Array

(TIB11 – Struktur Data)

Pertemuan 5, 6

Sub-CPMK

- Mahasiswa mampu menggunakan array untuk menyimpan dan mengakses data (C3, A3)

Materi:

1. *Array* dan Dimensi
2. *Index Array*
3. *Record* Dengan *Array*

1. *Array* dan Dimensi

Array n Dimensi

1.1. Pengertian *Array*

- *Array* atau Larik adalah sejumlah data secara berurutan
- Mempunyai susunan elemen yang sama
- Setiap *array* dapat diakses menggunakan indeks yang menyatakan urutan penempatan data pada *array*
- Secara umum, *Array* adalah sekumpulan item-item data yang homogen yang dapat dipilih menggunakan indeks pada saat program dijalankan
- *Array* secara sederhana dibentuk dari tipe data primitif, sehingga membentuk sederetan data dengan tipe data yang sama
- *Array* dapat dibentuk dari struktur /*record*

1.2. Tipe Data *Array*

- Tipe data *array* adalah jenis data yang mewakili kumpulan elemen (nilai atau variabel), masing-masing dipilih oleh satu atau beberapa indeks (kunci identifikasi) yang dapat dihitung pada run time selama eksekusi program. Koleksi seperti ini biasanya disebut variabel *array*, nilai *array*, atau *array* sederhana.
- Dengan analogi dengan konsep matematis vektor dan matriks, tipe *array* dengan satu dan dua indeks sering disebut tipe vektor dan tipe matriks.

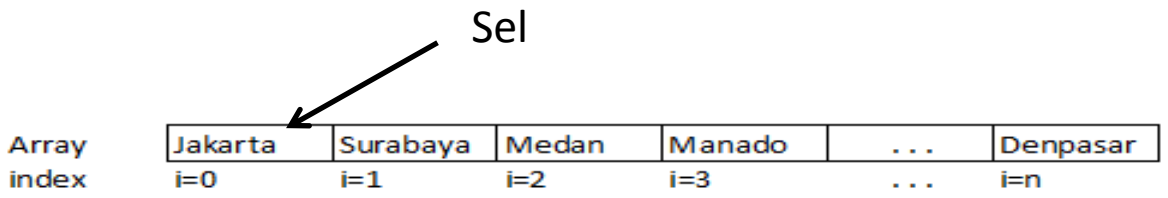
1.3. Struktur Data *Array*

- Struktur data *array*, atau hanya *array*, adalah struktur data yang terdiri dari kumpulan elemen (nilai atau variabel), masing-masing diidentifikasi oleh setidaknya satu indeks *array* atau kunci. Sebuah *array* disimpan sehingga posisi masing-masing elemen dapat dihitung dari tupel indeksnya dengan rumus matematika.
- Jenis struktur data yang paling sederhana adalah *array* linier, disebut juga *array* satu dimensi.

1.4. Dimensi *Array*

- *Array* dapat tersusun dalam 1 dimensi, 2 dimensi, 3 dimensi bahkan lebih
- *Array* dengan 1 dimensi biasa digunakan untuk menyatakan himpunan atau sejumlah *record*
- *Array* dengan 2 dimensi biasa digunakan untuk menyatakan sekumpulan himpunan matriks atau tabel
- *Array* dengan 3 dimensi biasa digunakan untuk menyatakan sekumpulan matriks atau tabel

1.4.1. *Array* 1 Dimensi



- *Apapun cara anda menggambarkan, Array 1 dimensi memiliki struktur yang sama, sekumpulan sel dengan indeksnya*
- *Array* dengan 1 dimensi biasa digunakan untuk menyatakan himpunan atau sejumlah *record*
- *Setiap sel / elemen dari array dialamati oleh nomor indeks*

