A R R A Y

A. Pendahuluan

Deskripsi Singkat

Bab ini akan mengemukakan bahasan tentang definisi array, deklarasi array, mengakses elemen array, array sebagai tipe data bentukan, array konstan, array sebagai parameter, array multidimensi.

Relevansi

Pembahasan pada bab ini sangat penting dipahami, karena materi array ini memberikan manfaat yaitu efisiensi program. Materi array sangat berkaitan dengan materi lainnya dalam sebuah pemrograman terstruktur, karena array dapat digunakan dan dikombinasikan dengan bahasan lain dalam sebuah program.

Tujuan Instruksional Khusus

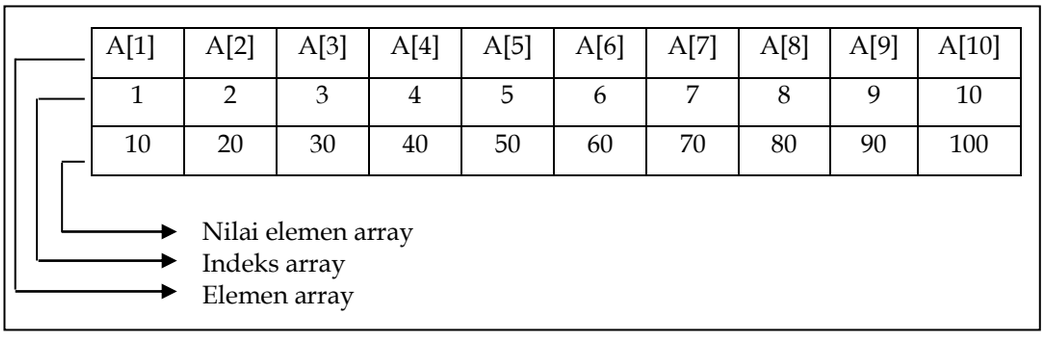
Mahasiswa mampu membuat program dengan menggunakan array.

B. Penyajian

Bagi para pemrogram, efisiensi program merupakan hal utama yang harus diperhatikan, baik itu dalam hal kecepatan jalannya program, memori yang digunakan, banyak baris kode yang dituliskan dan juga ketepatan algoritma yang digunakan. Salah satu komponen yang harus dikuasai untuk memperoleh program yang baik adalah pengetahuan tentang array.

1) Definisi Array

Array (larik) adalah sebuah variabel yang dapat menyimpan lebih dari satu nilai sejenis (memilikii tipe data sama). Hal ini berbeda dengan variabel biasa yang hanya mampu menampung satu buah nilai. Setiap nilai yang disimpan di dalam array disebut dengan elemen array, sedangkan nilai urut yang digunakan untuk mengakses elemennya disebut dengan indeks array. Sebagai contoh, misalkan terdapat array A yang memiliki 10 buah elemen nilai yang bertipe integer, maka dapat dipresentasikan sebagai berikut :



Gambar 1.1 Komponen Array (sumber :Rahardjo)

Setiap elemen array di atas menyimpan nilai bertipe integer dan akan menempati alamat memori yang berbeda, hal ini akan menyebabkan array tersebut memiliki ukuran 40 byte, yang berasal dari 10 x 4. Nilai 10 menunjukkan banyaknya elemen array sedangkan nilai 4 merupakan ukuran dari tipe data integer (dalam 32 bit).

2) Deklarasi Array

Sama seperti variabel lain, array juga dideklarasikan di dalam bagian deklarasi variabel. Bila akan didefinisikan sebagai tipe bentukan, maka array juga akan dideklarasikan di bagian definisi tipe (di bawah kata kunci type). Dalam bahasa Pascal, pendeklarasian array dilakukan dengan menggunakan kata kunci array dan tipe data yang akan disimpan di dalamnya, selai itu juga harus disertai dengan batas-batas indeksnya yang diapit oleh tanda *bracket* ([ ]). Berikut ini bentuk umum pendeklarasian array.

```
NamaArray : array [Indeks Awal . . IndeksAkhir] of tipe data;
```

Gambar 1.2 Bentuk Umum Pendeklarasian array(sumber:Rahardjo)

Sebagai contoh, apabila kita ingin mendeklarasikan array dengan nama A yang berisi 10 buah elemen bertipe integer, maka kita harus mendeklarasikannya dengan cara berikut.

```
Var
A : array [1 . . 10] of integer;
```

