STRUKTUR DATA

Array

Array merupakan bagian dasar pembentukan suatu struktur data yang lebih kompleks. Hampir setiap jenis struktur data kompleks dapat disajikan secara logik oleh array.

- **Array** : Suatu himpunan hingga elemen yang terurut dan homogen, atau dapat didefinisikan juga sebagai pemesanan alokasi memory sementara pada komputer.
- **Terurut** : elemen tersebut dapat diidentifikasi sebagai element pertama, kedua, dan seterusnya sampai elemen ke-n.
- **Homogen** : setiap elemen data dari sebuah array tertentu haruslah mempunyai tipe data yang sama.

Karakteristik Array :

1. Mempunyai batasan dari pemesanan alokasi memory (bersifat statis)
2. Mempunyai type data sama (bersifat Homogen)
3. Dapat diakses secara acak.
4. Berurutan (terstruktur).

Array mempunyai dimensi :

1. Array Dimensi Satu (Vektor)
2. Array Dimensi Banyak.
    - Dimensi Dua (Matriks/Tabel)
    - Dimensi Tiga (Kubik).

ARRAY DIMENSI SATU

- Merupakan bentuk yang sangat sederhana dari array.
- Setiap elemen array mempunyai subskrip/indeks.
- Fungsi indeks/subskrip ini antara lain :

1. Menyatakan posisi elemen pada array tsb.
2. Membedakan dengan elemen lain.

Penggambaran secara fisik Array A(1:N) :

A(1)	A(2)	A(3)	A(4)	…	A(N)

Ket : A : nama array

1,2,3,4,…,N : indeks / subskrip

- Secara umum Array Dimensi Satu A dengan tipe T dan subskrip bergerak dari L sampai U ditulis : **A(L:U) = (A(I)); I =L , L+1, L+2, …, U**

  Keterangan : L : batas bawah indeks / lower bound

  U : batas atas indeks / upper bound

  A : nama Array
- Banyaknya elemen array disebut **Rentang atau Range A(L:U) = U – L + 1**
- Range khusus untuk array Dimensi Satu yang mempunyai batas bawah indeks L=1 dan batas atas U=N, maka **Range A adalah A(1:N) =(N – 1) + 1 = N**

Contoh :

Data hasil pencatatan suhu suatu ruangan setiap satu jam, selama periode 24 jam ditulis dalam bentuk Array Dimensi Satu menjadi

Misal :

nama arraynya Suhu, berarti elemennya dapat kita tulis sebagai Suhu(I), dengan batas bawah 1 dan batas atas 24.

Suhu(I):menyatakan suhu pada jam ke-I dan 1<=I<=24

Range Suhu(1:24)=(24-1)+1=24

ARRAY DIMENSI BANYAK

- Array Dimensi Banyak (Multi-Dimensional ARRAY) :

  suatu array yang elemen –elemennya berupa array juga.
- **Array Dimensi Dua perlu dua subskrip/indeks** :

  a. Indeks pertama untuk menyatakan posisi baris

b. Indeks kedua untuk menyatakan posisi kolom

- Secara umum Array Dimensi Dua B dengan elemen-elemen bertype data T dinyatakan sbb : **B(L1:U1,L2:U2)={B(I,J)}**

  L1<=1<=U1, L2<=1<=U2, dan setiap elemen B(I,J) bertype data T

  Keterangan : B = nama array
  L1 = batas bawah indeks baris
  L2 = batas bawah indeks kolom
  U1 = batas atas indeks baris
  U2 = batas atas indeks kolom

- Jumlah elemen baris dari array B adalah **(U2 - L2 + 1)**
- Jumlah elemen kolom dari array B adalah **(U1 – L1 + 1)**
- Jumlah total elemen array adalah **(U2 – L2 + 1) * (U1 – L1 + 1)**
- Suatu array B yang terdiri atas M elemen dimana elemennya berupa array dengan N elemen, maka dapat digambarkan sbb:

	1	2	3	……	J	….. N
1						
2						
3						
…						
I					B(I,J)	
…						
M						

- Array diatas dituliskan :

  B(1:M, 1:N) = B(I,J) ;

  Untuk I = 1,2,3,…….,M dan

  J = 1,2,3,……., N

- Jumlah elemen array B : M * N
- Array B berukuran / berorder : M * N

CROSS SECTION

- **Cross Section** dari array berdimensi dua adalah suatu himpunan yang anggotanya adalah elemen-elemen dalam satu baris saja atau satu kolom saja.
- **Notasinya : ***

Misal

Array B(1:M ;1:N) = {B(I,J)};

I = 1,2,3,……..,M dan

J = 1,2,3,……..,N

Maka suatu Cross Section :

