BAB 2

ARRAY & RECORD

Array adalah suatu himpunan hingga elemen, terurut dan homogen.

ARRAY DIMENSI SATU

Vektor adalah bentuk yang sederhana dari array, yang merupakan array dimensi satu.

Array N dapat kita bayangkan :

N(1)	N(2)	N(3)	...	N(I)

Subskrip atau index dari suatu elemen menunjukkan posisi/urutan elemen dalam array.

BENTUK UMUM

Misal : Array N dengan tipe data T dan subskrip bergerak dari L sampai U, maka array N dapat ditulis : N(L:U)

Banyaknya elemen adalah : U - L + 1

DEKLARASI ARRAY DALAM BAHASA PEMROGRAMAN

Misalkan : Hasil pencatatan temperatur suhu ruangan dalam 1 hari (24 jam)

1	2	3					24
28	30	29				...	30

Disimpan dalam array TEMP sebagai berikut : TEMP (1:24)

Deklarasi :

PASCAL

```
var TEMP : Array[1..24] of integer;
```

BASIC

```
DIM TEMP(24)
```

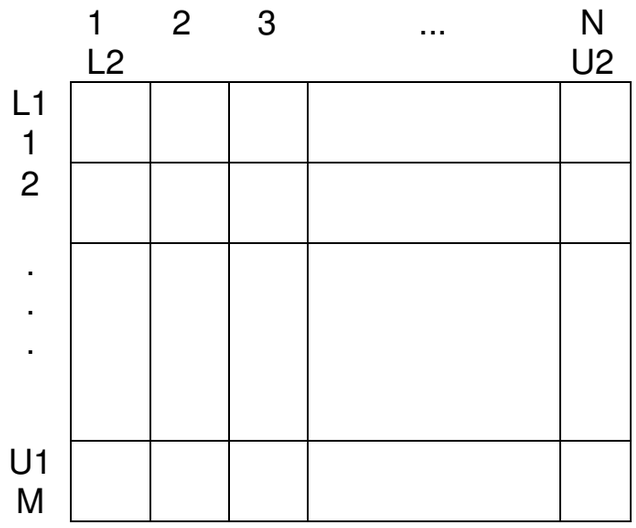
COBOL

```
01  TABEL-TEMP
    02  TEMP OCCURS 24 TIMES PIC 99.
```

ARRAY DIMENSI DUA

Adalah suatu array yang setiap elemennya merupakan tipe data array pula.

Jika array B terdiri dari M elemen, yang setiap elemennya terdiri dari suatu array dengan N elemen, maka array B dapat digambarkan sebagai berikut :



Memiliki 2 index (baris dan kolom).
Dalam hal ini kita perlu memberi 2 harga subskrip untuk mengidentifikasikan masing-masing elemen pada array dimensi dua, yaitu :

- Subskrip **pertama** menunjukkan **baris** dari array,
- Sedangkan subskrip **kedua** menunjukkan **kolom** dari array.

BENTUK UMUM

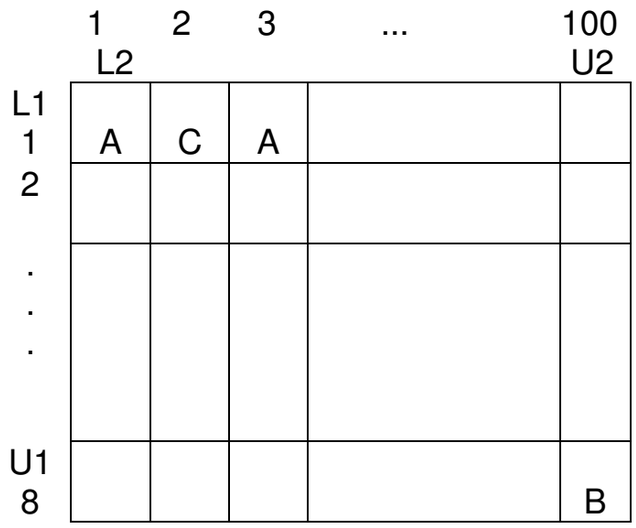
Misal :
Array B dengan tipe data T, subskrip baris dari L1 sampai U1, subskrip kolom dari L2 sampai U2, ditulis sebagai berikut :

```
B(L1:U1,L2:U2)
```

Banyaknya elemen adalah : (U1 - L1 +1) * (U2 - L2 +1)

DEKLARASI ARRAY DALAM BAHASA PEMROGRAMAN

Misal : Menyajikan nilai ujian dari 100 mahasiswa tingkat 2 sebanyak 8 mata kuliah.



