BAB IX
STRUKTUR

Tujuan :

1. Menjelaskan cara mendeklarasikan struktur
2. Menjelaskan cara menginisialisasi struktur
3. Menjelaskan cara mengakses elemen struktur
4. Menjelaskan pembentukan array dari struktur *(array of struct)*
5. Menjelaskan tentang hubungan antara struktur dengan fungsi
6. Menjelaskan tentang hubungan antara struktur dengan pointer

Struktur adalah pengelompokan variabel-variabel yang bernaung dalam satu nama yang sama. Berbeda dengan array yang berisi kumpulan variabel-variabel yang bertipe sama dalam satu nama, maka suatu struktur dapat terdiri atas variabel-variabel yang berbeda tipenya dalam satu nama struktur. Struktur biasa dipakai untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah kesatuan (dalam bahasa PASCAL, struktur disebut dengan *record*).

Variabel-variabel yang membentuk suatu struktur, selanjutnya disebut sebagai elemen dari struktur atau *field*. Dengan demikian dimungkinkan suatu struktur dapat berisi elemen-elemen data berbeda tipe seperti *char, int, float, double*, dan lain-lain. Contoh sebuah struktur adalah informasi data tanggal (**date**) yang berisi :

- **day**
- **month**, dan
- **year**

9.1 Mendefinisikan & Mendeklarasikan Struktur

Suatu struktur didefinisikan dengan menggunakan kata kunci *struct*. Contoh pendefinisian sebuah tipe data struktur :

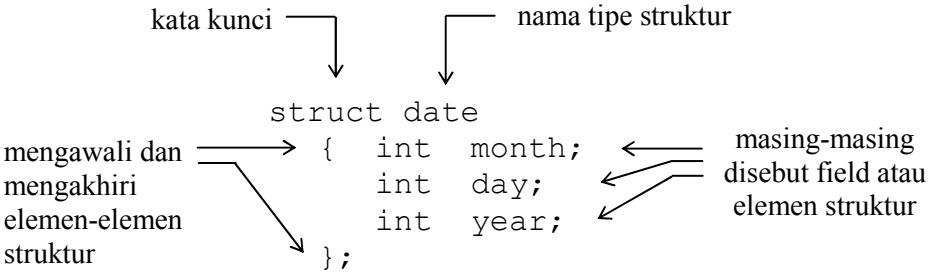
```
struct date {
       int  month;
       int  day;
       int  year;
};

struct date {
       int  month, day, year;
};
```

yang mendefinisikan sebuah tipe data struktur bernama **date** yang memiliki tiga buah elemen *(field)* berupa : **- day**

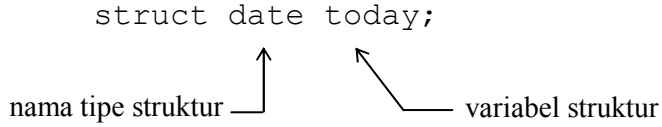
- month

- year



Gambar 9.1 Pendefinisian tipe struktur

Untuk mendeklarasikan sebuah variabel **today** yang bertipe struktur **date** pernyataan yang diperlukan adalah sebagai berikut:



Gambar 9.2 Pendeklarasian variabel bertipe struktur

Pernyataan di atas menyatakan bahwa variabel **today** bertipe struktur **date**.

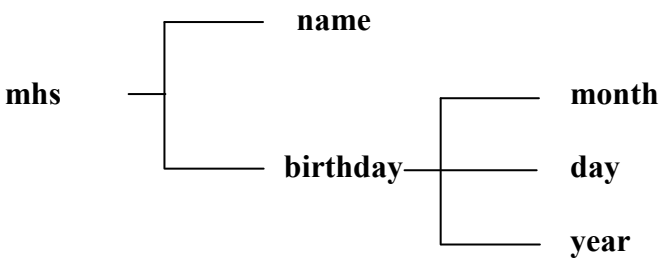
Dalam mendefinisikan sebuah struktur, elemen yang terkandung di dalamnya bisa juga berupa sebuah struktur, contoh :

```
struct date {
      int  month, day, year;
};

struct student {
      char name[30];
      struct date birthday;
};

struct student mhs;//deklarasi var mhs
```

