Bab 8
Array

Array merupakan sekumpulan variable dengan tipe yang sama. Elemen dalam array diakses dengan menggunakan indeks-nya.

Ketika kita membutuhkan sejumlah variable dengan tipe yang sama, sulit bagi kita untuk mendeklarasikannya satu persatu :

```
int a,b,c,d,e,f,g,h,i,j;
```

Akan lebih mudah jika kita mendeklarasikannya sekaligus :

```
int angka[10];
```

Masalah lain timbul ketika kita ingin memproses variable tersebut. Misalnya kita ingin mencetak seluruh isi variable :

```
printf("%d ", a);
printf("%d ", b);
printf("%d ", c);
printf("%d ", d);
printf("%d ", e);
printf("%d ", f);
printf("%d ", g);
printf("%d ", h);
printf("%d ", i);
printf("%d ", j);
```

Dengan array, kita cukup menuliskan :

```
for(i=0; i<10; i++)
  printf("%d ", angka[i]);
```

Deklarasi Array

Cara mendeklarasikan sebuah array adalah seperti berikut :

```
<tipe variable> <nama variable>[<jumlah>]
```

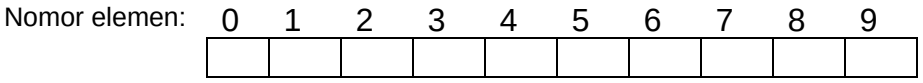
Contoh pendeklarasian array :

```
int angka[10];
float nilai_siswa[100];
char abc[5];
```

Misalnya contoh pertama, `int angka[10]`, berarti kita membuat 10 buah variable bertipe `int` yang diberi nama `angka`.

Mengakses Elemen Array

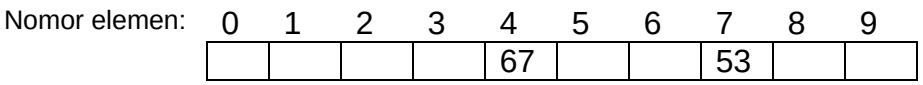
Untuk menggunakan variable-variable dalam array, kita gunakan penomoran yang disebut indeks. Pada bahasa C, penomoran array selalu dimulai dari 0 dan berakhir pada n-1.



Pada contoh kita diatas, `int angka[10],` kita sudah memiliki sebuah array dengan 10 elemen. Elemen pertama adalah `angka[0],` elemen kedua adalah `angka[1],` dst. Elemen terakhirnya adalah `angka[9].`

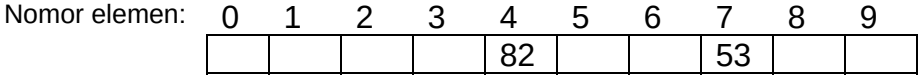
Elemen-elemen array ini dapat kita gunakan seperti variable pada umumnya. Misalnya kita isikan elemen array ke-4 dengan angka 67, kemudian isikan elemen array ke-7 dengan 53.

```
angka[4] = 67;
angka[7] = 53;
```



Kita juga dapat mengubah isinya, misalnya array ke-4 kita jumlahkan dengan 15.

```
angka[4] = angka[4] + 15;
```



Pada awal dideklarasikan, sama seperti variable-variable lain, array tidak memiliki isi yang jelas. Pada beberapa compiler, variable yang baru dibuat langsung diisi dengan nilai nol (0), namun ada juga compiler yang tidak memberi nilai awal, jadi jangan pernah mengasumsikan “nilai variable mula-mula pasti nol” !

Jika kita membutuhkan sebuah array yang bernilai awal nol, kita bisa menggunakan perintah `for` seperti berikut :

```
for(i=0; i<10; i++)
  angka[i] = 0;
```

Perintah diatas berarti mengulang sebanyak 10x sambil mengisikan variable `angka[i]` dengan `0`. Perulangan diatas sama saja dengan kita menuliskan :

```
angka[0] = 0;
angka[1] = 0;
```

```
angka[2] = 0;
angka[3] = 0;
angka[4] = 0;
angka[5] = 0;
angka[6] = 0;
angka[7] = 0;
angka[8] = 0;
angka[9] = 0;
```

Array n Dimensi

Array yang kita bahas diatas disebut array 1 dimensi, karena menggunakan sebuah indeks. Ada kalanya kita perlu array 2 dimensi, 3 dimensi, atau bahkan lebih.

Misalnya kita memiliki daftar nilai siswa SMA sebagai berikut :

No Absen	Nilai
1	7.5
2	8
3	6.5
4	6
5	6.5
6	7
7	8.5
8	8
9	7
10	8.5

Kita bisa saja memasukan data tersebuk ke array bertipe float sebagai berikut :

```
float nilai_siswa[10];
```

Nomor elemen:

0	1	2	3	4	5	6	7	8	9
7.5	8	6.5	6	6.5	7	8.5	8	7	8.5

* Karena array dalam C selalu mulai dari indeks 0, maka kita memasukan nilai-nilai tersebut ke array nomor absen-1.

