BAB VI
ARRAY

Tujuan :

1. Menjelaskan tentang array berdimensi satu
2. Menjelaskan tentang array berdimensi dua
3. Menjelaskan tentang array berdimensi banyak
4. Menjelaskan tentang inisialisasi array tak berukuran.
5. Menjelaskan array sebagai parameter fungsi

Dalam beberapa literatur, array sering disebut (diterjemahkan) sebagai larik. Array adalah kumpulan dari nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama. Nilai-nilai data di suatu array disebut dengan elemen-elemen array. Letak urutan dari elemen-elemen array ditunjukkan oleh suatu *subscript* atau indeks.

Array bisa 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 table
. Array berdimensi tiga *(three-dimensional array)* mewakili bentuk suatu ruang.

6.1 Array Berdimensi Satu

6.1.1 Mendeklarasikan Array Berdimensi Satu

Suatu array berdimensi satu dideklarasikan dalam bentuk umum berupa :

```
tipe_data nama_var[ukuran];
```

dengan :

- tipe_data : untuk menyatakan tipe dari elemen array, misalnya *int, char, float*.
- nama_var : nama variabel array
- ukuran : untuk menyatakan jumlah maksimal elemen array.

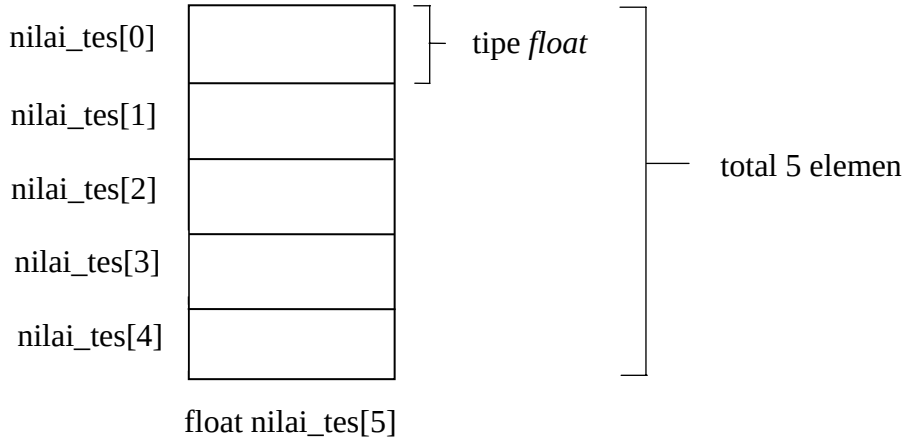
Contoh pendeklarasian array :

```
float nilai_tes[5];
```

menyatakan bahwa array **nilai_tes** mengandung 5 elemen bertipe *float*.

6.1.2 Mengakses Elemen Array Berdimensi Satu

Pada C, data array akan disimpan dalam memori yang berurutan. Elemen pertama mempunyai indeks bernilai 0. Jika variabel **nilai_tes** dideklarasikan sebagai array dengan 5 elemen, maka elemen pertama memiliki indeks sama dengan 0, dan elemen terakhir memiliki indeks 4. Gambar 6.1 di bawah ini menjelaskan urutan komponen dalam array.



Gambar 6.1 Array berdimensi satu

Bentuk umum pengaksesan array adalah sbb :

nama_var[indeks]

sehingga, untuk array **nilai_tes**, maka :

nilai_tes[0] → elemen pertama dari **nilai_tes**

nilai_tes[4] → elemen ke-5 dari **nilai_tes**

Contoh :

```
nilai_tes[0] = 70;                      /* contoh 1 */
scanf("%f", &nilai_tes[2]);             /* contoh 2 */
```

Contoh pertama merupakan pemberian nilai 70 ke **nilai_tes[0]**. Sedangkan contoh 2 merupakan perintah untuk membaca data bilangan dari keyboard dan diberikan ke **nilai_tes[2]**. Pada contoh 2 ini

```
&nilai_tes[2]
```

berarti "alamat dari **nilai_tes[2]**". Perlu diingat bahwa *scanf()* memerlukan argumen berupa alamat dari variabel yang digunakan untuk menyimpan nilai masukan.