Diagram struktur data dari variabel **mhs** dapat digambarkan sbb :



Gambar 9.3. Struktur data dari variabel **student**

9.2 Mengakses Elemen Struktur

Elemen dari suatu variabel struktur dapat diakses dengan menyebutkan nama variabel struktur diikuti dengan operator titik (‘**.**’) dan nama dari elemen strukturnya. Cara penulisannya sebagai berikut

variabel_struktur.nama_field

Untuk memberikan data nama ke *field* **name** dari variabel **student** di atas, maka pernyataan yang diperlukan misalnya adalah :

```
strcpy(mhs.name, "MUHAMMAD IHSAN");
```

Pada pernyataan di atas, `mhs.name` dapat dibaca sebagai "*field* **name** dari **mhs**". Contoh berikut merupakan instruksi untuk mengisikan data pada *field* **birthday** :

```
mhs.birthday.day = 10;
```

Sedangkan untuk mendapatkan isi suatu *field* dari variabel struktur, contohnya :

- `tgl = mhs.birthday.day;`
- `puts(mhs.name);`

Contoh pertama merupakan instruksi untuk memberikan isi dari *field* **day** ke variabel **tgl**. Sedangkan contoh kedua merupakan instruksi untuk menampilkan isi dari *field* **name**.

Program berikut merupakan contoh yang melibatkan variabel struktur. Mula-mula field dari struktur diisi dengan suatu data, kemudian isinya ditampilkan.

```
/* File program : student1.c
Mengisi field dr variabel struktur kemudian menampilkannya */

#include <stdio.h>
#include <string.h>

struct date {       /* definisi global dari tipe date */
    int month;
    int day;
    int year;
};

struct student{     /* definisi global dari tipe student */
    char name[30];
    struct date birthday;
};

/* deklarasi global dari variabel mhs*/
struct student mhs;
main()
{
    /* memberikan nilai kepada field dari struktur mhs */
    strcpy(mhs.name, "MUHAMMAD IHSAN");
    mhs.birthday.month = 8;
    mhs.birthday.day = 10;
    mhs.birthday.year = 1970;

    /* menampilkan isi semua field dari struktur mhs */
    printf("Name     : %s\n", mhs.name);
    printf("Birthday : %d-%d-%d\n",mhs.birthday.month,
        mhs.birthday.day, mhs.birthday.year);
}
```

Contoh eksekusi :

```
Name     : MUHAMMAD IHSAN
Birthday : 8-10-1970
```

9.3 Menginisialisasi Struktur

Sebuah struktur juga bisa diinisialisasi pada saat dideklarasikan. Hal ini serupa dengan inisialisasi array, yaitu elemen-elemennya dituliskan di dalam sepasang kurung kurawal (‘**{ }**‘) dengan masing-masing dipisahkan dengan koma. Deklarasi struktur didahului dengan kata kunci *static*, contoh

```
static struct zodiak bintang =
  {"Sagitarius", 22, 11, 21, 12};
```

Selengkapnya perhatikan contoh program di bawah ini.

```
/* File program : zodiak.c
Menentukan zodiak berdasarkan data tanggal lahir masukan */

#include <stdio.h>

main()
{
    struct zodiak {
        char nama[11];
        int tgl_awal;
        int bln_awal;
        int tgl_akhir;
        int bln_akhir;
    };

    static struct zodiak bintang =
    {"Sagitarius", 22, 11, 21, 12};
    int tgl_lhr, bln_lhr, thn_lhr;

    printf("Masukkan tgl lahir Anda (XX-XX-XXXX): ");
    scanf("%d-%d-%d",&tgl_lhr, &bln_lhr, &thn_lhr);

    if((tgl_lhr >= bintang.tgl_awal && bln_lhr ==
        bintang.bln_awal) || (tgl_lhr <= bintang.tgl_akhir &&
        bln_lhr == bintang.bln_akhir))
        printf("Bintang Anda adalah %s\n", bintang.nama);
    else
        printf("Bintang Anda bukan %s\n", bintang.nama);
}
```

