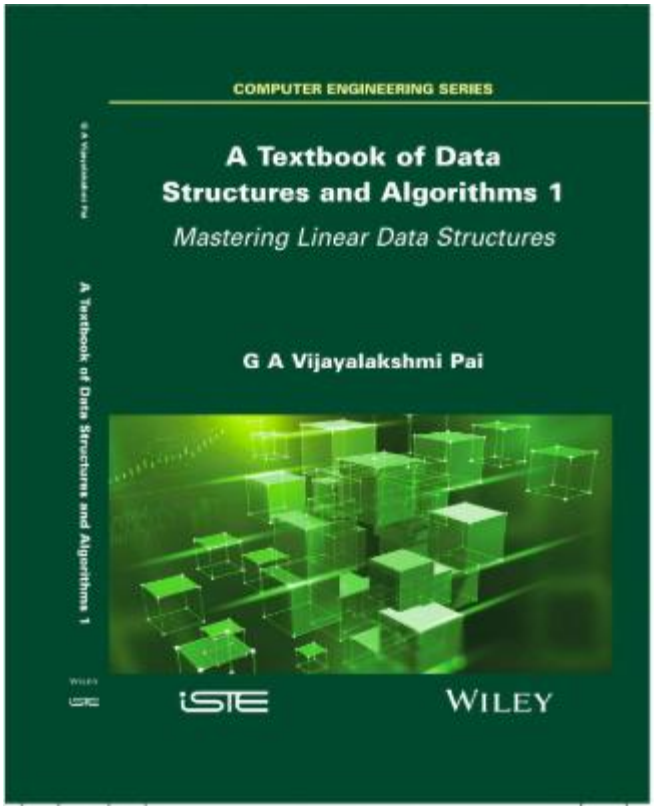
TIB29 – Struktur Data dan Algoritma

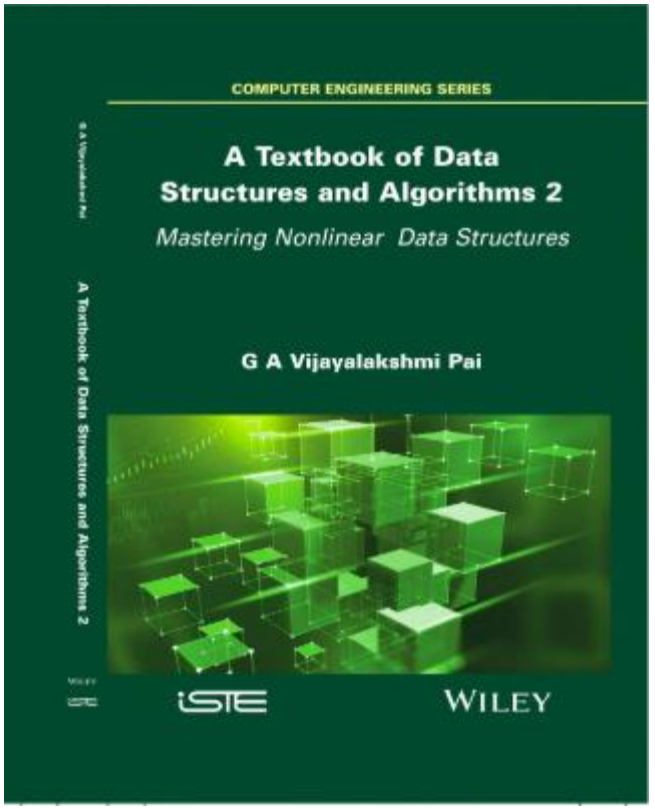
PERINGATAN HAK CIPTA

PERINGATAN HAK CIPTA

Array

Diadopsi Dari Sumber:





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

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.