Pada kode tersebut, indeks array diulai dari satu. Perlu diperhatikan bahwa bahasa Pascal berbeda dengan bahasa C yang indeks array-nya selalu dimulai dari nol. Pada bahasa Pascal, indeks array dapat dimulai dari bilangan berapapun. Selain itu, indeks array juga dapat bertipe karakter maupun tipe enumerasi. Berikut ini contoh-contoh kode yang dapat digunakan untuk mendeklarasikan 10 buah elemen array bertipe integer sebagai pengganti kode di atas.

```
Var
A1 : array [0 . . 9] of integer;
A2 : array [5 . . 15] of integer;
A3 : array ['a' . . 'j'] of integer;
A4 : arrat ['A' . . 'J'] of integer;
```

Dalam bahasa Pascal, tersedia dua buah fungsi yang dapat digunakan untuk mengambil indeks terendah dan tertinggi dari sebuah array, yaitu fungsi Low dan High. Adapun parameter dari kedua fungsi tersebut adalah nama array yang akan dicari indeksnya. Perhatikan contoh kode berikut.

```
Var
A: array [1 . . 100] of integer;
terendah, tertinggi : integer;
Begin
```

```
terendah := Low (A);      {akan menghasilkan nilai 1}
tertinggi := High (A)     {akan menghasilkan nilai 100}
 . .
end.
```

3) Mengakses Elemen Array

Setelah mengetahui cara pendeklarasian array, selanjutnya kita harus mengetahui bagaimana cara untuk memanipulasi array tersebut. Langkah pertama yang harus dilakukan adalah mengisikan nilai ke dalam elemen-elemen array bersangkutan. Bentuk umum untuk pengisian elemen array adalah sebagai berikut.

```
NamaArray [indeks] := nilai;
```

Gambar 1.3 Bentuk Umum Pengisian Elemen Array(sumber:Rahardjo)

Untuk lebih memahaminya, coba perhatikan contoh kode di bawah ini.

```
Var
A: array [1..100] of integer;
Begin
A[1] :=1;          {mengisi elemen pertama dengan nilai 1}
A[2] :=2;          {mengisi elemen kedua dengan niali 2}
A[3] :=3;          {mengisi elemen ketiga dengan niali 3}
....
A[100] :=100;      {mengisi elemen keseratus dengan nilai 100}
end.
```

Kode tersebut akan melakukan pengisian 100 elemen array dengan nilai 1 sampai 100 sehingga kode tersebut akan lebih sederhana apabila dituliskan dengan menggunakan struktur pengulangan seperti yang terlihat pada kode berikut.

```
Var
A: array [1..100] of integer;
i  : integer;
Begin
For  i:= 1 to 100 do
Begin
A[1] := i;
end;
End.
```

4) Mengapa Harus Menggunakan Array

Bagi seorang pemula, mungkin akan mucul pertanyaan mengapa kita perlu mendeklarasikan array? Untuk menjawab pertanyaan tersebut, coba perhatikan contoh kasus berikut.

Apabila kita akan membuat program untuk menyimpan sekumpulan data, misalnya data-data hasil penelitian yang berupa bilangan, dimana jumlah dari data tersebut puluhan, ratusan atau bahkan ribuan, apakah akan menggunakan variabel sebanyak data yang ada? Jawabannya tentu tidak, karena hal tersebut merupakan hal yang sangat tidak efisien. Sebagai contoh, asumsikan bahwa banyak data tersebut.

```
Var
N1, n2, n3, n4, n5, n6, n7, n8, n9, n10 : real;
Begin
Writeln('masukkan data ke-1 :  '); readln(n1);
Writeln('masukkan data ke-2 :  '); readln(n2);
Writeln('masukkan data ke-3 :  '); readln(n3);
Writeln('masukkan data ke-4 :  '); readln(n4);
Writeln('masukkan data ke-5 :  '); readln(n5);
Writeln('masukkan data ke-6 :  '); readln(n6);
Writeln('masukkan data ke-7 :  '); readln(n7);
Writeln('masukkan data ke-8 :  '); readln(n8);
Writeln('masukkan data ke-9 :  '); readln(n9);
Writeln('masukkan data ke-10 :  '); readln(n10);
End.
```

Hal ini tentu akan merepotkan diri kita. Apabila dilihat, program di atas memang masih pendek karena datanya hanya 10, bagaimana bila ratusan ata bahkan ribuan?

