BAB II

STRUKTUR BAHASA C

2.1. StrukturPenulisanBahasa C

Sebagaimanatradisidalambelajarbahasakomputeradalahdimulaidenganmembuat program Hello World, perhatikankodingberikut :

```
// Menampilkantulisan Hello World
#include <stdio.h>
void main() {
    printf(“Hello World”);
}
```

Jika program diatasdijalankan, maka di layarakantercetaktulisan "Hello World!",Marilahkitaanalisabagian program tersebutsatu per satu.

// Menampilkantulisan Hello World

Iniadalahbaris comment (komentar/keterangan). Semuabaris yang dimulaidengantanda (//) dianggapsebagai comment.

#include <stdio.h>

Perintah yang dimulaidengantanda (#) adalah directives (petunjuk) untuk preprocessor.Barisinitidakakandieksekusi, tapimerupakan directives bagi compiler. Dalamhalinikalimat #include <iostream.h>memberitahukankepada preprocessor compiler untuk include (memasukan) header file standariostream. File iniberisideklarasi input-output standar library didalamBahasa C yang diperlukanpadabagianberikutnyadalam program ini.

void main ()

Barisinimerupakandeklarasi main function (fungsiutama). Main function merupakantitikdimanadimana program akanmulaidijalankan. mainselaludiikutiolehtandakurung () karenamerupakansebuah function. Padabahasa Csemuamodular adalahfunction.Isi dari function main diapitdengantandakurawal({}).

printf (“Hello World”);

printfmerupakan standard output padabahasa C (biasanyakelayar), yang dalamhaliniakanmencetaktulisan "Hello World". printfdideklarasikandalam header file stdio.h, sehinggauntukmemanfaatkannyaperlu di #include.

Catatan :setiapperintah C++ diakhiridengankarakter semicolon (;).

2.2. Komentar/Keterangan

Ketikaandasedangmenulissebuah program, segalanyaadalahjelastentangpada yang sedangandakerjakan.Tetapisetelahsatubulan, andakembalimelihat program tersebut, mungkinsajaandatelahlupadanmerasabingung.Untukmengurangikeraguantersebut, andaperlumembuatkomentarsecukupnyapadakodeanda.Komentar/keteranganadalahbagiandari source code yang akandiabaikanoleh compiler.

Ada duacarapenulisankomentarpadabahasa C:

```
// bariskomentar
/* blokkomentar */
```

Yang pertamaadalahmengawalisetiapbariskomentardengantanda (//), sedangkanbentukkeduamengawalikomentardengan /* dandiakhiridengan */ ,jeniskomentarinicocokuntukkomentar/keterangan yang lebihdarisatubaris.

Perhatikancontohberikut :

```
/* Menampilkantulisan Hello World
dan Hello Bandung */
#include <iostream.h>
void main () {
    printf("Hello World \n"); // menampilkan Hello World
    printf("Hello Bandung"); // menampilkan Hello Bandung
}
```

2.3. Variable

Cobabayangkankalausayamemintaandauntukmengingatangka 8, danangka 2, kemudiansayamemintaandauntukmenjumlahkanangkapertamadengan 1, sehinggamenjadi 9 (8+1), danakhirnyahasilnyadikurangidenganbilangankedua (2).

Proses diatassecarakomputerdapatditulissebagaiberikut :

```
a = 8;
b = 2;
a = a + 1;
result = a - b;
```

Dalamhalini a dan b adalah variable, jadi variable di program komputermenyerupai variable padamatematika.Sehingga, kitadapatmendefinsikan variable sebagaibagiandarimemoriuntukmenyimpannilai, dalamhalini a danb disebutsebagai identifier (pengenal).

2.4. Identifiers

Suatu identifier harusmemenuhisyaratberikut :

a) Terdiridarihuruf, angkadan symbol garisbawah(_).
b) Untuk compiler tertentumaksimum 32 karakter.
c) Tidakbolehpakaispasiatautandabacalainnya .
d) Bolehdiawalidengan symbol garisbawah(_), tetapibiasanyadicadangkanuntuk external link.
e) Tidakbolehsamadengan keyword (kata kunci).