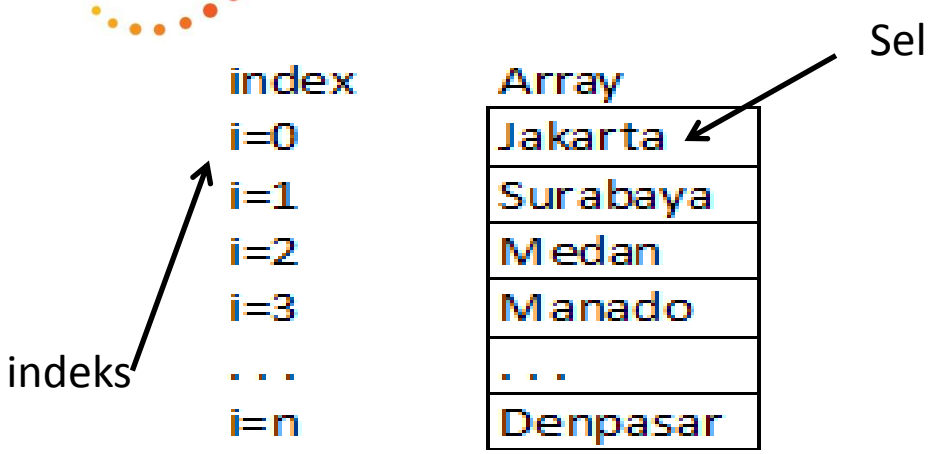
1.3 Struktur Data *Array*

- 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.

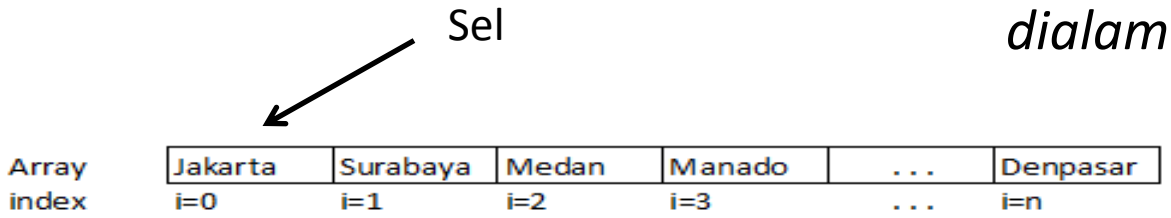
1.4 Dimensi *Array*

- *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*



1.4.1 *Array* 1 Dimensi (Lanj.)

Declaration

```
table1d=[] #phyton
int table1d[10]; //C
var table1d : array[1..10] of integer //pascal
//deklarasi array berukuran 10 dengan indeks 0..9
//Catatan: pada C index array dimulai dari 0
```

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
```

1.4.2 *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

1.4.2.1 Array 2 Dimensi Phyton

- Array 2D dimensi pada Phyton tidak memerlukan deklarasi.
- Array 2 Dimensi Pada Phyton disusun berdasarkan Row Mayor.
- Indeks dimensi pertama merupakan baris, indeks dimensi kedua merupakan kolom.
- Gambar disamping merupakan contoh berbagai cara mengakses elemen-elemen pada array 2 Dimensi pada Phyton.

```
m = [[1,2,3],[4,5,6],[7,8,9]]

print(m)
print(m[0])
print(m[1])
print(m[2])
print(m[0][0],m[0][1],m[0][2])
print(m[1][0],m[1][1],m[1][2])
print(m[2][0],m[2][1],m[2][2])