B(5,*) = {B(5,1), B(5,2), B(5,3),……, B(5,N)}

B(*,5) = {B(1,5), B(2,5), B(3,5),……, B(M,5)}

TRANSPOSE

- **Transpose** dari suatu array berdimensi dua adalah menukar posisi indeksnya (menukar posisi baris menjadi kolom atau kolom menjadi baris).
- Transpose suatu array dari B **dinotasikan B^T**
- **B adalah array dua dimensi, B(I,J) maka B^T (J,I)**
- A adalah array dua dimensi yang berorder M x N memepunyai transpose (A^T) N x M

ARRAY DIMENSI TIGA

- Banyaknya indeks yang diperlukan array dimensi tiga adalah 3
- Pada umumnya, suatu array berdimensi N memerlukan N indeks untuk setiap elemennya.
- **Secara acak array berdimesi N ditulis sbb:**

  A(L1:U1, L2:U2, …….., LN:UN) = (A(I1,I2,……,IN)) dengan $L_K <= I_K <= U_K$, k = 1,2,3,…, N

Contoh :

Penyajian data mengenai jumlah mahsiswa Manajemen Informatika Universitas Gunadarma berdasarkan tingkat, untuk kelas pagi dan malam dan jenis kelamin.

Jawab :

MHS = nama array

I = 1,2,3,4,5 (tingkat 1/5)

J = 1,2 (1 = pagi; 2 = malam)

K = 1,2 (1 = pria; 2= wanita)

MHS (1:5, 1:2,1:2)

Jadi :

- MHS(3,2,2)

  jumlah mahasiswa tingkat 3 MI kelas malam untuk jenis kelamin wanita

- Cross Section MHS (1,*,2)

  jumlah mahasiswa tingkat 1 MI kelas pagi atau malam dan berjenis kelamin wanita.

MAPPING KE STORAGE DARI ARRAY

Skema penyajian dapat dievaluasi berdasarkan 4 karakteristik yaitu :

1. Kesederhanaan dari akses elemen
2. Mudah ditelusuri
3. Efisiensi dari utilitas storage
4. Mudah dikembangkan

ARRAY SATU DIMENSI

Misal:

Diberikan array dengan nama B yang mempunyai indeks 1 s/d N yaitu A(1:N). Cara untuk menyimpan array tersebut adalah sedemikian sehingga urutan secara fisik dari elemen-elemen adalah sama dengan urutan logik dari elemen.

Untuk mengetahui Address Awal (Starting Address) dari elemen suatu array A(I) perlu diketahui :

1. Address Awal dari array yang bersangkutan
2. Ukuran dari masing-masing elemen array

- Address Awal dari array dinyatakan dengan notasi Address Awal yaitu **B (Base Location)**
- Masing-masing elemennya menggunakan ruang sebanyak **S byte**.
- Address awal dari elemen ke-I dari array A(1:N) adalah :
  B + (I – 1) * S
- Address Awal yang mempunyai batas bawah tidak sama dengan satu ,elemen ke-I dari array A(L:U) adalah : **B + (I– L) * S**

Misal :

A (1:7), address awal dari elemen A(5) adalah : B + (5 –1) * S
A (3:8), address awal dari elemen A(6) adalah : B + (6 - 3) * S
A (-3:4), address awal dari elemen A(2) adalah : B + (2 –(- 3)) * S

ARRAY MULTI DIMENSI

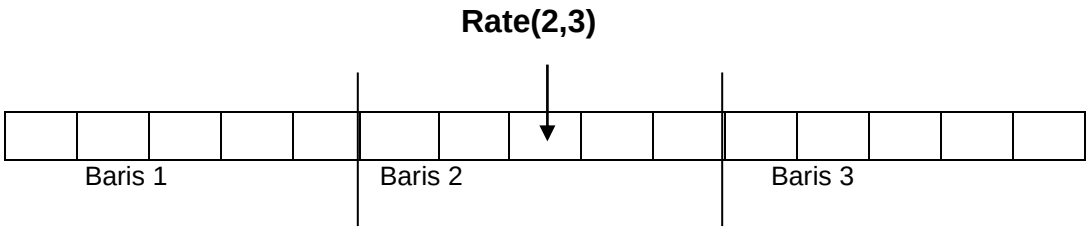
Memori komputer linier, maka untuk memetakan array dimensi banyak ke storage harus dilinierkan.
Alternatif untuk pelinieran tersebut adalah :

Row Major

Biasanya digunakan COBOL, PASCAL
Menyimpan pertama kali baris pertama dari array, kemudian baris kedua, ketiga dst
Array Rate (1:3 , 1:5)