Contoh eksekusi :

```
Masukkan tgl lahir Anda (XX-XX-XXXX): 23-11-1972
Bintang Anda adalah Sagitarius
```

9.4 Array dan Struktur

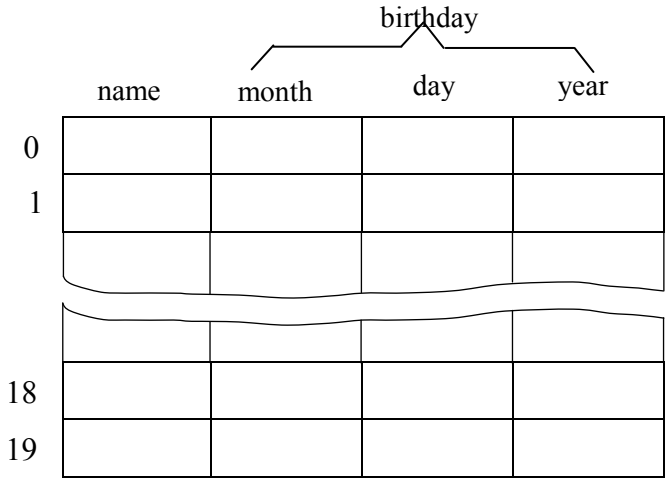
Elemen-elemen dari suatu array juga dapat berbentuk sebuah struktur. Misalnya array yang dipakai untuk menyimpan sejumlah data siswa *(struct student).* Array struktur berdimensi satu ini membentuk suatu tabel, dengan barisnya menunjukkan elemen dari array-nya dan kolomnya menunjukkan elemen dari struktur. Dalam hal ini maka deklarasi yang dibutuhkan adalah sebagai berikut :

```
#define MAKS 20
     .
     .
     .
struct date {           /* definisi dari tipe date */
     int month;
     int day;
     int year;
};

struct student {  /* definisi dari tipe student */
     char name[30];
     struct date birthday;
};

/* deklarasi dari variabel array mhs */
struct student data_mhs[MAKS];
```

yang artinya, mendeklarasikan array **data_mhs** yang memiliki elemen yang bertipe *struct* **student** sebanyak **MAKS** Setelah array **data_mhs** dideklarasikan, maka ruang yang disediakan ditunjukkan dalam gambar 9.4 di bawah ini.



Gambar 9.4 Array dari struktur

Elemen-elemen dari array stuktur tersebut bisa diakses dengan cara sebagai berikut :

```
  for (i=0; i<MAKS; i++)
  {
    printf("Name          : ");
    fgets(data_mhs[i].name,sizeof data_mhs[i].name,stdin);
    printf("Birthday (mm-dd-yyyy): ");
    scanf("%d-%d-%d", &data_mhs[i].birthday.month,
        &data_mhs[i].birthday.day,
        &data_mhs[i].birthday.year);
    printf("\n");

    /* hapus sisa data dalam penampung keyboard */
    fflush(stdin);
  };
```

Selengkapnya perhatikan contoh program di bawah ini.

```
/* File program : student2.c
Array struktur untuk menyimpan data-data student */

#include <stdio.h>

#define MAKS 20

struct date {       //definisi global dr tipe date
  int month;
  int day;
  int year;
};

struct student {            //definisi global dr tipe student
  char name[30];
  struct date birthday;
};

//deklarasi global dari variabel student
struct student data_mhs[MAKS];

main()
{
  int i=0, sudah_benar, jml;
  char lagi;

  //memasukkan data
  do
  {
    printf("Name             : ");
```

```
    fgets(data_mhs[i].name,sizeof data_mhs[i].name,stdin);

    printf("Birthday (mm-dd-yyyy): ");
    scanf("%d-%d-%d",
      &data_mhs[i].birthday.month,
      &data_mhs[i].birthday.day,
      &data_mhs[i].birthday.year);
    printf("\n");

    i++;

    printf("Mau memasukkan data lagi [Y/T] ? ");
    do
    {
      lagi = getchar( );        //baca tombol
      sudah_benar = (lagi == 'Y') || (lagi== 'y')||
        (lagi == 'T') || (lagi == 't');
    } while(! sudah_benar);

    //hapus sisa data dalam penampung keyboard
    fflush(stdin);
    printf("\n");
  } while(lagi == 'Y' || lagi == 'y');

  jml = i;

  //menampilkan data
  printf("DATA SISWA\n");
  for (i=0; i<jml; i++)
  {
    printf("%d. Name        : %s", i+1, data_mhs[i].name);
    printf("   Birthday : %d-%d-%d\n\n",
      data_mhs[i].birthday.month,
      data_mhs[i].birthday.day,
      data_mhs[i].birthday.year );
  };
}
```

