**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 dapat diillustrasikan sebagai berikut:

|  |  |  |
| --- | --- | --- |
| 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)

**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

 nama\_pointer = ↑Simpul

 Simpul = Record

 medan\_data : tipedata

 medan\_sambungan : Namapointer

 EndRecord

 nama\_var\_pointer : nama\_pointer

Istilah – istilah

* Simpul, terdiri dari dua bagian :
1. Bagian/medan data (info)
2. Bagian/medan sambungan (pointer yang menunjuk kesimpul berikutnya)
* 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 terakhir

|  |  |
| --- | --- |
| 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
	* + **List kosong {Awal=nil}**

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

Baru

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

Baru

Akhir

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

Awal

Baru

* + - **List tidak kosong {Awal ≠ Nil}**

Mula-mula keadaan list sebagai berikut:

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 di tengah
	* + List Kosong {Awal = Nil} (Sama dengan penyisipan di depan)
		+ List tidak kosong {Awal ≠ Nil}

**Mula-mula keadaan list sebagai berikut**:

Akhir

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

Awal

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

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

Asumsi data : 3 akan disisipkan setelah data ‘4’

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

Baru

Akhir

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

Awal

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

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

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

Baru

Bantu

Akhir

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

Awal

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

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

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

Baru

Akhir

Awal

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

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

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

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

1. Penyisipan di akhir
	* + List kosong {Awal = Nil} (Sama dengan penyisipan di depan)
		+ List tidak kosong (Awal ≠ Nil}

**Mula-mula keadaan list sebagai berikut**:

Akhir

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

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

Awal

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

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

Baru

Akhir

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

Awal

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

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

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

Baru

Akhir

Baru

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

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

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

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

Awal

Procedure sisip\_depan\_single(Input elemen : tipedata, I/O awal, akhir : nama\_pointer)

{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}

{F.S. : menghasilkan satu simpul yang disisipkan di depan pada single linked list}

Kamus :

 baru : nama\_pointer

Algoritma :

 Alloc(baru)

 baru↑.info 🡨 elemen

 If (awal = nil)

 Then

 baru↑.next 🡨 nil

 akhir 🡨 baru

 Else

 baru↑.next 🡨 awal

 EndIf

 awal 🡨 baru

EndProcedure

Procedure sisip\_belakang\_single(Input elemen : tipedata, I/O awal, akhir : nama\_pointer)

{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}

{F.S. : menghasilkan satu simpul yang disisipkan di belakang pada single linked list}

Kamus :

 baru : nama\_pointer

Algoritma :

 Alloc(baru)

 baru↑.info 🡨 elemen

 baru↑.next 🡨 nil

 If (awal = nil)

 Then

 awal 🡨 baru

 Else

 akhir↑.next 🡨 baru

 EndIf

 akhir 🡨 baru

EndProcedure

Procedure sisip\_tengah\_single(Input elemen : tipedata, I/O awal, akhir : nama\_pointer)

{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}

{F.S. : menghasilkan satu simpul yang disisipkan di tengah pada single linked list}

Kamus :

 baru,bantu : nama\_pointer

 ketemu : boolean

 datasisip : tipedata

Algoritma :

 If (awal = nil)

 Then

 Alloc(baru)

 baru↑.info 🡨 elemen

 baru↑.next 🡨 nil

 awal 🡨 baru

 akhir 🡨 baru

 Else

 Input(datasisip)

 bantu 🡨 awal

 ketemu 🡨 false

 While (not ketemu and bantu ≠ nil) do

 If (datasisip = bantu↑.info)

 Then

 ketemu 🡨 true

 Else

 bantu 🡨 bantu↑.next

 EndIf

 EndWhile

 If (ketemu)

 Then

 Alloc(baru)

 baru↑.info 🡨 elemen

 If (bantu = akhir)

 Then

 sisip\_belakang\_single(elemen,awal,akhir)

 Else

 baru↑.next 🡨 bantu↑.next

 bantu↑.next 🡨 baru

 EndIf

 Else

 Output(“Data yang akan disisipkan tidak ada”);

 EndIf

 EndIf

EndProcedure

Procedure isi\_elemen(Output elemen : tipedata)

{I.S. : user memasukkan data yang akan disisipkan (elemen)}

{F.S. : mengahsilkan data yang akan disisipkan (elemen)}

Kamus :

Algoritma :

 Input(elemen)

EndProcedure

{Algoritma Utama}

Algoritma\_penyisipan\_Single\_Linked\_List

Kamus :

 {prototype}

Procedure isi\_elemen(Output elemen : tipedata)

 Procedure sisip\_depan\_single(Input elemen : tipedata, I/O awal, akhir : nama\_pointer)

 Procedure sisip\_belakang\_single(Input elemen : tipedata, I/O awal, akhir : nama\_pointer)

 Procedure sisip\_tengah\_single(Input elemen : tipedata, I/O awal, akhir : nama\_pointer)

{deklarasi}

 nama\_pointer = ↑simpul

 simpul = Record

 < info : tipedata,

 next : nama\_pointer>

 EndRecord

 awal, akhir : nama\_pointer {pointer penunjuk}

 menu : integer

 elemen : tipedata

Algoritma :

 {penciptaan list}

awal 🡨 nil

 akhir 🡨 nil

 {penyisipan}

 Repeat

 output(“Menu Pilihan”)

 output(“==========”)

 output(“1. Sisip Depan”)

 output(“2. Sisip Tengah”)

 output(“3. Sisip Belakang”)

 output(“0. Keluar”)

 Input(menu)

 Depend on (menu)

 (menu = 1) : isi\_elemen(elemen)

 sisip\_depan\_single(elemen, awal, akhir)

 (menu = 2) : isi\_elemen(elemen)

 sisip\_tengah\_single(elemen, awal, akhir)

 (menu = 3) : isi\_elemen(elemen)

 sisip\_belakang\_single(elemen, awal, akhir)

 EndDepend

 Until (menu = 0)