Key word standarmenurut ANSI-C++ yang tidakbolehandagunakansebagai identifier:

asm, auto, bool, break, case, catch, char, class, const, const_cast, continue, default, delete, do, double, dynamic_cast, else, enum, explicit, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, operator, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_cast, struct, switch, template, this, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t

Operator lainnya yang tidakbolehdigunakansebagai identifier:

and, and_eq, bitand, bitor, compl, not, not_eq, or, or_eq, xor, xor_eq

Sangatpentingdiingat: Bahasa C++ language adalah "case sensitive", haliniberartibahwa identifier yang ditulisdenganhurufbesardankeciladalahtidaksama. Jadi, variable HASILtidaksamadengan variable hasilataupun variable Hasil.

2.5. Tipe Data

Tipe data merupakanbagian program yang paling pentingkarenatipe data mempengaruhisetiapinstruksi yang akandilaksanakanoleh computer. Misalnyasaja 5 dibagi 2 bisasajamenghasilkanhasil yang berbedatergantungtipedatanya.Jika 5 dan 2

bertipe integer makaakanmenghasilkannilai 2, namunjikakeduanyabertipe float makaakanmenghasilkannilai 2.5000000. Pemilihantipe data yang tepatakanmembuat proses operasi data menjadilebihefisiendanefektif.

Dalambahasa C terdapat lima tipe data dasar, yaitu :

NO	Tipe Data	Ukuran	Range (Jangkauan)	Format	Keterangan
1	char	1 byte	- 128 s/d 127	%c	Karakter/String
2	int	2 byte	- 32768 s/d 32767	%i , %d	BilBulat
3	float	4 byte	- 3.4E-38 s/d 3.4E+38	%f	BilPecahan
4	double	8 byte	- 1.7E-308 s/d 1.7+308	%lf	BilPecahanGanda
5	void	0 byte			TidakBertipe

ContohProgram :

```
#include <stdio.h>
#include <conio.h>
void main(){
    int x;
    float y;
    char z;
    double w;
    clrscr();  // untukmembersihkanlayar
    x = 10;  // variable x diisidengan 10
    y = 9.45;  // variable y diisidengan 9.45
    z = 'C';   // variable z diisidengankarakter "C"
    w = 3.45E+20;  // variable w diisidengan 3.45E+20
    printf("Nilaidari x adalah : %i\n", x); // Menampilkanisi variable x
    printf("Nilaidari y adalah : %f\n", y); // Menampilkanisi variable y
    printf("Nilaidari z adalah : %c\n", z); // Menampilkanisi variable z
    printf("Nilaidari w adalah : %lf\n", w); // Menampilkanisi variable w
    getch();
}
```

2.6. Konstanta

Suatukonstantaadalahekspresi yang memilikinilaitetap yang dapatberupa data type Integer, Floating-Point, Character dan String.

2.6.1. Defined Constants (#define)

Andadapatmendefinisikannamatertentupadakonstanta yang seringandagunakandenganmenggunakanprepocessor directive #define preprocessor directive yang memiliki syntax berikut :

```
#define identifier value
```

Sebagaicontoh :

```
#define PI 3.14159265
#define NEWLINE '\n'
#define WIDTH 100
```

Setelahdefinisidiatas, kitadapatmemakainyasepertiberikut:

```
circle = 2 * PI * r;
printf("%d",WIDTH);
```

Sesuatu yang harisdiingatadalahbahwa directive #define bukanmerupakanbarisperintah, tetapimerupakan directive untuk preprocessor, sehinggaandatidakperlumemberikan semicolon (;) padaakhirannya.

2.6.2. Declared Constants (const)

Andadapatjugamenggunakanawalanconstuntukmendeklarasikankonstantaden gan type tertentusebagaimanaandalakukanpadavariabel:

```
constint width = 100;
const char tab = '\t';
const zip = 12440;
```

Dalam hal ini, jika type tidak ditentukan (seperti pada contoh terakhir), maka compiler akan mengasumsikan type-nya sebagai int.

