

BAB IV STRUKTUR

Struktur adalah sekumpulan variabel yang masing-masing dapat berbeda tipe, dan dikelompokkan ke dalam satu nama (menurut Pascal, struktur juga dikenal sebagai record). Struktur membantu mengatur data-data yang rumit, khususnya dalam program yang besar, karena struktur membiarkan sekelompok variabel diperlakukan sebagai satu unit daripada sebagai entity yang terpisah.

Salah satu contoh struktur tradisional adalah record daftar gaji karyawan, dimana karyawan digambarkan dengan susunan lambang seperti nama, alamat, nomor jaminan sosial, gaji dan sebagainya. Beberapa dari lambang tersebut biasanya berupa struktur, nama mempunyai komponen begitu juga alamat dan gaji.

Struktur ini sering digunakan untuk mendefinisikan suatu record data yang disimpan di dalam file. Struktur termasuk ke dalam tipe data yang dibangkitkan (derived data type), yang disusun dengan menggunakan obyek tipe lain.

Contoh :

```
struct mhs
{
    char *nama;
    char *nim;
    int tts, tas;
    float akhir;
    char aksara;
}
```

Kata kunci struct menunjukkan definisi struktur, dan identitas mhs menunjukkan structure tag. Dengan demikian terdapat tipe data baru bernama struct mhs, yang terdiri dari nama mahasiswa, nilai tes tengah semester, tes akhir semester, nilai akhir, dan huruf aksara, yang masing-masing disebut dengan field.

Dapat dituliskan dengan :

```
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 tts, tas, akhir, dan huruf aksara dari 100 mahasiswa.

Sehingga dapat ditulis :

```
struct mhs
{
    char *nama;
    char *nim;
    int tts, tas;
    float akhir;
    char aksara;
} x, y[100], *z;
```

Inisialisasi juga dapat dilakukan dengan :

```
struct mhs x = { "Garfield", 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 (misal x) maka digunakan operator titik, sedangkan jika variabel pointer (misal z) digunakan operator panah.

Contoh :

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

DASAR STRUKTUR

Misal ada permasalahan grafis yang melibatkan koordinat x dan y. Objek dasar yang akan dibuat struktur adalah titik koordinatnya, yang diasumsikan sebagai koordinat x dan y dan keduanya bilangan bulat.

Deklarasi dari koordinat x dan y adalah :

```
struct point
{
    int x;
    int y;
};
```

kata kunci struct mengenalkan deklarasi struktur yang mana deklarasi list terlampir di kurung kurawal { }. Nama pilihan yang disebut structure tag mengikuti kata struct.

Deklarasi struktur yang tidak diikuti oleh variabel list tidak menyediakan tempat penyimpanan; deklarasi struktur hanya menjelaskan template atau bentuk struktur. Kalau deklarasi di tag, tag dapat digunakan dalam definisi contoh struktur. Sebagai contoh, memberikan deklarasi point diatas.

```
struct point pt;
```

Variabel pt yang berupa struktur tipe struct poin. Sebuah struktur dapat diletakkan di depan dengan mengikuti definisinya dengan daftar inisialisasi, masing-masing adalah lambang konstanta.

STRUKTUR DAN FUNGSI

Operasi yang sering diterapkan pada struktur adalah proses menyalin atau menunjukkan struktur sebagai unit, menggunakan alamatnya dan mengakses anggotanya. Copy dan assignment mencakup memberi argumen ke fungsi dan menghasilkan nilai dari fungsinya juga. Struktur tidak bisa dibandingkan.

Struktur dapat diletakkan di awal oleh daftar value konstanta dan otomatis juga dapat ditempatkan di awal oleh operasi assignment. Sebuah struktur otomatis mungkin juga diletakkan di depan oleh tugas atau oleh panggilan fungsi yang menghasilkan struktur jenis yang tepat.

Untuk menghubungkan nama struktur dan nama anggota digunakan simbol "."

