**ARRAY DINAMIS**

Sebelumnya telah dijelaskan mengenai variable bertipe array (array statis), suatu tipe data yang bersifat statis (urutan dan ukuran sudah pasti). Kelemahan dari array statis adalah penggunaan ruang memori yang sudah digunakan tidak dapat dihapus apabila nama variable array tersebut sudah tidak digunakan kembali dalam suatu program (penyebab kemubaziran).

Untuk pemecahannya maka digunakan struktur data dinamis dengan menggunakan variable dinamis. Variabel dinamis tidak dapat dideklarasikan secara eksplisit seperti halnya variable statis dan tidak dapat ditunjuk oleh identifier secara langsung, tetapi dapat ditunjuk secara khusus oleh variable dinamis yaitu **POINTER.**

Deklarasi secara umum untuk tipe data POINTER adalah sebagai berikut :

Type

Pengenal = Simpul

Simpul = Type

1.

2.

(Nama var) : (Type data)



Keterangan :

Pengenal : Nama pengenal yang menyatakan data berupa pointer

Simpul : Menyatakan nama simpul

Type : Tipe data dari simpul

* : Tanda yang menyatakan bahwa pengenal memiliki tipe data pointer

**Contoh** **:**

* 1. Kamus :

Type

Point = Data

Data = Record

< Nama\_Mhs : String,

Jurusan : String,

Semester : String >

Endrecord

DataMhs : Point

* 1. Kamus :

Jumlah\_data : Integer

Nama\_Siswa : String

Penjelasan:

* DataMhs merupakan variabel bertipe pointer
* Ketika program dikompilasi, variable DataMhs akan menempati lokasi tertentu dalam memori dan variabel tersebut belum menunjuk kesuatu simpul.
* Pointer yang belum menunjuk kesuatu simpul, nilainya dinyatakan sebagai nil guna mengalokasikan simpul dalam memori.
* Statemen yang digunakan untuk membuat sebuah simpul adalah **Alloc (variabel)**
* Catatan : Variabel merupakan nama peubah yang bertipe pointer
* Misal : Alloc (DataMhs); {Berarti kita membuat sebuah simpul yang ditunjuk oleh DataMhs}

Nil

|  |  |  |  |
| --- | --- | --- | --- |
| ? | ? | ? |  |

DataMhs

Keterangan : Masih tanda tanya karena belum terisi data

**Operasi pada Pointer**

1. Mengcopy pointer, artinya bahwa sebuah simpul akan ditunjuk oleh lebih dari sebuah pointer. Contoh :

Kamus :

Type

Point = Data

Data = Record

< Nama\_Mhs : String

Jurusan : String>

Endrecord

T1, T2 : Point;

T1. NamaMhs ‘Benny’



T1. Jurusan ‘Informatika’



|  |  |  |
| --- | --- | --- |
| ? | ? |  |

Alloc (T1); T1

|  |  |  |
| --- | --- | --- |
| ? | ? |  |

Alloc (T2); T2

Akibatnya simpul diatas menjadi :

|  |  |  |
| --- | --- | --- |
| Benny | Informatika |  |

T1

|  |  |  |
| --- | --- | --- |
| ? | ? |  |

T2

Apabila T2  T1

|  |  |  |
| --- | --- | --- |
| Benny | Informatika |  |

T1

|  |  |  |
| --- | --- | --- |
| ? | ? |  |

T2

1. Mengcopy isi simpul, artinya bahwa dua/lebih simpul ditunjuk leh pointer yang berbeda tetapi mempunyai isi yang sama. Apabila T2   T1; Maka

|  |  |  |
| --- | --- | --- |
| Benny | Informatika |  |

T1

|  |  |  |
| --- | --- | --- |
| Benny | Informatika |  |

T2

1. Menghapus pointer

Apabila pointer dihapus, maka lokasi semula yang ditempati oleh simpul yang ditunjuk oleh pointer tersebut akan bebas sehingga bisa digunakan oleh variable lain.

Bentuk : Dealloc (Variabel)

Misal : Dealloc (T1)

Dealloc (T2)

**LINKED LIST**

* Struktur ini terdiri dari rangkaian elemen yang saling berhubungan / berkaitan, dimana setiap elemen dihubungkan dengan elemen lainnya oleh sebuah pointer.
* Pointer, sel yang nilainya merupakan alamat sel yang lain dimana sel yang lain itu dapat berupa data atau berupa pointer juga
* Setiap elemen dalam linked list selalu berisi pointer
* Deklarasi Linked List

Type

Point = Data



Data = Record

<Info : char

Next : point >

Endrecord

Awal, Akhir: Point

Type

Namapointer = Namadata

Namadata = Record

Data : tipe data

Namasambungan:Namapointer

Endrecord

NamaVarPointer : NamaPointer

Istilah – istilah

* Simpul, terdiri dari dua bagian :

1. Bagian data (medan info)
2. Bagian pointer yang menunjuk kesimpul berikutnya (medan penyambung)

* Awal (First), variable yang berisi alamat yang menunjuk lokasi simpul pertama linked list

|  |  |
| --- | --- |
| A |  |

Awal

* Nil / Null, Tidak bernilai yaitu menyatakan tidak mengacu kealamat manapun.

|  |
| --- |
|  |

Nil

* Akhir, sebuah simpul yang tidak menunjuk pada simpul berikutnya

|  |  |
| --- | --- |
| A |  |

|  |  |
| --- | --- |
| B |  |

Awal

Akhir

* Linked List kosong dikenali dengan

Awal 🡨 nil

Awal

* Elemen terakhir linked list dikenali dengan : Akhir.  Next 🡨 Nil
* Linked List terdiri dari 3 macam yaitu :

1. Single Linked List
2. Double Linked List
3. Circular Linked List
   1. **Single Linked List**

Adalah linked list dengan simpul berisi satu link / pointer yang mengacu ke simpul berikutnya. Skema Simpul Single Linked List :

|  |  |
| --- | --- |
|  |  |

Operasi – operasi pada single linked list :

* 1. Penciptaan

Awal dan akhir diberi nil.

