditunjukkan pada Gambar 1 di bawah ini.



Gambar 1 Proses Kompilasi-*Linking* dari program C

Struktur Penulisan Program C

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



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.*

Pengenalan Fungsi-Fungsi Dasar

a. Fungsi *main()*

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’);
}
```

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()*.

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
   untuk multiple lines
*/
#include <stdio.h>

main()
{
 printf("Coba\n");     //Ini komentar satu baris
}
```

C. PERCOBAAN

Implementasikan semua menggunakan bahasa pemrograman C

1. Ketik dan jalankan 6 contoh program yang ada dalam buku ***Programming in C – Stephen G. Kocha****n, Chapter 3* ***Compiling and Running Your First Program.*** Bandingkan keluaran yang didapatkan dengan keluaran yang sudah ditampilkan di bawah setiap contoh program tersebut

2. Tuliskan sebuah program untuk menghasilkan tampilkan kalimat-kalimat berikut ini

```
1. In C, lowercase letters are significant.
2. main is where program execution begins.
3. Opening and closing braces enclose program statements in a
   routine.
4. All program statements must be terminated by a semicolon.
```

3. Tebaklah, apa keluaran dari program berikut ini

```
#include <stdio.h>
int main (void)
{
      printf ("Testing...");
      printf ("....1");
      printf ("...2");
      printf ("..3");
      printf ("\n");
      return 0;
}
```

4. Tuliksan program yang mengurangkan 15 dari 87 dan menampilkan hasilnya, dengan pesan/kalimat yang sesuai

5. Temukan kesalahan sintaks pada program berikut. Selanjutnya ketik ulang jalankan programnya setelah memperbaiki semua kesalahan yang ditemukan

```
#include <stdio.h>

int main (Void)
(
      INT sum;
      /* COMPUTE RESULT
      sum = 25 + 37 - 19
      /* DISPLAY RESULTS //
      printf ("The answer is %i\n" sum);
      return 0;
}
```

6. Tebaklah, apa keluaran yang dihasilkan dari program berikut ini

```
#include <stdio.h>
int main (void)
{
      int answer, result;
      answer = 100;
      result = answer - 10;
      printf ("The result is %i\n", result + 5);
      return 0;
}
```

D. LAPORAN RESMI

1. Cetak listing program yang anda buat

2. Kerjakan soal-soal di bawah ini, dan sertakan jawaban Anda pada Laporan Resmi

   a. Berapakah nilai jawaban yang ditampilkan oleh program di bawah ini :

```
main()
{
  int jawab, hasil;
  jawab = 100;
  hasil = jawab - 10;

  printf("Jawabannya adalah %d\n", hasil + 6);
}
```

   b. Apakah keluaran dari potongan program di bawah ini

```
main()
{
   int value1, value2, sum;

   value1 = 35;
   value2 = 18;
   sum = value1 + value2;

   printf("The sum of %d and %d is %d\n", value1,value2,sum);
}
```

c. Program di bawah ini tidak berhasil di-compile karena masih terdapat beberapa kesalahan. Temukan paling sedikit 6 buah kesalahannya. Selanjutnya tampilkan keluaran, setelah program ini berhasil dijalankan.

```
main ()
{
  INT jumlah;

  /* PERHITUNGAN HASIL
  jumlah = 25 + 37 - 19;

  /* TAMPILKAN HASIL
  printf("Berapa hasil perhitungan 25 + 37 - 19 ?\n);
  printf("Jawabannya adalah %d\n" jumlah);
}
```

d. Buatlah program yang menerima masukan dua buah bilangan. Tampilkan keluaran berupa jumlah, rata-rata dan kuadrat dari kedua bilangan yang dimasukkan.

e. Program di bawah ini seharusnya menampilkan keluaran satu baris sbb :

```
c * c = 25,000000
```

Namun, belum berhasil karena masih ada beberapa kesalahan. Temukan minimal 3 kesalahan dalam program tersebut.

```
#include <Studio.h>
main ()
{
  float a, b, c;

  a = 3;
  b = 4.0;

  c = a * a + b * b
  printf("c * c = %d", c);
}
```

3. Berilah kesimpulan hasil praktikum.