Contoh :

```

/*Program : struct1.cpp*/
#include <stdio.h>
struct time
{
    int jam;
    int min;
};
struct rencana {
    struct time awal;
    struct time akhir;
    int y;
    int z;
};
struct rencana kerja = { 11,22,33,44,5,6 };

funct(struct rencana oo);

main()
{
    kerja.akhir.min = 40;
    kerja.z = 66;
    printf("proses main sebelum ke fungsi\n%d %d %d %d %d %d\n",
    kerja.awal.jam, kerja.awal.min, kerja.akhir.jam, kerja.akhir.min, kerja.y,kerja.z);

    funct(kerja);    /* pengiriman struktur kerja ke fungsi */

    printf("proses main sesudah ke fungsi\n%d %d %d %d %d %d\n", kerja.awal.jam,
    kerja.awal.min, kerja.akhir.jam, kerja.akhir.min, kerja.y,kerja.z);
}

funct(struct rencana oo)
/* nilai struktur kerja disalinkan ke oo */
{
    printf("dalam fungsi (a)\n%d %d %d %d %d %d\n", oo.awal.jam, oo.awal.min,
    oo.akhir.jam,oo.akhir.min, oo.y, oo.z);
    oo.awal.jam = 111;
    oo.y = 555;    /* ubah nilai dalam fungsi */

    printf("dalam fungsi (a)\n%d %d %d %d %d %d\n", oo.awal.jam, oo.awal.min,
    oo.akhir.jam,oo.akhir.min, oo.y, oo.z);
}

```

Bila program dijalankan maka :

```

proses main sebelum ke fungsi
11 22 33 40 5 66
dalam fungsi (a)
11 22 33 40 5 66
dalam fungsi (a)
111 22 33 40 555 66
proses main sesudah ke fungsi
11 22 33 40 5 66

```

ARRAY DALAM STRUKTUR

Array disini berfungsi untuk menyimpan nama dan bilangan bulat yang akan digunakan dalam proses perhitungan.

Contoh :

// Program : struct2.cpp

```
#include <iostream.h>
#include <string.h>
#include <stdlib.h>

struct movies_t {
    char title [50];
    int year;
} mine, yours;

void printmovie (movies_t movie);

int main ()
{
    char buffer [50];

    strcpy (mine.title, "Finding Nemo");
    mine.year = 2003;

    cout << "Masukkan judul film favorit: ";
    cin.getline (yours.title,50);
    cout << "Masukkan tahun: ";
    cin.getline (buffer,50);
    yours.year = atoi (buffer);

    cout << "Judul film favorit yang ada:\n ";
    printmovie (mine);
    cout << "Judul film favorit kamu adalah:\n ";
    printmovie (yours);
    return 0;
}

void printmovie (movies_t movie)
{
    cout << movie.title;
    cout << " (" << movie.year << ")\n";
}
```

Bila program diatas dijalankan maka hasilnya adalah :

```
Masukkan judul file favorit : Avp
Masukkan tahun : 2004
Judul film favorit yang ada :
Finding Nemo (2003)
Judul film favorit kamu adalah :
Avp (2004)
```

Dapat dilihat deklarasi array dalam struktur terletak pada char title[50]; dimana dalam title dapat menyimpan judul film sebanyak 50 tempat dan title ini terletak pada :

```
struct movies_t {
    char title [50];
    int year;
} mine, yours;
```

Contoh :

// Program : struct3.cpp

// array of structures

#include <iostream.h>

#include <stdlib.h>

#define N_MOVIES 5

```
struct movies_t {
    char title [50];
    int year;
} films [N_MOVIES];
```

```
void printmovie (movies_t movie);
```

```
int main ()
```

```
{
    char buffer [50];
    int n;
    for (n=0; n<N_MOVIES; n++)
    {
        cout << "Masukkan judul film: ";
        cin.getline (films[n].title,50);
        cout << "Masukkan tahun : ";
        cin.getline (buffer,50);
        films[n].year = atoi (buffer);
    }
    cout << "\nFilm yang menjadi favorit kamu:\n";
    for (n=0; n<N_MOVIES; n++)
        printmovie (films[n]);
    return 0;
}
```

```
void printmovie (movies_t movie)
```

```
{
    cout << movie.title;
    cout << " (" << movie.year << ")\n";
}
```

Bila program diatas dijalankan maka hasilnya adalah :

Masukkan judul file favorit : Matrix

Masukkan tahun : 2002

Masukkan judul file favorit : Alien Vs Predator

Masukkan tahun : 2004

...