Contoh eksekusi :

```
Name                   : Salsabila
Birthday (mm-dd-yyyy) : 10-25-1979

Mau memasukkan data lagi [Y/T] ? y

Name                   : Wildan
Birthday (mm-dd-yyyy) : 4-16-1974

Mau memasukkan data lagi [Y/T] ? t
```

```
DATA SISWA
1. Name      : Salsabila
   Birthday  : 10-25-1979

2. Name      : Wildan
   Birthday  : 4-16-1974
```

Di samping cara pendeklarasian di atas, struktur juga dapat dideklarasikan dalam berbagai bentuk yang lain, di antaranya sbb :

```
struct date {
    int month, day, year;
} today, tomorrow;

struct student {
    char name[30];
    struct date birthday;
} data_mhs[MAKS];
```

yaitu mendefinisikan struktur **date**, sekaligus mendeklarasikan variabel **today** dan **tomorrow** dengan tipe struktur **date**. Demikian juga mendefinisikan struktur **student**, sekaligus mendeklarasikan variabel array **data_mhs** sebanyak **MAKS** elemen dengan tipe struktur **student**. Atau cara lainnya mendefinisikan, mendeklarasikan sekaligus menginisialisasi struktur, sebagai berikut :

```
struct date {
    int month, day, year;
} today = {5,14,2001};
```

9.5 Struktur dan Fungsi

Melewatkan sebuah struktur untuk menjadi parameter sebuah fungsi dapat dilakukan sama dengan pengiriman parameter berupa variabel biasa. Fungsi yang mendapat kiriman parameter tersebut juga bisa mengirimkan hasil baliknya yang juga berupa sebuah struktur *(pass by reference)*.

9.5.1 Melewatkan Elemen Struktur ke dalam Fungsi

Melewatkan parameter berupa elemen struktur dapat dilakukan sebagaimana pengiriman parameter berupa variabel biasa, dapat dilakukan baik secara nilai *(pass by value)* maupun secara acuan *(pass by reference)*.

```
/* File program : cetak1.c
Melewatkan elemen struktur sbg parameter fungsi scr nilai */

#include <stdio.h>

void cetak_tanggal(int, int, int);

main()
{
    struct date {       /* definisi lokal dari tipe date */
        int month;
        int day;
        int year;
    } today;

    printf("Enter the current date (mm-dd-yyyy): ");
    scanf("%d-%d-%d", &today.month, &today.day, &today.year);

    cetak_tanggal(today.month, today.day, today.year);
}

void cetak_tanggal(int mm, int dd, int yy)
{
    static char *nama_bulan[] = {
        "Wrong month", "January", "February", "March",
        "April", "May", "June", "July", "August",
        "September", "October", "November", "December"
    };

    printf("Todays date is %s %d, %d\n\n",
            nama_bulan[mm],dd,yy);
}
```

Contoh eksekusi :

```
Enter the current date (mm-dd-yyyy): 5-29-2001
Todays date is May 29, 2001
```

Tampak bahwa elemen dari struktur dilewatkan ke fungsi memakai bentuk pengaksesan elemen struktur, berupa :

```
cetak_tanggal(today.month, today.day, today.year);
```

