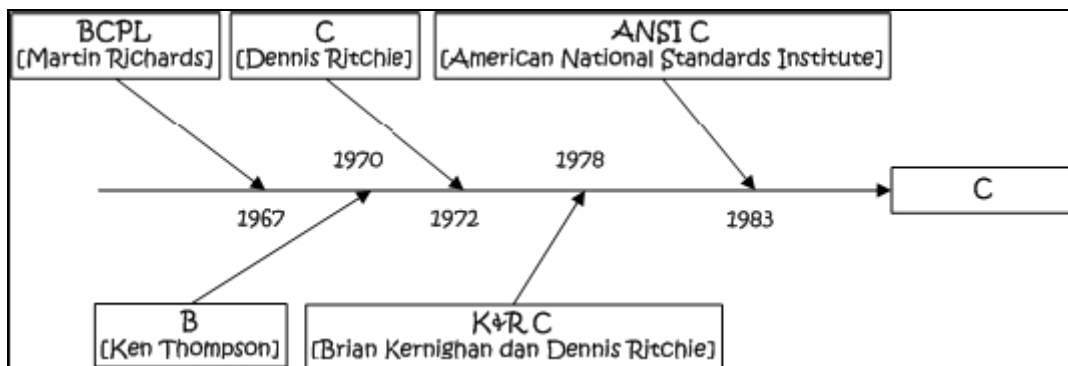


# BAB 1

## KONSEP DASAR BAHASA C

### 1. Sejarah dan Standar C

Akar dari bahasa C adalah bahasa BCPL yang dikembangkan oleh Martin Richard 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 1972-an di Bell Telephone Laboratories Inc. (sekarang adalah AT&T Bell Laboratories).



Gambar 1.1 Sejarah Bahasa C

Kepopuleran bahasa C membuat versi-versi dari bahasa ini banyak dibuat untuk komputer mikro. Untuk membuat versi-versi tersebut standar, ANSI (*American National Standards Institute*) kemudian menetapkan standar ANSI untuk bahasa C. Standar ANSI ini

didasarkan dari standar UNIX yang diperluas. Standar ANSI menetapkan sebanyak 32 buah kata-kata kunci (*keyword*) standar. Ke 32 kata kunci ini adalah :

Tabel 1.1 *Keyword dalam bahasa C*

<b>auto</b>	<b>break</b>	<b>case</b>	<b>char</b>	<b>const</b>	<b>continue</b>	<b>default</b>	<b>do</b>
<b>double</b>	<b>else</b>	<b>enum</b>	<b>extern</b>	<b>float</b>	<b>for</b>	<b>goto</b>	<b>if</b>
<b>int</b>	<b>long</b>	<b>register</b>	<b>return</b>	<b>short</b>	<b>signed</b>	<b>sizeof</b>	<b>static</b>
<b>struct</b>	<b>switch</b>	<b>typedef</b>	<b>union</b>	<b>unsigned</b>	<b>void</b>	<b>volatile</b>	<b>while</b>

## 2. Menjalankan Bahasa C dengan Turbo C++

Dengan asumsi di komputer kita sudah ada *software* bahasa C yaitu Turbo C++ yang tersimpan dalam folder TC, maka untuk menjalankan Turbo C++ dengan cara sebagai berikut :

1. Siapkan folder untuk menyimpan data program C kita misal kita buat folder di **D:\Naya**
2. Buka Windows Explorer dan buka folder TC
3. Buka folder BIN
4. Cari file *execution* (.EXE) yang bernama TC.EXE
5. Untuk menjalankannya maka double klik file tersebut, sehingga akan tampil menu utama Turbo C++ seperti gambar berikut ini.