for i in m:
  print(i)
for i in m:
  for j in i:
    print(j)
for i in range(0,3):
  for j in range(0,3):
    if j<2:
      print(m[i][j], end=' ')
    else:
      print(m[i][j])
```

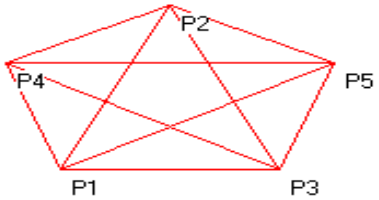
```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
1 2 3
4 5 6
7 8 9
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9
```

1.4.2.2 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.

1.4.2.3 Contoh Implementasi 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	☑	☐	☑	☐	☐

1.4.2.3 Contoh Implementasi 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)

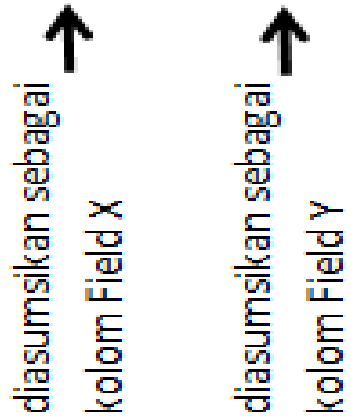
1.4.2.3 Contoh Implementasi Matriks (Lanj.)

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.

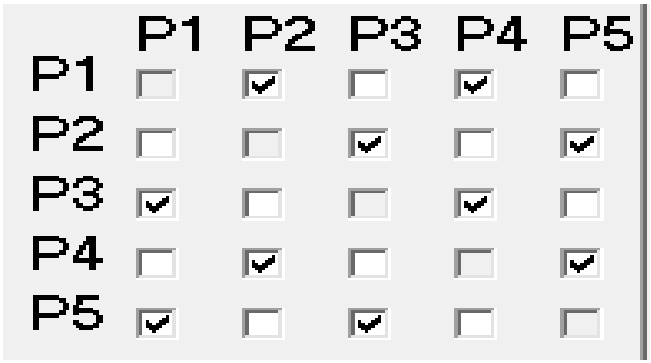
Implementasi dengan Arrad 2D

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



1.4.2.3 Contoh Implementasi 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.



