II. ALOKASI & REPRESENTASI

2.1. ALOKASI.

Alokasi adalah proses "*pemesanan*" ruang memori untuk ditempati oleh data atau variabel. Sebaliknya dealokasi adalah proses "*pembebasan*" ruang memori yang tadinnya sudah dipesan (dan diisi oleh nilai data atau variabel tertentu). Pembebasan, artinya bahwa ruang memori tersebut selanjutnya dapat dipakai lagi untuk keperluan lain, atau bahkan dipesan kembali.
Alokasi pada dasarnya dapat dilakukan secara statik atau dinamik.

ALOKASI STATIK :

- dipesan sekaligus dengan ukuran tetap selama kehidupannya (meskipun pada beberapa sistem dapat di realokasi, atau diubah), tanpa dapat dibebaskan (pada beberapa sistem dapat dibebaskan).
- pemesanan tempat dilakukan pada saat "*inisiasi*" bukan tepat pada saat "*akan dipakai*".
- biasanya sifat alokasi "*contigous*", artinya menempati suatu range ruang memori secara menerus (berurutan), tanpa selang.
- pada umumnya proses alokasi (*dealokasi dan realokasi*) berada diluar jangkauan kendali program.

ALOKASI DINAMIK :

- mungkin saja dipesan sedikit demi sedikit sesuai dengan kebutuhan pada setiap tahapan program.
- sifat alokasi bisa *tidak "contiguous"* (bersebelahan).
- proses alokasi (dealokasi dan realokasi) berada sepenuhnya didalam jangkauan kendali program.

CARA MENYATAKAN alokasi, dealokasi dan realokasi pada setiap sistem pemrograman hampir tidak pernah sama, biasanya tergantung dari :
- level dan jenis bahasa pemrograman
- jenis arsitektur komputer

Bahkan dalam suatu lingkungan pemrograman tertentu, alokasi kadang kadang tidak perlu dinyatakan dengan sengaja, sehingga bersifat *implisit* dan *transparan*. Oleh karena itu sukar dilacak apakah termasuk kategori statik atau dinamik.

Secara umum, objek yang terlibat dalam proses alokasi adalah :
- *tipe data*
- *ukuran* ruang memori dalam satuan jumlah data
- suatu *variabel pointer*

Sebagai hasil proses alokasi :
- telah disisihkan ruang memori dengan ukuran sesuai permintaan
- sebuah variabel pointer yang menunjuk posisi *"awal"* ruang memori tsb.

Dalam pembahasan selanjutnya jika diperlukan akan ditulis dengan notasi sbb :

p ← alloc (ukuran, tipe-data)

dimana p adalah sebuah variabel pointer.

CONTOH APLIKASI :

MASALAH :

- sebuah algoritma bertujuan menerima sekelompok data nilai numerik integer dari keyboard.
- diinginkan dapat memanfaatkan memori secara ekonomik, akan tetapi juga dapat beradaptasi dengan bervariasinya jumlah data.
- juga dikehendaki bahwa alokasi dilakukan sepenuhnya dibawah kendali program.

BENTUK ALGORITMA :

```
read(keyboard) ndata                    {terima dari keyboard, jumlah data yang akan di "entry" }
p ← alloc ( ndata, integer )            { alokasi contiguous sebesar jumlah data }

i ← 1
while  i <= ndata                       { terima ndata buah data }
    do  read(keyboard) (p+i-1) ↑        { baca dari keyboard, tampung di lokasi yang  berjarak (
                                        i - 1 ) dari lokasi yang ditunjuk oleh pointer p }
        i ← i + 1                       { tambah nilai i }
endwhile
```

2.2. REPRESENTASI.

Dalam pemrograman, data yang diproses seringkali sangat kompleks, sedangkan struktur dalam tempat penyimpanan komputer sangat sederhana. Karena itu dibutuhkan suatu "abstraksi" dari perepresen-tasian data yang kompleks, supaya pemrogram dapat memusatkan struktur datanya terhadap "lojik" dari program.
Maka dikenal tiga struktur data, yaitu :