Untuk mengatasi masalah ini, seharusnya kita menggunakan array untuk menyimpan data-data tersebut sehingga program akan jauh lebih sederhana dan mudah dalam pengerjaannya. Berikut ini perbaikan program di atas apabila kita menampung data-datanya ke dalam sebuah array.

```
Const max = 10;
Var
n : array [1 . . max] of real;
i : integer;
begin
for i:= 1 to max do
writeln('Masukkan data ke-', i,'  :  '); readln(n[i]);
end.
```

Apabila ternyata data berjumlah 100 atau 1000, maka kita hanya perlu mengganti nilai dari konstanta **max** di atas dengan nilai yang sesuai. Alasan seperti inilah yang menyebabkan kita perlu untuk menggunakan array.

5) Array Sebagai Tipe Data Bentukan

Array juga dapat digunakan sebagai tipe data bentukan, yaitu dengan cara mendeklarasikannya di bagian definisi tipe data, yaitu bagian yang diawali dengan kata kunci **type**.

```
Type
Bilangan = array [1 . . 100] of integer;
Vokal = array [1 . . 5] of char;
```

Setelah selesai mendefinisikan array tersebut, maka kita dapat menggunakannya untuk mendeklarasikan variabel lain di dalam program. Berikut ini contoh pendeklarasian variabel yang menggunakan tipe bentukan berupa array.

```
Var
X : bilangan;
Vowel : vokal;
```

Pada ontoh di atas kita mendeklarasikan variabel dengan nama x yang bertipe Bilangan, sedangkan variabel vowel bertipe vokal. Hal ini menyebabkan variabel tersebut juga dapat diperlukan sebagai array. Berikut ini contoh kode yang menunjukkan penggunaan variabel-variabel tersebut.

```
Begin
X[1[ := 1;
X[2] := 2;
. . .
Vowel [1] := 'a';
Vowel [2] := 'i';
. . .
End.
```

6) Array Konstan

Nilai yang terkandung di dalam sebuah array dapat bernilai konstan, artinya nilai-nilai tersebut tidak dapat diubah. Untuk melakukan hal tersebut, kita harus mendeklarasikan array bersangkutan dengan kata kunci **const**. Berikut ini bentuk umum pendeklarasiannya.

```
Const
NamaArray : array [indexAwal . . indeksAkhir] of tipe_data =
        Nila1, nilai2, ...);
```

Gambar 1.4 Bentuk Umum Pendeklarasian Array Konstan (sumber:Rahardjo)

Perlu diperhatikan bahwa banyaknya nilai konstan yang dituliskan diatas harus sesuai dengan banyaknya elemen array yang didefinisikan. Sebagai contoh, apabila kita ingin mendeklarasikan array dengan jumlah elemen 5, maka nilai konstan yang diisikan juga haruslah berjumlah 5. Perhatikan contoh kode berikut.

```
Const
A : array [1 . . 5] of char = ('A' , 'B' , 'C' , 'D' , 'E');
```

Oleh karena array A di atas bersifat konstan, maka kita tidak dapat menggantikan nilainya dengan nilai lain, seperti yang ditunjukkan oleh kode di bawah ini.

```
A [1]     := 'V' ; {SALAH, karena elemen A [1] selalu bernilai 'A'}
A [2]     := 'W' ; {SALAH, karena elemen A [2] selalu bernilai 'B'}
A [1]     := 'X' ; {SALAH, karena elemen A [3] selalu bernilai 'C'}
A [1]     := 'Y' ; {SALAH, karena elemen A [4] selalu bernilai 'D'}
A [1]     := 'Z' ; {SALAH, karena elemen A [5] selalu bernilai 'E'}
```

Hal ini menunjukkan bahwa array konstan nilainya hanya dapat dibaca, namun tidak untuk diubah. Agar lebih memahami konsepnya, perhatikan contoh implementasi dari array konstan berikut ini.

```
Function HariSekarang : string;
Const
```

```
Hari : array[0 . .6] of string[6] =
        ('Minggu', 'Senin', 'Selasa', 'Rabu', 'Kamis', 'Jumat', 'Sabtu');
var
thn, bln, hr, indeksHari : word;
begin
getDate(thn, bln, hr, indeksHari);
HariSekarang := Hari [indeksHari];
End;
```