Selengkapnya perhatikan contoh program di bawah ini

```
/* File program : nilai_tes.c
Pemakaian array utk menyimpan sejumlah nilai tes */

#include <stdio.h>
#define MAKS 5

main()
{
     int i;
     float total = 0, rata;
     float  nilai_tes[MAKS];       /* deklarasi array */

     for(i=0; i < MAKS; i++)  /* pemasukan data nilai_tes */
     {
          printf("Nilai tes ke-%d : ", i+1);
          scanf("%f", &nilai_tes[i]);
          /* menghitung jumlah seluruh nilai */
          total = total + nilai_tes[i];
     }
     rata = total / MAKS;     /* hitung nilai rata-rata */

     /* cetak nilai rata-rata */
     printf("\nNilai rata-rata = %g\n", rata);
}
```

Contoh eksekusi :

```
Nilai tes ke-1 : 56.5
Nilai tes ke-2 : 67.75
Nilai tes ke-3 : 80
Nilai tes ke-4 : 77
Nilai tes ke-5 : 78.5

Nilai rata-rata = 71.95
```

6.1.3 Inisialisasi Array Berdimensi Satu

Sebuah array dapat diinisialisasi sekaligus pada saat dideklarasikan. Untuk mendeklarasikan array, nilai-nilai yang diinisialisasikan dituliskan di antara kurung kurawal ({}) yang dipisahkan dengan koma.

```
/* File program : jhari.c */

#include <stdio.h>

main()
{
     int bulan, tahun, jhari;
     int jum_hari[12] =
     {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

     puts("MEMPEROLEH JUMLAH HARI");
     puts("PADA SUATU BULAN DAN SUATU TAHUN");
     puts("-------------------------------");
     printf("Masukkan bulan (1..12)     : ");
     scanf("%d", &bulan);
     printf("Masukkan tahunnya     : ");
     scanf("%d", &tahun);

     if(bulan == 2)
          if(tahun % 4 == 0)
               jhari = 29;
          else
               jhari = 28;
     else
          jhari = jum_hari[bulan-1];
     printf("\nJumlah hari dalam bulan %d tahun %d adalah %d
          hari\n", bulan, tahun, jhari);
}
```

Contoh eksekusi :

```
MEMPEROLEH JUMLAH HARI
PADA SUATU BULAN DAN SUATU TAHUN
-------------------------------
Masukkan bulan (1..12)   : 2
Masukkan tahunnya        : 1988

Jumlah hari dalam bulan 2 tahun 1988 adalah 29 hari
```

Pada program **jhari.c** di atas

```
int jum_hari[12] =
{31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

merupakan instruksi untuk mendeklarasikan array **jum_hari** yang memiliki 12 elemen yang bersifat statis dan sekaligus melakukan inisialisasi terhadap masing-masing elemen array.

```
/* File program : inisial.c */

#include <stdio.h>

main()
{
     int i;
     int values[] = {1,2,3,4,5,6,7,8,9};
     char word[] = {'H','e','l','l','o'};

     for(i = 0; i < 9; ++i )
          printf("values[%d] is %d\n", i, values[i]);

     printf("\n");
     for(i = 0; i < 6; ++i )
          printf("word[%d] is %c\n", i, word[i]);
}
```

Contoh eksekusi :

```
values[0] is 1
values[1] is 2
values[2] is 3
values[3] is 4
values[4] is 5
values[5] is 6
values[6] is 7
values[7] is 8
values[8] is 9