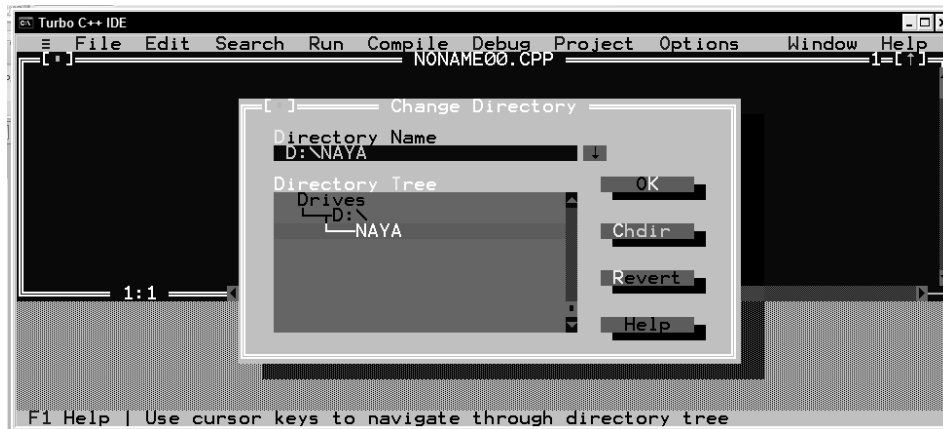


Gambar 1.2. *Editor Turbo C++*

6. Dari tampilan ini, maka tekan Enter atau klik OK. Maka editor Turbo C++ akan tampil. Disinilah tempat kita menyetikkan semua program C++ kita. Anda melihat disini ada menu **F**ile, **E**dit, **S**earch, **R**un, **C**ompile, **D**ebug, **P**roject, **O**ption, **W**indows dan **H**elp. Untuk memilih menu tersebut bisa dengan mouse atau dengan menekan ALT dan huruf depan dari menu yang ingin kita pilih. Misal **ALT+F** untuk memilih File. dst.
7. Sebelum menyetikkan program, sebaiknya arahkan direktori kerja ke folder yang sudah kita siapkan sebelumnya (langkah 1) dalam contoh ini adalah D:\Naya, dengan langkah sebagai berikut :
  - a. Pilih menu *File*

- b. Pilih *Change Dir...*
- c. Pilih *Drive* (Tekan Enter atau double klik)
- d. Pilih D (Tekan Enter atau double klik)
- e. Cari folder Naya (Tekan Enter atau double klik)
- f. Tekan OK

Maka folder D:\Naya sudah aktif sebagai folder kerja, artinya pada saat penyimpanan program dan pemanggilan program akan langsung mengarah pada folder D:\Naya, sehingga file program kita tidak tercampur dengan data file yang lain. Lakukan langkah ini setiap kali memulai Turbo C++.



Gambar 1.3. *Change Directory*

8. Mulailah mengetik program. Perintah-perintah penting dalam C++ dalam pengoperasian program antara lain :

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

- a. F1 → Help (Menu pertolongan)
- b. F2 → Untuk menyimpan program (*Save*)
- c. F3 → Untuk membuka program (*Open*)
- d. ALT-F3 → Untuk menutup program (*Close*)
- e. F5 → Melebarkan editor (*Maximize*)
- f. ALT-F5 → Untuk melihat hasil *running* program
- g. ALT-F9 → Mengkompile (*Compile*)
- h. F9 → Mengecek error program (*Make*)
- i. CTRL-F9 → Untuk menjalankan program (*Run*)
- j. F10 → Mengaktifkan menu
- k. ALT-X → Keluar dari Turbo C++
- l. Tombol-tombol yang lain akan dipelajari kemudian yaitu selama praktikum.

### 3. Alasan-alasan Menggunakan Bahasa C

Beberapa alasan mengapa bahasa C banyak digunakan, diantaranya adalah sebagai berikut :

1. Bahasa C tersedia hampir di semua jenis komputer
2. Kode bahasa C sifatnya adalah portabel

Aplikasi yang ditulis dengan bahasa C untuk suatu komputer tertentu dapat digunakan di komputer lain hanya dengan sedikit modifikasi.

3. Bahasa C hanya menyediakan sedikit kata-kata kunci

4. Proses *executable program* bahasa C lebih cepat

5. Dukungan pustaka yang banyak

Keandalan bahasa C dicapai dengan adanya fungsi-fungsi pustaka.