1.4.2. *Array* 2 Dimensi

- Array 2 dimensi akan memiliki struktur yang sama pada masing-masing selnya, baik sel-sel yang tersusun secara horisontal, maupun sel-sel yang tersusun secara vertikal

	j=0	j=1	j=2	j=3	. . .	j=n
i=0	Jakarta	Surabaya	Medan	Manado	. . .	Denpasar
i=1	New York	Manhattan	California	Kentucky	. . .	Washington
i=2	Tokyo	Osaka	Kyoto	Hiroshima	. . .	Nagasaki
i=3	Bangkok	Pattaya	Chiangmai	Mukdahan	. . .	Krbai
. . .	. . .	. . .	. . .	. . .	. . .	. . .
i=n	Beijing	Shanghai	Guangzhou	Shenzhen	. . .	Chengdu

1.5. Implementasi Tabel

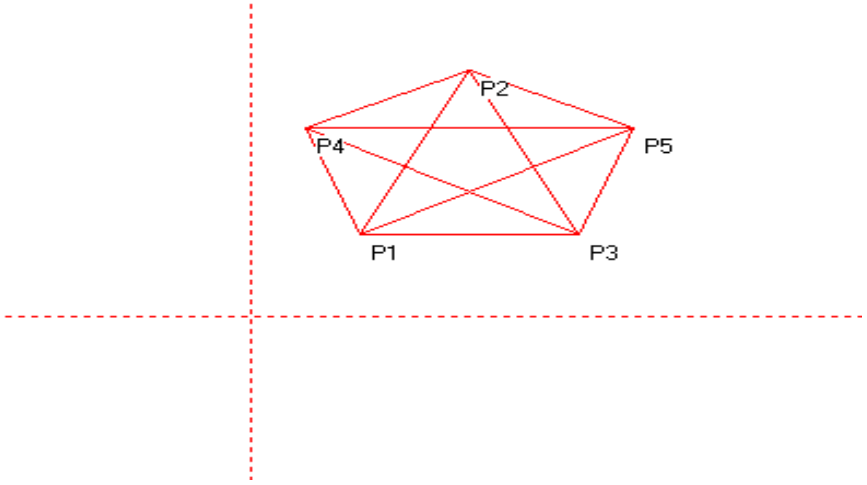
- Sebuah tabel dapat diterapkan dengan *struct* yang diimplementasikan dengan *array* 1D
- Tabel dengan *struct* yang dideklarasikan pada *array* 1D menjadikan *fields* sebagai kolom dan *index array* sebagai baris

1.6. Implementasi Matriks

- Matriks dapat diterapkan dengan menggunakan *array* 2 dimensi
- Implementasi matriks dengan *struct* yang dideklarasikan pada *array* 1 dimensi untuk menyatakan dapat juga dilakukan dengan *fields* merupakan *array* juga untuk menyatakan kolom

1.7. Contoh Implementasi Tabel dan Matriks

- Berdasarkan matriks arah, maka garis-garis yang terhubung dari tiap titik yang diwakili indeks baris ke titik yag diwakili indek kolom dapat ditentukan
- Pemberian tanda checked pada contoh menandai adanya garis yang terhubung



Input Koordinat

P1	50	50
P2	100	150
P3	150	50
P4	25	115
P5	175	115

Input Arah

	P1	P2	P3	P4	P5
P1	☐	☑	☐	☑	☐
P2	☐	☐	☑	☐	☑
P3	☑	☐	☐	☑	☐
P4	☐	☑	☐	☐	☑
P5	☑	☐	☑	☐	☐

1.7. Contoh Implementasi Tabel dan Matriks (Lanj.)

Pada Contoh ditampilkan sebuah grafik dengan koordinat-koordinat dan arah vektornya.

- P1 dengan koordinat(50,50)
- P2 dengan koordinat(100,150)
- P3 dengan koordinat(150,50)
- P4 dengan koordinat(25,115)
- P5 dengan koordinat(175,115)

1.7. Contoh Implementasi Tabel dan Matriks (Lanj.)

- Jika digambarkan sebagai tabel maka tabel vektor adalah sbb

Input Koordinat

P1	50	50
P2	100	150
P3	150	50
P4	25	115
P5	175	115

Tabel Vektor

Vektor	X	Y
P1	50	50
P2	100	150
P3	150	50
P4	25	115
P5	175	115

1.7. Contoh Implementasi Tabel dan Matriks (Lanj.)

Dengan menggunakan *Record* yang dideklarasikan dengan *array*, maka tabel vektor dapat digambarkan dengan cara