word[0] is H
word[1] is e
word[2] is l
word[3] is l
word[4] is o
```

Perhatikan, pada contoh **inisial.c** di atas, pendeklarasian nama variabel array tidak disertai ukuran yang mengindikasikan besarnya array. Dalam kondisi seperti ini, C akan menginisialisasi ukuran array tersebut sejumlah elemen yang diberikan di dalam kurung kurawal pada saat proses inisialisasi. Sehingga array **values** terdiri atas 9 elemen dan array **word** memiliki 5 elemen.

6.1.4 Beberapa Variasi dalam Mendeklarasikan Array

Ada beberapa variasi cara mendeklarasikan sebuah array (dalam hal ini yang berdimensi satu), di antaranya adalah sebagai berikut :

- `int  numbers[10];`
- `int numbers[10] = { 34, 27, 16 };`
- `int numbers[] = { 2, -3, 45, 79, -14, 5, 9, 28, -1, 0 };`
- `char text[] = "Welcome to New Zealand.";`
- `float radix[12] = { 134.362, 1913.248 };`
- `double radians[1000];`

6.2 Array Berdimensi Dua

Data seperti yang disajikan pada Tabel 6-1, dapat disimpan pada sebuah array berdimensi dua. Dimensi pertama dari array digunakan untuk menyatakan kode program kursus dan dimensi kedua untuk menyatakan tahun kursus.

Tabel 6-1. Data Kelulusan Siswa Pada Sebuah Kursus Komputer

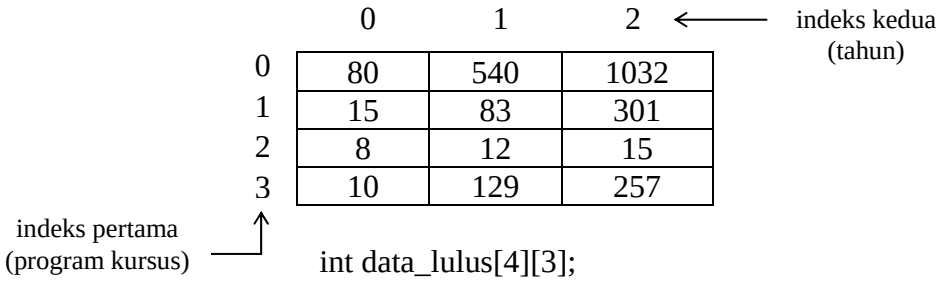
Tahun / Program	1998	1999	2000
1. (INTRO)	80	540	1032
2. (BASIC)	15	83	301
3. (PASCAL)	8	12	15
4. (C)	10	129	257

6.2.1 Mendeklarasikan Array Berdimensi Dua

Pendeklarasian yang diperlukan untuk menyimpan data kelulusan siswa pada Tabel 6-1 adalah:

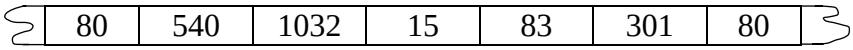
```
int data_lulus[4][3];
```

Nilai 3 untuk menyatakan banyaknya tahun dan 4 menyatakan banyaknya program kursus. Gambar 6.2 memberikan ilustrasi untuk memudahkan pemahaman tentang array berdimensi dua.



Gambar 6.2 Array berdimensi dua

Sama halnya pada array berdimensi satu, data array aka ditempatkan pada memori yang berurutan. Perhatikan Gambar 6.3.



Gambar 6.3 Model penyimpanan array dimensi dua pada memori

6.2.2 Mengakses Elemen Array Berdimensi Dua

Array seperti **data_lulus** dapat diakses dalam bentuk

data_lulus[indeks pertama, indeks kedua]

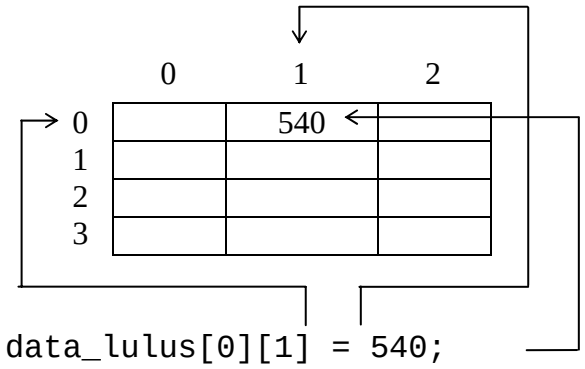
Contoh :

(1) `data_lulus[0][1] = 540;`

merupakan instruksi untuk memberikan nilai 540 ke array **data_lulus** untuk indeks pertama = 0 dan indeks kedua bernilai 1.

(2) `printf("%d",data_lulus[2][0]);`

merupakan perintah untuk menampilkan elemen yang memiliki indeks pertama = 2 dan indeks kedua = 0.



Gambar 6.4. Pemberian nilai ke array berdimensi dua

Perhatikan contoh program di bawah ini.

```
/* File program : lulus.c
Contoh pemakaian array berdimensi dua */

#include <stdio.h>

main( )
{
     int tahun, kode_program;
     int data_lulus[4][3] ;
```

```
/* Memberikan data ke array */
     data_lulus[0][0] = 80;
     data_lulus[0][1] = 540;
     data_lulus[0][2] = 1032;

     data_lulus[1][0] = 15;
     data_lulus[1][1] = 83;
     data_lulus[1][2] = 301;

     data_lulus[2][0] = 8;
     data_lulus[2][1] = 12;
     data_lulus[2][2] = 15;

     data_lulus[3][0] = 10;
     data_lulus[3][1] = 129;
     data_lulus[3][2] = 257;

