BAB VIII
STRUKTUR, UNION, ENUMERASI, dan MANIPULASI BIT

STRUKTUR

Struktur (*structures*) adalah sekumpulan variabel yang masing-masing dapat berbeda tipe, dan dikelompokkan ke dalam satu nama. Struktur ini sering digunakan untuk mendefinisikan suatu rekord data yang disimpan di dalam file.

Struktur termasuk ke dalam tipe data yang dibangkitkan (*derived data type*), yang disusun dengan menggunakan obyek tipe lain. Perhatikan definisi struktur berikut:

```
struct mhs {
    char *nama;
    char *nim;
    int uts, uas;
    float akhir;
    char mutu;
}
```

Kata kunci **struct** menunjukkan definisi struktur, dan identifikasi **mhs** menunjukkan *structure tag*. Dengan demikian terdapat tipe data baru bernama **struct mhs**, yang terdiri dari nama mahasiswa, nilai ujian tengah semester, akhir semester, nilai akhir, dan huruf mutu, yang masing-masing disebut dengan **field**. Oleh karena itu, jika ingin mendeklarasikan variabel dengan tipe tersebut, dapat dituliskan seperti contoh berikut:

```
struct mhs x, y[100], *z;
```

Variabel **x** adalah variabel tunggal, **y** adalah variabel array dengan 100 lokasi memori, dan **z** adalah variabel pointer, yang kesemuanya masing-masing berisi field di atas. Jadi, variabel **y** adalah daftar nama, nilai uts, uas, akhir, dan huruf mutu dari 100 mahasiswa.

Penulisan deklarasi tersebut dapat juga ditulis sekaligus seperti di bawah ini:

```
struct mhs {
    char *nama;
    char *nim;
    int uts, uas;
    float akhir;
    char mutu;
} x, y[100], *z;
```

Inisialisasi terhadap variabel struktur ini dapat dilakukan seperti contoh berikut:

```
struct mhs x = { "Asterix", 80, 60, 76.8, 'A' };
```


Untuk mengakses anggota dari struktur digunakan salah satu dari dua operator, yaitu operator titik (.), atau operator panah (->) tergantung tipe variabel yang dideklarasikan. Jika variabel tunggal (misalnya x) maka digunakan operator titik, sedangkan jika variabel pointer (misalnya z) maka digunakan operator panah, seperti yang terdapat pada dua pernyataan berikut:

```
printf("%s", x.nama);
printf("%s", z->nama);
```

TYPEDEF

Kata kunci **typedef** merupakan mekanisme untuk membuat sinonim atau alias dari tipe data yang telah didefinisikan sebelumnya. Sebagai contoh, pernyataan:

```
typedef  struct mhs MHS;
```

mendefinisikan tipe data baru bernama **MHS** sebagai sinonim untuk **struct mhs**. Dengan demikian, pernyataan **struct mhs** untuk selanjutnya dapat diganti dengan **MHS** saja.

UNION

Sama seperti **struct**, **union** juga merupakan tipe data yang dibangkitkan, dimana anggota-anggotanya menggunakan secara bersama-sama ruang penyimpanan memori yang sama, berbeda dengan struktur yang masing-masing variabel menempati lokasi memori yang berbeda. Jumlah bytes yang digunakan untuk menyimpan **union** adalah sedikitnya cukup untuk menyimpan data terbesar yang ditangani. Oleh karena itu, tipe **union** ini umumnya digunakan untuk menangani satu, dua, atau tiga variabel dengan tipe yang mirip. Sebagai contoh:

```
union nilaiUjian {
    int uts, uas;
    float akhir;
}
```

Inisialisasi, deklarasi, dan pengolahan terhadap tipe **union** ini sama dengan **struct** yang telah dijelaskan pada bagian sebelumnya.