- Label Vektor dapat diwakili dengan Indeks *array*
- Kolom X diwakili dengan *fields* X pada tiap *record*
- Kolom Y diwakili dengan *fields* Y pada tiap *record*

Implementasi Dengan Record dan Array

Indeks	X	Y
1	50	50
2	100	150
3	150	50
4	25	115
5	175	115

1.7. Contoh Implementasi Tabel dan Matriks (Lanj.)

Dengan menggunakan *array* 2 Dimensi maka tabel vektor dapat digambarkan dengan cara:

- Label Vektor dapat diwakili dengan Indeks array dimensi pertama
- Kolom X diwakili *array* dimensi ke 2 indeks ke 1
- Kolom Y diwakili *array* dimensi ke 2 indeks ke 2

Implementasi dengan Arrad 2D

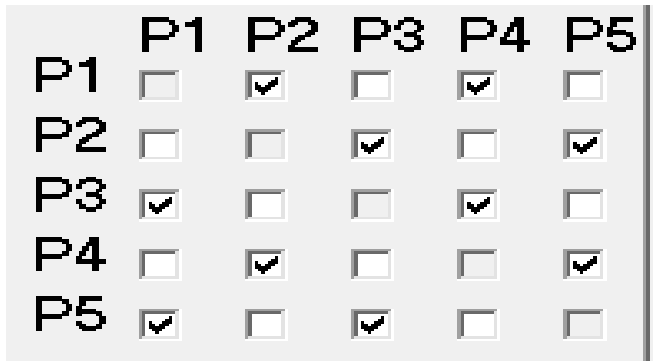
Indeks	1	2
1	50	50
2	100	150
3	150	50
4	25	115
5	175	115

diasumsikan sebagai kolom Field X

diasumsikan sebagai kolom Field Y

1.7. Contoh Implementasi Tabel dan Matriks (Lanj.)

- Arah vektor diimplementasikan dengan matriks arah (ini akan dibahas pada materi 8 mengenai *Graph*), dimana baris merupakan asal vektor dan kolom merupakan tujuan.



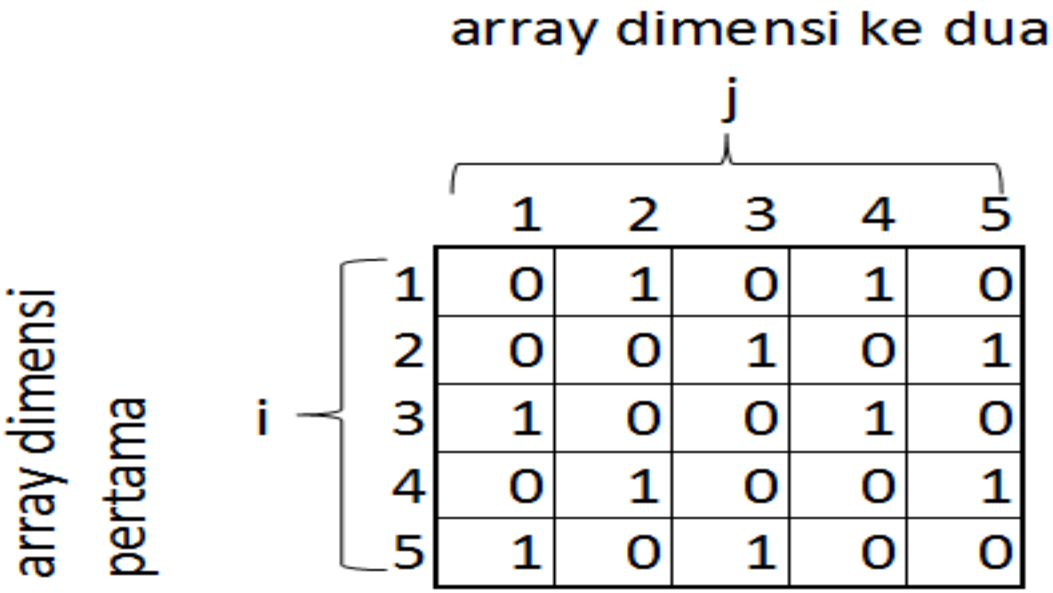
	P1	P2	P3	P4	P5
P1	0	1	0	1	0
P2	0	0	1	0	1
P3	1	0	0	1	0
P4	0	1	0	0	1
P5	1	0	1	0	0