     /* proses utk memperoleh informasi jml siswa yg lulus */
     printf("Masukkan tahun dr data yg ingin anda ketahui ");
     printf("(1998..2000) : ");
     scanf("%d", &tahun);
     printf("Masukkan kode program kursus yang ingin anda
          ketahui");
     printf("(1 = INTRO, 2 = BASIC, 3 = PASCAL, 4 = C)  : ");
     scanf("%d", &kode_program);
     printf("\nTotal kelulusan program tsb = %d\n",
          data_lulus[kode_program - 1][tahun - 1998] );
}
```

Contoh eksekusi :

```
Masukkan tahun dr data yg ingin anda ketahui (1998…2000) :
2000
Masukkan kode program kursus dari data yang ingin anda
ketahui (1 = INTRO, 2 = BASIC, 3 = PASCAL, 4 = C)  : 3

Total kelulusan program tsb = 15
```

6.2.3 Inisialisasi Array Berdimensi Dua

Gambar berikut memberikan penjelasan tentang inisialisasi yang dilakukan terhadap array berdimensi dua :

0	1	1	1	1	1	0	0
0	1	0	0	0	1	0	0

0	1	0	0	0	1	0	0
1	1	1	1	1	1	1	0
1	1	0	0	0	0	1	0
1	1	0	0	0	0	1	0
1	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0

Gambar 6.5 Representasi inisialisasi array berdimensi dua

Untuk itu, deklarasi dan inisialisasi yang dilakukan berupa :

```
int huruf_A[8][8] = {
        { 0, 1, 1, 1, 1, 1, 0, 0 } ,
        { 0, 1, 0, 0, 0, 1, 0, 0 } ,
        { 0, 1, 0, 0, 0, 1, 0, 0 } ,
        { 1, 1, 1, 1, 1, 1, 1, 0 } ,
        { 1, 1, 0, 0, 0, 0, 1, 0 } ,
        { 1, 1, 0, 0, 0, 0, 1, 0 } ,
        { 1, 1, 0, 0, 0, 0, 1, 0 } ,
        { 0, 0, 0, 0, 0, 0, 0, 0 }
    };
```

atau bisa juga ditulis sebagai berikut :

```
int huruf_A[8][8] =
   {  0, 1, 1, 1, 1, 1, 0, 0,
      0, 1, 0, 0, 0, 1, 0, 0,
      0, 1, 0, 0, 0, 1, 0, 0,
      1, 1, 1, 1, 1, 1, 1, 0,
      1, 1, 0, 0, 0, 0, 1, 0,
      1, 1, 0, 0, 0, 0, 1, 0,
      1, 1, 0, 0, 0, 0, 1, 0,
      0, 0, 0, 0, 0, 0, 0, 0
   };
```

Contoh program berikut memenfaatkan data yang ada pada array **huruf_A** untuk membentuk karakter A dengan ukuran besar. Setiap nilai satu pada array akan diganti dengan karakter ber-ASCII 219 (DBh) dan nilai 0 akan diganti dengan karakter spasi.

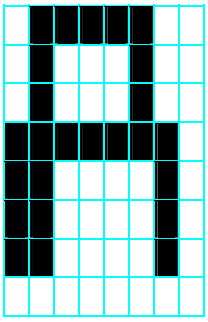
```
/* File program : hurufA.c
Contoh inisialisasi array dimensi dua  */

#include <stdio.h>
```

```
main()
{
     int  i,j;
     int huruf_A[8][8] = {
          { 0, 1, 1, 1, 1, 1, 0, 0 } ,
          { 0, 1, 0, 0, 0, 1, 0, 0 } ,
          { 0, 1, 0, 0, 0, 1, 0, 0 } ,
          { 1, 1, 1, 1, 1, 1, 1, 0 } ,
          { 1, 1, 0, 0, 0, 0, 1, 0 } ,
          { 1, 1, 0, 0, 0, 0, 1, 0 } ,
          { 1, 1, 0, 0, 0, 0, 1, 0 } ,
          { 0, 0, 0, 0, 0, 0, 0, 0 }
     };

     for(i = 0; i < 8; i++)
     {
          for(j = 0; j < 8; j++)
               if(huruf_A[i][j] !=0  )
                    putchar('\xDB\n');
               else
                    putchar ('  ');               /* spasi */
               putchar('\n');
     }
}
```

Contoh eksekusi :



6.3. Array Berdimensi Banyak.

C memungkinkan untuk membuat array yang dimensinya lebih dari dua. Bentuk umum pendeklarasian array berdimensi banyak :

```
tipe nama_var[ukuran 1][ukuran2}…[ukuranN];
```

sebagai contoh :