Pada contoh di atas kita membuat sebuah fungsi untuk mendapatkan nama hari sesuai dengan tanggal sekarang (hari ini). Berikut ini contoh program lain yang akan menunjukkan penggunaan array konstan.

```
Program ArrayKonstan;
Uses crt;
Const
Bulan : array [1 . . 12] of string =
          ('Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni', 'Juli', 'Agustus',
           'September', 'Oktober', 'Nopember', 'Desember');
var
noBulan : integer;
begin
clrscr;
write('Masukkan nomor bulan :'); readln(noBulan);
write('Nama bulan ke-', noBulan, 'adalah ', Bulan[noBulan]);
readln;
end.
```

Contoh hasil yang akan diberikan oleh program di atas adalah sebagai berikut.

```
Masukkan nomor bulan : 3
Nama bulan ke-3 adalah Maret
```

7) Array Sebagai Parameter

Pada kasus-kasus pemrograman tertentu kita juga dituntut untuk menggunakan array sebagai parameter sebuah prosedur ataupun fungsi. Hal ini sering kita jumpai pada saat kita akan melakukan pencarian maupun pengurutan dari sekumpulan data. Berikut ini contoh penggunaan array di dalam sebuah prosedur.

```
Type
Bilangan = array[1..100] of integer;
Procedure inputArray[a:bilangan; N:integer);
Var
i : integer;
Begin
For i := 1 to N do
Write ('masukkan elemen array ke-', i); readln(A[i]);
End;
```

Pada contoh di atas kita te;ah membuat prosedur yang memiliki parameter bertipe array. Prosedur tersebut akan digunakan untuk melakukan pengisian elemen array sebanyak N, dimana $1 \leq N \leq 100$.

Perlu diperhatikan bahwa array yang dilewatkan sebagai parameter ini harus dideklarasikan terlebih dahulu. Berikut ini contoh penggunaan array yang tidak diperbolehkan oleh kompiler.

```
Procedure InputArray [A:array[1..100] of integer; N);                {salah}
```

Berikut ini kode yang merupakan perbaikan dari kode sebelumnya.

```
Procedure InputArray [A:array of integer; N);                {benar}
```

Untuk lebih memahaminya, perhatikan contoh program di bawah ini dimana kita akan menggunakan array sebagai parameter.

```
Program ParamArray;
Uses crt;
Const max = 100;]type
Bilangan = array [1 . . max] of integer;
Procedure InputArray[A:bilangan; N:integer);
Var
i:integer;
Begin
Writeln (’Memasukkan data :’);
For i : 1 to N do
Write(’Masukkan nilai A[’, i,  ’] : ’); readln(A[i]);
End;
Procedure OutputArray (A:bilangan; N:integer);
Var
i:integer;
begin
writeln(’Menampilkan data :’);
for i : 1 to N do
write (’A[’, i,  ’] = ’, A[i]);
end;
var
Arr:bilangan;
count:integer;
Begin
Clrscr;
Write (’Masukkan banyaknya elemen array :’); readln(count);
Writeln;
OutputArray (Arr, count);
Readln;
End.
```

Contoh hasil yang akan diberikan dari program di atas adalah sebagai berikut.

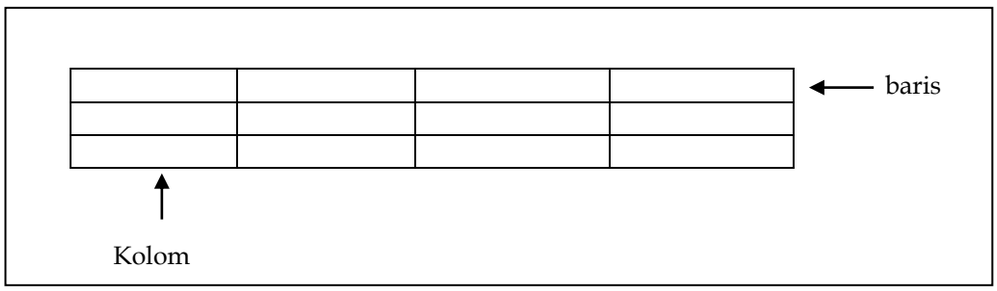
```
Masukkan banyaknya elemen array : 3

Memasukkan data :
Masukkan nilai A[1] : 10
Masukkan nilai A[2] : 20
Masukkan nilai A[3] : 30

Menampilkan data :
A[1] = 10
A[2] = 20
A[3] = 30
```

8) Array Multidimensi

Sejauh ini kita baru membahas mengenai array berdimensi satu. Pada bagian ini kita akan mempelajari bagaimana cara mendeklarasikan dan memanipulasi data yang terdapat pada array multidimensi. Agar lebih mempermudah pembahasan, materi ini akan dibagi menjadi dua kelompok, yaitu array dua dimensi dan array tiga dimensi.