	1	2	3	4	5
1					
2			Rate(2,3)		
3					

Menjadi



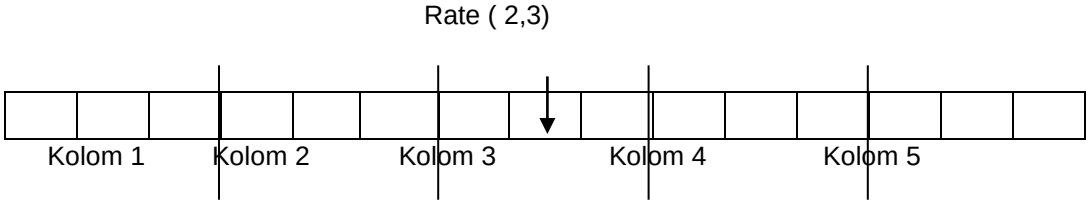
- Array A(I,J) dari array yang didefinisikan sebagai array
  A(L1:U1 , L2:U2), mempunyai
  Address awal : B + (I – L1) * (U2 – L2 +1) * S + (J – L2) * S

Contoh :

- Array A(1:3, 1:5) dan elemen A(2,3) mempunyai address awal :
  B + (2-1) * (5-1+1) * S + (3-1) * S
  B + 5 * S + 2 * S
  B + 7 * S
- Array A(2:4, 3:5) dan elemen A(3,4) mempunyai address awal :
  B + (3-2) * (5-3+1) * S + (4-3) * S
  B + 1 * 3 * S + 1 * S
  B + 4 * S

Column Major

Biasanya digunakan FORTRAN
Menyimpan Kolom pertama dari array kemudian kolom kedua, ketiga, dst
Menjadi



- Array A(I,J) dari array yang didefinisikan sebagai array
  A(L1:U1 , L2:U2) mempunyai
  Address awal : B + (J – L2) * (U1 – L1 +1) * S + (I – L1) * S

Contoh :

- Array A(1:3, 1:5) dan elemen A(2,3) mempunyai address awal :
  B + (3-1) * (3-1+1) * S + (2-1) * S
  B + 6 * S + 1 * S
  B + 7 * S
- Array A(2:4, 3:5) dan elemen A(3,4) mempunyai address awal :
  B + (4-3) * (4-2+1) * S + (3-2) * S
  B + 1 * 3 * S + 1 * S
  B + 4 * S

ARRAY SEGITIGA (TRINGULAR ARRAY)

Ada 2 macam
1. Upper Tringular
   Elemen dibawah diagonal utama adalah 0
2. Lower Tringular
   Elemen diatas diagonal utama adalah 0

- Suatu array dimana elemen diagonalnya juga nol disebut **Strictly (upper/lower) Tringular.**
- Pada array Lower Tringular dengan N baris, jumlah maksimum elemen <> 0 pada baris ke-I adalah I
- **Total elemen <> 0 adalah** $\sum_{K=1}^{N} I = (N * (N+1)) /2$
- Untuk N kecil : tidak ada masalah

- Untuk N besar :
  1. Elemen yang sama dengan nol tidak usah disimpan dalam memori
  2. Pendekatan terhadap masalah ini adalah dengan pelinieran array dan hanya menyimpan array yang tidak nol.

1. Misal

   A array segitiga atas berorder N x N

   B array bersegitiga bawah dengan order (N-1) x (N-1)

   A dan B dapat disimpan sebagai array C berorder N x N

   C (I,J) = A(I,J) untuk I <= J

   C(I+1,J) = B(I,J) untuk I >= J

2. Misal

   A array segitiga atas berorder N x N

   B array bersegitiga bawah dengan order N x N

   A dan B dapat disimpan sebagai array C berorder N x (N + 1)

   C (I,J+1) = A(I,J) untuk I <= J

   C(I,J) = B(I,J) untuk I >= J

3. Misal

   A dan B keduanya merupakan array segitiga atas

   Maka untuk menyimpannya secara bersama-sama dengan melakukan transpose terhadap salah satu array tersebut.

   Array C berorder N x (N+1)

   C (I,J+1) = A(I,J) untuk I <= J

   C(J,i) = B(I,J) untuk I >= J

SPARSE ARRAY

- Suatu type khusus yang lain dari array
- Dikatakan Sparse atau jarang karena elemen-elemen yang tidak nolnya relatif lebih sedikit jumlahnya

- Setiap elemen bukan nol pada sparse array dua dimensi dapat direpresentasikan dalam format **(Row-Subscript, Column-subscript, value).**
- Triple ini dapat diurut berdasarkan Row-Subscript Major dan Colum-Subscript Minor dan disimpan dalam bentuk vektor.
- Penyajian lain dari Sparse Array adalah dengan menggunakan daftar berkait/ Linked List.