```
int data_huruf[2][8][8];
```

merupakan pendeklarasian array **data_huruf** sebagai array berdimensi tiga.

Sama halnya dengan array berdimensi satu atau dua, array berdimensi banyak juga bisa diinisialisasi. Contoh inisialisasi array berdimensi tiga :

```
int data_huruf [2][8][8] =
     { { { 0, 1, 1, 1, 1, 1, 0, 0 } ,
         { 0, 1, 0, 0, 0, 1, 0, 0 } ,
         { 0, 1, 0, 0, 0, 1, 0, 0 } ,
         { 1, 1, 1, 1, 1, 1, 1, 0 } ,
         { 1, 1, 0, 0, 0, 0, 1, 0 } ,
         { 1, 1, 0, 0, 0, 0, 1, 0 } ,
         { 1, 1, 0, 0, 0, 0, 1, 0 } ,
         { 0, 0, 0, 0, 0, 0, 0, 0 }
       },
       { {1, 1, 1, 1, 1, 1, 0, 0 } ,
         {1, 0, 0, 0, 0, 1, 0, 0 } ,
         {1, 0, 0, 0, 0, 1, 0, 0 } ,
         {1, 1, 1, 1, 1, 1, 1, 0 } ,
         {1, 1, 0, 0, 0, 0, 1, 0 } ,
         {1, 1, 0, 0, 0, 0, 1, 0 } ,
         {1, 1, 1, 1, 1, 1, 1, 0 } ,
         {0, 0, 0, 0, 0, 0, 0, 0 }
       } };
```

atau bisa juga ditulis menjadi

```
int data_huruf [2][8][8] =
          0, 1, 1, 1, 1, 1, 0, 0,
          0, 1, 0, 0, 0, 1, 0, 0,
          0, 1, 0, 0, 0, 1, 0, 0,
          1, 1, 1, 1, 1, 1, 1, 0,
          1, 1, 0, 0, 0, 0, 1, 0,
          1, 1, 0, 0, 0, 0, 1, 0,
          1, 1, 0, 0, 0, 0, 1, 0,
          0, 0, 0, 0, 0, 0, 0, 0,
          1, 1, 1, 1, 1, 1, 0, 0,
          1, 0, 0, 0, 0, 1, 0, 0,
          1, 0, 0, 0, 0, 1, 0, 0,
          1, 1, 1, 1, 1, 1, 1, 0,
          1, 1, 0, 0, 0, 0, 1, 0,
          1, 1, 0, 0, 0, 0, 1, 0,
          1, 1, 1, 1, 1, 1, 1, 0,
```

```
            0, 0, 0, 0, 0, 0, 0, 0
          };
```

Program berikut merupakan pengembangan dari Program 8 – 4, digunakan untuk menampilkan dua buah huruf dengan ukuran besar.

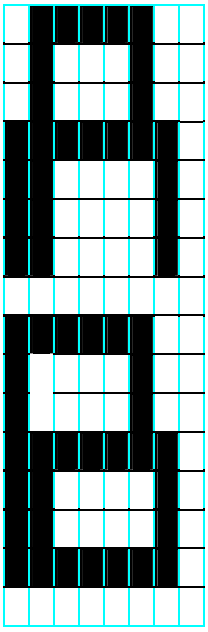
```
/* File program: data_huruf.c
Contoh inisialisasi array dimensi tiga  */

# include <stdio.h>

main()
{
     int i, j, k;
     int data_huruf[2][8][8] = {
          {{ 0, 1, 1, 1, 1, 1, 0, 0 },
          { 0, 1, 0, 0, 0, 1, 0, 0 },
          { 0, 1, 0, 0, 0, 1, 0, 0 },
          { 1, 1, 1, 1, 1, 1, 1, 0 },
          { 1, 1, 0, 0, 0, 0, 1, 0 },
          { 1, 1, 0, 0, 0, 0, 1, 0 },
          { 1, 1, 0, 0, 0, 0, 1, 0 },
          { 0, 0, 0, 0, 0, 0, 0, 0 }
          },
          {{1, 1, 1, 1, 1, 1, 0, 0 },
          {1, 0, 0, 0, 0, 1, 0, 0 },
          {1, 0, 0, 0, 0, 1, 0, 0 },
          {1, 1, 1, 1, 1, 1, 1, 0 },
          {1, 1, 0, 0, 0, 0, 1, 0 },
          {1, 1, 0, 0, 0, 0, 1, 0 },
          {1, 1, 1, 1, 1, 1, 1, 0 },
          {0, 0, 0, 0, 0, 0, 0, 0 }
          }
     };