2.7. Karakter Escape

Karakter escape merupakan karakter spesial yang tidak dapat diekspresikan kecuali dalam source code, seperti newline (\n) atau tab (\t). Berikut ini adalah daftar dari kode escape :

- \a : untuk bunyi bell (alert)
- \b : mundur satu spasi (backspace)
- \f : ganti halaman (form feed)
- \n : ganti baris baru (new line)
- \r : ke kolom pertama, baris yang sama (carriage return)
- \v : tabulasi vertical
- \0 : nilai kosong (null)
- \' : karakter petik tunggal
- \" : karakter petik ganda
- \\ : karakter garis miring

2.8. Deklarasi

Deklarasi diperlukan bila kita akan menggunakan pengenal (identifier) dalam program. Identifier dapat berupa variable, konstanta dan fungsi.

2.8.1. Deklarasi Variabel

Bentuk umum pendeklarasian suatu variable adalah :

```
Nama_tipe nama_variabel;
```

Contoh :

```
int x;   // Deklarasi x bertipe integer
char y, huruf, nim[10]; // Deklarasi variable bertipe char
float nilai;   // Deklarasi variable bertipe float
double beta;  // Deklarasi variable bertipe double
int array[5][4];  // Deklarasi array bertipe integer
char *p;   // Deklarasi pointer p bertipe char
```

2.8.2. Deklarasi Fungsi

Fungsi merupakan bagian yang terpisah dari program dan dapat diaktifkan atau dipanggil di manapun di dalam program. Fungsi dalam bahasa C ada yang sudah disediakan sebagai fungsi pustaka seperti printf(), scanf(), getch() dan untuk menggunakannya tidak perlu dideklarasikan. Fungsi yang perlu dideklarasikan terlebih dahulu adalah fungsi yang dibuat oleh programmer. Bentuk umum deklarasi sebuah fungsi adalah :

```
Tipe_fungsi nama_fungsi(parameter_fungsi);
```

Contohnya :

```
float luas_lingkaran(int jari);
void tampil();
int tambah(int x, int y);
```

2.9. Operator

Sebagaimana pada Pascal untuk mengoperasikan variable dan konstanta kita membutuhkan operator, adapun operator yang disediakan oleh C++ adalah sebagai berikut :

2.9.1. Operator Penugasan

Operator Penugasan (Assignment operator) dalam bahasa C berupa tanda sama dengan (“=”).

Contoh :

```
nilai = 80;
A = x * y;
```

Artinya : variable “nilai” diisi dengan 80 dan variable “A” diisi dengan hasil perkalian antara x dan y.

2.9.2. Operator Aritmatika

Bahasa C menyediakan lima operator aritmatika, yaitu :

- * : untuk perkalian
- / : untuk pembagian
- % : untuk sisa pembagian (modulus)
- + : untuk pertambahan
- - : untuk pengurangan

Catatan : operator % digunakan untuk mencari sisa pembagian antara dua bilangan.

Misalnya :

```
9 % 2 = 1
9 % 3 = 0
9 % 5 = 4
9 % 6 = 3
```

Contoh Program 1 :

```
#include<stdio.h>
#include<conio.h>
void main() {
    clrscr();  // untuk membersihkan layar
    printf("Nilai dari 9 + 4 = %i", 9 + 4);   /* mencetak hasil 9 + 4 */
    printf("Nilai dari 9 - 4 = %i", 9 - 4);  /* mencetak hasil 9 - 4 */
    printf("Nilai dari 9 * 4 = %i", 9 * 4);  /* mencetak hasil 9 * 4 */
    printf("Nilai dari 9 / 4 = %i", 9 / 4);  /* mencetak hasil 9 / 4 */
    printf("Nilai dari 9 \% 4 = %i", 9 % 4); /* mencetak hasil 9 % 4 */
    getch();
}
```