STACK

❑ *LINEAR LIST*

- ♦ Linear list adalah suatu struktur data yang merupakan himpunan terurut dari satuan data atau dari record.
- ♦ Elemen yang terdapat dalam daftar disebut simpul/node.
- ♦ Daftar disebut Linear karena elemen nampak seperti baris, bahwa setiap simpul (kecuali simpul pertama dan terakhir) selalu memiliki elemen penerus langsung (suksesor) dan elemen pendahulu langsung (predesor).
- ♦ Misalnya didefinisikan suatu linear list A yang terdiri atas T buah elemen sebagai berikut :

  A=[a1,a2,……, aT]

  Jika T = 0 , maka A dikatakan sebagai **"Null List" (list hampa).**
- ♦ Suatu elemen dapat dihapus (delete) dari sembarang posisi pada linear list .
- ♦ Suatu elemen baru dapat dimasukkan (insertion) kedalam list dan dapat menempati sembarang posisi pada list tersebut.
- ♦ Jadi suatu linear list dapat berkurang atau bertambah setiap saat

  Contoh : file merupakan linier list yang elemen-elemennya berypa record.

❑ *DEFINISI STACK*

STACK adalah suatu bentuk khusus dari linear list di mana operasi penyisipan dan penghapusan atas elemen-elemennya hanya dapat dilakukan pada satu sisi saja yaitu posisi akhir dari list. Posisi ini disebut puncak atau disebut sebagai **"TOP(S)".**

- ♦ Prinsip Stack adalah **LIFO (Last In First Out)** atau Terakhir masuk pertama keluar.

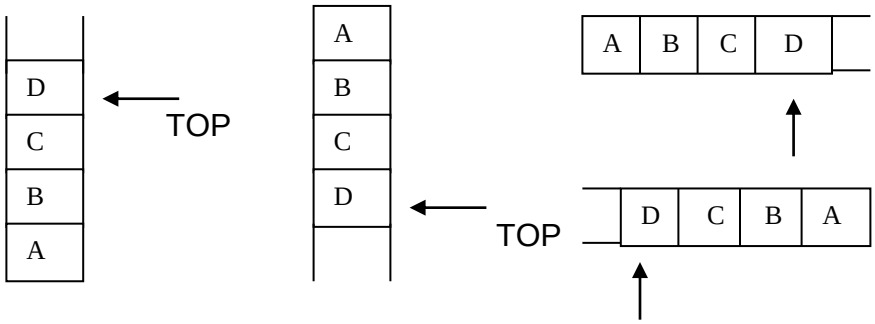
  Setiap elemen tidak dapat dikeluarkan (POP keluar) sebelum semua elemen diatasnya dikeluarkan.
- ♦ Elemen teratas (puncak) dari stack dinotasikan sebagai TOP(S)

  Misal diberikan stack S sebagai berikut :

S= [S2,S2,……, ST] ➔ maka TOP(S) = ST

- Untuk menunjukkan jumlah elemen suatu stack digunakan notasi **NOEL(S).**
  Dari stack diatas maka NOEL(S) = T.
  NOEL(S) menghasilkan nilai integer.

Jika diberikan sebuah stack S = [A,B,C,D] maka stack S ini dapat digambarkan sebagai berikut :



❑ *OPERASI PADA STACK*

1. CREATE (STACK)
2. ISEMPTY (STACK)
3. PUSH (ELEMEN, STACK)
4. POP (STACK)

➢ CREATE(S)

Operator ini berfungsi untuk membuat sebuah stack kosong (menjadi hampa) dan didefinisikan bahwa

NOEL (CREATE (S)) = 0 dan

TOP (CREATE(S)) = null / tidak terdefinisi

➢ ISEMPTY(S)

Operator ini berfungsi untuk menentukan apakah suatu stack adalah stack kosong (hampa) atau tidak . Operasinya akan bernilai boolean dengan definisi sebagai berikut :

ISEMPTY(S) = true, jika S adalah stack kosong atau NOEL(S) = 0

False, jika S bukan stack kosong atau NOEL(S) $\neq$ 0

Catatan : ISEMPTY(CREATE(S)) = true

- **PUSH(E,S)**
  - Operator ini berfungsi untuk menambahkan satu elemen ke dalam stack . Notasi yang digunakan adalah **PUSH(E,S)**
    Artinya : menambahkan elemen E ke dalam stack S
  - Elemen yang baru masuk ini akan menempati posisi TOP jadi **TOP(PUSH(E,S)) = E**
  - Akibat dari operasi ini jumlah elemen dalam stack akan bertambah, artinya NOEL (S) menjadi lebih besar atau stack menjadi tidak kosong (ISEMPTY(PUSH(E,S)) = false)
- **POP(S)**
  - Operator ini berfungsi untuk mengeluarkan satu elemen dari dalam stack, notasinya **POP(S)**
  - Elemen yang keluar dari dalam stack adalah elemen yang berada pada posisi TOP.
  - Akibat dari operasi ini jumlah elemen stack akan berkurang atau NOEL(S) berkurang 1 dan elemen pada posisi TOP akan berubah.
  - Operator ini tidak dapat digunakan pada stack kosong, artinya **POP(CREATE(S)) = error condition dan**
    POP(PUSH(E,S)) = S