1. Definisi Fungsional, yaitu pendefinisian struktur data dan operator-operatornya yang berlaku pada struktur tersebut.
2. Representasi Lojik, yaitu spesifikasi dari struktur, yang menyangkut nama type dan spesifikasi semua operator.
   Representasi lojik tidak bergantung pada memori komputer. Struktur ini memudahkan pemrogram untuk merancang data dan algoritma.
3. Representasi (implementasi) fisik, yaitu spesifikasi dari struktur data sesuai dengan implemantasinya dalam memory komputer.
   Pada dasarnya hanya ada dua macam implementasi fisik, yaitu :
   - Representasi Kontigu, yaitu sekumpulan data yang penempatannya dalam memory benar-benar secara fisik adalah kontigu, setiap elemen ditaruh secara berurutan dengan elemen lainnya. Struktur ini biasa disebut dengan struktur yang statis.
   - Representasi fisik berkait , yaitu sekumpulan data yang penempatannya pada memory komputer dapat terpencar-pencar , namun dapat ditelusuri berkat adanya informasi berupa alamat yang menghubungkan elemen yang satu dengan yang lainnya. Struktur ini biasa disebut dengan struktur yang dinamis.

Dalam dunia sistem informasi, khususnya pada daerah masalah pemrograman dan struktur data, representasi merupakan suatu aspek yang penting. Suatu bentuk konseptual selalu harus di implementasikan melalui suatu cara representasi tertentu.
Sebuah bentuk konseptual bisa direpresentasikan melalui *lebih dari 1 cara*. Pemilihan representasi merupakan salah satu dari sarana untuk mencapai performansi yang lebih baik.

CONTOH ILUSTRATIF ke 1 :

MASALAH :

- sebuah algoritma bertujuan menerima sekumpulan data integer dari keyboard kemudian melakukan "sorting" terhadap data tersebut diatas.
- suatu alasan tertentu menginginkan bahwa setelah proses selesai data tersebut diatas tidak kehilangan aspek "urutan kedatangan", atau dalam hal ini urutan entrynya dari keyboard.

SOLUSI KONSEPTUAL :
algoritma harus menciptakan 2 macam struktur :

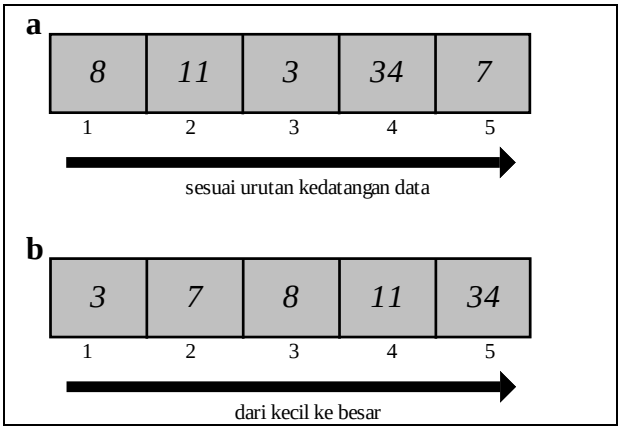
1. deretan terurut sesuai kedatangan
2. deretan terurut sesuai besar nilai integer

Keduanya merupakan struktur *LINIER (GARIS)*.

REPRESENTASI :
Ada 2 cara representasi untuk solusi konseptual tersebut.

CARA-1

melalui 2 buah array, misalnya a dan b.
Array a berisi deretan nilai integer, dengan *urutan sesuai kedatangan data* mulai dari a[1] yang paling awal, a[2], a[3] ...dst.
Array b berisi deretan nilai integer yang sama akan tetapi *terurut dari kecil ke besar*, dari b[1] yang paling kecil, b[2], b[3] .... dst.
Dalam gambar misalnya saja :