6. C adalah bahasa yang terstruktur

Bahasa C mempunyai struktur yang baik sehingga mudah untuk dipahami. C mempunyai fungsi-fungsi sebagai program bagiannya.

7. Selain bahasa tingkat tinggi, C juga dianggap sebagai bahasa tingkat menengah.

Bahasa C mampu menggabungkan kemampuan bahasa tingkat tinggi dengan bahasa tingkat rendah.

8. Bahasa C adalah kompiler

Karena C sifatnya adalah kompiler, maka akan menghasilkan *executable program* yang banyak dibutuhkan oleh program-program komersial.

Selain alasan tersebut ada beberapa alasan lain mengapa menggunakan bahasa C, yaitu:

**a. C adalah bahasa pemrograman yang memiliki portabilitas tinggi.**

Program C yang kita tulis untuk satu jenis platform, bisa kita kompilasi dan jalankan di platform lain dengan tanpa ataupun hanya sedikit perubahan. Ini bisa diwujudkan dengan adanya standarisasi ANSI untuk C.

**b. C adalah bahasa pemrograman dengan kata kunci (keyword) sedikit.**

Kata kunci disini adalah merupakan fungsi ataupun kata dasar yang disediakan oleh kompilasi suatu bahasa pemrograman. Hal ini membawa pengaruh semakin mudahnya

kita menulis program dengan C. Pengaruh lain dari sedikitnya kata kunci ini adalah proses eksekusi program C yang sangat cepat.

Adapun kekurangan yang biasa di Bahasa C antara lain :

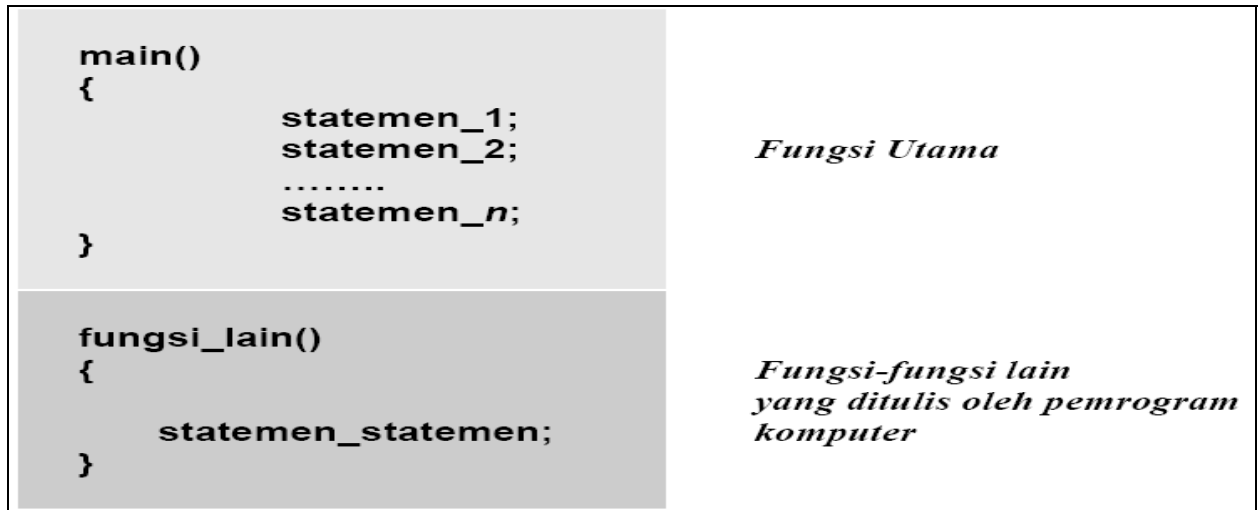
1. Banyaknya Operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai.
2. Bagi pemula pada umumnya akan kesulitan menggunakan pointer.