Untuk mengambil nilai siswa dengan nomor absen 5, kita dapat melihat isi array `nilai_siswa[5-1]`.

Bagaimana jika kita memiliki daftar nilai beberapa kelas sebagai berikut :

Kelas 1-1		Kelas 1-2		Kelas 1-3	
No Absen	Nilai	No Absen	Nilai	No Absen	Nilai
1	7.5	1	7	1	8.5
2	8	2	7	2	8
3	6.5	3	8	3	7.5
4	6	4	8.5	4	8

5	6.5	5	6	5	6
6	7	6	6.5	6	6
7	8.5	7	7	7	7.5
8	8	8	6	8	7
9	7	9	8	9	8
10	8.5	10	7.5	10	7.5

Akan sulit jika kita memasukan nilai-nilai ini kedalam array 1 dimensi.

```
float nilai_siswa[30];
```

Nomor elemen:	0	1	2	3	4	5	6	7	8	9
	7.5	8	6.5	6	6.5	7	8.5	8	7	8.5

Nomor elemen:	10	11	12	13	14	15	16	17	18	19
	7	7	8	8.5	6	6.5	7	6	8	7.5

Nomor elemen:	20	21	22	23	24	25	26	27	28	29
	8.5	8	7.5	8	6	6	7.5	7	8	7.5

Jika kita ingin melihat nilai siswa kelas 1-1 absen 5, mungkin kita dapat langsung melihat isi array `nilai_siswa[5-1].` Tapi timbul masalah jika ingin mengambil nilai siswa kelas 1-2 absen 7, kita harus melihat isi array `nilai_siswa[16]`.

Akan lebih mudah jika kita menggunakan array 2 dimensi sebagai berikut :

```
float nilai_siswa[3][10];
```

dimensi 2 / dimensi 1	0	1	2	3	4	5	6	7	8	9
0	7.5	8	6.5	6	6.5	7	8.5	8	7	8.5
1	7	7	8	8.5	6	6.5	7	6	8	7.5
2	8.5	8	7.5	8	6	6	7.5	7	8	7.5

Ketika kita membutuhkan nilai siswa 1-2 absen 7, kita dapat melihat nilai array `nilai_siswa[2-1][7-1].` Atau dengan lebih umum bisa dikatakan :

untuk melihat nilai siswa pada kelas X absen Y, lihatlah isi array `nilai_siswa[X-1][Y-1].`

Mengapa harus dikurangi dengan 1 ? Ini dikarenakan array pada bahasa C selalu dimulai dengan nomor indeks 0, sehingga nilai siswa absen 1 disimpan pada array ke-0 dan nilai-nilai kelas 1 disimpan pada array ke-0.

Bagaimana jika kita ingin menyimpan data nilai untuk siswa SMA kelas 1, 2, 3, dimana masing-masing tingkat memiliki kelas A,B,C dan masing-masing kelas memiliki 40 siswa ?
Kita dapat membuat array 3 dimensi :

- Dimensi pertama untuk menunjukan kelas berapa.
    - Indeks 0 untuk kelas 1
    - Indeks 1 untuk kelas 2
    - Indeks 2 untuk kelas 3
- Dimensi kedua untuk menunjukan kelas A,B,atau C.
    - Indeks 0 untuk kelas A
    - Indeks 1 untuk kelas B
    - Indeks 2 untuk kelas C
- Dimensi ketiga untuk menunjukan nomor absen 1-40.
    - Indeks 0 untuk absen 1
    - Indeks 1 untuk absen 2
    - …
    - Indeks 39 untuk absen 40

Maka array kita akan seperti ini :

```
float nilai_siswa[3][3][40];
```

Misalnya untuk melihat nilai siswa kelas 3-A nomor absen 34, kita mengakses array `nilai_siswa[2][0][33]`.

Cobalah !
Bagaimana jika kita ingin menyimpan nilai siswa SMP dan SMA ? Masing-masing jenjang memiliki 3 tingkat kelas, masing-masing tingkat memiliki kelas A,B,C,D. Masing-masing kelas memiliki 40 siswa. Coba buatlah sebuah array 4 dimensi untuk kasus ini =).

Deklarasi Array n Dimensi
Mirip dengan deklarasi array 1 dimensi, untuk mendeklarasikan array n dimensi digunakan sintaks seperti berikut :

```
<tipe var> <nama var>[<jml dimensi1>][<jml dimensi2>]…[<jml dimensi n>]
```

Contoh membuat array 2 dimensi :

```
float nilai_siswa[3][10];
```

Contoh membuat array 3 dimensi :

```
float nilai_siswa[3][3][40];
```