     for(i = 0; i < 2; i++)             /* Tampilkan huruf */
     {
          for(j = 0; j < 8; j++)
          {
               for(k = 0; k < 8; k++)
                    if(data_huruf[i][j][k] != 0)
                         putchar('\xDB');
                    else
                         putchar(' ');  /* spasi */
                    printf("\n");       /* pindah baris */
          }
          printf("\n");                 /* pindah baris */
     }
}
```

Contoh Eksekusi :



6.4. Inisialisasi Array Tak Berukuran

Inisialisasi array yang tak berukuran dapat dilakukan untuk array berdimensi satu atau lebih. Untuk array berrdimensi lebih dari satu, dimensi terkirilah yang boleh tak berukuran. Dengan cara ini tabel dalam array dapat diperluas atau dikurangi tanpa mengubah ukuran array. Sebagai contoh :

tak berukuran

```
int skala[] =
      { 1, 2, 4, 6, 8 };
```

merupakan pendeklarasian array berdimensi satu yang tak berukuran. Secara otomatis

skala[0] bernilai 1

skala[1] bernilai 2

skala[2] bernilai 4

skala[3] bernilai 6

skala[4] bernilai 8

Contoh lain :

```
char konversi[][2] =
        { 'A', 'T',
          'E', 'M',
          'I', 'V',
          'O', 'S',
          'U', 'J',
         };
```

Pada contoh ini,

konversi[0][0] bernilai 'A'

konversi[0][1] bernilai 'T'

konversi[1][0] bernilai 'E'

konversi[1][1] bernilai 'M'

……..

Contoh berikut akan menyandikan suatu kalimat yang dimasukkan melalui keyboard dengan menggunakan data yang ada pada tabel **konversi.** Misal, huruf A akan diganti menjadi T, huruf T akan diganti menjadi A.

```
/* File program : sandi.c
contoh inisialisasi array tak berukuran */

#include <stdio.h>
#include <stdlib.h>

#define JUM_KOLOM 2
#define MAX_KAR 256

main()
{
     char konversi[][JUM_KOLOM] = {
          'A', 'T',
          'a', 't',
          'E', 'M',
          'e', 'm',
          'I', 'V',
          'i', 'v',
          'O', 'S',
          'o', 's',
          'U', 'J',
          'u', 'j'
     };
```

```
     char kalimat[MAX_KAR], karakter;
     int i = 0, j, jum_kar, jum_penyandi;

     printf("Masukkan sebuah kalimat dan akhiri dengan
               ENTER\n");
     printf("Kemudian kalimat tsb akan saya sandikan\n");
     printf("Kalimat:  ");

     /* Memasukkan data karakter ke array kalimat */
     while((kalimat[i] = getchar()) != '\n')
          i++;
     jum_kar = i ;

     /* sandikan dan menampilkan ke layar */
     printf("\nHasil setelah disandikan: ");
     jum_penyandi = sizeof(konversi) / JUM_KOLOM;

     for(i = 0; i < jum_kar; i++)
     {
          karakter = kalimat [i];
          for(j = 0; j < jum_penyandi; j++)
          {
               if(karakter == konversi[j][0])
               {
                    karakter = konversi[j][1];
                    break; /* keluar dari for terdalam */
               }
               if(karakter == konversi[j][1])
               {
                    karakter = konversi[j][0];
                    break; /* keluar dari for terdalam */
               }
          }
          putchar (karakter);
     }
     printf("\n\n");
}
```

Contoh Eksekusi :

```
Masukkan sebuah kalimat dan akhiri dengan ENTER
Kemudian kalimat tsb akan saya sandikan
Kalimat : Saya belajar