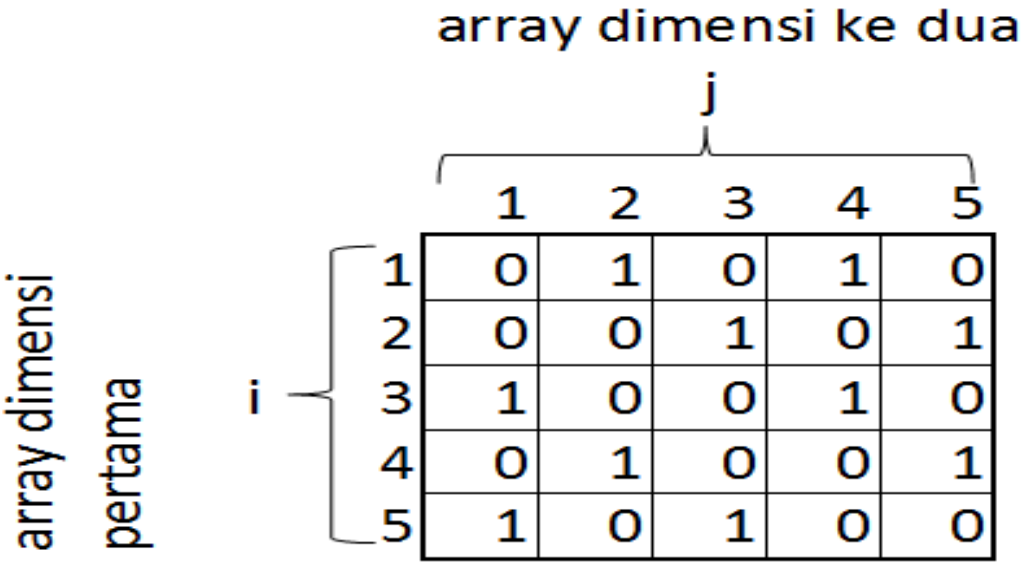
	P1	P2	P3	P4	P5
P1	0	1	0	1	0
P2	0	0	1	0	1
P3	1	0	0	1	0
P4	0	1	0	0	1
P5	1	0	1	0	0

1.4.2.3 Contoh Implementasi 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

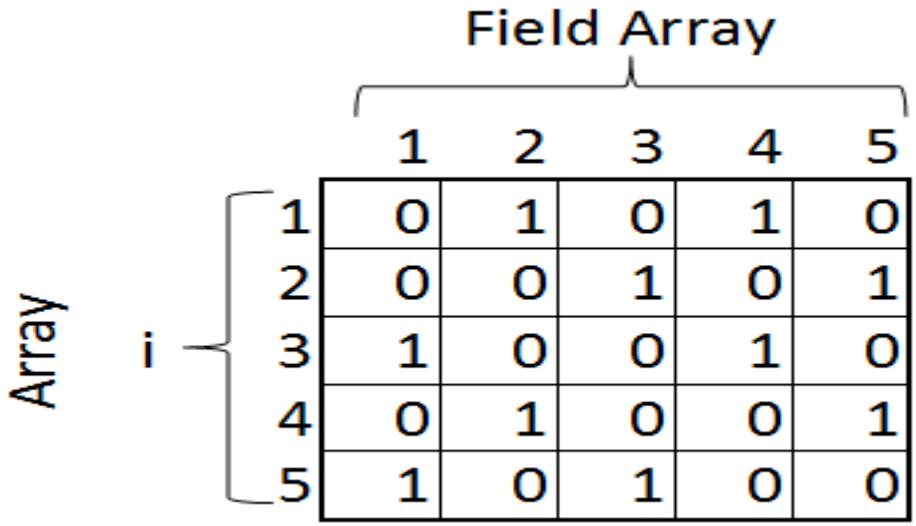


1.4.2.3 Contoh Implementasi Matriks (Lanj.)

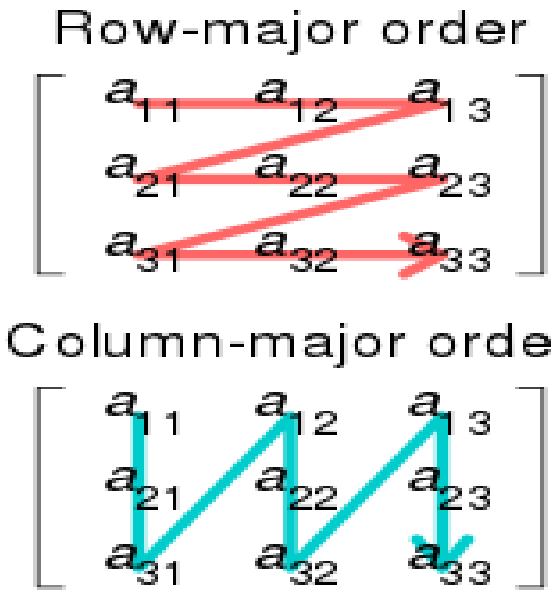
Dengan menggunakan *record* yang berisi *field array* yang dideklarasikan dengan *array* 1 Dimensi maka matriks arah dapat digambarkan dengan cara:

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

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



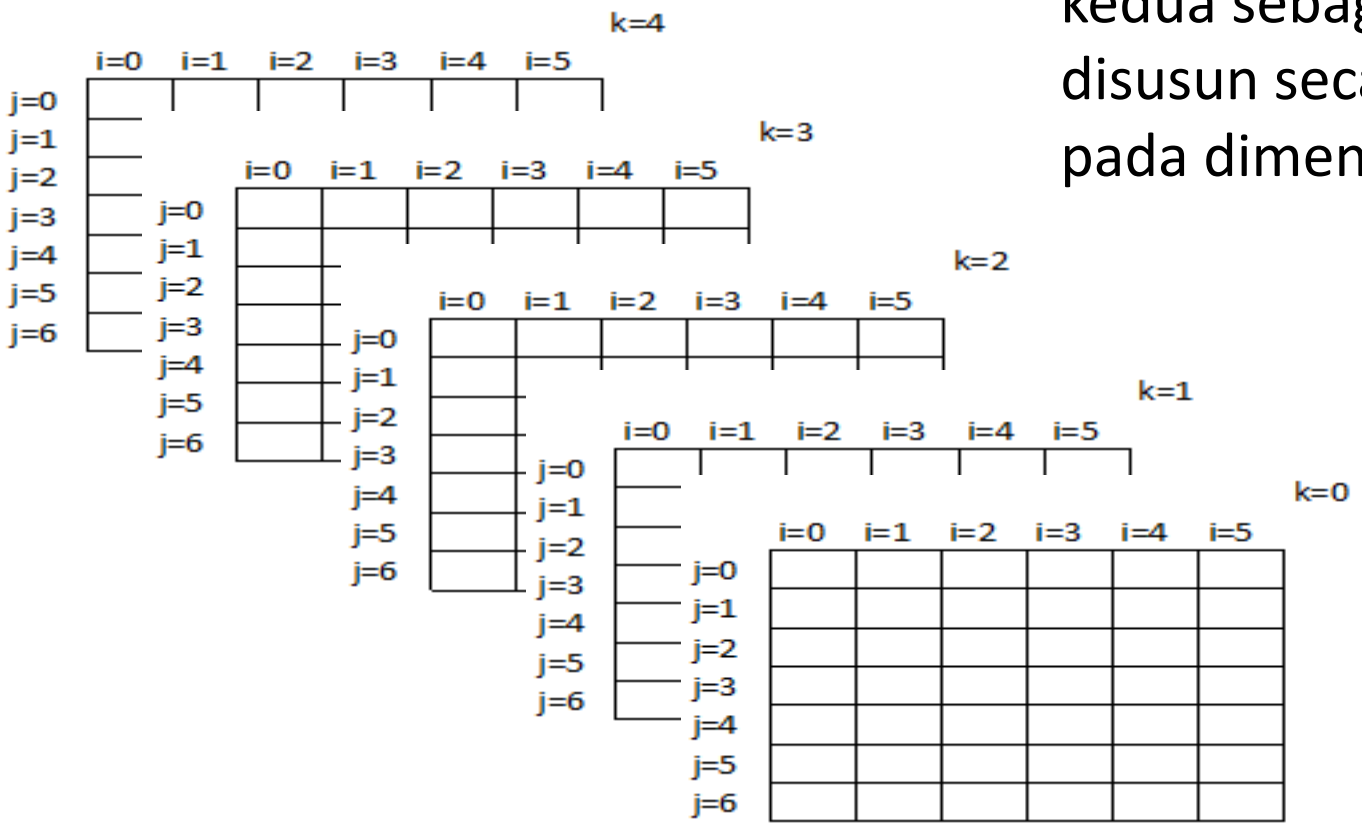
1.4.2.4 *Row- and Column-major Order*



- 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 dibawahnya.
- 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.

1.5 *Array* 3 Dimensi

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



1.6 Deklarasi Array

- Untuk mendeklarasikan Array pada pemrograman konvensional dapat dilakukan dengan syntax:

```
tipedata namarray[ukuranarray];
```

- Phyton tidak memerlukan ukuran array, karena array pada phyton bersifat dinamis
- Karena phyton tidak melakukan deklarasi variable, maka deklarasi array kosong pada phyton dapat diwakili dengan syntax

```
namaArray = [] #dengan kurung siku kosong saja
```

Contoh:

```
pool=[] #phyton tidak menentukan ukuran array
```

1.6 Deklarasi Array (Lanj.)

- Contoh mendeklarasikan array integer dengan nama pool dengan ukuran 10

```
pool=[] #phyton tidak menentukan
ukuran array
int pool[10];//C++
var //pascal
  pool:array[1..10] of integer;
```

1.7 Deklarasi Beberapa Array Dengan Tipe Data Yang Sama

(catatan: untuk Bahasa pemrograman generasi 3)

- Untuk mendeklarasikan beberapa Array sekaligus dapat dilakukan dengan cara sbb:

```
tipedata namarray_1[ukuranarray_1],
namarray_2[ukuranarray_2], …,
namarray_n[ukuranarray_n];
```

- Contoh

  Mendeklarasikan tiga array integer dengan nama pool, temp dan dat dengan ukuran 10

```
int pool[10], temp[10], dat[10];
```

1.8 Deklarasi Array Multi Dimensi

- Pada Phyton, Deklarasi Array multi dimensi dapat dilakukan dengan cara yang sama seperti deklarasi array 1 dimensi.
- Nantinya penambahan data dilakukan per dimensi keduanya
- Contoh

```
matriks = []
matriks.append([1,0,0])
matriks.append([0,1,0])
matriks.append([0,0,1])
```

1.8 Deklarasi Array Multi Dimensi (Lanj.)

- Pada Bahasa pemrograman generasi 3, Deklarasikan Array multi dimensi dapat dilakukan hanya dengan menambahkan [ukuranarray] pada dimensi berikutnya dapat dilakukan dengan cara sbb:

```
tipedata
namarray[ukuranarray_D1][ukuranarray_D2]…[ukuranar
ray_Dn];
```

- Contoh

  Mendeklarasikan array integer 2 dimensi dengan nama matriks dengan ukuran 10x5

```
int matriks[10][5]; //C++
```

1.9 Manipulasi Array

Beberapa perintah untuk manipulasi array:

- Menghapus elemen ke n dari array:

  namaArray.pop(n)

- Menambahkan elemen ke array pada offset terakhir:

  namaArray.append(data)

- Menambahkan elemen pada posisi tertentu

  namaArray.insert(posisi, data)

- Mengganti data elemen array pada indeks ke i:

  namaArray[i] = databaru

1.10 Contoh 1 (menampilkan array/list)

- Gambar disamping contoh array dan mengaksesnya
- 9 data di inisialisasikan pada array m
- len(m) : menghitung Panjang array
- m.pop(n) menghapus array pada indeks ke n

```
m = [1,2,3,4,5,6,7,8,9]

print(m)
print(m[0])
print(m[1])
print(m[2])
print("Panjang array mula-mula:",len(m))
for i in m:
  print(i)
for i in range(0,9):
  print(m[i], end=' ')
print()
m.append(10) #menambahkan data ke list/array
for i in range(0,10):
  print(m[i], end=' ')
print("\nPanjang array sekarang:",len(m))
m.pop(9)
for i in range(0,len(m)):
  print(m[i], end=' ')
print("\nPanjang array sekarang:",len(m))
m.pop(5)
for i in range(0,len(m)):
  print(m[i], end=' ')
print("\nPanjang array sekarang:",len(m))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
1
2
3
Panjang array mula-mula: 9
1
2
3
4
5
6
7
8
9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
Panjang array sekarang: 10
1 2 3 4 5 6 7 8 9
Panjang array sekarang: 9
1 2 3 4 5 7 8 9
Panjang array sekarang: 8
```

1.11 Contoh 2 (modifikasi array/list)