1.7. Contoh Implementasi Tabel dan Matriks (Lanj.)

Dengan menggunakan *array* 2 Dimensi maka matriks arah dapat digambarkan dengan cara:

- *Array* dimensi pertama digunakan sebagai indeks baris
- *Array* dimensi kedua sebagai indeks kolom
- Dapat juga diterapkan dengan cara sebaliknya.

Implementasi dengan array 2 dimensi

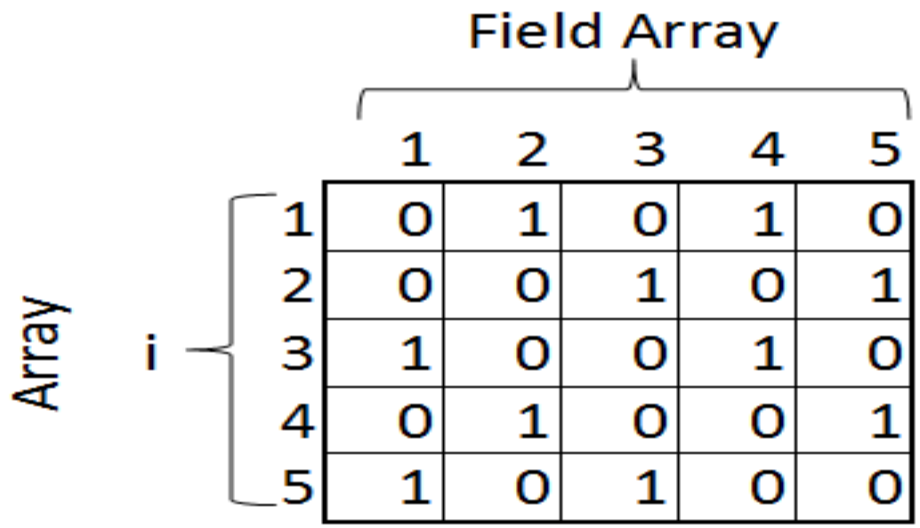


1.7. Contoh Implementasi Tabel dan Matriks (Lanj.)

Dengan menggunakan *record* yang berisi *field array* yang dideklarasikan dengan *array* 1 Dimensi maka matriks arah dapat digambarkan dengan cara:

- *Array* sebagai baris
- *Field* berupa *array* sebagai kolom pada tiap barisnya

Struct berisi *field array* yang dideklarasikan sebagai *array* 1 dimensi

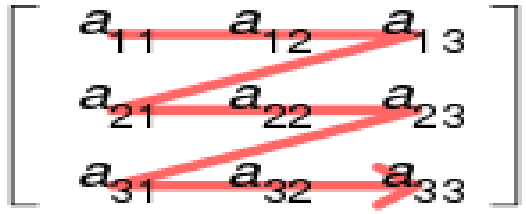


1.8. Contoh *Array* 2 D dalam C

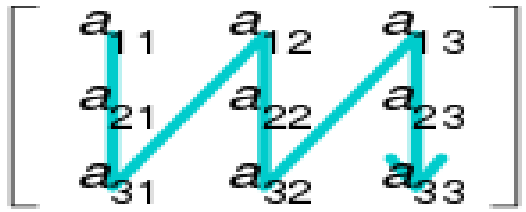
```
#include <stdio.h>

int main(void)
{
  int matrix[2][3] = {{1,2,3},{7,8,9}};

  printf("Isi array matrix: \n");
  printf("%d %d %d \n", matrix[0][0], matrix[0][1], matrix[0][2]);
  printf("%d %d %d \n", matrix[1][0], matrix[1][1], matrix[1][2]);

  return 0;
}
```

