BAB 1
Struct, Array, dan Pointer

Tujuan :

1. Mahasiswa memahami apakah yang dimaksud dengan struktur data.
2. Mahasiswa memahami apakah yang dimaksud dengan algoritma.
3. Mengingat kembali array, struktur, pointer dalam bahasa C.

1.1 Pengenalan Struktur Data

Struktur data adalah sebuah skema organisasi, seperti struktur dan array, yang diterapkan pada data sehingga data dapat diinterprestasikan dan sehingga operasioperasi spesifik dapat dilaksanakan pada data tersebut

1.2 Pengenalan Algoritma

Algoritma adalah barisan langkah-langkah perhitungan dasar yang mengubah masukan (dari beberapa fungsi matematika) menjadi keluaran. Contoh :

Perkalian

Input : integer positif *a* , *b*

Output : *a* X *b*

Algoritma perkalian :

Contoh kasus : *a* = 365, *b* = 24

Metode 1 : 365 * 24 = 365 + (365 * 23)
= 730 + (365 * 22)
…..
= 8760 + (365 * 0)
= 8760

Metode 2 :

```
  3 6 5
    2 4
-------
  1 4 6 0
  7 3 0
-------
  8 7 6 0
```

Manakah algoritma yang lebih baik ?

1.3 Array

Array adalah organisasi kumpulan data homogen yang ukuran atau jumlah elemen maksimumnya telah diketahui dari awal. Array umumnya disimpan di memori komputer secara kontigu (berurutan). Deklarasi dari array adalah sebagai berikut:

int A[5]; artinya variabel A adalah kumpulan data sebanyak 5 bilangan bertipe integer.

Operasi terhadap elemen di array dilakukan dengan pengaksesan langsung. Nilai di masing-masing posisi elemen dapat diambil dan nilai dapat disimpan tanpa melewati posisi-posisi lain.

Terdapat dua tipe operasi, yaitu:

1. Operasi terhadap satu elemen/posisi dari array
2. Operasi terhadap array sebagai keseluruhan

Dua operasi paling dasar terhadap satu elemen/posisi adalah

1. Penyimpanan nilai elemen ke posisi tertentu di array
2. Pengambilan nilai elemen dari posisi tertentu di array

1.3.1 Penyimpanan dan Pengambilan Nilai

Biasanya bahasa pemrograman menyediakan sintaks tertentu untuk penyimpanan dan pengambilan nilai elemen pada posisi tertentu di array.

Contoh:

A[10] = 78, berarti penyimpanan nilai 78 ke posisi ke-10 dari array A

C = A[10], berarti pengambilan nilai elemen posisi ke-10 dari array A

1.3.2 Keunggulan dan Kelemahan Array

Keunggulan array adalah sebagai berikut:

1. Array sangat cocok untuk pengaksesan acak. Sembarang elemen di array dapat diacu secara langsung tanpa melalui elemen-elemen lain.
2. Jika berada di suatu lokasi elemen, maka sangat mudah menelusuri ke elemen-elemen tetangga, baik elemen pendahulu atau elemen penerus
3. Jika elemen-elemen array adalah nilai-nilai independen dan seluruhnya harus terjaga, maka penggunaan penyimpanannya sangat efisien

Kelemahan array. Array mempunyai fleksibilitas rendah, karena array mempunyai batasan sebagai berikut:

1. Array harus bertipe homogen. Kita tidak dapat mempunyai array dimana satu elemen adalah karakter, elemen lain bilangan, dan elemen lain adalah tipe-tipe lain
2. Kebanyakan bahasa pemrograman mengimplementasikan array statik yang sulit diubah ukurannya di waktu eksekusi. Bila penambahan dan pengurangan terjadi terus-menerus, maka representasi statis
   - Tidak efisien dalam penggunaan memori
   - Menyiakan banyak waktu komputasi
   - Pada suatu aplikasi, representasi statis tidak dimungkinkan

1.4 Pointer

Misalnya kita ingin membuat beberapa penunjuk ke blok penyimpan yang berisi integer. Deklarasi pada C adalah:

```
int *IntegerPointer;
```

Tanda asterik (*) yang berada sebelum nama variable IntegerPointer menandakan ‘pointer pada suatu int’. Jadi deklarasi diatas berarti ‘definisikan sebuah tipe yang terdiri dari pointer bertipe integer yang bernama IntegerPointer’. Apabila didepannya ditambahkan typedef sebagai berikut

```
Typedef int *IntegerPointer;
```

Berarti IntegerPointer merupakan suatu tipe pointer berbentuk integer.

Apabila akan mendeklarasikan dua variable A dan B sebagai penunjuk ke bilangan integer :

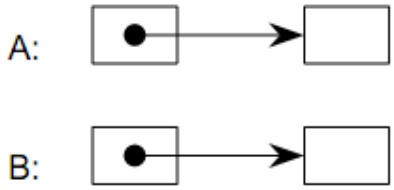
```
IntegerPointer A, B;
```

Berarti kompiler C akan berisi nilai dari variable A dan B yang 'menunjuk ke integer'.