ENUMERASI

Program C menyediakan tipe data yang dapat didefinisikan oleh pemrogram yang disebut dengan enumerasi. Enumerasi, didefinisikan dengan menggunakan kata kunci **enum**, adalah sekumpulan konstanta integer yang direpresentasikan dengan identikasi tertentu. Nilai dalam enum dimulai dari 0, dapat diubah dengan nilai lainnya, dan menaik dengan penambahan 1 untuk nilai selanjutnya. Sebagai contoh, enumerasi berikut:

```
enum bulan {JAN, PEB, MAR, APR, MEI, JUN, JUL, AGU, SEP, OKT, NOP, DES};
```

akan menciptakan tipe baru yaitu **enum bulan**, yang secara otomatis menunjukkan deret nilai 0 untuk JAN hingga 11 untuk DES. Nilai bulan ini dapat diubah menjadi 1 hingga 12 dengan cara sebagai berikut:

```
enum bulan {JAN = 1, PEB, MAR, APR, MEI, JUN, JUL, AGU, SEP, OKT, NOP, DES};
```

Program berikut menyajikan contoh penggunaan tipe enumerasi. Silakan ditelaah dan diduga keluarannya.

```
#include <stdio.h>

enum bulan {JAN = 1, PEB, MAR, APR, MEI, JUN, JUL, AGU, SEP, OKT, NOP, DES};

main() {
    enum bulan Bulan;
    char *namaBulan[] = {"", "Januari", "Pebruari", "Maret", "April", "Mei", "Juni", "Juli",
                         "Agustus", "September", "Oktober", "Nopember", "Desember" };

    for ( Bulan = JAN ; Bulan <= DEC ; Bulan++ )
        printf( "%2d%11s\n", Bulan, namaBulan[Bulan] );
    return 0;
}
```

OPERATOR BITWISE

Semua data direpresentasikan di dalam komputer sebagai deretan digit biner atau *binary digit*, yang lebih sering disingkat sebagai **bit**, yang masing-masing bernilai **0** atau **1**. Pada sebagian besar sistem komputer, 8 bit membentuk satu **byte**, yaitu penyimpanan baku bagi sebuah variabel bertipe **char**. Tipe data lainnya disimpan dalam bytes yang lebih besar, misalnya **int** membutuhkan 2 bytes, **float** membutuhkan 4 bytes, dan sebagainya.

Operator bitwise digunakan untuk memanipulasi bit-bit dari operand-operand integral, yaitu **char**, **short**, **int**, dan **long**; serta kedua **signed** dan **unsigned**. Tabel berikut menunjukkan operator bitwise yang dikenal oleh C.

Operator		Deskripsi
&	bitwise AND	bernilai **1** bila hubungan kedua operand bernilai **1**
|	bitwise inclusive OR	bernilai **1** jika sedikitnya satu operand bernilai **1**
^	Bitwise exclusive OR	bernilai **1** jika tehap hanya satu operand yang bernilai **1**
<<	left shift	geser bit pada operand pertama ke arah **kiri** sebanyak nilai yang ditentukan pada operand kedua
>>	right shift	geser bit pada operand pertama ke arah **kanan** sebanyak nilai yang ditentukan pada operand kedua
~	one's complement	Semua bit **0** diganti **1**, dan bit **1** diganti **0**

Untuk memahami manipulasi bit ini, perhatikan dan telaah keluaran yang dihasilkan oleh program berikut:

```
#include <stdio.h>

main() {
    unsigned x;
    void displayBits(unsigned);

    printf( "Ketik bilangan bulat tak bertanda : "); scanf( "%u", &x);
    displayBits(x);
    return 0;
}

void displayBits (unsigned value) {
    unsigned c, displayMask = 1 << 15;

    printf ( "%7u = ", value );

    for ( c = 1 ; c <= 16 ; c++ ) {
        putchar ( value & displayMask ? '1' : '0' ) ;

        if ( c % 8 == 0 )
            putchar ( ' ' );
    }

    putchar ( '\n' );
}
```