Bagian 2

2 Array

"Naiklah ke puncak gunung, maka akan kau lihat jalan – jalan lain menuju ke sana yang belum pernah kau ketahui sebelumnya"

-Me

Bab ini akan membahas tentang *array* atau larik. Pembahasan dimulai dari penjelasan singkat tentang *array*, cara pendeklarasian, dan cara memanipulasinya. Kemudian dilanjutkan dengan bahasan mengenai *array* multidimensi. Terakhir adalah contoh studi kasus penggunaan *array* untuk membuat program statistik sederhana yang dapat melakukan perhitungan rata – rata, pencarian nilai terbesar dan terkecil.

2.1 Pengertian

Fungsi variabel di dalam program adalah untuk menyimpan suatu nilai tertentu yang dapat diubah-ubah. Tiap variabel mempunyai nama dan tipe. Hanya data yang bertipe sama dengan tipe variabel yang dapat disimpan di dalam variabel tersebut.

Dalam beberapa literatur *array* sering diartikan larik. *Array* merupakan koleksi data dengan setiap elemen data menggunakan nama yang sama dan tiap elemen data bertipe sama. Setiap elemen *array* dapat diakses melalui *indeks*.

Berdasarkan jumlah dimensi indeks dalam sebuah variabel, *array* dibedakan menjadi array berdimensi satu dan *array* berdimensi banyak.

Terdapat beberapa hal yang perlu diperhatikan apabila akan memasukkan deretan data dalam variabel bertipe *array*:

1. Mengetahui tipe data yang digunakan dalam variabel *array*. Variabel array numerik hanya dapat menerima data numerik dan variabel array string hanya dapat menerima data karakter. String merupakan kumpulan dari karakter, biasanya merupakan kumpulan dari huruf dalam alfabet. String didefinisikan sebagai kumpulan dari tipe data *char*. Dalam bahasa C string didefinisikan sebagai sekumpulan *char* yang diakhiri dengan *null character* (*C-style string*).
2. Banyaknya data harus lebih kecil atau sama dengan besarnya ukuran *array*.
3. Struktur perulangan lazim digunakan untuk membaca nilai – nilai dalam *array*.
4. Banyaknya indeks yang digunakan menunjukkan banyaknya ruang memori yang dialokasikan. Supaya tidak terjadi pemborosan ruang memori, maka banyaknya indeks harus disesuaikan dengan banyaknya data.

Deklarasi dari suatu array akan memberitahu compiler berapa jumlah elemen yang dikandung oleh array tersebut dan tipe data dari *array* tersebut.

Penentuan ukuran dimensi suatu *array* didasarkan dari beberapa pertimbangan, diantaranya:

1. Model data dari kasus yang akan diselesaikan.
2. Model pengorganisasian data untuk kemudahan dalam pengolahan data.

2.2 Array 1 Dimensi

2.2.1 Deklarasi

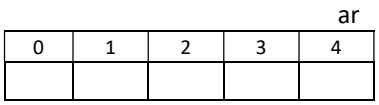
Pendeklarasian array 1 dimensi ditulis seperti pendeklarasian variabel biasa dengan diikuti tanda *square bracket* []. Ukuran array ditulis di dalam tanda *square bracket*. Berikut ini adalah bentuk umum pendeklarasiannya.

```
TipeData NamaArray[Ukuran];
```

Contoh:

```
int ar[5];
```

Deklarasi di atas berarti bahwa `ar` adalah sebuah array bertipe int dengan ukuran 5. Ilustrasi visual deklarasi di atas adalah sebagai berikut.



Gambar 2-1 Ilustrasi Array

Saat dideklarasikan `ar` akan disediakan 5 area untuk menampung elemen. Tiap – tiap elemen memiliki index dari 0 sampai 4.

Catatan:

> *Index atau subscript pada array digunakan untuk menandai tiap – tiap nilai yang tersimpan di dalamnya. Satu nilai yang tersimpan memiliki satu index yang unik. Index array selalu dimulai dari 0 sehingga untuk array dengan panjang n akan memiliki index terakhir n-1. Penjelasan lebih lanjut mengenai index ini akan dibahas pada bab berikutnya* (§3.2.1).