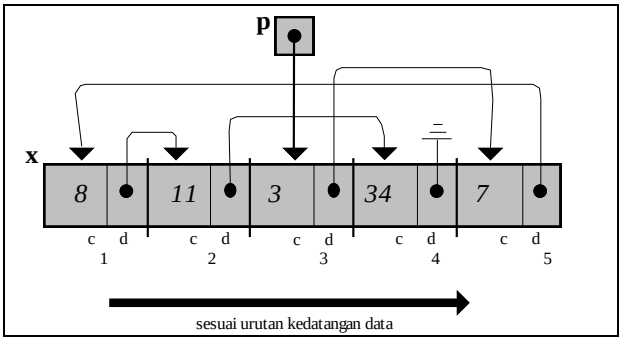


CARA-2

didefinisikan suatu tipe data kombinasi disebut *ttt*, yang terdiri dari 2 bagian :

bagian ke 1 berjenis integer disebut *c*
bagian ke 2 berjenis pointer disebut *d*

Array *x* adalah *array* jenis *ttt*, sedangkan *p* adalah *variabel pointer*, berisi nilai sedemikian rupa sehingga tercipta gambaran sbb :



Perhatikan bahwa :
- urutan *x[1].c , x[2].c* .... dst, mencerminkan urutan sesuai kedatangan.
- pointer *p* menunjuk elemen yang terkecil, dan
  pointer *x[ ].d* menunjuk ke elemen *berikut* pada urutan dari *kecil ke besar*.

Pengamatan terhadap kedua cara representasi tersebut memperlihatkan bahwa :

- cara-1 mempergunakan cara representasi dengan alokasi sequential. Urutan pada kedua deretan diwakili oleh "*posisi fisik*" pada medan alokasi.
- cara-2 mempergunakan kombinasi 2 macam cara representasi. Struktur linier yang pertama (struktur linier sesuai urutan kedatangan data) direpresentasikan melalui alokasi sequential, dimana urutan diwakili oleh posisi. Sedangkan struktur linier yang kedua (struktur linier sesuai nilai dari kecil ke besar) direpresentasikan secara *"non-sequential"*, dimana urutan diwakili oleh arah pointer.
- cara-1 lebih *"simple"*, akan tetapi ada duplikasi data.
- cara-2 lebih *"efisien"*, karena tidak ada duplikasi data.

CONTOH ILUSTRATIF ke 2 :

MASALAH :

Perhatikanlah ketiga himpunan data/variabel sebagai berikut.

HIMPUNAN KE 1.

Ada sebuah array *x* dengan 5 buah elemen. Tipe datanya terdefinisikan terdiri dari 2 variabel : *kota* dan *nama* , keduanya berjenis string.

x[1].kota berisi data "Bandung",
x[2].kota berisi data "Jakarta" ...dst.

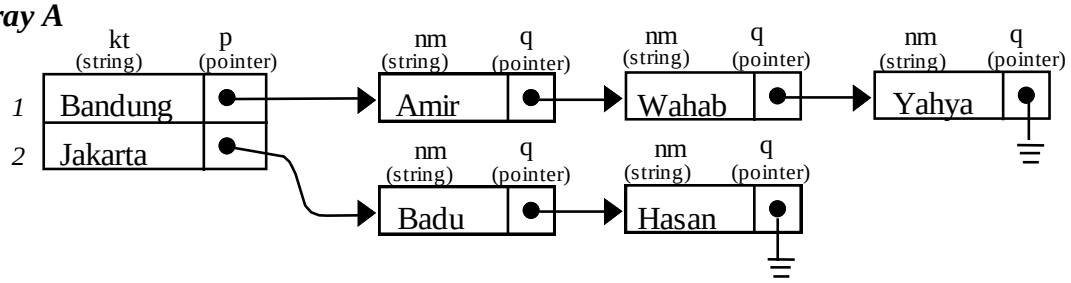
Sedangkan :
x[1].nama berisi data "Amir",
x[1].nama berisi data "Badu"...dst.

array X

	kota (string)	nama (string)
1	Bandung	Amir
2	Jakarta	Badu
3	Jakarta	Hasan
4	Bandung	Wahab
5	Bandung	Yahya

HIMPUNAN KE 2.