#### 4. Struktur Program C

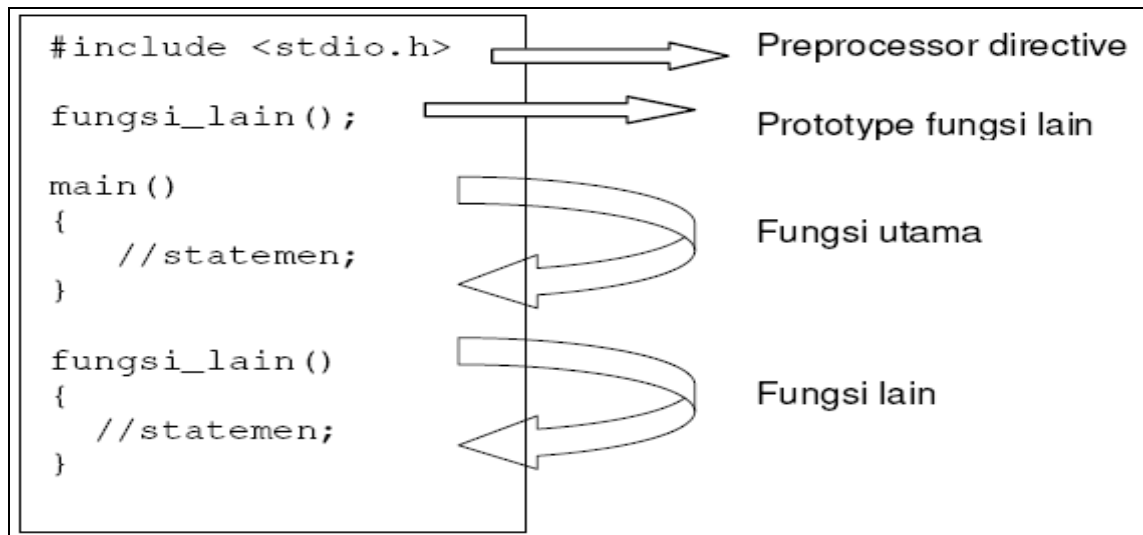
Untuk dapat memahami bagaimana suatu program ditulis, maka struktur dari program harus dimengerti terlebih dahulu, atau sebagai pedoman penulis program (*programmer*) bagaimana seharusnya program tersebut ditulis.

Struktur dari program C dapat dilihat sebagai kumpulan dari sebuah atau lebih fungsi-fungsi. Fungsi pertama yang harus ada di program C yang sudah ditentukan namanya, yaitu fungsi ***main()***. Artinya program C minimal memiliki satu fungsi (*fungsi main()*).

Fungsi-fungsi lain selain fungsi utama bisa dituliskan setelah atau sebelum fungsi utama dengan deskripsi prototype fungsi pada bagian awal program. Bisa juga dituliskan pada file lain yang apabila kita ingin memakai atau memanggil fungsi dalam file lain tersebut, kita harus menuliskan header file-nya, dengan preprocessor directive `#include`. File ini disebut file pustaka (*library file*). Struktur bahasa C dapat dilihat pada gambar berikut ini :



Gambar 1.4. Struktur program C



Gambar 1.5. Struktur program C Preprocessor Directive

Keterangan :

1. Dimulai dari tanda { hingga tanda } disebut tubuh fungsi / blok.
2. Tanda ( ) digunakan untuk mengapit argumen fungsi, yaitu nilai yang dilewatkan ke fungsi. Pada fungsi **main()** tidak ada argumen yang diberikan, maka tidak ada entri di dalam ( ).
3. Kata **void** menyatakan bahwa fungsi ini tidak memiliki nilai balik.
4. Tanda { menyatakan awal eksekusi program dan tanda } menyatakan akhir eksekusi program.
5. Didalam tanda { } bisa tergantung sejumlah unit yang disebut pernyataan (*statemen*).