9) Array Dua Dimensi

Array dua dimensi adalah array yang memiliki dua buah elemen bertipe array. Dengan kata lain, array dua dimensi memiliki dua buah subskrip, yang biasanya dipresentasikan dengan baris dan kolom. Untuk lebih memahami konsepnya, perhatikan terlebih dahulu gambar di bawah ini.



Gambar 1.5. Array dua dimensi (sumber:Rahardjo)

Pada gambar di atas, array memiliki 3 buah baris dan 4 buah kolom, sehingga jumlah elemennya adalah 3x4 = 12. Perlu diketahui bahwa keadaan tersebut sebenarnya hanya merupakan keadaan logik yang bertujuan untuk mempermudah pemahaman array dua dimensi. Pada kenyataannya, pengalamatan memori di komputer dari array dua dimensi tetap akan dipresentasikan dengan sebuah deretan larik yang memanjang (tidak berbentuk baris dan kolom). Pendeklarasian array dua dimensi di dalam bahasa Pascal dilakukan melalui bentuk umum di bawah ini.

```
NamaArray : array [1 . . banyakbaris, 1 . . banyakkolom] of tipe_data;
```

Sebagai contoh, apabila kita akan mendeklarasikan array dua dimensi dengan 3 buah baris dan 4 buah kolom dimana setiap elemennya bertipe integer, maka kita dapat menuliskan kode seperti berikut.

```
Array2D : array [1 . . 3, 1 . . 4] of integer;
```

Untuk melakukan pengaksesan terhadap elemen-elemennya, kita harus menuliskan indeks baris dan kolomnya. Sebagai contoh, apabila kita ingin mengisikan nilai 100 ke dalam elemen yang terdapat pada baris ke-2 kolom ke-3, maka kita harus menuliskannya sebagai berikut.

```
Array2D [2, 3] : = 100;
```

Berikut ini contoh program yang menunjukkan penggunaan array dua dimensi. Di sini kita akan membuat program yang dapat menjumlahkan dua buah matriks A dan B yang masing-masing berordo 2 x 3 (memiliki 2 baris dan 3 kolom).

```
Program JumlahMatriks;
Uses crt;
Const
Jbaris = 2;
Jkolom = 3;
```

```
Type
Matriks23 = array [1 . . Jbaris, 1 . . Jkolom] of integer;
Var
A,B,C : Matriks23;
j, k : integer;
begin
clrscr;
{mengisikan matriks A}
writeln (’Matriks A’);
for j: = 1 to Jbaris do begin
for k: = 1 to Jkolom do begin
write(’A[’, j, ’ , ’ , k, ’] = ,); readln(A[j, k]);
end;
writeln;
end;
writeln;
{mengisikan matriks B}
writeln(’Matriks B’);
for j: = 1 to Jbaris do begin
for k: = 1 to Jkolom do begin
write(’B[’, j, ’ , ’ , k, ’] = ,); readln(B[j, k]);
end;
writeln;
end;
writeln;
{melakukan penjumlahan matriks A dan B sekaligus menampilkan hasilnya ke layar}
writeln(’Hail Penjumlahan’);
for j: = 1 to Jbaris do begin
for k: = 1 to Jkolom do begin
C[j, k] : = A[j, k] + B[j, k];
write(’C[’, j, ’ , ’ , k, ’] = ,); readln(C[j, k]);
end;
writeln;
end;
readln;
end.
```

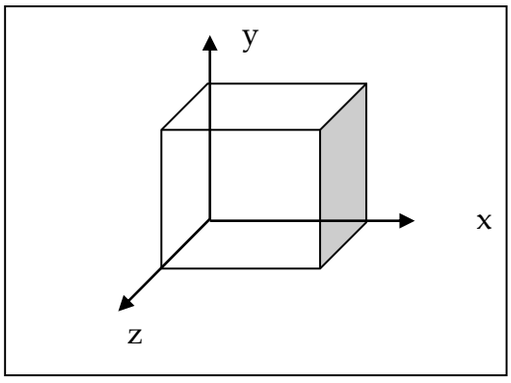
Contoh hasil yang akan diberikan oleh program di atas adalah sebagai berikut.

```
Matriks A
A[1, 1] = 1
A[1, 2] = 2
A[1, 3] = 3

A[2, 1] = 4
A[2, 2] = 5
A[2, 3] = 6

Matriks B
B[1, 1] = 3
B[1, 2] = 2
B[1, 3] = 1

B[2, 1] = 6
B[2, 2] = 5
B[2, 3] = 4


Hasil Penjumlahan
C[1, 1] = 4
C[1, 2] = 4
C[1, 3] = 4
```

```
C[2, 1] = 10
C[2, 2] = 10
C[2, 3] = 10
```