1.9. *Row- and Column-major Order*

Row-major order

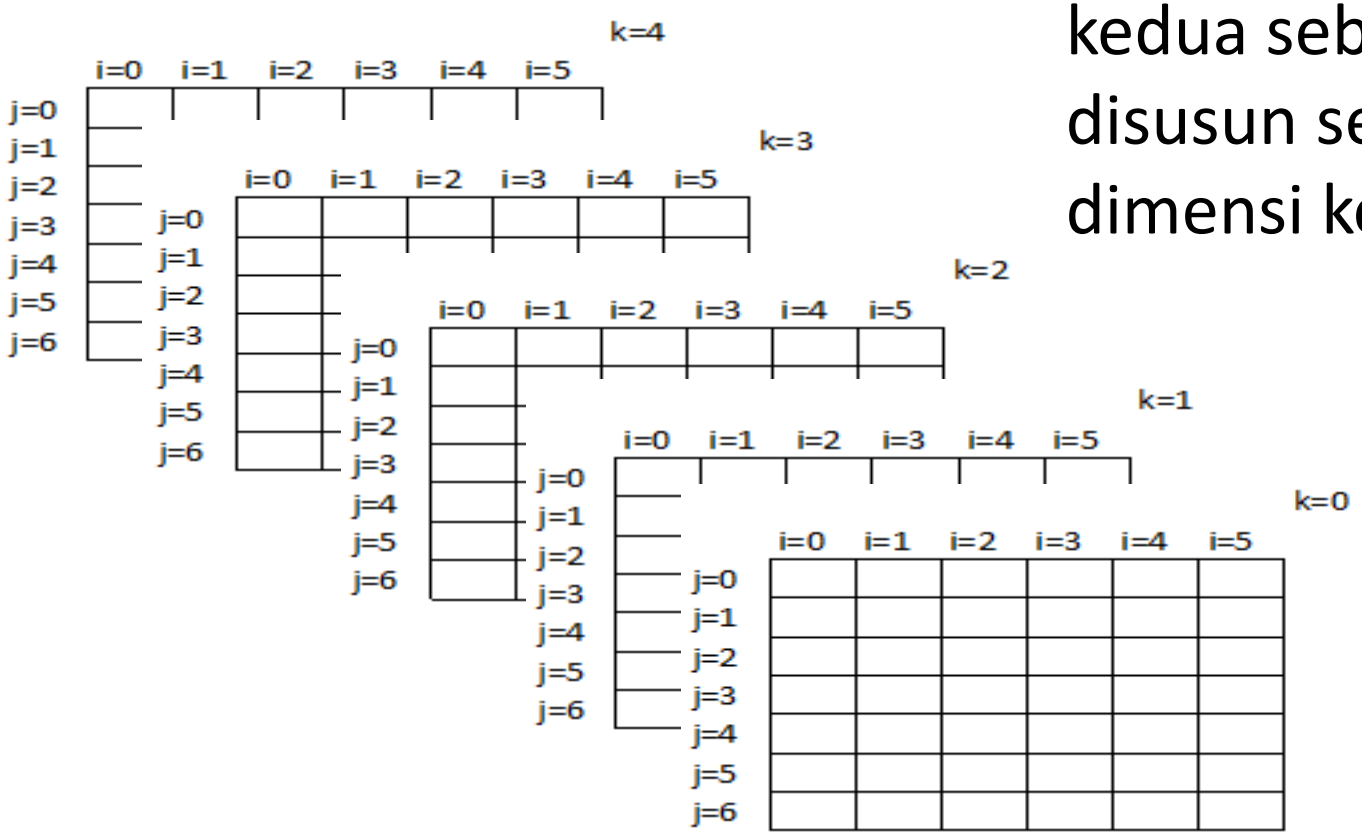


Column-major order



- Row Major Oder merupakan susunan pengaksesan sel-sel array 2 dimensi yang diakses berdasarkan sel-sel pada tiap baris, setelah semua sel pada suatu baris selesai diakses, baru pengaksesan dilanjutkan pada baris di bawahnya
- Column Major Oder merupakan susunan pengaksesan sel-sel array 2 dimensi yang diakses berdasarkan sel-sel pada tiap kolom, setelah semua sel pada suatu kolom selesai diakses, baru pengaksesan dilanjutkan pada kolom berikutnya