Catatan : TOP(PUSH(E,S)) = E

Queue

Adalah suatu bentuk khusus dari linear list dengan operasi penyisipan (insertion) hanya pada salah satu sisi (Rear/ belakang) dan operasi penghapusan (deletion) hanya diperbolehkan pada sisi lainnya (Front/ depan) dari list.

Antrean Q = [Q1, Q2, Q3,……., QT]

Front(Q) = bagian depan dari antrean Q

Rear(Q) = bagian belakang dari antrean Q

Noel(Q) = Jumlah elemen di dalam antrean (berharga integer)

Jadi : Front(Q) = QT

Rear(Q) = Q1

Noel(Q) = T

Antrean beroperasi secara **FIFO (First In First Out)** yang pertama masuk, yang pertama keluar.

- **Operasi dasar pada Antrean :**

1. CREATE(Q)

Operator untuk membentuk suatu antrean hampa

Q = [,…….,]

NOEL(CREATE(Q)) = 0

FRONT(CREATE(Q)) = tidak didefinisikan

REAR(CREATE(Q)) = tidak didefinisikan

2. ISEMPTY(Q)

Operator yang menentukan apakah antrean Q hampa atau tidak.

Operand dari operator ISEMPTY adalah antrean.

Hasilnya bertipe data Boolean

ISEMPTY(Q) =TRUE jika Q adalah antrean hampa (NOEL(Q) = 0)

FALSE jika Q bukan antrean kosong (NOEL(Q) $\neq$ 0)

3. INSERT(E,Q)

Operator yang menyisipkan elemen E ke dalam antrean Q

Catt : Elemen Q ditempatkan pada bagian belakang antrean dan antrean menjadi lebih panjang

Q = [A, B, C, D]

REAR(INSERT(E,Q)) = E

FRONT(Q) = A

NOEL(Q) = 5

ISEMPTY(INSERT(E,Q)) = FALSE

4. REMOVE(Q)

Operator yang menghapus elemen bagian depan dari antrean Q dan antrean menjadi lebih pendek

Jika NOEL(Q) = 0 maka

REMOVE(Q) = ERROR (UNDERFLOW)

- **Penyajian dari antrean :**

1. One way list
2. Array

- Menunjukkan bagaimana suatu antrean dalam array Queue dengan N elemen

1. Antrean mula-mula terdiri dari elemen AAA, BBB, CCC, DDD

AAA	BBB	CCC	DDD				…….	
1	2	3	4	5	6	7	8	9

FRONT(Q) = AAA : 1

REAR(Q) = DDD : 4

2. REMOVE(Q)

	BBB	CCC	DDD				…….	
1	2	3	4	5	6	7	8	9

FRONT(Q) = BBB : 2

REAR(Q) = DDD : 4

3. INSERT(EEE,Q)

	BBB	CCC	DDD	EEE			……..	
1	2	3	4	5	6	7	8	9

FRONT(Q) = BBB : 2

REAR(Q) = EEE : 5

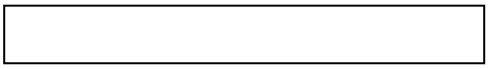
KESIMPULAN :

Untuk setiap kali penghapusan nilai FRONT bertambah

Untuk setiap kali penambahan nilai REAR akan bertambah

- Antrean yang disimpan dalam array dengan 5 lokasi memori sebagai array Sirkular.