Apabila nilai suatu elemen struktur diharapkan akan diubah oleh fungsi, maka yang dilewatkan haruslah berupa alamat dari elemen struktur *(pass by reference)*. Untuk keperluan ini, operator alamat ditempatkan di depan nama variabel struktur (bukan di depan nama elemen struktur).

```
/* File program : posisi1.c
Melewatkan elemen struktur sbg parameter fungsi scr acuan */

#include <stdio.h>

void tukar_xy(int *, int *);

main()
{
    struct koordinat {
        int x;
        int y;
    } posisi;

    printf("Masukkan koordinat posisi (x, y) : ");
    scanf("%d, %d", &posisi.x, &posisi.y);

    printf("x, y semula   = %d, %d\n", posisi.x, posisi.y);
    tukar_xy(&posisi.x, &posisi.y);
    printf("x, y sekarang = %d, %d\n", posisi.x, posisi.y);
}

void tukar_xy(int *a, int *b)
{
    int z;

    z = *a;
    *a = *b;
    *b = z;
}
```

Contoh eksekusi :

```
Masukkan koordinat posisi (x, y) : 34, 21
x, y semula   = 34, 21
x, y sekarang = 21, 34
```

9.5.2 Melewatkan Struktur ke dalam Fungsi

Pada program **cetak1.c** di atas misalnya, semua elemen dari struktur dikirimkan ke fungsi **cetak_tanggal()**, dengan maksud nilai elemen dari struktur akan ditampilkan di layar. Untuk keadaan seperti ini, lebih baik kalau parameter fungsi diubah menjadi bentuk struktur, sehingga parameter fungsi tidak lagi sebanyak tiga buah, melainkan hanya satu. Selengkapnya, perhatikan program di bawah ini.

```
/* File program : cetak2.c
Melewatkan struktur sebagai parameter fungsi */
#include <stdio.h>

struct date {           /* definisi global dari tipe date */
    int month;
    int day;
    int year;
};

void cetak_tanggal(struct date);

main()
{
    struct date today;

    printf("Enter the current date (mm-dd-yyyy): ");
    scanf("%d-%d-%d", &today.month, &today.day, &today.year);

    cetak_tanggal(today);
}

void cetak_tanggal(struct date now)
{
    static char *nama_bulan[] = {
        "Wrong month", "January", "February", "March",
        "April", "May", "June", "July", "August",
        "September", "October", "November", "December"
    };
    printf("Todays date is %s %d, %d\n\n",
        nama_bulan[now.month], now.day, now.year);
}
```

Contoh eksekusi :

```
Enter the current date (mm-dd-yyyy): 5-29-2001
Todays date is May 29, 2001
```

9.6 Struktur dan Pointer (Pointer ke Struktur)

Jika sebuah struktur mengandung banyak *field* dan diputuskan bahwa keseluruhan *field*-nya akan diubah oleh fungsi, maka cara yang efisien adalah dengan melewatkan *(passing)* alamat dari struktur. Dengan demikian pada pendefinisian fungsi, parameter formalnya berupa pointer yang menunjuk ke struktur.

Masalah pointer ke struktur dapat diterapkan dalam program **posisi1.c**. Argumen dari fungsi **`tukar_xy()`** dapat disederhanakan menjadi satu argumen saja, yakni sebagai berikut :

```
void tukar_xy(struct koordinat *pos_xy)
{
    int z;

    z = (*pos_xy).x;
    (*pos_xy).x = (*pos_xy).y;
    (*pos_xy).y = z;
}
```

Pada definisi fungsi di atas,

```
struct koordinat *pos_xy
```

menyatakan bahwa **pos_xy** adalah pointer yang menunjuk ke obyek bertipe struktur **koordinat**. Adapun penulisan :

```
(*pos_xy).x
```

menyatakan : elemen bernama **x** yang ditunjuk oleh pointer **pos_xy**

Perlu diperhatikan bahwa penulisan tanda kurung seperti pada contoh **(*pos_xy).x** merupakan suatu keharusan. Sebab

```
*pos_xy.x
```