Hasil setelah disandikan: Otyt bmltutr
```

Pada program diatas ,

```
sizeof(konversi)/JUM_KOLOM)
```

merupakan ungkapan untuk memperoleh banyaknya baris (ukuran dari dimensi pertama) dalam array **konversi.** Dengan cara ini tabel konversi dapat diperluas tanpa perlu memberitahu banyaknya jumlah dimensi pertama secara ekplisit.

6.5. Array Sebagai Parameter

Array juga dapat dilewatkan sebagai parameter fungsi. Sebagai contoh ditujukan pada program **sorting.c**. Program digunakan untuk memasukkan sejumlah data, kemudian data tersebut diurutkan naik *(ascending)* dan dicetak ke layar.

Untuk melakukan sorting (proses pengurutan data), cara yang dipakai yaitu metode *buble sort* (suatu metode pengurutan yang paling sederhana, dan memiliki kecepatan pengurutan yang sangat lambat).

Algoritma pada metode pengurutan ini adalah sebagai berikut :

1. Atur i bernilai 0
2. Bandingkan x[i] dengan x[j], dengan j berjalan dari i + 1 sampai dengan n-1.
3. Pada setiap pembandingan, jika x[i] > x[j], maka isi x[i] dan x[j] ditukarkan
4. Bila i < (n – 1), ulangi mulai langkah 2.

Catatan: i = indeks array
x = nama array untuk menyimpan data
n = jumlah data

Algoritma diatas berlaku untuk pengurutan menaik *(ascending)*. Untuk pengurutan menurun *(descending)*, penukaran dilakukan jika x[i] < x[j].

Penjelasan proses pengurutan terhadap 5 buah data ditunjukkan pada gambar 6.6.

Data semula (sebelum pengurutan) adalah

50,5 30,3 20,2 25,2 31,3

Setelah pengurutan menaik (secara *ascending*), hasil yang diharapkan berupa

20,2 25,2 30,3 31,3 50,5

i = 0 ⟶ j = 1

0	50.5
1	30.3
2	20.2
3	25.2
4	31.3

j = 2

30.3
50.5
20.2
25.2
31.3

j = 3

20.2
50.5
30.3
25.2
31.3

j = 4

20.2
50.5
30.3
25.2
31.3

i = 1 ⟶ j = 2

0	20.2
1	50.5
2	30.3
3	25.2
4	31.3

j = 3

20.2
30.3
50.5
25.2
31.3

j = 4

20.2
25.2
50.5
30.3
31.3

i = 2 ⟶ j = 3

0	20.2
1	25.2
2	50.5
3	30.3
4	31.3

j = 4

20.2
25.2
30.3
50.5
31.3

i = 3 ⟶ j = 4

0	20.2
1	25.2
2	30.3
3	50.5
4	31.3

Hasil akhir

20.2
25.2
30.3
31.3
50.5

Gambar 6.6 Proses pengurutan data secara ascending dengan metode Buble Sort

```
/* File program : sorting.c */
#include <stdio.h>

#define MAKS 20

void pemasukan_data(float [], int *);
void pengurutan_data(float [], int);
void penampilan_data(float [], int);


main()
{
     float data[MAKS];
     int jum_data;

     pemasukan_data(data, &jum_data);
     pengurutan_data(data, jum_data);
     penampilan_data(data, jum_data);
}

void pemasukan_data(float x[], int *pjumlah)
{
     int jum, i;

     printf("Jumlah data = ");
     scanf("%d", &jum);

     for(i=0; i<jum; i++)
     {
          printf("Data ke-%d : ", i+1);
          scanf("%f", &x[i]);
     }

     *pjumlah = jum;
}

void pengurutan_data(float x[], int jumlah)
{
     int i, j;
     float smtr;

     for(i=0; i<jumlah-1; i++)
          for(j=i+1; j<jumlah; j++)
               if(x[i] > x[j])          /* penukaran data */
```

```
                {
                     smtr = x[i];
                     x[i] = x[j];
                     x[j] = smtr;
                }
}

void penampilan_data(float x[], int jumlah)
{
     int i;

     printf("\nData setelah diurutkan :\n\n");
     for (i=0; i<jumlah; i++)
          printf("Data ke-%d : %g\n", i+1, x[i]);
}
```

Contoh Eksekusi :

```
Jumlah data = 5
Data ke-1 : 50.5
Data ke-2 : 30.3
Data ke-3 : 20.2
Data ke-4 : 25.2
Data ke-5 : 31.3
Data setelah diurutkan