Disimpan dalam array NILAI sebagai berikut :

NILAI(1:8, 1:100)

Deklarasi :

PASCAL

```
Var NILAI : array[1..8, 1..100] of char;
```

BASIC

```
Dim NILAI(8,100)
```

COBOL

```
01. TABEL-NILAI
    02. BRS OCCURS 8 TIMES
        03. KOLOM OCCURS 100 TIMES PIC X.
```

PEMETAAN KE STORAGE : ARRAY

DIMENSI SATU

Alamat awal dari memori yang dialokasikan bagi array.
Alamat awal dari array dinyatakan dengan B (Base Location), dan setiap elemen dari array menduduki S byte.
Alamat awal dari array dengan elemen ke-I adalah :

$$B + (I - L) * S$$

Contoh :

A(2:6)

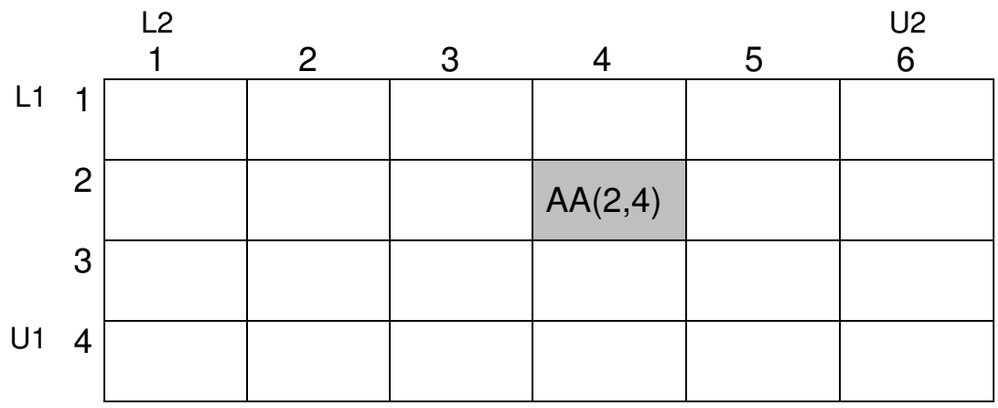
2	3	4	5	6
25	20	10	15	12

Alamat awal dari A(4) $\rightarrow$ I = 4
B + (I – L) * S
B + (4 – 2) * S = B + 2 * S

DIMENSI DUA

Memori komputer adalah linier.

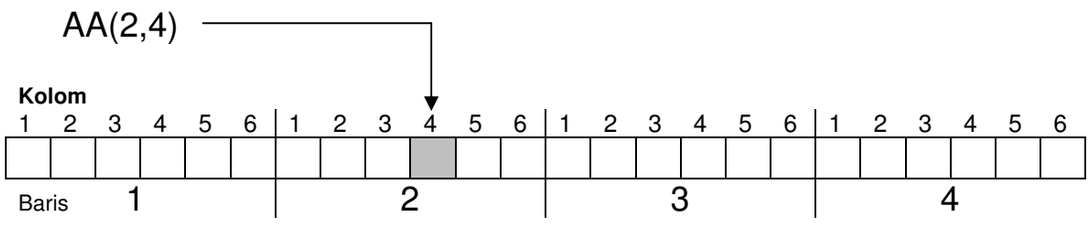
Contoh : AA(1:4, 1:6)



AA(2,4) ⟶ I = 2, J = 4

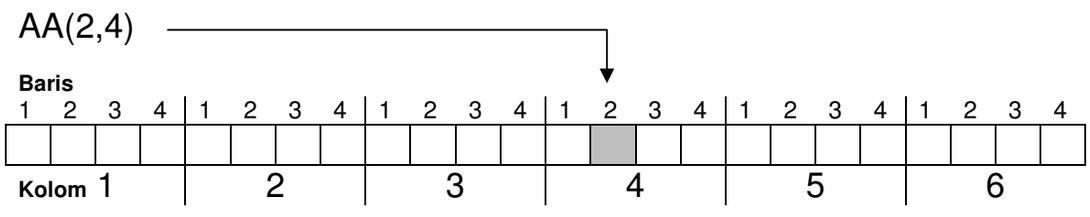
Pelinieran array dimensi banyak dengan cara :

1. Secara Baris (ROW-MAJOR ORDER)



B + (I - L1) * (U2 - L2 + 1) * S + (J - L2) * S

2. Secara Kolom (COLUMN-MAJOR ORDER)



B + (J - L2) * (U1 - L1 + 1) * S + (I - L1) * S

ARRAY DIMENSI TIGA

Adalah suatu array yang setiap elemennya merupakan tipe data array juga yang merupakan array dimensi dua.

