BAB III
STRUKTUR PROGRAM C

SEJARAH C

Bahasa pemrograman C disusun berdasarkan dua bahasa terdahulu, yaitu BCPL dan B. BCPL dikembangkan pada tahun 1967 oleh Martin Richards sebagai bahasa untuk menulis perangkat lunak sistem operasi dan kompilator (*compiler*). Ken Thompson memodelkan beberapa fitur di dalam bahasa B bersama-sama dengan rekan-rekannya yang menggunakan bahasa BCPL untuk membuat sistem operasi UNIX yang pertama di Bell Laboratories pada tahun 1970 dengan menggunakan komputer DEC PDP-7.

Bahasa C dikembangkan lebih lanjut dari bahasa B oleh Dennis Ritchi di Bell Laboratories dan pertama kali diimplementasikan dalam komputer DEC PDP-11 pada tahun 1972 yang menggunakan sistem operasi UNIX. Pada perkembangannya saat ini, hampir semua sistem operasi baru ditulis dalam C atau C++, dan telah menjadi bahasa pemrograman yang populer. C yang pertama kali muncul ini saat ini dikenal dengan nama "C tradisional" yang dipublikasikan oleh Kernighan dan Ritchi pada tahun 1978. Dalam perkembangannya telah muncul C dalam berbagai versi, antara lain ANSI C, Borland C, Turbo C, dan sebagainya.

TAHAPAN PEMROGRAMAN C

Pemrosesan program C umumnya melalui lima tahapan, yaitu *edit*, *preprocess/compile*, *link*, *load*, dan *execute*. Setiap tahap dilakukan satu per satu (bila menggunakan UNIX), atau ada beberapa tahap yang dapat digabungkan (misalnya menggunakan Turbo C dimana empat tahap terakhir dapat dilakukan dalam satu waktu sekaligus). Berikut adalah penjelasan dari setiap tahapan:

1. Tahap edit adalah proses penulisan program dengan menggunakan program editor dan kemudian disimpan ke dalam penyimpanan sekunder, misalnya disk. File program C diberi nama dengan menambahkan ekstensi **.c**. Dua program editor tersedia di dalam UNIX, yaitu **vi** dan **emacs**; sedangkan jika menggunakan Borland, Turbo, atau yang sejenis, program editor telah tersedia di dalamnya dalam satu paket.
2. Tahap selanjutnya adalah mengkompilasi program, dimana kompilator akan menerjemahkan program C yang telah ditulis ke dalam kode bahasa mesin (sering disebut *object code* = **.obj**). Di dalam sistem C, bagian program yang disebut *preprocessor* (dipanggil melalui *preprocessor directives*) secara otomatis diproses sebelum tahap penerjemahan dimulai.
3. Proses *link* dilakukan untuk mengkombinasikan kode obyek (obj) dengan pustaka C yang tersedia agar menghasilkan file program lengkap yang dapat dieksekusi oleh komputer. Di dalam UNIX, proses kompilasi dan link dilakukan bersamaan dengan menggunakan program yang bernama **cc** dan menghasilna file dengan ekstensi **.out**. Pada IBM PC, file yang dihasilkan mempunyai ekstensi **.exe**.
4. Tahap selanjutnya adalah memanggil file program hasil langkah (3) ke dalam memori untuk dapat dijalankan oleh komputer.
5. Tahap terakhir adalah mengeksekusi program tersebut.

STRUKTUR PROGRAM C

Perhatikan contoh program C di bawah ini.

```
/* PROGRAM MENCARI BILANGAN TERBESAR DARI 2 BILANGAN BULAT */

#include <stdio.h>                              /* preprocessor directives */

int maksimum(int, int);                         /* prototype functions */
void tulis(int);

main() {                                        /* main function */
    int Nilai1, Nilai2, NilaiMaks;              /* symbolic variables */
    printf("\nKetikkan dua bilangan bulat : ");
    scanf("%d %d", &Nilai1, &Nilai2);

    NilaiMaks = maksimum(Nilai1, Nilai2);       /* function call */
    tulis (NilaiMaks);

    return 0;
}

int maksimum(int x1, int x2) {                  /* user defined function */
    if (x1>x2)
        return x1;
        else
        return x2;
}

void tulis (int x) {                            /* user defined function */
    printf("\nHasilnya adalah : %d", x);
}
```

Di dalam program C, suatu aksi dilakukan dengan suatu **ekspresi** (*expression*), misalnya 'hasil = 9'. Ekspresi yang diakhiri dengan titik-koma (;) membentuk **pernyataan** (*statement*) seperti contoh berikut:

```
int Nilai;                                      /* declaration statement */
Nilai = 4 + 5;                                  /* assignment statement */
```

Pernyataan-pernyataan dapat dikelompokkan secara logika menjadi suatu **fungsi** (*function*). Setiap program C harus mengandung sedikitnya sebuah fungsi, yang disebut dengan **main()**. Pernyataan pertama yang dieksekusi oleh program adalah main().