Data ke-1 : 20.2
Data ke-2 : 25.2
Data ke-3 : 30.3
Data ke-4 : 31.3
Data ke-5 : 50.5
```

Pada fungsi **main( )**;

- `float data[MAKS];`

  merupakan instruksi untuk mendeklarasikan array data dengan elemen bertipe float dan jumlahnya sebanyak MAKS elemen.

- `pemasukan_data(data, &jum_data);`

  merupakan instruksi untuk memanggil fungsi **pemasukan_data ( )**. Pada pemanggilan fungsi, data merupakan array. Yang perlu diperhatikan, parameter array ditulis tanpa diawali dengan &, sekalipun tujuan dari pemanggilan fungsi yaitu untuk mengisikan data ke array. Sebab nama array tanpa kurung siku dalam parameter fungsi berarti ”alamat dari elemen indeks ke-0 dari array tsb”.

Sedangkan **&jum_data** berarti ”alamat dari jum_data”. Tanda & harus disertakan sebab variabel **jum_data** akan diisi oleh fungsi **pemasukan_data ().**

- `pengurutan_data(data, jum_data);`
  Merupakan instruksi untuk menjalankan fungsi **pengurutan _data (),** dalam hal ini **data** dilewatkan ke fungsi dengan referensi (memberikan alamat array), karena memang hal ini merupakan cara satu-satunya untuk melewatkan array. Sedangkan jumlah data dilewatkan ke fungsi dalam bentuk nilai (pemanggilan dengan nilai).
- `penampilan_data(data,jum_data);`
  Merupakan instruksi untuk memanggil fungsi `penampilan_data().`

Pada fungsi untuk pemasukan data, pengurutan data maupun penampilan data,

```
data[i]
```

menyatakan elemen **data** ke-i.

Beberapa hal tambahan yang perlu diketahui:

- Untuk menyatakan alamat dari suatu elemen array, bentuk umumnya adalah

  ```
  &nama_array[indeks]
  ```

  Misalnya,

  ```
  &data[1]
  ```

  menyatakan alamt dari elemen ke-1. Adapun

  ```
  &data[0]
  ```

  sama saja dengan:

  ```
  data
  ```

- Suatu array berdimensi satu dalam parameter formal dideklarasikan dengan bentuk

  ```
  tipe nama_array[]
  ```

dengan di dalam tanda kurung siku tak disebutkan mengenai jumlah elemen. Jumlah elemen dinyatakan dalam parameter tersendiri (atau dinyatakan dalam bentuk variabel eksternal). Untuk array berdimensi lebih dari satu, kurung siku terkirilah yang kosong.

Kesimpulan :

- Array adalah kumpulan dari nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama.
- Nilai-nilai data di suatu array disebut dengan elemen-elemen array.
- Letak urutan dari elemen-elemen array ditunjukkan oleh suatu *subscript* atau indeks.
- Array bisa berupa array berdimensi satu, dua, tiga atau lebih.
- Data array akan disimpan dalam memori yang berurutan, dengan elemen pertama mempunyai indeks yang bernilai 0.
- Sebuah array dapat diinisialisasi sekaligus pada saat dideklarasikan, caranya saat mendeklarasikan array, nilai-nilai yang diinisialisasikan dituliskan di antara kurung kurawal (`{}`) yang dipisahkan dengan koma.
- Array juga dapat dilewatkan sebagai parameter fungsi.

Latihan :

Buatlah potongan program untuk soal-soal di bawah ini

1. Deklarasikan sebuah variabel array (misalkan nama variabelnya = **letters**) yang mengalokasikan untuk 10 elemen bertipe *char*.

2. Masukkan karakter ‘Z’ pada elemen yang ke-empat dari array **letters.**

3. Gunakan loop *for* untuk menghitung nilai akumulasi (total) dari seluruh isi sebuah array integer (*array of int*, misalkan namanya = **numbers**) yang memliki 5 elemen.

4. Deklarasikan sebuah array multidimensi yang elemennya bertipe float (*array of float*, misalkan namanya = **balances**) yang memiliki 3 baris dan 5 kolom.

5. Gunakan loop *for* untuk menghitung nilai total dari seluruh isi array **balances** di atas.

6. Deklarasikan sebuah array karakter (*array of char*, mislakan namanya = **words**) dan sekaligus inisialisasikan dengan nilai string = `“Hello”`

7. Deklarasikan sebuah array karakter (*array of char*, mislakan namanya = **stuff**) dengan alokasi 50 elemen. Selanjutnya isikan dengan nilai string = `“Welcome”` pada body programnya (bukan pada saat deklarasi)

8. Gunakan statemen *printf()* untuk menampilkan (di layar) isi elemen ke-tiga dari sebuah array integer (*array of int*, misalkan namanya = **total**s)

9. Gunakan statemen *printf()* untuk menampilkan (di layar) string yang merupakan isi dari sebuah array karakter (*array of char*, misalkan namanya = **words**)

10. Gunakan statemen *scanf()* untuk membaca string masukan dari keyboard dan memasukkannya ke array **words.**

11. Tuliskan loop *for* untuk membaca 5 karakter (gunakan statemen *scanf()*) dan memasukkannya ke dalam array **words**, mulai dari indeks 0.