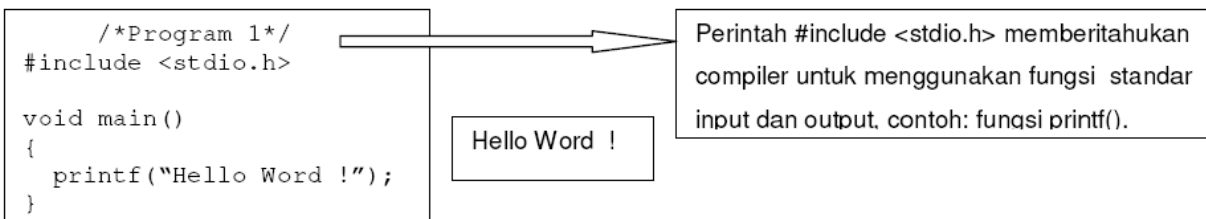
Umumnya pernyataan berupa instruksi untuk :

- a. Memerintah komputer melakukan proses menampilkan string ke layar.
- b. Menghitung operasi matematika.
- c. Membaca data dari keyboard.
- d. dll.

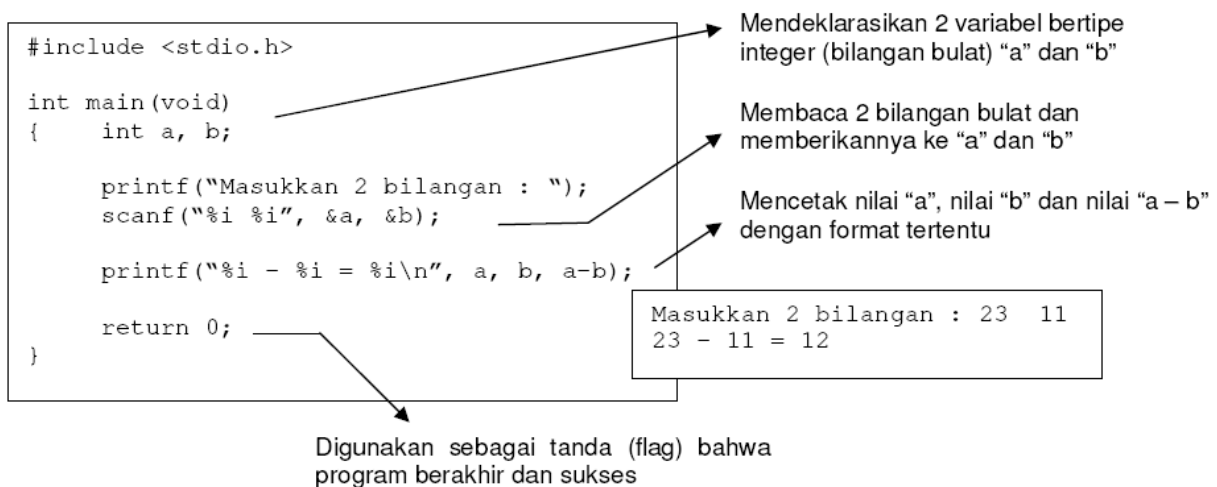
Bahasa C dikatakan sebagai bahasa pemrograman terstruktur, karena strukturnya menggunakan fungsi-fungsi sebagai program bagian (*subroutine*). Fungsi-fungsi selain fungsi utama merupakan program-program bagian. Fungsi-fungsi ini dapat ditulis setelah fungsi utama atau diletakkan di file pustaka (*library*).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

## 5. Program C Sederhana

Setelah mengetahui struktur dari suatu program C, berdasarkan struktur ini, maka dapat ditulis suatu program C yang sederhana dengan tidak mengalami banyak kesulitan. Berikut ini adalah suatu program C yang sederhana :



Program 1.1 Menampilkan Teks "Hello Word"