Sebuah fungsi terdiri dari empat bagian, yaitu tipe kembali (*return type*), nama fungsi (*function name*), daftar argumen (*argument list*), dan tubuh fungsi (*function body*). Tiga bagian pertama membentuk apa yang disebut dengan fungsi **prototipe** (*prototype function*). Daftar argumen dibatasi oleh sepasang kurung biasa (…), yang dapat terdiri dari nol atau lebih argumen yang masing-masing dipisahkan oleh tanda koma (,). Sedangkan tubuh fungsi dibatasi oleh sepasang kurung kurawal { … } dan terdiri dari baris-baris pernyataan.

PREPROCESSOR DIRECTIVES

Pada contoh program C sebelumnya, terdapat baris perintah sebagai berikut:

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

File stdio.h disebut file **header** yang mengandung informasi berupa variabel-variabel dan fungsi-fungsi yang diperlukan untuk penggunaan librari standard (*standard library*). Untuk mengakses variabel atau fungsi yang ada di dalam librari standard, kita harus menyertakan file header yang sesuai ke dalam program, yaitu dengan menggunakan apa yang disebut dengan *include directive*. Sedangkan program yang memproses directive tersebut disebut dengan ***preprocessor directives***.

Perhatikan contoh-contoh berikut:

```
#include <stdio.h>          /* file header tersimpan di default directory */
#include <stdlib.h>

#include "a:myfile.h"       /* file header tersimpan di drive a: */
```

KATA KUNCI

Kata kunci (keywords) adalah kata di dalam pemrograman yang tidak dapat digunakan untuk kepentingan lain kecuali yang telah ditetapkan oleh kompilator. Dengan demikian kita tidak dapat membuat variabel dengan nama seperti yang tercantum pada kata kunci, atau kepentingan-kepentingan lainnya. Ada 32 kata kunci di dalam C, yaitu:

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

TIPE DATA

Setiap program umumnya mempunyai data, dan setiap data memiliki tipe tertentu. Di dalam program C terdapat dua kelompok besar tipe data, yaitu *integral types* (terdiri dari **char**, **short**, **int**, dan **long**) dan *non-integral types* (yaitu **float** dan **double**). Penjelasan masing-masing tipe ini seperti yang diuraikan pada tabel berikut:

No.	Tipe	Keterangan
1.	char	berisi sebuah karakter atau kode ASCII : -128 hingga +127
2.	int	bilangan bulat : -32768 hingga 32767
3.	short	sama dengan int
4.	float	bilangan riil dengan ketelitian 10^{-38} hingga 10^{+38}
5.	double	bilangan riil dengan ketelitian 10^{-308} hingga 10^{+308}
6.	long	untuk meningkatkan kapasitas, misalnya: long int → -2147483648 hingga +2147483647

Tipe data short dan int dapat berbentuk signed maupun unsigned.

LITERAL CONSTANT

Suatu nilai data dalam program kadang-kadang berbentuk **literal** (hanya berupa nilai, misalnya 3, 4.15, dan sebagainya) dan bersifat **constant** (tidak dapat diubah). Setiap literal mempunyai tipe, misalnya: 3 bertipe int, 4.15 bertipe float, dan sebagainya. Nilai literal juga bersifat *nonaddressable* karena alamatnya tidak dapat diakses.

Ada tiga tipe literal constant ini, yaitu:

1. literal integer constants, untuk menuliskan konstanta bilangan bulat. Konstanta ini dapat dituliskan dalam notasi **desimal** (misalnya 20, -5), **oktal** yang dimulai dengan karakter '0' (misalnya 024), dan **heksadesimal** yang dimulai dengan karakter '0X' (misalnya 0X14). Jika tidak menggunakan deklarasi tipe apapun, otomatis konstanta yang dituliskan bertipe integer bertanda (*signed int*). Tipe *long* dan *unsigned* dapat ditambahkan di belakang konstanta, misalnya 123U, 325UL, 2L, dan sebagainya.
2. literal floating point constants, yaitu nilai konstanta jenis bilangan riil yang dapat dituliskan dalam notasi **ilmiah** (misalnya 1.0E-3 yang berarti 1.0×10^{-3}), dan notasi **desimal** (misalnya 3.14F, 0.0).
3. literal character constants, yaitu konstanta karakter yang ditulis di dalam tanda kutip ('..'), misalnya 'a', '2', ',', '' (*empty character*), dan sebagainya. Disamping itu, terdapat karakter yang bersifat **nonprintable character** (tidak dapat dicetak, melainkan hanya sebagai kode operasi tertentu), yang ditulis dengan **escape sequences**, seperti yang diuraikan pada tabel berikut.