10) Array Tiga Dimensi

Array tiga dimensi merupakan array yang memiliki tiga buah subskrip dan lebih kompleks apabila dibandingkan dengan array dua dimensi. Di sini, subskrip yang ada akan dipresentasikan dengan sumbu x, y dan z atau panjang, lebar dan tinggi seperti yang ditunjukkan oelh gambar berikut.



Gambar 1.6. Array tiga dimensi (sumber : Rahardjo)

Berikut ini bentuk umum yang digunakan untuk mendeklarasikan array tiga dimensi di dalam bahasa Pascal.

```
NamaArray : array [1 . . xMaks, 1 . . yMaks, 1 . . zMaks] of tipe_data;
```

Gambar 1.7 Bentuk Umum Pendeklarasian Array Tiga Dimensi(sumber:Rahardjo)

Sebagai contoh apabila kita ingin mendeklarasikan array yang memiliki panjang 2, lebar 3 dan tinggi 4 elemen bertipe integer, maka kita akan menuliskannya seperti di bawah ini.

```
Array3D : array [1 . . 2, 1 . . 3, 1 . . 4] of integer;
```

Jumlah elemen yang terdapat di dalam array tersebut adalah 2 x 3 x 4 = 24. Sekarang apabila kita ingin mengisikan nilai 100 ke dalam elemen yang berada pada posisi panjang ke-2, lebar ke-3 dan tinggi ke-1, maka kita akan melakukannya melalui kode berikut.

```
Array3D : [2, 3, 1] : = 100;
```

Berikut ini adalah contoh yang akan menunjukkan cara pengaksesan elemen di dalam array 3 dimensi.

```
Program AksesArray3D;
Uses crt;
Const
xMaks = 2;
yMaks = 2;
zMaks = 2;
Type
Array3D = array [1 . . xMaks, 1 . . yMaks,1 . . zMaks] of integer;
```

```
Var
A: Array3D;
i, j, k : integer;
begin
clrscr;

{mengisikan nilai ke dalam array A}
x := 1;
for i := 1 to xMaks do begin
  for j := 1 to  jMaks do begin
    for k := 1 to zMaks do begin
      A[i, j, k] := x;
      inc(x);
    end;
  end;
end;

{menampilkan isi yang terdapat dalam array A}
for i := 1 to xMaks do begin
  for j := 1 to  jMaks do begin
    for k := 1 to zMaks do begin
      write('A[ ' , i,  ',' , j, ',' , z, ' ] = ',  A[ i, j, k] , ' ' );
    end;
  end;
  writeln;
end;
readln;
end.
```

Hasil yang akan diberikan oleh program diatas adalah sebagai berikut:

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

C. Penutup

Array merupakan sebuah variabel yang dapat menyimpan lebih dari satu nilai yang memiliki tipe data sama. Hal ini berbeda dengan variabel biasa yang hanya mampu menampung satu buah nilai. Setiap nilai yang disimpan di dalam array disebut dengan elemen array, sedangkan nilai urut yang digunakan untuk mengakses elemennya disebut dengan indeks array.

Apabila akan membuat program untuk menyimpan sekumpulan data, misalnya data-data hasil penelitian yang berupa bilangan, dimana jumlah dari data tersebut puluhan, ratusan atau bahkan ribuan, apakah akan menggunakan variabel sebanyak data yang ada? Jawabannya tentu tidak, karena hal tersebut merupakan hal yang sangat tidak efisien. Penggunaan array dalam program akan membuat program lebih efisien dan mudah dipahami.

LATIHAN

(a) Buat program dengan menggunakan array untuk menginput 10 nilai, bandingkan dan tampilkan nilai terbesar.

(b) Buat program dengan menggunakan array konstan untuk memasukkan nomor bulan dan menampilkan nama bulan.