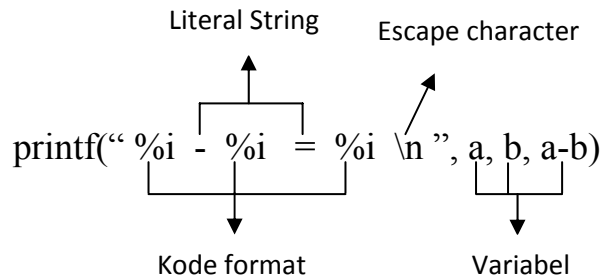


Program 1.2. Memasukkan 2 bilangan bulat dan mengurangkannya

**Pembahasan kedua program:**

1. Untuk memberikan keterangan program, suatu komentar dapat dituliskan bebas dimanapun dalam program C. Komentar atau keterangan program diawali dengan tanda `/*` dan diakhiri dengan `*/`. Contohnya : `/* Program 1 */`
2. Kedua program di atas menggunakan fungsi input dan output standar yaitu fungsi `printf()` dan `scanf()`, kedua fungsi tersebut telah disediakan oleh C dan merupakan file pustaka. Supaya fungsi tersebut dapat dikenali oleh program maka prototype dari fungsi-fungsi tersebut harus disebutkan dengan menggunakan *predecessor directive* **#include**. Prototype fungsi `printf()` dan `scanf()` terdapat pada file judul (*header file*) `stdio.h` (*extension file .h* menyatakan suatu *header file*).
3. Semua variabel yang akan digunakan dalam program C harus terlebih dahulu dideklarasikan atau dengan kata lain diperkenalkan. Deklarasi variabel ini dipergunakan untuk pemberitahuan tentang tipe data dan nama dari variabel yang akan digunakan. Contoh program ke-2 menggunakan dua buah variabel yaitu **a** dan **b**, dimana keduanya merupakan variabel yang bertipe integer (hanya dapat menampung bilangan bulat).
4. Salah satu cara untuk menampilkan hasil di layar adalah dengan menggunakan statemen yang dibentuk dari fungsi standar **printf()**. Dari kedua program di atas dapat dilihat bahwa fungsi `printf()` dapat dibagi menjadi dua bagian, yaitu bagian yang ditulis diantara tanda petik dua (“....”) dan bagian yang dituliskan di luar tanda petik dua. Bagian yang dituliskan di luar tanda petik akan menampilkan nilai dari variabel.

Pada contoh program 2:



Sedangkan untuk bagian yang berada diantara tanda petik dua adalah sebagai berikut:

- a. Kode format.

Kode format menunjukkan format dari variabel yang akan ditampilkan nilainya.

- b. Literal string.

Literal string adalah suatu konstanta string yang akan ditampilkan sesuai dengan apa yang dituliskan.

- c. Escape character. Karakter escape merupakan suatu konstanta karakter yang diawali dengan tanda back slash ( \ ). Karakter escape “\n” ang digunakan fungsi printf() digunakan untuk menggeser posisi kursor turun satu baris kembali ke kolom pertama.

5. Salah satu cara untuk memasukkan data dari keyboard adalah dengan menggunakan fungsi pustaka *scanf()*. Pada fungsi *scanf()*, bagian yang ditulis diantara tanda petik dua adalah kode format dan yang ditulis di luar tanda petik dua adalah variabel yang akan menerima nilai yang diketikkan dari keyboard. Untuk fungsi *scanf()*, nama variabelnya harus diawali operator pointer **&**, sehingga pada contoh program 2 di atas variabel **a** dan

variabel **b** dituliskan menjadi **&a** dan **&b**. Penggunaan operator pointer **&** tidak berlaku untuk variabel yang string.

## 6. File Judul (*Header File*)

File judul (*header file*) merupakan file yang berisi dengan *prototype* (judul, nama dan sintak) dari sekumpulan fungsi-fungsi pustaka tertentu. Jadi file ini hanya berisi dengan *prototype* dari fungsi-fungsi pustaka, sedangkan fungsi-fungsi pustakanya sendiri disimpan di file pustaka (*library file* dengan *extention file* .LIB). Misalnya *prototype* dari fungsi-fungsi pustaka **printf()** dan **scanf()** terdapat di file judul **stdio.h**, sehingga jika fungsi-fungsi ini akan digunakan di program, maka nama file judulnya harus dilibatkan dengan menggunakan preprocessor **#include**. File judul **stdio.h** berisi *prototype* fungsi-fungsi pustaka untuk operasi *input* dan *output standar*. Ada dua cara untuk melibatkan file judul di suatu program C, yaitu sebagai berikut :

```
#include <stdio.h>
```

dan

```
#include "stdio.h"
```

## 7. Komentar Program

Komentar merupakan bagian terpenting dari program. Kehadirannya sangat membantu pemrograman ataupun orang lain dalam memahami program, karena berupa penjelasan-penjelasan mengenai program atau bagian-bagian program. Hal ini penjelasannya bisa berupa :