Film yang menjadi favorit kamu :

Matrix (2002)

Alien Vs Predator (2004)

Deklarasi array dalam struktur terletak pada char title [50]; dan #define N_Movies 5, sehingga 5 buah data film harus dimasukkan.

POINTER DALAM STRUKTUR

Misalkan sebuah pointer yaitu ptpelajar, yang menunjuk kepada sebuah data yang mempunyai struktur PELAJAR seperti berikut :

```
struct PELAJAR *ptpelajar;
```

Seperti pada pointer yang lain, deklarasi tersebut tidak menyediakan sebarang tempat untuk record PELAJAR. Perlu dibuat record baru yang fungsinya menggantikan pointer. Misalnya pelajar_baru.

```
ptpelajar = &pelajar_baru;
```

Dengan kondisi tersebut, pointer ptpelajar boleh digunakan untuk menggantikan tempat alamat pelajar_baru, dan pointer ptpelajar ini ditunjukkan dengan menggunakan simbol ->.

Contoh :

```
ptpelajar->nama = Jill Valentine;
ptpelajar->nim = 672009001;
ptpelajar->fakultas = Teknologi Informasi;
ptpelajar->tahun = 2009;
ptpelajar->alamat = Salatiga;
```

Sama dengan :

```
*ptpelajar.nama = Jill Valentine;
*ptpelajar.nim = 672009001;
*ptpelajar.fakultas = Teknologi Informasi;
*ptpelajar.tahun = 2009;
*ptpelajar.alamat = Salatiga;
```

Contoh :

// Program : struct4.cpp

```
#include <iostream.h>
struct time {
    int jam;
    int min;
};
struct rencana {
    struct time *awal; /* penunjuk bagi struktur */
    struct time *akhir;
};
struct time jk = { 1,2 };
struct time kl = { 3,4 };
struct rencana kerja = { &jk,&kl };

main()
{
    kerja.akhir->min = 37;
    cout << kerja.awal->jam << " " << kerja.awal->min << " "
    << kerja.akhir->jam << " " << kerja.akhir->min << endl;
}
```

Bila program diatas dijalankan maka hasilnya adalah :

```
1 2 3 37
```

STRUKTUR YANG MENUNJUK DIRINYA SENDIRI

Struktur yang mempunyai sifat menunjuk dirinya sendiri dapat dilihat pada contoh deklarasi berikut ini :

```
struct tnode {
    char *perkataan;
    int *perkiraan;
    struct tnode *kiri;
    struct tnode *kanan;
};
```

TYPEDEF

Kata kunci typedef merupakan mekanisme untuk membuat sinonim atau alias dari tipe data yang telah didefinisikan sebelumnya.

Contoh :

```
typedef struct mhs MHS;
```

Dari deklarasi tersebut dapat didefinisikan sebuah tipe data baru bernama MHS sebagai sinonim untuk struct mhs. Pernyataan struct mhs dapat diganti dengan MHS saja.

Contoh :

```
typedef int PANJANG;
```

Jenis PANJANG boleh digunakan untuk deklarasi variabel yang lain.