2.2.2 Inisialisasi

Ada berbagai cara untuk memberikan nilai awal terhadap array 1 dimensi:

1. Statis, pemberian nilai dilakukan serentak saat deklarasi. Berikut ini adalah contoh pemberian nilai secara statis:

   ```
   int ar[] = {4, 8, 9, 2, 7};
   int ar[]{4, 8, 9, 2, 7}; // atau menggunakan direct
                            // initialization list
   ```

   Ukuran array secara implisit akan disesuaikan dengan jumlah elemen yang dimasukkan. Namun, bisa juga pemberian ukuran array dilakukan secara eksplisit seperti berikut ini:

   ```
   int ar[5] = {4, 8, 9, 2, 7};
   int ar[5]{4, 8, 9, 2, 7}; // direct initialization
                             // list
   ```

   Setiap elemen akan otomatis memiliki index sesuai urutan yang dimulai dari index 0. Pendeklarasian di atas bila dilakukan satu persatu adalah seperti berikut:

   ```
   ar[0] = 4;
   ar[1] = 8;
   ar[2] = 9;
   ar[3] = 2;
   ar[4] = 7;
   ```

   Ilustrasi visual dari contoh pemberian nilai (inisialisasi) di atas adalah sebagai berikut:

ar

0	1	2	3	4
4	**8**	**9**	**2**	**7**

Gambar 2-2 Inisialisasi Array

2. Dinamis, pemberian nilai dilakukan setelah pendeklarasian yang biasanya berasal dari input pengguna. Berikut ini adalah contoh kode untuk pemberian nilai secara dinamis:

```
for(int i = 0; i < 5; ++i) {
    cout << "elemen ke-" << i << ": ";
    cin >> ar[i];
}
```

Ketika array dideklarasikan dan belum mendapatkan input maka tiap elemen akan berisi nilai yang tidak diketahui, kecuali bila array (dan variabel tunggal) dideklarasikan dalam lingkup global maka nilai *default*-nya adalah 0;

2.2.3 Pengaksesan

Sama halnya dengan pemberian nilai, setiap elemen dalam array dapat diakses dengan merujuk pada nomor *index*-nya. Berikut ini adalah contoh kode untuk menampilkan nilai setiap elemen dalam array:

```
cout << "elemen ke-0: " << ar[0] << endl;
cout << "elemen ke-1: " << ar[1] << endl;
cout << "elemen ke-2: " << ar[2] << endl;
cout << "elemen ke-3: " << ar[3] << endl;
cout << "elemen ke-4: " << ar[4] << endl;
```

Pengaksesan elemen di atas dapat juga dilakukan menggunakan struktur perulangan seperti berikut ini:

```
for(int i = 0; i < 5; ++i)
  cout << "elemen ke-" << i << ": "
       << ar[i] << endl;
```

Kedua potongan kode di atas akan menghasilkan output yang sama seperti berikut:

```
elemen ke-0: 4
elemen ke-1: 8
elemen ke-2: 9
elemen ke-3: 2
elemen ke-4: 7
```

Selain menggunakan struktur `for` tradisional pengaksesan elemen dapat pula menggunakan *range-based-for* seperti berikut ini:

```
for(auto n : ar) {
  cout << n << " ";
```

Struktur di atas dapat dibaca sebagai *"untuk semua n dalam ar, lakukan ..."*.

2.3 Array Multidimensi

Array multidimensi adalah array yang memiliki index lebih dari satu atau berordo lebih dari satu. Contoh penggunaan array multidimensi adalah matriks. Contoh lain adalah array 2 dimensi yang direpresentasikan dalam bentuk tabel seperti diilustrasikan Tabel 2-1 berikut ini:

Tabel 2-1 Representasi Array 2 Dimensi

	0	1	2
0	d	z	x
1	c	v	b
2	a	m	n
3	s	d	f

Ilustrasi tabel di atas dapat dibaca sebagai array 2 dimensi dengan ukuran 4 x 3. Dimensi pertama menyatakan baris dan dimensi kedua adalah kolom. Sebagai contoh nilai 'm' dapat dirujuk pada posisi baris ke-2 kolom ke-1.

Berdasar ilustrasi tabel di atas dapat dikatakan bahwa array multidimensi adalah array di dalam array. Tabel adalah sebuah array yang berisi baris – baris, dan di setiap baris adalah array yang berisi kolom – kolom.

2.3.1 Deklarasi

Pendeklarasian array multidimensi sama seperti pada array 1 dimensi dengan jumlah pasangan square bracket lebih dari satu yang menyatakan dimensinya. Secara umum benutk pendeklarasiannya adalah sebagai berikut:

```
TipeData NamaArray[Ukuran1]…[UkuranN]
```

Sebagai contoh berikut ini adalah pendeklarasian array 2 dimensi dengan ukuran 4 x 3:

```
char ar2d[4][3];
```

Deklarasi di atas menghasilkan array bertipe `char` dengan nama `ar2d`

2.3.2 Pemberian Nilai

Seperti halnya dalam array 1 dimensi pemberian nilai pada array multidimensi dapat dilakukan melalui 2 cara yaitu:

1. Statis, pemberian nilai dilakukan pada saat pendeklarasian. Berikut ini adalah contoh pemberian nilai pada array 2 dimensi:

```
char ar2d[4][3] = {
    {'d', 'z', 'x'},
    {'c', 'v', 'b'},
    {'a', 'm', 'n'},
    {'s', 'd', 'f'}
};
```