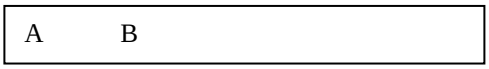
1. Pada Awal Hampa



1 2 3 4 5

FRONT = 0
REAR = 0

2. A dan B dimasukkan

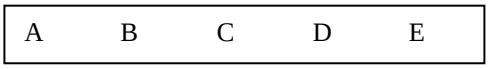


1 2 3 4 5

FRONT : 1

REAR : 2

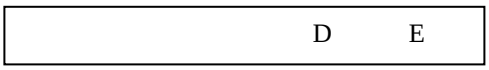
3. C, D , dan E dimasukkan



1 2 3 4 5

FRONT :1
REAR : 5

4. A, B, dan C dihapus

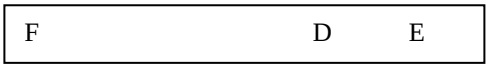


1 2 3 4 5

FRONT : 4

REAR :5

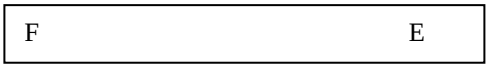
5. F dimasukkan



1 2 3 4 5

FRONT : 4
REAR : 1

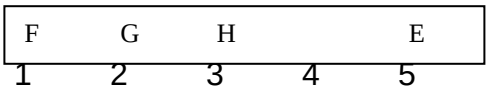
6. D dihapus



1 2 3 4 5

FRONT : 5
REAR :1

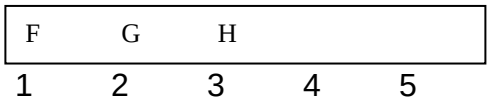
7. G dan H dimasukkan



1 2 3 4 5

FRONT : 5
REAR : 3

8. E dihapus



FRONT :1
REAR : 3

- **ALGORITMA QINSERT**

QINSERT(QUEUE, N, FRONT, DATA)

1. {Apakah antrean penuh}

   Jika FRONT = 1 dan REAR = N atau Jika FRONT = REAR+1, maka Write : OVERFLOW, return.

2. Jika FRONT = NULL maka FRONT := 1, REAR := 1

   Dalam hal ini

   Jika FRONT = N maka REAR := 1

   Dalam hal lain

   REAR := REAR + 1

3. QUEUE(REAR) := DATA (masukkan elemen baru)
4. Return

- **ALGORITMA QDELETE**

QDELETE(QUEUE, N, FRONT, REAR, DATA)

1. {Apakah antrean kosong}

   Jika FRONT = NULL, maka Write : UNDERFLOW, return.

2. DATA := QUEUE(FRONT)
3. (FRONT mendapat nilai baru)

   Jika FRONT =REAR maka FRONT := NULL

   REAR := NULL

   Dalam hal ini

   Jika FRONT := N maka FRONT := 1

   Dalam hal lain

   FRONT := FRONT + 1

4. Return

LINK LIST

PENDAHULUAN

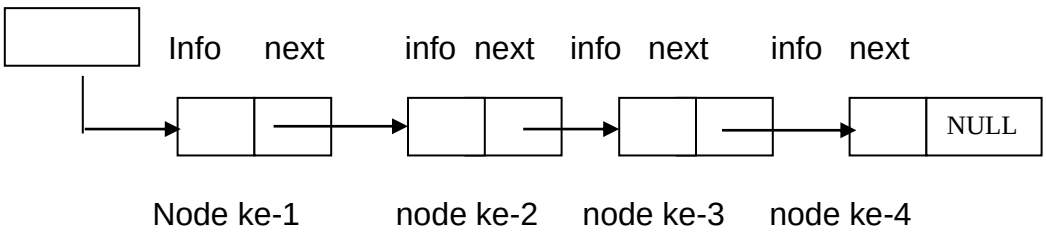
- Dalam suatu linier list kita dapat melakukan operasi penyisipan atau penghapusan atas elemen-elemennya pada sembarang posisi.
- Misalkan ada 1500 item yang merupakan elemen dari suatu linier list. Jika elemen ke-56 akan kita keluarkan, maka elemen ke-1 s/d elemen ke-55 tidak akan berubah posisinya pada linier list tersebut. Tetapi elemen ke–57akan menjadi elemen ke-56, elemen ke-58 akan menjadi elemen ke-57 dst. Selanjutnya, jika kita sisipkan satu elemen pada posisi setelah elemen ke-41, maka elemen ke-42 s/d elemen ke-1500 akan berubah posisinya.

- Untuk menyatakan keadaan diatas diperlukan suatu konsep yang berbeda dengan konsep sekuensial sebelumnya.
  Linked list merupakan suatu cara non-sekuensial yang digunakan untuk merepresentasikan suatu data.