Akhir

Awal

* 1. Penyisipan

1. Penyisipan didepan

{awal=nil}

|  |  |
| --- | --- |
|  |  |

Baru

|  |  |
| --- | --- |
| 1 |  |

Baru

Akhir

|  |  |
| --- | --- |
| 1 |  |

Awal

Baru

{Awal ≠ Nil}

Akhir

|  |  |
| --- | --- |
| 1 |  |

Awal

|  |  |
| --- | --- |
| 3 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 4 |  |

Baru

Akhir

|  |  |
| --- | --- |
| 1 |  |

Awal

|  |  |
| --- | --- |
| 3 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 3 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 4 |  |

Baru

Akhir

Awal

|  |  |
| --- | --- |
| 4 |  |

|  |  |
| --- | --- |
| 3 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 1 |  |

Baru

1. Penyisipan ditengah

{Awal = Nil} Sama dengan penyisipan didepan

{Awal ≠ Nil}

Akhir

|  |  |
| --- | --- |
| 1 |  |

Awal

|  |  |
| --- | --- |
| 4 |  |

|  |  |
| --- | --- |
| 2 |  |

Asumsi data : 3 akan disisipkan setelah data ‘4’

|  |  |
| --- | --- |
| 3 | Bantu |

Baru

Akhir

|  |  |
| --- | --- |
| 1 |  |

Awal

|  |  |
| --- | --- |
| 4 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 3 |  |

Baru

Bantu

Akhir

|  |  |
| --- | --- |
| 1 |  |

Awal

|  |  |
| --- | --- |
| 4 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 3 |  |

Baru

Akhir

Awal

|  |  |
| --- | --- |
| 4 |  |

|  |  |
| --- | --- |
| 3 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 1 |  |

1. Penyisipan diakhir

{Awal = Nil} Sama dengan penyisipan didepan

{Awal ≠ Nil}

Akhir

|  |  |
| --- | --- |
| 3 |  |

Awal

|  |  |
| --- | --- |
| 1 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 4 |  |

Baru

Akhir

|  |  |
| --- | --- |
| 3 |  |

Awal

|  |  |
| --- | --- |
| 1 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 4 |  |

Baru

Akhir

Baru

|  |  |
| --- | --- |
| 1 |  |

|  |  |
| --- | --- |
| 2 |  |

|  |  |
| --- | --- |
| 3 |  |

|  |  |
| --- | --- |
| 4 |  |

Awal

* 1. Penghapusan

1. Penghapusan diawal

Penghapusan data di awal adalah proses menghapus elemen pertama (awal), sehingga variable awal akan berpindah ke elemen data berikutnya. Ada 3 kondisi yang perlu diperhatikan yaitu kondisi linked list masih kosong, kondisi linked list hanya memiliki 1 data, dan kondisi linked list yang memiliki data lebih dari 1 elemen

* Kondisi linked list kosong



Pada kondisi ini proses penghapusan tidak bisa dilakukan

* Kondisi linked list memiliki hanya 1 data{Satu simpul}

Akhir

|  |  |
| --- | --- |
| A |  |

Awal

Berikan satu variable Bantu yang sama menunjuk ke simpul awal/akhir

Awal

Akhir

|  |  |
| --- | --- |
| A |  |

Bantu

Kemudian berikan nilai null untuk variable awal dan variable akhir

Simpan terlebih dahulu nilai dari elemennya kedalam suatu variabel

|  |  |
| --- | --- |
| A |  |

Bantu Awal Akhir

|  |  |
| --- | --- |
| A |  |

Bantu

Setelah itu simpul Bantu dapat dihapus

* Kondisi linked list memiliki data lebih dari 1 data



Kemudian alamat data awal diisikan ke suatu variabel pembantu (phapus).



setelah itu pindahkan awal ke data berikutnya.



Setelah itu hapus/hancurkan data di posisi phapus. Sehingga linked list menjadi seperti di bawah ini.



1. Penghapusan ditengah

Penghapusan akan dilakukan pada data 3 dari banyak data sebanyak 4.



Kemudian cari posisi elemen sebelum elemen posisi hapus, kemudian simpan dalam variabel bantu.



Kemudian simpan alamat elemen posisi hapus dalam suatu variabel dengan nama bantu2.



Kemudian pindahkan field next dari bantu ke alamat yang ditunjuk oleh field next dari bantu2.



Hapus elemen data yang ditunjuk dengan bantu2.



Setelah langkah tersebut, maka elemen telah terhapus.



1. Penghapusan diakhir

Kondisi linked list memiliki lebih dari 1 data.



Karena posisi hapus adalah data terakhir, maka nanti posisi akhir harus pindah ke posisi sebelumnya. Oleh karena itu harus dicari posisi data sebelum data terakhir, sebut dengan variabel bantu.



Kemudian pindahkan variable akhir ke simpul yang ditunjuk oleh varibel Bantu dan variable Bantu pindahkan ke simpul yang ditunjuk oleh variable akhir field next-nya.

Akhir

Awal

Bantu

|  |  |
| --- | --- |
| 5 |  |

|  |  |
| --- | --- |
| 9 |  |

|  |  |
| --- | --- |
| 15 |  |

|  |  |
| --- | --- |
| 20 |  |

Simpan elemen variable simpul Bantu. Kemudian lakukan penghapusan pada simpul yang ditunjuk variable Bantu, sehingga hasil akhirnya seperti dibawah ini.

