Pengenalan Struktur Data dan Algoritma

BAB 1

Materi

1.1 Pengenalan Struktur Data

1.2 Pengenalan Algoritma

1.3 Array
- 1.3.1 Tipe Operasi Array
- 1.3.2 Penyimpanan dan Pengambilan nilai Array
- 1.3.3 Keunggulan Array
- 1.3.4 Kelemahan Array

1.4 Pointer

1.5 Struktur (Struct)
- 1.5.1 Mendeklarasikan Struktur
- 1.5.2 Mengakses Elemen Struktur

1.6 Kesimpulan

1.1 Pengenalan Struktur Data

Skema organisasi, misal struktur dan array yang diterapkan pada data

Sehingga data dapat diinterpretasikan sedemikian rupa

Operasi spesifik dapat dilakukan pada data tersebut

1.2 Pengenalan Algoritma

Barisan langkah perhitungan dasar yang mengubah masukan menjadi keluaran,

Masukan tersebut dapat dari beberapa fungsi matematis

Contoh algoritma Perkalian

Contoh kasus : a = 365, b = 24

Metode ke 1

365 * 24 = 365 + (365 * 23)
= 730 + (365 * 22)
.....
= 8760 + (365 * 0)
= 8760

Metode ke 2

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

Manakah 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 computer secara kontinyu (berurutan).

Misal : 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.

1.3.1 Tipe Operasi Array

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

1.3.2 Penyimpanan dan Pengambilan nilai Array

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

1.3.3 Keunggulan Array

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 elemenelemen 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.

1.3.4 Kelemahan Array

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’.

Pointer (lanjutan)

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 :

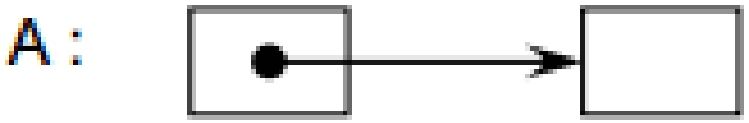
IntegerPointer A, B;

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

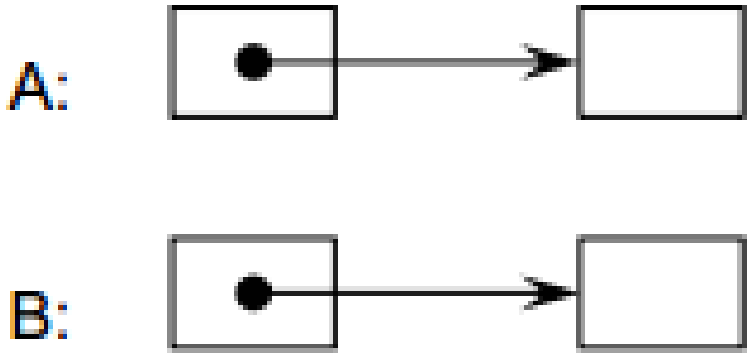
Pointer (lanjutan)

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));



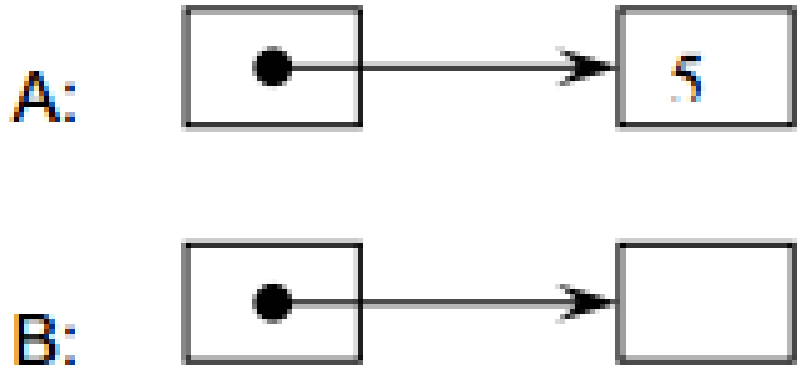
B = (int *) malloc (sizeof(int));



Pointer (lanjutan)

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;*



Hubungan Pointer dengan Linked List

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 (Struct)

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

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 variable struktur dipisahkan dengan tanda koma.

1.5.1 Mendeklarasikan Struktur (Lanjutan)

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.

1.5.1 Mendeklarasikan Struktur (Lanjutan)

Pendefnisian dan pendeklarasian struktur dapat juga ditulis sebagai berikut:

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

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;

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.

Terima Kasih

Pertanyaan?