Untuk membuat beberapa penunjuk ke beberapa penyimpan integer yang kosong dan untuk membuat A dan B menunjuk tempat tersebut, digunakan prosedur dinamis untuk alokasi penyimpan yang disebut malloc

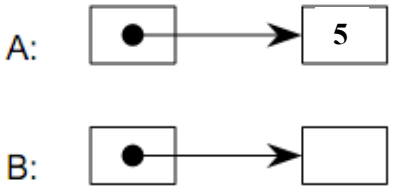
```
A = (IntegerPointer *) malloc (sizeof(int));
```


```
B = (int *) malloc (sizeof(int));
```



Misalnya kita akan menyimpan integer 5 pada blok penyimpan yang ditunjuk pointer pada variable A. Untuk menuimpan angka 5 pada blok penyimpan integer itu melalui pointer A, digunakan pernyataan :

*A = 5;



Linked list adalah salah satu struktur data yang paling fundamental. Linked list terdiri dari sejumlah kelompok elemen (*linked*) dengan urutan tertentu. Linked list sangat berguna untuk memelihara sekelompok data, semacam array, tetapi linked list lebih menguntungkan dalam beberapa kasus. Linked list lebih efisien dalam proses penyisipan (*insertion*) dan penghapusan (*deletion*). Linked list juga menggunakan pengalokasian penyimpan secara dinamis, dimana penyimpan dialokasikan pada saat waktu berjalan (*runtime*).

1.5 Struktur

Struktur adalah koleksi dari variabel yang dinyatakan dengan sebuah nama, dengan sifat setiap variabel dapat memiliki tipe yang berlainan. Struktur biasa dipakai untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah satu kesatuan.

Contoh sebuah struktur adalah informasi data tanggal, yang berisi: tanggal, bulan dan tahun.

1.5.1 Mendeklarasikan Struktur

Contoh pendefinisian tipe struktur adalah sebagai berikut:

```
struct data_tanggal {
  int tanggal;
  int bulan;
  int tahun;
};
```

yang mendefinisikan tipe struktur bernama data_tanggal, yang terdiri dari tiga buah elemen (field) berupa : tanggal, bulan dan tahun. Pendefnisian dan pendeklarasian struktur dapat juga ditulis sebagai berikut:

```
struct data_tanggal {
  int tanggal;
  int bulan;
  int tahun;
} tgl_lahir;
```

Bentuk umum dalam mendefinisikan dan mendeklarasikan struktur adalah sebagai berikut

```
struct nama_tipe_struktur {
  tipe field1;
  tipe field2;
  ...
  ...
  tipe fieldn;
}variabel_struktur1, ... , variabel_strukturM;
```

Masing-masing tipe dari elemen struktur dapat berlainan. Adapun variabel_struktur1 sampai dengan variabel_strukturM menyatakan bahwa variabel struktur yang dideklarasikan bisa lebih dari satu. Jika ada lebih dari satu variabel, antara variabel struktur dipisahkan dengan tanda koma.

1.5.2 Mengakses Elemen Struktur

Elemen dari struktur dapat diakses dengan menggunakan bentuk

```
variabel_struktur.nama_field
```

Antara variabel_struktur dan nama_field dipisahkan dengan operator titik (disebut operator anggota struktur). Contoh berikut merupakan instruksi untuk mengisikan data pada field tanggal

```
tgl_lahir.tanggal = 30;
```

1.6 Kesimpulan

1. Struktur data adalah sebuah skema organisasi yang diterapkan pada data sehingga data dapat diinterprestasikan dan sehingga operasi-operasi spesifik dapat dilaksanakan pada data tersebut
2. Apabila kita membuat program dengan data yang sudah kita ketahui batasnya, maka kita bisa menggunakan array (tipe data statis), namun apabila data kita belum kita ketahui batasnya, kita bisa menggunakan pointer (tipe data dinamis)
3. Untuk sekumpulan data dengan tipe data yang berlainan, namun merupakan satu-kesatuan, kita dapat menggunakan struktur untuk merepresentasikannya

2. LATIHAN PRAKTIKUM

Percobaan 1: Penggunaan array pada bilangan fibonacci

```
#include <stdio.h>
#define MAX 10

int fibo[MAX];

void main()
{
    int i;

    fibo[1] = 1;
    fibo[2] = 1;

    for (i=3;i<=MAX;i++)
        fibo[i]=fibo[i-2]+fibo[i-1];

    printf("%d Bilangan Fibonaci Pertama adalah : \n",MAX);
    for (i=1;i<MAX;i++)
        printf("%d-",fibo[i]);
}
```

Percobaan 2: Program mengubah isi variabel melalui pointer

```
#include <stdio.h>

main()
{
    int y, x = 87;
    int *px;

    px = &x;
    y = *px;

    printf("Alamat x    = %p\n", &x);
    printf("Isi px      = %p\n", px);
    printf("Isi x       = %d\n", x);
    printf("Nilai yang ditunjuk oleh px = %d\n", *px);
    printf("Nilai y     = %d\n", y);
}
```