1.10. *Array* 3 Dimensi



- Array Dimensi pertama dan kedua sebagai matriks yang disusun secara berurutan pada dimensi ketiga

1.11. *Array* 1D (Pascal)

Declaration

```
namaArray : array [awalIndex..akhirIndex] of tipevariabel;
```

```
Var
  table1d : array[1..10] of integer;
```

Assignment → `table1d[8] := 1000;`

`//mengisi array ke 8 dengan 1000`

Accessing → `temp := table1d[8];`

`//mengambil array ke 8 dan menyimpan pada variabel temp`

1.12. *Array* 2D (Pascal)

Declaration

namaArray : *array* [awalIndexDimensi1..akhirIndexDimensi1, awalIndexDimensi2..akhirIndexDimensi2] of tipevariabel;

//contoh array 2D berukuran 5 x 10

```
Var
  table2d : array[1..5, 1..10] of byte;
```

Assignment → `table2d[1,3] := 88;`

`//mengisi array baris 1 kolom 3 dengan 88`

Accessing → `temp := table2d[1,3];`

`//mengambil array baris 1 kolom 3 dan menyimpan pada variabel temp`

1.13. *Array* 3D (Pascal)

Declaration

namaArray : *array* [awalIndexDimensi1..akhirIndexDimensi1, awalIndexDimensi2..akhirIndexDimensi2, awalIndexDimensi3..akhirIndexDimensi3] of tipevariabel;

//contoh array 3D berkuran 5 x 4 x 6

```
Var
      table3d : array [1..5, 1..4, 1..6] of byte;
```

Assignment → `table3d[1,4,3] := 100;`

`//mengisi array baris 1 kolom 4 tabel 3 dengan 100`

Accessing → `temp := table3d[1,4,3];`

`//mengambil array baris 1 kolom 4 tabel 3 dan menyimpan pada variabel temp`

1.14. *Array* 1D (C)

Declaration

```
tipeVariabel namaArrayname[ukuranArray];
//deklarasi array berukuran 10 dengan indeks 0..9
//Catatan: pada C index array dimulai dari 0
int table1d[10];
```

Assignment

```
table1d[7] = 1000; //mengisi array ke 7 dengan 1000
```

Accessing

```
temp = table1d[7]; //mengambil data dari array ke 7 dan menyimpan pada
variabel temp
```

1.15. *Array* 2D (C)

Declaration

```
tipeVar namaArrayname[ukuranArraydimensi1][ukuranArraydimensi2];
//deklarasi array berukuran 10 x 20 bertipe data int
int table2d[10][20];
```

Assignment

```
//mengisi array baris 1 kolom 4 dengan 88
table3d[1][4] = 88;
```

Accessing

```
//mengambil data array pada baris 1 kolom 4 dan menyimpan pada variabel temp
   temp = table2d[1][4];
```

1.16. *Array* 3D (C)

Declaration

```
tipeVar namaArray[ukuranArrayDim1][ukuranArrayDim2] [ukuranArrayDim3];
```

```
//deklarasi array berukuran 10 x 20 x 5 bertipe int
   int table3d[10][20][5];
```

Assignment

```
//mengisi baris 1 kolom 4 tabel 5 dengan 100
   table3d[1][4][5] = 100;
```

Accessing

```
//mengambil data array pada baris 1 kolom 4 tabel 5
//dan menyimpan pada variabel temp
   temp = table3d[1][4][5]; //Accessing
```

2. Index *Array*

- Untuk menunjuk suatu sel pada *array* dilakukan dengan memberikan nilai pada indeks dari *array* yang dituju
- Indeks *array* diperlukan untuk mengakses alamat sel
  - Misal untuk mengakses sel pada indeks ke 9
  - Maka kita cukup mencantumkan varArray[9]
- Indeks *array* dapat diganti dengan variabel bertipe kardinal
  - Misal untuk mengakses sel pada indeks ke 9, maka kita dapat menyimpan data 9 tersebut ke variabel n
  - Kemudian kita cukup mencantumkan varArray[n]