   Atau dapat dilakukan satu persatu seperti berikut ini:

```
ar2d[0][0] = ‘d’;
ar2d[0][1] = ‘z’;
ar2d[0][2] = ‘x’;
ar2d[1][0] = ‘c’;
ar2d[1][1] = ‘v’;
ar2d[1][2] = ‘b’;
ar2d[2][0] = ‘a’;
ar2d[2][1] = ‘m’;
ar2d[2][2] = ‘n’;
ar2d[3][0] = ‘s’;
ar2d[3][1] = ‘d’;
ar2d[3][2] = ‘f’;
```

2. Dinamis, pemberian nilai dilakukan dari input pengguna. Berikut ini adalah contoh kode untuk pemberian nilai secara dinamis:

```
for(int i = 0; i < 4; ++i)
{
   for(int j = 0; j < 3; ++j)
   {
      cout << “elemen [“ << i << “, “ << j << “]:” ;
      cin >> ar2d[i][j];
    }
}
```

Berdasar potongan kode di atas dapat diketahui bahwa perulangan luar (counter i) digunakan untuk index dimensi pertama dan perulangan dalam (counter j) digunakan untuk index dimensi kedua.

Berikut ini adalah contoh lain pemberian nilai secara dinamis pada array 3 dimensi bertipe `char` yang bernama `ar3d` dengan ukuran 4 x 3 x 2:

```
for(int i = 0; i < 4; ++i)
{
   for(int j = 0; j < 3; ++j)
   {
      for(int = 0; k < 2; ++k)
      {
         cout << “elemen [“ << i << “, “
                            << j << “, “
                            << k << “]: “;
         cin >> ar3d[i][j][k];
      }
   }
}
```

2.3.3 Pengaksesan

Elemen – elemen array multidimensi dapat diakses seperti pada array 1 dimensi yaitu dengan merujuk pada nomor – nomor *index* tiap elemen.

Pengaksesan dapat dilakukan satu demi satu atau serentak menggunakan struktur perulangan. Berikut ini adalah contoh mengakses elemen pada array 2 dimensi:

```
for(int i = 0; i < 4; ++i)
{
    for(int j = 0; j < 3; ++j)
        cout << arr2d[i][j] << “\t” ;
```

```
            cout << '\n';
        }
```

Contoh output potongan kode di atas adalah sebagai berikut:

```
d       z       x
c       v       b
a       m       n
s       d       f
```

2.4 Studi Kasus: Statistik Sederhana

Program untuk perhitungan statistik sederhana yang dapat mencari data terbesar, terkecil, dan nilai rata – rata. Ketentuan

1. Data disimpan dalam array berukuran maksimal 20
2. Input dilakukan secara dinamis oleh pengguna melalui keyboard dengan jumlah maksimal 20 data (boleh kurang)

Penyelesaian:

```
/**
 * program statistik sederhana
 * menampilkan data terbesar, terkecil, dan rata - rata
 *
**/

#include <iostream>

using namespace std;

int get_max(int ar[], int sz);
int get_min(int ar[], int sz);
double get_avg(int ar[], int sz);

int main()
{
        int data[20];
        int n;

        cout << "banyak data (max 20): ";
        cin >> n;

        for(int i = 0; i < n; ++i)
        {
                cout << "data ke-" << i << ": ";
                cin >> data[i];
        }

        cout << "max: " << get_max(data, n) << endl;
        cout << "min: " << get_min(data, n) << endl;
        cout << "avg: " << get_avg(data, n) << endl;

        return 0;
}

int get_max(int ar[], int sz)
{
        int imax{0};
```

```
	for(int i = 1; i < sz; ++i)
		if(ar[i] > ar[imax])
			imax = i;

	return ar[imax];
}

int get_min(int ar[], int sz)
{
	int imin{0};

	for(int i = 1; i < sz; ++i)
		if(ar[i] < ar[imin])
			imin = i;

	return ar[imin];
}

double get_avg(int ar[], int sz)
{
	double sum{0};

	for(int i = 0; i < sz; ++i)
		sum += ar[i];

	return sum / sz;
}
```

2.5 Latihan

1. Buatlah program untuk menghitung standar deviasi dari sejumlah data yang dimasukkan.
2. Buatlah program menggunakan array 2 dimensi sehingga menampilkan output seperti berikut ini:

```
x 0 0 0 0 0 0 0
0 x 0 0 0 0 0 0
0 0 x 0 0 0 0 0
0 0 0 x 0 0 0 0
0 0 0 0 x 0 0 0
0 0 0 0 0 x 0 0
0 0 0 0 0 0 x 0
0 0 0 0 0 0 0 x
```