Contoh :

```
PANJANG len, maxlen;
PANJANG *lengths[];
```

Sama dengan :

```
int len, maxlen;
int *lengths[];
```

Contoh :

// Program : struct5.cpp

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

```
main()
```

```
{
```

```
    static struct s1 {
        char c[4], *s;
    } s1 = { "abcd", "fghi" };
```

```
    static struct s2 {
        char *cp;
        struct S1 ss1;
    } s2 = { "klmn", { "pqrs", "uvwxy" } };
```

```
    printf("s1.c[2] = %c, *s1.s = %c\n", s1.c[2], *s1.s);
    printf("s1.c = %s s1.s = %s\n", s1.c, s1.s);
    printf(" s2.cp = %s s2.ss1.s = %s\n", s2.cp, s2.ss1.s);
    printf("++s2.cp = %s ++s2.ss1.s = %s\n", ++s2.cp, ++s2.ss1.s);
```

```
}
```

Bila program tersebut dijalankan maka :

```
s1.c[2] = c, *s1.s = f
s1.c = abcd s1.s = fgghi
s2.cp = klmn s2.ss1.s = uvwxy
++s2.cp = lmn ++s2.ss1.s = vwxy
```

Penggunaan typedef terletak pada variabel s1 dan s2 yang mempunyai panjang karakter yang berbeda dan nantinya pada waktu dicetak akan menghasilkan output sesuai kondisi pada program.

UNION

Sama seperti struct, union juga merupakan tipe data yang dibangkitkan, dimana anggotanya menggunakan secara bersama 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. Tipe union umumnya digunakan untuk menangani satu, dua, atau tiga variabel dengan tipe yang mirip.

Contoh :

```
//Program:struct6.cpp
#include <iostream.h>
typedef union int_or_float {
    int n;
    float x;
} number;

main()
{
    number temp;
    temp.n = 4444;
    cout << "(a) temp.n = " << temp.n << " temp.x = " << temp.x << endl;
    temp.x = 4444.0;
    cout << "(b) temp.n = " << temp.n << " temp.x = " << temp.x << endl;
    temp.n = 4444;
    cout << "(c) temp.n = " << temp.n << " temp.x = " << temp.x << endl;
}
```

Bila program dijalankan, maka :

```
(a) temp.n = 4444 temp.x = 0.574484
(b) temp.n = -8192 temp.x = 4444
(c) temp.n = 4444 temp.x = 4418.169922
```

Output pada baris (b), nilai temp.n berubah karena tempatnya telah digunakan untuk menempatkan temp.x. Hal sebaliknya terjadi kepada nilai temp.x pada baris (c).

Output program akan berbeda jika deklarasi union tadi diganti dengan deklarasi struktur berikut :

```
typedef struct int_or_float {
    int n;
    float x;
} number;
```

Hasilnya :

```
(a) temp.n = 4444  temp.x = -0.00000
(b) temp.n = 4444  temp.x = 4444.00000
(c) temp.n = 4444  temp.x = 4444.00000
```

ENUMERASI

C++ menyediakan tipe data yang dapat didefinisikan oleh pemrogram disebut dengan enumerasi. Enumerasi, didefinisikan dengan menggunakan kata kunci enum, adalah sekumpulan konstanta integer yang direpresentasikan dengan identifikasi tertentu.

Nilai dalam enum dimulai dari 0, dapat diubah dengan nilai lainnya, dan menaik dengan penambahan 1 untuk nilai selanjutnya.

Contoh :

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

Deklarasi tersebut 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 :

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

Contoh :

```
/* Program : struct7.cpp*/
```

```
#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 <= 12 ; Bulan++ )
        printf( "%2d%11s\n", Bulan, namaBulan[Bulan] );
    return 0;
}
```

Bila program tersebut dijalankan maka :

1	Januari
2	Februari
3	Maret
4	April
5	Mei
6	Juni
7	Juli
8	Agustus
9	September
10	Oktober
11	November
12	Desember

CONTOH SOAL :

Soal 1

Buatlah program dengan struktur untuk menampilkan data karyawan yang berjumlah 5 orang lengkap dengan nama dan no Id-nya.

Contoh ada di file : dataid.exe

Save dengan nama file : st1_nim (4 digit nim terakhir)