No.	Karakter	Penjelasan
1.	\n	new line
2.	\t	horizontal tab
3.	\r	carriage return
4.	\a	alert, sound bell system

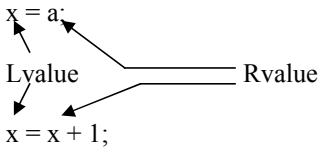
5.	\\	backslash
6.	\"	double quote

4. string literal constants, yang terdiri dari nol atau lebih karakter, dan ditulis di dalam tanda kutip dobel (double quotes), yaitu "…..", misalnya "" (null string='\0'), "a" (sama dengan 'a','\0') , "komputer" (sama dengan 'k','o','m','p','u','t','e','r','\0'), dan sebagainya.

SYMBOLIC VARIABLE

Setiap program umumnya mengandung satu atau lebih variabel untuk menyimpan suatu nilai, misalnya 'Nilai1', 'Nilai2', 'NilaiMaks', 'x1', dan 'x2' contoh program C sebelumnya. Identifikasi variabel ini dibuat oleh pengguna sendiri dan bersifat *addressable* karena alamatnya dapat diakses melalui program.

Ada dua nilai yang berkaitan dengan sebuah variabel, yaitu Rvalue dan Lvalue. Rvalue adalah nilai data yang disimpan pada lokasi memori, sedangkan Lvalue adalah alamat dari lokasi memori dimana nilai data itu disimpan. Perhatikan contoh berikut:



Di dalam sebuah program, setiap variabel yang dibuat harus dideklarasikan dengan menentukan tipe variabel yang bersangkutan. Berikut adalah contoh deklarasi variabel:

```
int hari, bulan, tahun;
unsigned long jarak;
int nilai = 0;
double harga = 15.9, potongan = 0.2;
float gaji;
unsigned x = abs(-108);
```

MENULIS DAN MEMBACA

Fungsi standard yang sering digunakan di dalam program C untuk menuliskan nilai ke layar dan membaca nilai dari papan ketik adalah **printf** dan **scanf**. Kedua fungsi ini dapat dilihat penggunaannya pada contoh program C sebelumnya, yaitu:

```
printf("\nKetikkan dua bilangan bulat : ");
printf("\nHasilnya adalah : %d", x);
scanf("%d %d", &Nilai1, &Nilai2);
```


Fungsi printf yang pertama mempunyai satu argumen, yang bertujuan untuk menuliskan suatu string kalimat 'Ketikkan dua bilangan bulat :' di layar. Kalimat ini diawali dengan dengan karakter '\n' yang berarti bahwa kursor terlebih dahulu dipindahkan ke baris baru sebelum menuliskan kalimat tersebut. Sedangkan fungsi printf yang kedua mempunyai dua argumen, dimana argumen pertama adalah string kalimat yang akan ditulis ke layar, dan argumen kedua adalah nilai suatu variabel yang akan ditulis (x). Nilai x ini ditulis dengan menggunakan notasi desimal (karakter '%d' pada argumen pertama).

Fungsi kedua adalah scanf yang digunakan untuk membaca nilai suatu variabel yang diketikkan dari papan ketik saat program diproses. Pada contoh ini, nilai yang dibaca terdiri dari dua buah dan masing-masing akan dimasukkan ke dalam variabel Nilai1 dan Nilai2. Karakter '%d' menunjukkan bahwa nilai yang diketik harus dalam notasi desimal, sedangkan karakter '&' menunjukkan bahwa argumen fungsi scanf berupa **reference variable** (akan dibahas pada bab selanjutnya).

OPERATOR ARITMATIKA

Program komputer banyak melakukan perhitungan aritmatika, yang terdiri dari **operator** (+, -, *, /, dan %) dan **operand**. Misalnya ekspresi 3 + 7 terdiri dari operator + dan dua buah operand (disebut *binary*), yaitu 3 dan 7. Disamping itu, ada pula operator yang hanya diterapkan pada sebuah operand (disebut *unary*), misalnya -5, +12.7, dan sebagainya. Operator aritmatika beserta notasinya seperti tercantum pada tabel berikut:

Operasi	Operator	Ekspresi Aljabar	Ekspresi C
Penjumlahan	+	f + 7	f + 7
Pengurangan	-	p - c	p - c
Perkalian	*	bm	b * m
Pembagian	/	x : y	x / y
Modulo	%	r mod s	r % s