1. Tujuan / fungsi program
2. Saat program dibuat atau direvisi
3. Keterangan-keterangan lain tentang kegunaan sejumlah pernyataan dalam program.

Pada C suatu komentar diawali dengan tanda ( /\* ) dan diakhiri dengan tanda ( /\* ). Semua string atau tulisan yang terletak sesudah tanda /\* hingga akhir baris dengan sendirinya akan diperlukan sebagai keterangan. Bagi kompilator hal ini tidak berguna dan akan diabaikan pada saat kompilasi.

*Contoh :*

```
/*-----*/  
/* PROGRAM LATIHAN PERTAMA          */  
/* Program menggunakan perintah printf dan scanf      */  
/*-----*/
```

## 8. Statement

Suatu statemen (*statement*) adalah pernyataan yang menyebabkan suatu tindakan akan dilakukan oleh komputer. Tindakan tersebut dapat berupa tindakan untuk menghitung, menampilkan hasil, menerima input data, mengendalikan proses program dan lain-lain. Suatu statemen di bahasa C ditulis dengan diakhiri oleh tanda titik koma (;).

Contoh :

```
X = X + 1;  
printf(" Nilai X = %f\n",X);
```

## 9. Karakter Pembentuk Program C

Program C ditulis dengan menggunakan sebagian dari karakter ASCII (*American Standard Code for Information Interchange*), yaitu:

1. Huruf besar: A – Z
2. Huruf kecil: a – z
3. Angka: 0 – 9
4. Karakter khusus

Tabel 1.2. Karakter Khusus

!	“	#	\$	%	&
‘	(	)	*	+	,
-	.	/	<	=	>
?	[	\	]	^	_
{	}	~	;	:	spasi

## 10. Penulisan Program C

Dalam menuliskan program dalam bahasa C spasi ke dalam ataupun spasi baris dan tab dapat digunakan untuk memudahkan pembacaan dan pembuatan program. Penggunaan spasi kosong ataupun baris kosong tidak akan mempengaruhi program karena compiler C akan mengabaikan spasi atau baris yang kosong. Penulisan dengan tanpa memberikan spasi ataupun jarak baris juga benar, tetapi penulisan seperti itu tidak terlalu baik karena selain menyulitkan pembacaan program juga menyulitkan programmer dalam penelusuran program, terutama untuk program dengan jumlah baris atau perintah yang banyak.

Bandingkan kedua program ini:

```
#include <stdio.h>
main(){ int a;
printf("Masukkan bilangan : "); scanf("%d",&a);
printf("\n%d kuadrat = %d", a, a*a); }
```

Masukkan bilangan : 5

5 kuadrat = 25

```
#include <stdio.h>

main()
{
    int a;

    printf("Masukkan bilangan : ");
    scanf("%d",&a);

    printf("\n%d kuadrat = %d", a, a*a);
}
```

Program 1.3. *Gaya Penulisan Program C*

Kedua program di atas mempunyai perintah yang sama dan akan menghasilkan keluaran yang sama, tetapi jika dilihat sekilas maka program pertama agak sulit dibaca dan dimengerti, sedangkan pada program kedua selain tampilan program terlihat lebih menarik juga memudahkan dalam pembacaan dan pemahaman program.