DEFINISI

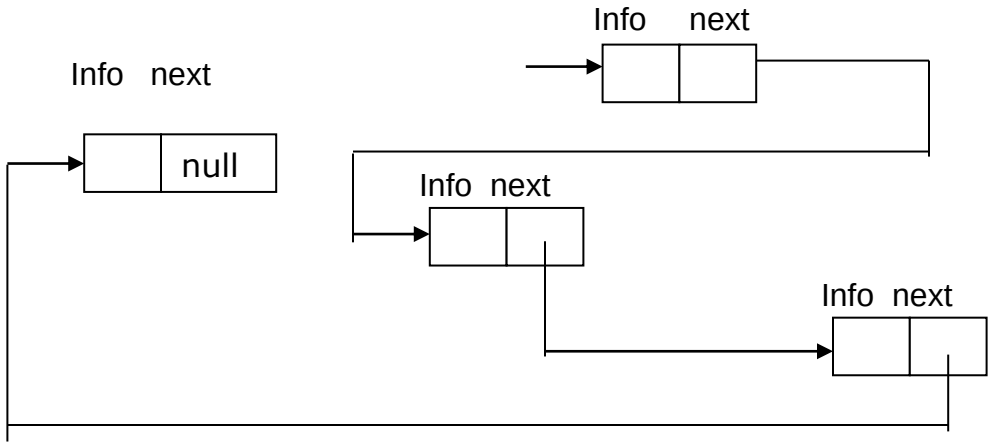
- ***Linked list (one way list)*** adalah suatu kumpulan elemen data (yang disebut sebagai node) dimana urutannya ditentukan oleh suatu pointer.
- Setiap elemen (node) dari suatu linked list terdiri atas dua bagian, yaitu:
  - INFO berisi informasi tentang elemne data yang bersangkutan.
  - NEXT (link field/next pointer field), berisi alamat dari elemen (node) selanjutnya yang dituju.

Berikut ini sebuah contoh linked list yang terdiri atas 4 node:



Pada node ke-4 field NEXT–nya berisi ***NULL ,*** artinya node ke-4 tsb adalah node terakhir.

- Node-node dalam linked list tidak harus selalu digambarkan paralel seperti pada gambar diatas. Linked list pada contoh diatas dapat pula digambarkan seperti berikut ini:



CATATAN :

- Ada dua hal yang menjadi kerugian dengan representasi suatu data dengan linked list ini,
  yaitu:
  1. Diperlukan ruang tambahan untuk menyatakan/tempat field pointer.
  2. Diperlukan waktu yang lebih banyak untuk mencari suatu node dalam linked list.
- Sedangkan keuntungannya adalah :
  1. Jenis data yang berbeda dapat di-link
  2. Operasi REMOVE atau INSERT hanya dilakukan dengan mengubah pointer-nya saja .

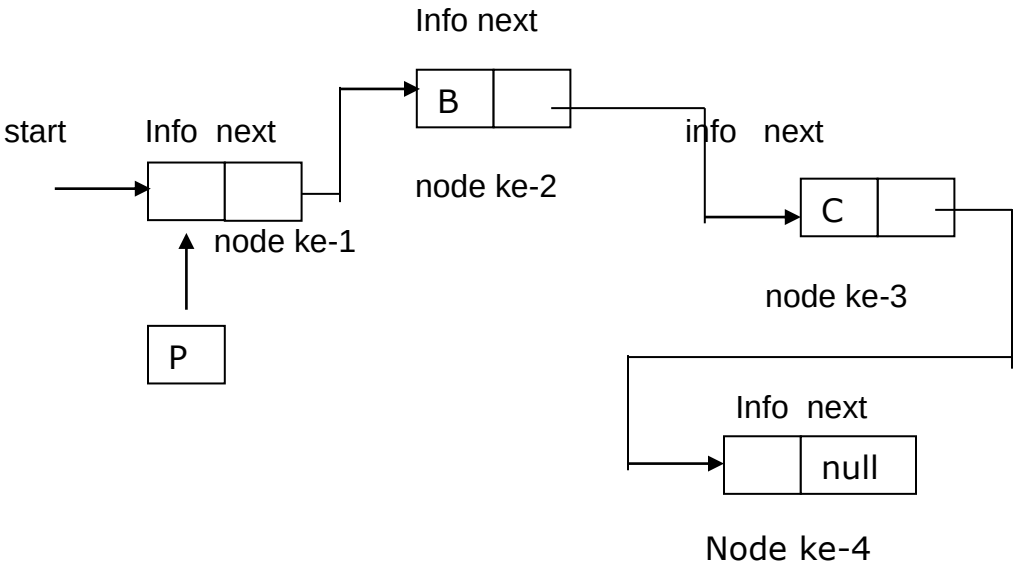
OPERASI DASAR PADA LINKED LIST

- Ada beberapa aturan yang didefinisikan pada operasi didalam linked list yaitu:
  - Jika P adalah suatu variabel pointer, maka nilainya adalah alamat atau lokasi dari variabel lain yang dituju.