Contoh Program 2 :

```
/* Penggunaan operator % untuk mencetak deret bilangan genap antara 1 – 100 */
#include<stdio.h>
#include<conio.h>
void main() {
    int bil;
    clrscr();    // untuk membersihkan layar
```

```
    for (bil=1; bil<100; bil++) {
        if(bil % 2 == 0)   //periksa apakah 'bil' genap
        printf("%5.0i", bil);
    }
    getch();
}
```

2.9.3. Operator Hubungan (Perbandingan)

Operator Hubungan digunakan untuk membandingkan hubungan antara dua buah operand (sebuah nilai atau variable). Operator hubungan dalam bahasa C :

Operator	Arti	Contoh	
<	Kurang dari	X < Y	Apakah X kurang dari Y
<=	Kurang dari sama dengan	X <= Y	Apakah X kurang dari sama dengan Y
>	Lebih dari	X > Y	Apakah X lebih besar dari Y
>=	Lebih dari sama dengan	X >= Y	Apakah X lebih besar sama dengan Y
==	Sama dengan	X == Y	Apakah X sama dengan Y
!=	Tidak sama dengan	X != Y	Apakah X tidak sama dengan Y

2.9.4. Operator Logika

Jika operator hubungan membandingkan hubungan antara dua buah operand, maka operator logika digunakan untuk membandingkan logika hasil dari operator-operator hubungan.

Operator logika ada tiga macam, yaitu :

- && : Logika AND (DAN)
- || : Logika OR (ATAU)
- ! : Logika NOT (INGKARAN/BUKAN)

2.9.5. Operator Bitwise

Operator bitwise digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori.

Operator bitwise dalam bahasa C :

- << : Pergeseran bit ke kiri
- >> : Pergeseran bit ke kanan
- & : Bitwise AND
- ^ : Bitwise XOR (exclusive OR)
- | : Bitwise OR
- ~ : Bitwise NOT

2.9.6. Operator Unary

Operator Unary merupakan operator yang hanya membutuhkan satu operand saja. Dalam bahasa C terdapat beberapa operator unary, yaitu :

Operator	Maksud	Letak	Contoh	Equivalen
-	Unary minus	Sebelum operator	A + -B * C	A + (-B) * C
++	Peningkatan dengan penambahan nilai 1	Sebelum dan sesudah	A++	A = A + 1
--	Penurunan dengan pengurangan nilai 1	Sebelum dan sesudah	A--	A = A – 1
sizeof	Ukuran dari operand dalam byte	Sebelum	sizeof(I)	
!	Unary NOT	Sebelum	!A	
~	Bitwise NOT	Sebelum	~A	
&	Menghasilkan alamat memori operand	Sebelum	&A	
*	Menghasilkan nilai dari pointer	Sebelum	*A	

Catatan Penting:

Operator peningkatan ++ dan penurunan -- jika diletakkan sebelum atau sesudah operand terdapat perbedaan. Perhatikan contoh berikut :

Contoh Program 1 :

```
/* Perbedaan operator peningkatan ++ yang diletakkan di depan dan dibelakang
operand */
#include <stdio.h>
#include <conio.h>
void main() {
    int x, nilai;
    clrscr();
    x = 5;
    nilai = ++x;   /* berarti x = x + 1; nilai = x;  */
    printf("nilai = %d, x = %d\n", nilai, x);
    nilai = x++;   /* berarti nilai = x;  nilai = x + 1;  */
    printf("nilai = %d, x = %d\n", nilai, x);
    getch();
}
```

Contoh Program 2 :

```
#include <stdio.h>
#include <conio.h>
void main() {
    int b, nilai;
    clrscr();  // untuk membersihkan layar
    b = 15;
    nilai = --b;   /* berarti b = b - 1; nilai = b;  */
    printf("nilai = %d, b = %d\n", nilai, b);
    nilai = b--;   /* berarti nilai = b;  b = b + 1;  */
    printf("nilai = %d, b = %d\n", nilai, b);
    getch();
}
```