Suatu statemen di program C dapat ditulis dalam beberapa baris penulisan. Akhir dari suatu baris yang menggunakan tanda ‘\’ menunjukkan bahwa baris selanjutnya merupakan baris sambungannya. Contoh:

```
printf("ini adalah contoh pemecahan statemen karena statemen \  
yang cukup panjang untuk dituliskan dalam satu baris saja \n");
```

Atau dapat ditulis sebagai berikut :

```
printf("ini adalah contoh pemecahan statemen karena statemen" \  
"yang cukup panjang untuk dituliskan dalam satu baris saja \n");
```

**Hal yang penting:**

- Setiap statemen atau perintah dalam bahasa C harus diakhiri titik koma (;).
- C merupakan bahasa dengan format yang cukup bebas, maksudnya anda dapat memecah perintah menjadi dua baris tetapi jangan meletakkan titik koma diantaranya. Selain itu dalam satu baris juga diperbolehkan terdiri dari dua atau lebih perintah, tetapi ingat harus meletakkan titik koma untuk memisahkan perintah-perintah tersebut.
- Bahasa C merupakan bahasa yang *case sensitive*. Maksudnya adalah penulisan huruf besar akan dianggap berbeda dengan huruf kecil. Contoh, kata **int** tidak sama dengan **INT** dan tidak sama dengan **Int**.
- Semua kata kunci (keywords) dan fungsi-fungsi pustaka standar (standard library function) menggunakan huruf kecil.
- Komentar dalam program dapat ditulis dengan diawali tanda /\* dan diakhiri tanda \*/ atau hanya

**11. Preprocessor Directive #define**

Bahasa C menyediakan *preprocessor directive #define* untuk mendefinisikan konstanta, makro maupun nama. *Preprocessor directive #define* dapat diletakkan di dalam program yang sama atau diletakkan di file terpisah dengan programnya.

Contoh penggunaan *preprocessor directive* **#define** untuk mendefinisikan sebuah konstanta nama dengan nilai tetap “Naya Kartika Ramadhani”.

```
#include <stdio.h>
#define nama "Naya Kartika Ramadhani"
main()
{
    printf(nama);
}
```

Naya Kartika Ramadhani

### Latihan Program 1

```
#include <stdio.h>
#include <conio.h>

/*
  Perintah pencetakan string ke layar
  Sintax : printf("pesan yang tampil");
  NB : \n berfungsi untuk meletakkan pencetakan pada baris baru
*/
main()
{
    clrscr();
    printf("Selamat Datang di Turbo C++ \n");
    printf("Di Universitas Bina Darma \n");
    printf("Palembang");
    getch();
    return 0;
}
```

**Soal 1.A**

1. Dalam perkembangan bahasa C, ada tokoh dengan nama Ken Thompson, siapakah dia dan apa peranannya dalam mengembangkan bahasa C.
2. Apa yang dimaksud dengan *keyword* bahasa C++?
3. Sebutkan 5 *keyword* bahasa C++!
4. Untuk menjalankan Turbo C maka kita eksekusi file TC.EXE. Dimanakah letak file TC.EXE tersebut dalam *Software* Turbo C++?
5. Sebutkan menu-menu utama dalam Turbo C++!
6. Mengapa kita perlu menyiapkan folder kerja dalam mengoperasikan Turbo C++?
7. Bagaimana cara mengarahkan area kerja ke folder kerja kita (misal : D:\DataC) ?
8. Jelaskan fungsi tombol F9 dan kombinasinya dalam Turbo C++!
9. Sebutkan beberapa alasan mengapa kita menggunakan bahasa C++?
10. Tuliskan struktur penulisan program bahasa C!
11. Apa yang dimaksud dengan *Header File*?
12. Bagaimana cara kita membuat komentar dalam program?
13. Bahasa C merupakan bahasa yang *case sensitive*, apa maksudnya?
14. Sebutkan ketentuan dalam membuat atau menentukan nama variabel (*identifier*) ?
15. Jelaskan apa fungsi dari *preprocessor directive #define*?

**Soal 1.B**

Buat program untuk menampilkan Biodata diri Anda!

```
-----  
                        BIODATA  
-----  
Nama      : <nama>  
Alamat    : <alamat>  
Tempat Lahir : <tmp_lahir>  
Tanggal Lahir : <tgl_lahir>  
Hobby     : <hobby>  
-----
```