- Operasi yang didefinisikan pada suatu variabel pointer adalah:
  1. Test apakah sama dengan NULL
  2. Test untuk kesamaan denganvariabel pointer lain
  3. Menetapkan sama dengan NULL
  4. Menetapkan menuju ke node lain

- Notasi yang didefinisikan sehubungan dengan operasi diatas adalah
  1. NODE (P), artinya node yang ditunjuk oleh pointer P
  2. INFO (P), artinya nilai INFO dari node yang ditunjuk pointer P
  3. NEXT (P), artinya hubungan (link) selanjutnya dari node yang ditunjuk oleh pointer P

Sebagai contoh, perhatikan linked list dibawah ini:



NODE (P) = node yang ditunuk oleh P yaitu node pertama

INFO (P) = A

NEXT (P) = node kedua

INFO (NEXT(NEXT(P))) = C

MENGHAPUS SUATU NODE DARI LINKED LIST (REMOVE)

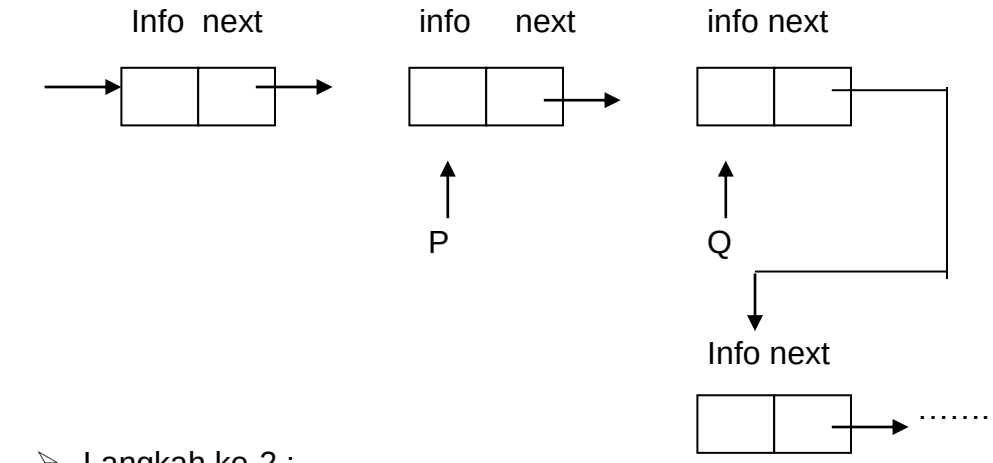
- Untuk menghapus node dalam linked list digunakan procedure ***FREENODE.***

- Jika Q adalah suatu variabel pointer, maka FREENODE (Q) akan menyebabkan node yang ditunjuk oleh variabel poinnter Q dihapus dalam linked list.

  Perhatikan linked list berikut :

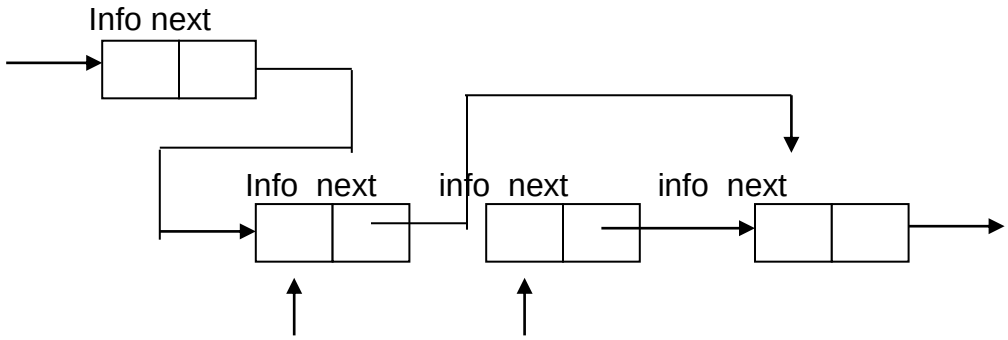
  - Langkah ke-1 :

    Q := Next (P)



  - Langkah ke-2 :

    Next (P) := Next (Q)



  - Langkah ke-3 :

    Freenode (Q)

    Procedure Freenode (Q)

    (a) Next (Q) := Avail

    (b) Info (Q) := Null

    (c) Avail := Q

MENYISIPKAN SUATU NODE KEDALAM LINKED LIST

- Untuk menyisipkan node dalam linked list digunakan procedure *GETNODE*
- Jika NEW adalah suatu variabel pointer, maka GETNODE (NEW) akan menyebabkan node yang ditunjuk oleh variabel pointer NEW disisipkan kedalam linked list.

  Perhatikan linked list berikut:

  Procedure Getnode (NEW)

  If Avail = Null

  Then out-of-free-space

(a) else begin

  Getnode := Avail

(b) Avail := Next (Avail)

(c) Next (Getnode) := Null;

  End;

- **Algoritma menyisipkan sebuah node** :
  (a) Getnode (NEW)
  (b) Info (NEW) := Name;
  (c) Q := Next (P)
  (d) Next (P) := NEW
  (e) Next (NEW) := Q