1.1. Tipe Data Untuk Indeks *Array*

- Tipe data untuk *index array* diperlukan suatu variabel bertipe *cardinal* seperti int, longInt, smallint dan char

3. *Record* Dengan *Array*

3.1. *Record With Array* - PASCAL

- Record definition

```
Type
   RecordName = Record
     Var1Name : vartype;
     Var2Name : vartype;
     VarnName : vartype;
   End;
```

- Array declaration

```
DataCell : array[1..250] of RecordName;
```

- Assignment

```
DataCell[ArrayNum].VarName := value;
```

- Accessing

```
DataCell[ArrayNum].VarName
```

- Example

```
//record definition
Type
TheCell=Record
   Name : string;
   Age : Integer;
End;
//Declaration
Var
     DataMhs : array[1..250] of TheCell;

Begin
   //Assignment
   DataMhs[1].Name := ”Doraemon”;
   DataMhs[1].Age := 19;
   //Accessing
   writeln(DataMhs[1].Name);
   writeln(DataMhs[1].Age);
End.
```

3.2. *Record With Array - C*

- Record definition

```
struct StructName
{
    vartype Var1Name;
    vartype Var2Name;
    vartype VarNName;
};
```

- Array declaration

```
struct StructName DataCell[ArraySize];
```

- Assignment

```
DataCell[ArrayNum].VarName = value;
```

- Accessing

```
DataCell[ArrayNum].VarName
```

- Example

```
//Struct definition
struct TheCell
{
    char Name[10];
    int Age;
};
//Declaration
struct TheCell DataMhs[250];

void main()
{
    //Assignment
    strcpy(DataMhs[1].Name,"Doraemon");
    DataMhs[1].Age = 19;
    //Accessing
    printf("%s", DataMhs[1].Name);
    printf("%d", DataMhs[1].Age);
}
```

Ringkasan

- *Array* atau Larik adalah sejumlah data secara berurutan yang mempunyai susunan elemen yang sama
- *Array* dapat diakses menggunakan indeks yang menyatakan urutan penempatan data pada *array*
- *Array* dapat dibentuk dari struktur / *record*
- *Array* dapat tersusun dalam 1 dimensi, 2 dimensi, 3 dimensi bahkan lebih
- *Array* dengan 1 dimensi biasa digunakan untuk menyatakan himpunan atau sejumlah *record*
- *Array* dengan 2 dimensi biasa digunakan untuk menyatakan sekumpulan himpunan matriks atau tabel
- *Array* dengan 3 dimensi biasa digunakan untuk menyatakan sekumpulan matriks atau tabel

Contoh Array dan mengaksesnya

```
#include <stdio.h>
#include <conio.h>
//Contoh array dengan record
//-----------------------------------------------
struct TheCell
{
  char nama[10];
  int umur;
};
struct TheCell DataMhs[100];

/* program utama */
void main()
{
  int n, i;
  do{
    //clrscr();
    printf("Jumlah Data : ");scanf("%d", &n);
  }while (n>10);
  printf("\nEntry Serial Data\n\n");
```

```
 for(i=0;i<=n-1;i++)
  {
    printf("Data ke %d: \n",i);
    printf("  Nama : ");scanf("%s",&DataMhs[i].nama);
    printf("  Umur : ",i);scanf("%d", &DataMhs[i].umur);
  }
 printf("\n");
 printf("Teman-teman ku : \n");  for(i=0;i<=n-1;i++)
 {
   printf("%s (%d tahun) \n",DataMhs[i].nama,DataMhs[i].umur);
 }
 //printf("\n");}
```

Silahkan ganti printf dengan cout dan scanf dengna cin jika versi C anda tidak mensupportnya

Terimakasih

TUHAN Memberkati Anda

Teady Matius Surya Mulyana (tmulyana@bundamulia.ac.id)