ARRAY

Array atau larik adalah sebuah tipe data bentukan atau terstruktur yang terdiri dari sejumlah komponen dengan tipe yang sama. Dengan array dapat menyimpan banyak data dengan satu nama variabel array. Jumlah komponen ditunjukkan dengan nilai indeks atau dimensi dari array.

A. Deklarasi dan Pemrosesan Array

Array dapat berupa array berdimensi satu, dua, tiga atau lebih.

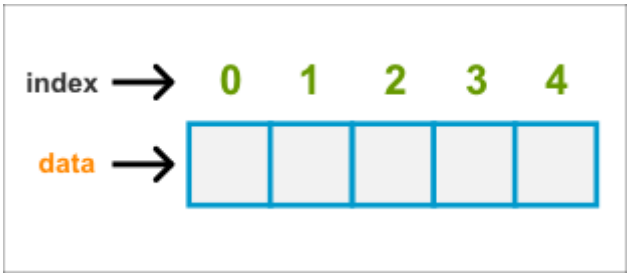
- Array berdimensi satu (one-dimensional array) mewakili bentuk suatu vektor.
- Array berdimensi dua (two-dimensional array) mewakili bentuk dari suatu matriks atau tabel
- Array berdimensi tiga (three-dimensional array) mewakili bentuk suatu ruang.

Contoh deklarasi array berdimensi satu :

var m : array[1..5] of integer;

keterangan

- M adalah nama array satu dimensi
- [1..5] adalah indeks array; yang dapat menampung maksimal 5 (lima) buah data
- indeks dimulai dari 0
- integer adalah tipe data array M



Gambar Ilustrasi Array

Dari deklarasi tersebut di atas dapat digambarkan sebuah array dengan nama M mempunyai lima elemen atau dapat menyimpan lima buah data bertipe sama yaitu integer : ***m[1], m[2], m[3], m[4], m[5].***

Dan berikut psudo-code untuk menginputkan data array: ***read(m[1]); read(m[2]); read(m[3]); read(m[4]); read(m[5])***

Sebenarnya kapan array perlu digunakan? Jawabannya, jika kita memerlukan penyimpanan sementara untuk data-data yang bertipe sama di dalam memori, untuk selanjutnya data-data tersebut dimanipulasi (dihitung atau diterapkan oleh proses lainnya)

Contoh algoritma tanpa array:

Deklarasi

x,i : integer;

{baca 6 buah nilai yang bertipe integer dan simpan di x}
for i ← 1 to 6 do
read(x)
endfor

```
 {cetak setiap nilai x}
 for i ← 1 to 6 do
 write(x)
endfor
```

Bila runtunan nilai x yang dibaca dari keyboard adalah : 20, 30, 40, 50, 60, 70 maka output dari algoritma di atas adalah: 70 70 70 70 70 70
Karena variabel x hanya dapat menampung satu buah nilai, dan nilai yang disimpan oleh x adalah selalu nilai yang terakhir yaitu 70, maka nilai 70 itulah yg akan dicetak pada setiap kali pengulangan.

Sekarang bandingkan dengan algoritma yang menggunakan array x yang berisi 6 buah data / elemen.
Algoritma dengan array:

```
Deklarasi
        x : array [1..6] of integer
        i : integer

        {baca 6 buah nilai integer simpan di x}
         for i ← 1 to 6 do
                read(x[i])
         endfor
         {cetak setiap nilai x}
         for I ← 1 to 6 do
         write(x[i])
         endfor
```

Keluaran dari algoritma di atas akan sesuai dengan data yang diinputkan yaitu: 20 30 40 50 60 70.
Selama pelaksanaan program, elemen array tetap menyimpan nilai-nilai yang dimaksud. Hal ini berguna jika kita ingin menggunakan nilai-nilai di dalam array tersebut untuk diproses lebih lanjut di bagian lain dalam algoritma berikut ini:

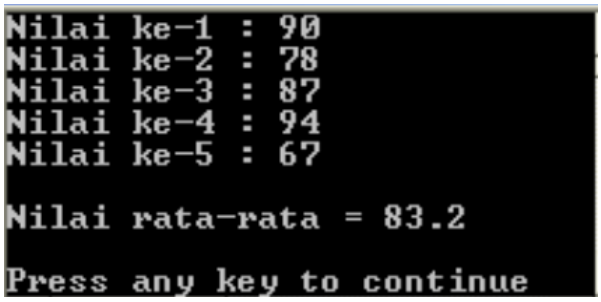
Contoh algoritma menggunakan array :

```
        x : array [1..6] of integer
        i,jumlah : integer

         {baca 6 buah nilai integer simpan di x}
         for i ← 1 to 6 do
                read(x[i])
         endfor

         {cetak setiap nilai x}
         for I ← 1 to 6 do
         write(x[i])
         endfor

        {pada bagian ini elemen array digunakan lagi, untuk menghitung nilai rerata seluruh elemen}
        Jumlah ← 0
        For I ← 1 to 6 do
        Jumlah ← jumlah + x[i]
        Endfor
        Write(jumlah/6)
```

Contoh program lengkap untuk mengakses tiap elemen array:

```
#include <stdio.h>
#define MAKS 5
main()
{
int i;
float total=0, rata, nilai[MAKS];
for(i=0; i<MAKS; i++)
{
printf("Nilai ke-%d : ", i+1);
scanf("%f", &nilai[i]);
total = total + nilai[i]; //hitung jml total nilai
}
rata = total / MAKS; //hitung nilai rata2
//cetak nilai rata-rata
printf("\nNilai rata-rata = %g\n", rata);
}
```

Output program :



Gambar 7.2. Output Program Array

Sebuah array juga dapat diinisialisasi sekaligus pada saat dideklarasikan. Untuk mendeklarasikan array, nilai-nilai yang diinisialisasikan dituliskan di antara kurung kurawal ({}) yang dipisahkan dengan koma. Berikut contoh program inisialisasi array untuk menampilkan jumlah hari dalam setiap bulannya.

```
main()
{
int bln, thn, jhari;
int jum_hari[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30,
 31, 30, 31};
printf("Masukkan bulan (1..12) : ");
scanf("%d", &bln);
printf("Masukkan tahunnya : ");
scanf("%d", &thn);
    if(bln == 2)
    if(thn % 4 == 0) //thn kabisat
    jhari = 29;
```

```
else
    jhari = 28;
else
    jhari = jum_hari[bln-1];
printf("\nJumlah hari dalam bulan %d tahun %d
adalah %d hari\n",bln,thn,jhari);}
```

B. Matriks : Array Berdimensi dua

Array dapat dikategorikan sebagai tipe data terstruktur. Elemen array dapat distrukturkan lagi menjadi matriks. Matriks pada umumnya dikenal dengan array berdimensi dua (array berindeks dua). Jika didalam array 1 dimensi hanya menggunakan sebuah tanda [] (bracket), maka pada array 2 dimensi atau matriks terdapat 2 tanda [] tersebut. Matriks dapat digambarkan seperti tabel di mana terdapat baris dan kolom.

Berikut bentuk umum deklarasi array berdimensi dua (matriks) :

tipe_data nama_array[jumlah elemen baris][jumlah elemen kolom];

Contoh mendeklarasikan sebuah matriks bertipe data integer yang di dalamnya terdapat 3 baris dan 4 kolom.

int A[3][4];

Inisialisasi nilai array 2 dimensi (matriks) tersebut dapat dibuat manual atau diinput oleh pengguna saat program dijalankan. Karena array 2 dimensi mempunyai lebih dari satu bentuk index array, maka dalam inisialisasinya perlu menggunakan tanda {} untuk membentuk baris array, contoh sebagai berikut :

int A[3][4]={{3,4,8,0},{3,9,2,1},{6,3,0,2}};

Ilustrasi inisialisasi matriks di atas dapat digambarkan lewat tabel berikut:

Tabel 7.1. Ilustrasi Matriks Beserta Nilai Tiap Elemennya

A	Kolom Ke 0	Kolom Ke 1	Kolom Ke 2	Kolom Ke 3
Baris Ke 0	A [0] [0]	A [0] [1]	A [0] [2]	A [0] [3]
Baris Ke 1	A [1] [0]	A [1] [1]	A [1] [2]	A [1] [3]
Baris Ke 2	A [2] [0]	A [2] [1]	A [2] [2]	A [2] [3]

A	Kolom Ke 0	Kolom Ke 1	Kolom Ke 2	Kolom Ke 3
Baris Ke 0	3	4	8	0
Baris Ke 1	3	9	2	1
Baris Ke 2	6	3	0	2

C. Record : Array Bertipe Terstruktur

Array dapat menyimpan sejumlah elemen bertipe terstruktur (rekaman / record). Record disusun oleh satu atau lebih field. Tiap field menyimpan data dari tipe dasar tertentu atau tipe bentukan. Record dalam Bahasa C++ dikenal dengan nama struct (lihat Kembali pembahasan tipe data bentukan pada bab IV di atas).

Contoh deklarasi record atau struct dengan nama “mahasiswa” yang mempunyai field : nama, prodi, ipk dengan tipe data berbeda-beda :

```
struct mahasiswa{
   string nama;
   string prodi;
   float ipk;};
```

Untuk memproses matriks digunakan algoritma berstruktur perulangan bersarang agar dapat membaca tiap nilai yang terkandung di dalam elemen-elemennya, per baris dan per kolomnya. Berikut algoritmanya dalam notasi *pseudo-code* :

Deklarasi
i : integer; {indeks baris}
j : integer; {indeks kolom}

Algoritma

```
for i ← 1 to bar do
  for j ← 1 o kol do
    write(M[i,j]);
endfor
endfor
```

Translasi algoritma di atas ke dalam program lengkapnya:

```
#include <iostream>
#include <conio.h>
using namespace std;
int main() {

        int M[3][4]={{3,4,8,0},{3,9,2,1},{6,3,0,2}};
                for (int i=0;b<3;i++) {
                for (int j=0;k<4;j++) {
                        cout<<M[i][j]<<” “;
                }
                cout << endl;
               }
     getch();
   }
```

Program untuk memproses record atau struct :

```
#include <iostream>
#include <string>
using namespace std;

 struct mahasiswa{
   string nama;
   string prodi;
   float ipk;
};
```

```
int main(){
    mahasiswa mhs;
    mhs.nama=”Wurijanto “;
   mhs.prodi=” Informatika”;
   mhs.ipk=3.50;
   cout<<”Nama    : “<<mhs.nama<<endl;
   cout<<”Jurusan : “<<mhs.prodi<<endl;
   cout<<”IPK     : “<<mhs.ipk<<endl;
  return 0;
}
```

Selain matriks dan record, tipe data **string** pada dasarnya merupakan array karakter dengan panjang tertentu. Karena string adalah array, maka elemen-elemennya yang berupa karakter juga dapat diakses melalui indeks.

Contoh : ‘prodi’ → s[1] = ‘p’
s[2] =’r’
s[3] = ‘o’
s[4] =’d’
s[5] = ‘i’

Contoh penerapan array karakter atau string, dapat dilihat dalam program menghitung panjang sebuah string s berikut ini. Program tersebut menggunakan fungsi yang diberi nama “panjang” :

```
int panjang(char s[])
{ int i;
  i = 0;
  while (s[i] != ‘\0’)
 {
    i = i + 1 ;
 }
  return i
}
```

Pada program tersebut di atas, selama string s diakhiri dengan karakter null ‘\0’ maka program menemukan akhir string (akhir kata yang diinputkan). Panjang string dihitung dengan membaca elemen-elemen string sampai ditemukan karakter null tersebut.

Bahasa C++ sudah menyediakan fungsi standar untuk mengembalikan (menghitung) panjang sebuah string, yaitu *strlen*. Fungsi ***strlen()*** ini didefinisikan di dalam file header *<string.h>* , contoh : *n = strlen(s);*