Contoh :
Penyajian data mengenai banyaknya mahasiswa dari 20 perguruan tinggi di Jakarta, berdasarkan tingkat (1 sampai 5), dan jenis kelamin (pria atau wanita). Misalkan array tersebut dinamakan MHS. Ambil subskrip pertama, tingkat = 1, 2, ..., 5; subskrip kedua, jenis kelamin (pria = 1, wanita = 2), dan subskrip ketiga, perguruan tinggi adalah K = 1, 2, ..., 20. Jadi MHS(4,2,17) menyatakan jumlah mahasiswa tingkat 4, wanita, dari perguruan tinggi 17.

CROSS SECTION (Penampang Array Berdimensi-2)

Adalah pengambilan salah satu subskrip.
Misal : Baris = tetap/konstan
Kolom = berubah-ubah (*)

Contoh : B(*,4) = semua elemen pada kolom ke-4.
B(2,*) = semua elemen pada baris ke-2.

Pengertian cross-section pada array dimensi banyak, adalah sama seperti pada array dimensi dua.

Misal :
MHS(4,*,17) = jumlah mahasiswa tingkat 4 dari perguruan tinggi 17 (masing-masing untuk pria dan wanita).
MHS(*,*, 3) = jumlah mahasiswa untuk masing-masing tingkat, pria dan wanita, dari perguruan tinggi 3.

TRANSPOSE dari array dimensi-2.

Adalah penulisan baris menjadi kolom atau kolom menjadi baris.
Notasi : Array B (I,J), transpose dari array B adalah B^T = (J,I)

Contoh :

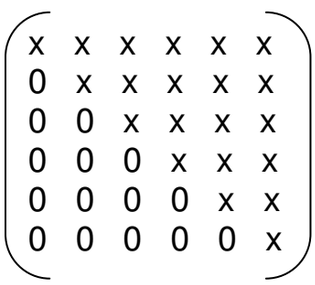
$$A = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{pmatrix} \qquad A^T = \begin{pmatrix} 0 & 3 \\ 1 & 4 \\ 2 & 5 \end{pmatrix}$$

$A(1,1) = A^T(1,1)$ $A(2,1) = A^T(1,2)$
$A(1,2) = A^T(2,1)$ $A(2,2) = A^T(2,2)$
$A(1,3) = A^T(3,1)$ $A(2,3) = A^T(3,2)$

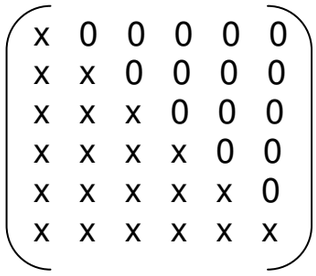
TRIANGULAR ARRAY (ARRAY SEGITIGA)

Triangular array dapat berupa :
1. Upper Triangular
   semua elemen di bawah diagonal utama = 0
2. Lower Triangular
   semua elemen di atas diagonal utama = 0

Array berukuran 6 x 6



Upper Triangular



Lower Triangular

Jumlah baris (N) besar, elemen 0 tidak perlu disimpan dalam memori.

Pendekatan :
1. Pelinieran array
2. Menyimpan bagian/elemen $\neq 0$.

Untuk lower triangular, jumlah maksimum elemen $\neq 0$ pada baris ke-I = I

Total elemen $\neq$ o tidak lebih dari : (Untuk upper dan lower triangular)

$$\sum_{I=1}^{N} I = 1/2\, N(N+1)$$

Upper Triangular T disimpan secara baris dalam array dim-1, S.

T(1,1) → S(1)	T(2,2) → S(n+1)
T(1,2) → S(2)	
.	.
.	.
.	.
T(1,n) → S(n)	T(n,n) → S(1/2n (n+1))

Program dengan lebih dari 1 array triangular kita dapat menyimpan 2 array sekaligus.

Contoh 1 :

Array A (upper) : N x N
B (lower) : (N-1) x (N-1)

→ C : N x N
C(I,J) = A(I,J) untuk I ≤ J
C(I+1, J) = B(I,J) untuk I ≥ J

Misal :
Array A (upper) : 3 x 3
B (lower) : 2 x 2

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{pmatrix} \qquad B = \begin{pmatrix} 7 & 0 \\ 8 & 9 \end{pmatrix}$$

maka akan disimpan di C : 3 x 3

$$\longrightarrow \quad C = \begin{pmatrix} 1 & 2 & 3 \\ 7 & 4 & 5 \\ 8 & 9 & 6 \end{pmatrix}$$

Contoh 2 :

Array A (upper) : N x N
B (lower) : N x N

Disimpan bersama dalam array C : N x (N + 1)
C(I, J+1) = A(I,J) untuk I ≤ J
C(I, J) = B(I,J) untuk I ≥ J

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{pmatrix} \quad B = \begin{pmatrix} 7 & 0 & 0 \\ 8 & 9 & 0 \\ 11 & 12 & 13 \end{pmatrix}$$

$$\longrightarrow \quad C = \begin{pmatrix} 7 & 1 & 2 & 3 \\ 8 & 9 & 4 & 5 \\ 11 & 12 & 13 & 6 \end{pmatrix}$$

Misalkan sekarang ada 2 array, sama-sama Upper Triangular, yakni array A dan B. Kita dapat menyimpan bersama-sama dengan melakukan transpose terhadap salah satu array tersebut, misal A menjadi A^T. (Lower Triangular).