Tanda kurung '(…)' digunakan untuk memberi peringkat lebih tinggi dari proses pengolahannya, misalnya a*(b+c)+c*(d-e). Untuk suatu ekspresi yang mempunyai banyak operator, maka operator yang mana yang akan diproses terlebih dahulu tergantung pada dua hal, yaitu **precedence** dan **associativity**. Precendence menentukan urutan operasi dari operator-operator dalam ekspresi majemuk. Operator dengan tingkat precedence lebih tinggi akan dievaluasi terlebih dahulu. Sedangkan associativity menentukan arah evaluasi dari operator bila mempunyai precedence yang sama, apakah dari kiri ke kanan (disimbolkan **LR**), atau sebaliknya (disimbolkan **RL**). Tabel lengkap yang memuat precedence dan associativity dari semua operator yang dikenal di dalam program C akan dicantumkan pada bagian akhir dari bab ini.

OPERATOR LOGIKA, RELASIONAL, DAN KESAMAAN

Disamping operator aritmatika, juga dikenal operator logika, relasional, dan kesamaan, dimana ekspresi yang dibentuk menghasilkan dua nilai khas, yaitu salah (**false**, dalam C diberi nilai 0) dan benar (**true**, dalam C bernilai ≠ 0). Operator yang dimaksud seperti terlihat pada tabel berikut:

Aljabar	Operator C	Contoh	Arti
KESAMAAN			
=	= =	x = = y	x sama dengan y ?
≠	!=	x != y	x tidak sama dengan y ?
LOGIKA			
>	>	x > y	x lebih besar dari y ?
<	<	x < y	x lebih kecil dari y ?
≥	>=	x >= y	x lebih besar atau sama dengan y ?
≤	<=	x <= y	x lebih kecil atau sama dengan y ?
RELATIONAL			
∩	&&	(x>0) && (x<3)	x > 0 dan x < 3
∪	||	(x<0) | (x>3)	x < 0 atau x > 3
~	!	! (x < 0)	x tidak lebih kecil dari 0

OPERATOR PENUGASAN (*ASSIGNMENT OPERATOR*)

Pemberian suatu nilai konstanta dan/atau ekspresi ke dalam suatu variabel dibuat dengan menggunakan operator penugasan, yaitu sama dengan ('='). Operand sebelah kiri dari operator penugasan ini haruslah suatu Lvalue, dan tipe dari hasil adalah tipe dari operand sebelah kirinya. Contoh untuk hal ini adalah:

```
a = 5;
b = a = 10;
c = c + 5;
```

COMPOUND ASSIGNMENT OPERATOR

Pernyataan 'c = c + 5' di atas di dalam program C dapat disederhanakan menjadi c += 5. Operator '+=' yang digunakan untuk menyederhanakan tersebut disebut dengan compound assignment operator. Semua operator aritmatika dapat digunakan untuk model seperti ini, yaitu +, -, *, /, dan %. Disamping itu juga operator lainnya seperti <<, >>, &, dan ^ dapat digunakan dengan cara yang sama. Perhatikan contoh berikut:

```
c += 5;          /* artinya c = c + 5 */
x /= 2;          /* artinya x = x / 2 */
jumlah %= 2;     /* artinya jumlah = jumlah % 2 */
```


OPERATOR INCREMENT DAN DECREMENT

Program C memberikan kemudahan notasi untuk menambah atau mengurangi satu nilai pada suatu variabel bilangan bulat (int). Bentuk operator yang dapat digunakan ada dua, yaitu bentuk **prefix** (++var atau --var) yang berarti bahwa nilai ekspresinya sama dengan nilai variabel setelah diubah; dan bentuk **postfix** (var++ atau var--) yang berarti bahwa nilai ekspresinya sama dengan nilai variabel sebelum diubah. Perhatikan penjelasan pada contoh berikut:

```
c = 5;          /* c bernilai 5 */
c++;            /* c++ tetap bernilai 5, sedangkan c selanjutnya bernilai 6 */
++c;            /* ++c dan c itu sendiri bernilai 7 */
```

Untuk lebih jelasnya, proses program berikut dan perhatikan keluarannya.

```
#include <stdio.h>
main() {
    int a = 10;
    printf("\n%d", a += 3);
    printf("\n%d", a -= 3);
    printf("\n%d", a *= 3);
    printf("\n%d", a /= 3);
    printf("\n%d", a %= 3);
    printf("\n%d", a);
    printf("\n%d", ++a);
    printf("\n%d", a);
    printf("\n%d", a++);
    printf("\n%d", a);
    printf("\n%d", --a);
    printf("\n%d", a);
    printf("\n%d", a--);
    printf("\n%d", a);
    return 0;
}
```

LATIHAN 3.
Terjemahkan setiap jawaban algoritme pada LATIHAN 1 ke dalam bahasa pemrograman C dan proses program tersebut menggunakan komputer.