```
m = [1,2,3,4,5,6,7,8,9]
print("\n",m)
m.append(10)
print("\n",m)
m.insert(5, 100)
print("\n",m)
m[3] = 999
print("\n",m)
m.sort()
print("\n",m)
m.reverse()
print("\n",m)
m.pop(5)
print("\n",m)
m.remove(3)
print("\n",m)
n = [101,102,103,104,105]
m.extend(n)
print("\n\n",m)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[1, 2, 3, 4, 5, 100, 6, 7, 8, 9, 10]

[1, 2, 3, 999, 5, 100, 6, 7, 8, 9, 10]

[1, 2, 3, 5, 6, 7, 8, 9, 10, 100, 999]

[999, 100, 10, 9, 8, 7, 6, 5, 3, 2, 1]

[999, 100, 10, 9, 8, 6, 5, 3, 2, 1]

[999, 100, 10, 9, 8, 6, 5, 2, 1]


[999, 100, 10, 9, 8, 6, 5, 2, 1, 101, 102, 103, 104, 105]
```

1.12 Perintah-perintah List Phyton

- append(): Menambahkan pada akhir list.
- clear(): menghapus semua element dari list.
- copy(): mengambil seluruh isi array dan mengcopykan ke variabel lain.
- count(): menghitung elemen array dengan nilai tertentu.
- extend(): menambahkan elemen-elemen dari suatu list ke current list pada akhir list.
- index(): mengambil index dari elemen pertama dari suatu nilai tertentu.
- insert(): menyisipkan elemen pada posisi tertentu.
- pop(): menghapus elemen pada posisi tertentu.
- remove(): menghapus elemen dengan nilai tertentu.
- reverse(): membalikkan urutan list/array.
- sort(): mengurutkan list/array.

2. Index *Array*

2.1 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]

2.2. Tipe Data Untuk Indeks *Array*

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

2.3 Array Statis vs Array Dinamis

Array terbagi menjadi dua macam:

- Array Statis, array dengan ukuran yang sudah di deklarasikan, ukuran tidak dapat diubah-ubah, pada Bahasa pemrograman generasi 3, umumnya menggunakan array statis, karena variable dan array harus dideklarasikan sebelum digunakan.
- Array Dinamis: Array dengan ukuran yang dapat berubah-ubah, pada Bahasa pemrograman generas 4, umumnya variable tidak perlu di deklarasikan, sehingga Array umumnya berupa array dinamis.

3. *Record* Dengan *Array*

3.1 Pengertian *Record / Structure*

- Rekaman atau *record* atau *structure* adalah sekumpulan data yang disusun dari tipe data yang sama atau tipe data yang berbeda.
- Sebuah record berisi beberapa variabel lain yang ‘dipaketkan’. Konsep struktur data seperti ini sedikit mirip dengan konsep class dan object dalam *object oriented programming.*
- *Record*/*Structure* harus di definisikan terlebih dahulu.
- Hasil definisi *Record*/*Structure* diperlakukan seperti tipe data.
- Ketika akan digunakan, *Record*/*Structure* harus dideklarasikan dahulu pada sebuah variabel.

3.2 Record Dengan Array

- Record dapat dideklarasikan menjadi sebuah array.
- Dengan adanya record dalam suatu Array, maka record Array dapat diperlakukan sebagai suatu tabel.

3.3 *Record With Array* – Phyton

- Record definition

```
#Define Class
class beverage:
  def __init__(self, nama,
  harga):
    self.nama = nama
    self.harga = harga
```

- Array declaration

```
#membuat array objek dari class
   beverage
p = [] #list objek (array
   objek)
```

- Assignment

```
p.append(beverage("Bajigur"
   , 25000))
```

- Accessing

```
print(p[0].nama,
   p[0].harga)
```

3.4 *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.
```

3.5 *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);
}
```

3.6 Contoh Implementasi Tabel

- 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

3.6 Contoh Implementasi Tabel (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

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.

PERINGATAN HAK CIPTA

PERINGATAN HAK CIPTA

Terimakasih

TUHAN Memberkati Anda

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