Percobaan 3: Program mengakses dan mengubah isi suatu variabel pointer

```
#include <stdio.h>

main()
{
    float d = 54.5f, *pd;

    printf("Isi d mula-mula = %g\n", d);

    pd = &d;
    *pd += 10;

    printf("Isi d sekarang  = %g\n", d);
}
```

Percobaan 4: Penggunaan pointer untuk bilangan fibonacci

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

int *fibo;

void main()
{
    int i;

    fibo = (int *) malloc(MAX * sizeof(int));

    *(fibo + 1) = 1;
    *(fibo + 2) = 1;

    for (i=3;i<=MAX;i++)
        *(fibo + i)= (*(fibo + i - 2) + *(fibo + i - 1));

    printf("%d Bilangan Fibonaci Pertama adalah : \n",MAX);
    for (i=1;i<MAX;i++)
        printf("%d-",*(fibo+i));
}
```

Percobaan 5: Penggunaan struktur pada konversi koordinat polar ke koordinat catersian

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

struct polar {
    double r;
    double alpha;
};

struct kartesian {
    double x;
    double y;
};
```

```
void main()
{
    struct polar p1;
    struct kartesian k1;

    printf("Masukkan nilai r untuk koordinat polar : ");
    scanf("%lf",&p1.r);

    printf("Masukkan nilai alpha untuk koordinat polar : ");
    scanf("%lf",&p1.alpha);

    k1.x = p1.r * cos(p1.alpha);
    k1.y = p1.r * sin(p1.alpha);

    printf(
    "Nilai koordinat kartesian untuk koordinat polar r= %2.2lf alpha=
    %2.2lf adalah:\n",p1.r,p1.alpha);
    printf("x = %2.2lf  y = %2.2lf",k1.x,k1.y);
}
```

Percobaan 6: Program struktur dalam array

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

struct dtnilai
{
    char nrp[10];
    char nama[20];
    double nilai;
};

struct dtnilai data[10];
int j=0;

void tambah_data()
{
    char jawab[2];
    while(1)
    {
        fflush(stdin);
        printf("NRP :");scanf("%s",&data[j].nrp);
        printf("Nama    :");scanf("%s",&data[j].nama);
        printf("Nilai Test :");scanf("%lf",&data[j].nilai);

        printf("Ada data lagi(y/t):"); scanf("%s",&jawab);
```

```
        if((strcmp(jawab,"Y")==0)||(strcmp(jawab,"y")==0))
        {
            j++;continue;
        }
        else if ((strcmp(jawab,"T")==0)||(strcmp(jawab,"t")==0))
            break;
    }
}

void tampil()
{
    int i;
    printf("Data Mahasiswa yang telah diinputkan :\n");
    printf("NRP\tNama\tNilai\n");

    for (i=0;i<=j;i++)
    {
        printf("%s\t%s\t%6.2f\n",data[i].nrp,data[i].nama, data[i].nilai);
    }
}

void main()
{
    tambah_data();
    tampil();
}
```

3. TUGAS RUMAH

Tugas Rumah 1:
Aritmatika polinom

```
Masalah aritmatika polinom adalah membuat sekumpulan subrutin manipulasi
terhadap polinom simbolis (symbolic Polynomial).

Misalnya:

P1 = 6x8 + 8x7 + 5x5 + x3 + 15
P2 = 3x9 + 4x7 + 3x4 + 2x3 + 2x2 + 10
P3 = x2 + 5

Terdapat empat operasi aritmatika polinom dasar antara lain:

Penambahan:
P1 + P2 = 3x9 + 6x8 + 12x7 + 5x5 + 3x4 + 3x3 + 2x2 + 25

Pengurangan:
P1 - P2 = - 3x9 + 6x8 +  4x7 + 5x5 - 3x4 - x3 - 2x2 + 5

Perkalian:
P1 * P3 = 6x10 + 8x9 + 5x7 + x5 + 15x2 + 30x8 + 40x7 + 25x5 + 5x3 + 75 =
6x10 + 8x9 + 30x8 + 45x7 + 26x5 + 5x3 + 15x2 + 75

Turunan:
P2' = 27x8 + 28x6 + 12x3 + 6x2 + 4x


Representasikan bilangan polinom dengan array dan buatlah prosedur-prosedur
yang melakukan kelima operasi aritmatika di atas.
```

Tugas Rumah 2:
Bilangan kompleks

Bilangan kompleks berbentuk a + bi, dimana a dan b adalah bilangan nyata dan i2 = -1. Terdapat empat operasi aritmatika dasar untuk bilangan kompleks, yaitu:

Penambahan:
(a+bi) + (c+di) = (a+c) + (b+d)i

Pengurangan:
(a+bi) - (c+di) = (a-c) + (b-d)i

Perkalian:
(a+bi) * (c+di) = (ac-bd) + (ad+bc)i

Pembagian:
(a+bi) / (c+di) = [(ac+bd) / (a2+b2)] + [(bc-ad)/(c2+d2)]i

Tulis program yang membaca dua bilangan kompleks dan simbol operasi yang perlu dilakukan, kemudian lakukan operasi yang diminta. Gunakan struktur untuk merepresentasikan bilangan kompleks dan gunakan prosedur untuk implementasi tiap operasi.