mempunyai makna yang berbeda dengan

```
(*pos_xy).x
```

Ungkapan ***pos_xy.x** mempunyai makna yaitu : "yang ditunjuk oleh **pos_xy.x** " (sebab operator titik mempunyai prioritas yang lebih tinggi daripada operator *).

```
/* File program : posisi2.c
Fungsi parameternya berupa pointer yg menunjuk ke struktur */

#include <stdio.h>

struct koordinat
      { int x;
        int y;
      };

void tukar_xy(struct koordinat *);

main()
{
      struct koordinat posisi;

      printf("Masukkan koordinat posisi (x, y) : ");
      scanf("%d, %d", &posisi.x, &posisi.y);
      printf("x, y semula   = %d, %d\n", posisi.x, posisi.y);

      tukar_xy(&posisi);

      printf("x, y sekarang = %d, %d\n", posisi.x, posisi.y);
}

void tukar_xy(struct koordinat *pos_xy)
{
      int z;

      z = (*pos_xy).x;
      (*pos_xy).x = (*pos_xy).y;
      (*pos_xy).y = z;
}
```

Contoh eksekusi :

```
Masukkan koordinat posisi (x, y) : 34, 21
x, y semula   = 34, 21
x, y sekarang = 21, 34
```

Bentuk semacam :

```
(*pos_xy).x
```

dapat ditulis dengan bentuk lain menjadi

```
pos_xy->x
```

Dalam C operator -> (berupa tanda minus - diikuti dengan tanda lebih dari >) disebut sebagai **operator panah.** Dengan menggunakan operator panah, maka fungsi **tukar_xy()** dalam program **posisi2.c** dapat ditulis menjadi

```
void tukar_xy(struct koordinat *pos_xy)
{
    int z;

    z = pos_xy->x;
    pos_xy->x = pos_xy->y;
    pos_xy->y = z;
}
```

Kesimpulan :

- Struktur adalah pengelompokan variabel-variabel yang bernaung dalam satu nama yang sama, namun tipe datanya tidak harus sama.
- Variabel-variabel yang membentuk suatu struktur, selanjutnya disebut sebagai elemen dari struktur atau *field.*
- Suatu struktur didefinisikan dengan menggunakan kata kunci *struct*.
- Elemen dari suatu variabel struktur dapat diakses dengan menyebutkan nama variabel struktur diikuti dengan operator titik (‘**.**’) dan nama dari elemen strukturnya.
- Sebuah struktur juga bisa diinisialisasi pada saat dideklarasikan. Hal ini serupa dengan inisialisasi array, yaitu elemen-elemennya dituliskan di dalam sepasang kurung kurawal (‘**{ }**‘) dengan masing-masing dipisahkan dengan koma.
- Elemen-elemen dari suatu array juga dapat berbentuk sebuah struktur *(array of struct)*.
- Melewatkan sebuah struktur untuk menjadi parameter sebuah fungsi dapat dilakukan sama dengan pengiriman parameter berupa variabel biasa. Fungsi yang mendapat kiriman parameter tersebut juga bisa mengirimkan hasil baliknya yang juga berupa sebuah struktur *(pass by reference)*.
- Jika sebuah struktur mengandung banyak *field* dan diputuskan bahwa keseluruhan *field*-nya akan diubah oleh fungsi, maka cara yang efisien adalah dengan melewatkan *(passing)* alamat dari struktur. Dengan demikian pada pendefinisian fungsi, parameter formalnya berupa pointer yang menunjuk ke struktur *(pointer to struct)*.

Latihan :

Buatlah potongan program untuk soal-soal di bawah ini

1. Definisikan sebuah struktur (misalkan namanya = **record**) yang memiliki 3 buah *field* berupa sebuah integer (misalkan namanya = **loop**), sebuah array karakter dengan 5 elemen (misalkan namanya = **word**) dan sebuah *float* (misalkan namanya = **sum**).

2. Deklarasikan sebuah variabel struktur (misalkan namanya = **sample**) yang didefinisikan memiliki tipe struktur **record.**

3. Masukkan nilai 10 kepada field **loop** dari struktur **sample** yang bertipe struktur **record** tsb.

4. Tampilkan ke layar (menggunakan fungsi *printf()*) string yang tersimpan dalam array **word** dari struktur **sample.**

5. Definisikan sebuah struktur (misalkan namanya = **date**) yang memiliki 3 *field* bertipe *int* (misalkan namanya = **day, month** dan **year**). Kemudian tuliskan potongan program untuk memasukkan 5 buah tanggal yang disimpan dalam sebuah array struktur yang bertipe **date**.