SPARSE ARRAY

Suatu array yang sangat banyak elemen nol-nya dikenal sebagai sparse array.

Contoh :

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	1	0	0	2	0	0
2	0	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	4	0	0	0	0	0	0
6	0	0	0	0	0	0	0	2	0	0
7	0	0	0	0	0	0	0	0	0	0
8	2	1	0	0	0	0	0	0	0	0

Keuntungan :

Menyimpan elemen $\neq 0$ saja, disimpan sebagai TRIPEL, dengan bentuk : (sub baris, sub kolom, nilai elemen)
TRIPEL tersebut disimpan sebagai vektor.
Array disimpan sebagai TRIPEL

	Baris	**Kolom**	**Nilai**
V(1)	1	5	1
V(2)	1	8	2
V(3)	2	2	1
V(4)	3	1	1
V(5)	5	4	4
V(6)	6	8	2
V(7)	8	1	2
V(8)	8	2	1

Kekurangan :

Bila dilakukan up-dating, elemen = 0 $\rightarrow \neq 0$ atau $\neq 0 \rightarrow = 0$, menimbulkan kesulitan yaitu urutan vektor harus diperbaiki.

Misal :
1. V(2) diubah menjadi 0, urutan V(3) - V(8) $\rightarrow$ V(2) - V(7)
2. elemen dengan subskrip (4,6) menjadi 7
   V(5) $\rightarrow$ (4,6,7), urutan bergeser V(5) - V(8) $\rightarrow$ V(6) - V(9).

RECORD adalah suatu kumpulan elemen **hingga**, **terurut** dan **heterogen** sebagai suatu unit.

Elemen dari suatu record disebut **field**.
Field adalah suatu area dari record yang menggunakan suatu informasi tertentu.

Contoh 1:

NIP	NAMA	JABATAN	GAJI
85001	ARINI	ANALIS	700000

Di sini :

Field	Tipe data
NIP	String
NAMA	String
JABATAN	String
GAJI	Integer

DEKLARASI

PASCAL

```
Var recpeg : RECORD
             NIP : string[5];
             NAMA : string[20];
             JABATAN : string[10];
             GAJI : integer
             END;
```

COBOL

```
01  PEGAWAI.
    02   NIP        PIC X(5).
    02   NAMA       PIC X(20).
    02   JABATAN    PIC X(10).
    02   GAJI       PIC 9(7).
```

Contoh 2 :

NIP	NAMA		JABATAN	ALAMAT	GAJI
	KECIL	KELUARGA			
85001	ARINI	ISKANDAR	ANALIS	JL.JAMBU 29	700000

Di sini :

Field	Tipe data
NIP	String
NAMA	Record
JABATAN	String
ALAMAT	String
GAJI	Integer

Kecil

Keluarga

DEKLARASI

PASCAL

```
Var  recpeg : RECORD
                  NIP : string[5];
                  NAMA : RECORD
                             KECIL : string[10]
                             KEL    : string[15]
                             END;
                  JABATAN : string[10];
                  ALAMAT   : string[20];
                  GAJI        : integer
                  END;
```

COBOL

```
01  PEGAWAI.
    02  NIP          PIC X(5).
    02  NAMA.
        03  KECIL   PIC X(10).
        03  KEL     PIC X(15).
    02  JABATAN    PIC X(10).
    02  ALAMAT     PIC X(20).
    02  GAJI         PIC 9(7).
```

Contoh 3 :

NIP	NAMA	ABSEN HADIR					
		1	2	3	4	...	12
85001	ARINI	19	20	24	23		24

Di sini :

Field	Tipe data
NIP	String
NAMA	String
ABSEN	Array - integer

DEKLARASI

PASCAL

```
Var  recpeg : RECORD
                  NIP      : string[5];
                  NAMA   : string[20];
                  ABSEN : Array[1..12] of integer
                  END;
```

COBOL

```
01  PEGAWAI.
    02  NIP         PIC X(5).
    02  NAMA      PIC X(10).
    02  ABSEN OCCURS 12 TIMES
          03 JUMLAH     PIC 99.
```