array A



Ada sebuah *array a* tipe datanya didefinisikan terdiri dari 2 variabel sbb : *kt* berjenis *string* dan *p* berjenis *pointer*.
Array a tersebut mempunyai 2 buah elemen, dengan isi :
a[1].kt berisi string "Bandung", sedangkan *a[2].kt* berisi "Jakarta", atau dalam notasi formal :

a[1].kt="Bandung"
a[2].kt="Jakarta"

Pointer *a[1].p* menunjuk alamat suatu data yang tipenya terdefinisikan terdiri dari dari 2 variabel : *nm* berjenis *string* dan *q* berjenis *pointer*.
Bagian *nm* dari alamat data yang ditunjuk oleh *a[1].p* tersebut berisi string "Amir", atau dalam notasi formalnya :

a[1].p↑.nm="Amir"

Sedangkan bagian *q* nya menunjuk alamat data lain yang sejenis, dengan nilai nm="Wahab" (lihat gambar).
Demikian seterusnya sehingga terbentuk semacam "deretan" data :

"*Bandung*" →"*Amir*" →"*Wahab*" →"*Yahya*" (lihat gambar !)

Sama halnya, mulai dari *a[2].p* dstnya, terbentuk "deretan" data :

"*Jakarta*" →"*Badu*" →"*Hasan*" (lihat gambar !)

HIMPUNAN KE 3.

	array u (string)		***array v*** (integer)		***array w*** (string)
1	Amir	*1*	**1**	*1*	Bandung
2	Badu	*2*	**2**	*2*	Jakarta
3	Hasan	*3*	**2**		
4	Wahab	*4*	**1**		
5	Yahya	*5*	**1**		

Array string u mempunyai 5 buah elemen, array integer v mempunyai 5 buah elemen, dan array string w mempunyai 2 buah elemen.
Variabel u[1] berisi "Amir", v[1] bernilai 1, dan w[1] berisi "Bandung". Secara formal :

u[1]="Amir"
v[1]=1
w[1]="Bandung"

Perhatikan bahwa : nilai *v[1]* = nomor elemen *w* yang mempunyai hubungan dengan *u[1]* !
Dengan kata lain : *u[1]* punya hubungan dengan *w[v[1]]* !
Maka bila *i* adalah nilai integer tertentu : *u[i]* punya hubungan dengan *w[v[i]]*.

Pengamatan yang seksama terhadap ke 3 himpunan tesebut diatas akan menghasilkan kesimpulan bahwa ketiganya mempunyai kandungan data/informasi yang tepat sama. Dengan kata lain ketiganya merupakan cara representasi yang berbeda terhadap data/informasi yang sama.
Informasi tersebut adalah bahwa : Amir tinggal di Bandung, Badu tinggal di Jakarta, Hasan tinggal di Jakarta, Wahab tinggal di Bandung dan Yahya tinggal di Bandung !
Akan tetapi ketiganya *belum tentu* merupakan cara representasi yang berlainan dari struktur data yang sama !

2.3. STRUKTUR DATA ABSTRAK,

Struktur data abstrak (*Abstract Data Type/ADT*) adalah struktur data dengan definisi fungsional dan representasi lojik tertentu, yang implementasi fisiknya disesuaikan dengan persoalannya. Struktur data abstrak ini diperlukan untuk mempermudah merancang struktur data dari suatu persoalan dalam terminologi yang lebih mudah dimengerti oleh manusia, namun sulit direpresentasikan langsung oleh mesin riil.

Contoh struktur data abstrak yang “standard” di dalam bidang informatika :

- List Linier
- Multi Linked List
- Antrian (Queue)
- Tumpukan (stack)
- Pohon (tree)
- Graph

2.4. KESIMPULAN :

Ada 2 tahapan pemilihan representasi :

Tahap 1 : data/informasi dunia nyata $\Rightarrow$ struktur data konseptual

Tahap 2 : struktur data "konseptual" $\Rightarrow$ struktur data "detail"

Keduanya dianggap penting dalam bahasan mengenai struktur data ! Akan tetapi sebagai langkah awal untuk mempelajari struktur data perhatian lebih dipusatkan pada karakteristik struktur data "detail".