Algoritma & Pemrograman Saintifik

Gatot F. Hertono, Ph.D

Departemen Matematika

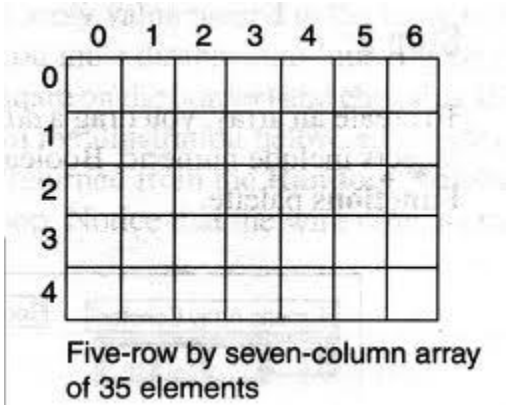
SCMA601401

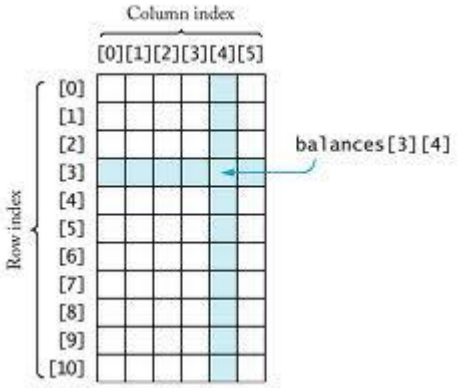
Definition

Arrays are "lists" of **related** values. Every value in the array is usually of the exact same type and *only differentiated by the **position*** in the array.

Source: http://www.cs.utah.edu/~germain/PPS/Topics/arrays.html

Examples:





$$\begin{bmatrix} 8 & 15 & -3 & 7 & 2 \\ -22 & -21 & 3 & -12 & 11 \\ 10 & 6 & 24 & 3 & -6 \\ 0 & -2 & 21 & 0 & 4 \\ 4 & 9 & 1 & -22 & -7 \end{bmatrix}$$

A

1	1000	30	-15	57	89	-25

PTN

UI	ITB	UGM	IPB	UNPAD	ITS	UNAIR

Index of an Array

Arrays have a single name, but many values.

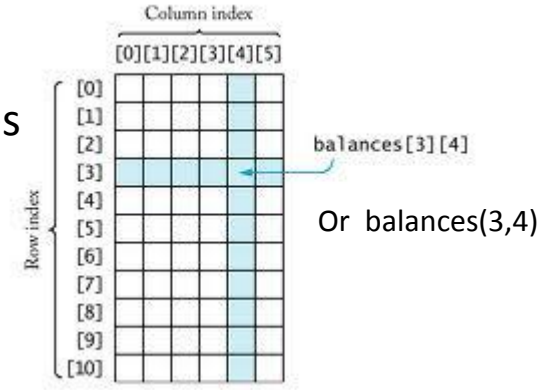
- ✓ How do we (the programmer) define which particular value we are interested in? (i.e. reference to a specific location in array) -> The answer is, we provide a numerical "index".
- ✓ The index into an array is a number representing which bucket is currently of interest.

Consider an array with the following 5 grades :

[100, 95, 99, 87, 90];

The **3rd value** in the array is 99.

balances



Or balances(3,4)

A

(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	1000	30	-15	57	89	-25

A(4) contains value -15

A(6) contains value 89

Notes: in C and Java arrays are indexed from 0, whereas Matlab is indexed from 1.

Creating and Assigning Value to Array

Direct

(in Matlab)

Arrays can be initialized using [] brackets:

grades = [99, 100, 50];

Create one-dimensional arrays of integer "grades" with 3 elements/components

MatrixA = [1.5, 2.0, 3.75; 0.5, -7.25, 80.111];

Create two-dimensional arrays of real numbers, MatrixA with 2 rows and 3 columns.

PTN = ['UI', 'ITB', 'UGM', 'IPB', 'UNAIR', 'ITS'];

Create one-dimensional arrays of string characters "grades" with 3 elements/components

Using Loop

```
for i=1:4
   for j=1:i
      A(i,j) = input('Elemen A:')
   end
end
```

```
A =
     1     0     0     0
     2     3     0     0
     4     5     6     0
     7     8     9    10
```

Referencing & Displaying an Element of Array

```
grades = ...;  % create an array of grades

for index = 1 to the length of the grades array
        print the grade at "index"  is grades(index)
end
```

```
%
% Sum up the values in a variabled named array_of_grades
%
total = 0;

for i = 1 : length(grades)  % the length function tells us the number of values in the array!!!
        total = total + grades(i);
end
```

Examples

```
% Finding the minimum and maximum elements in a list
%
L = [10, 23, -15, 125, 3, -43, 10, 125, -34];
n = length(L)    % compute number of elements in the list L

% initialization
MaxEl = -99999;
MinEl = 99999;

% Finding Max and Min elements
for i=1:n
    if (MaxEl <= L(i))
        MaxEl = L(i);
    end;
    if (MinEl >= L(i))
        MinEl = L(i);
    end;
end;

% Display Min and Max elements
disp('[Min, Max] = '), [MinEl, MaxEl]
```

Examples (cont.)

```
% Searching an element X in a list L
%
L = [10, 23, -15, 125, 3, -43, 10, 125, -34];
n = length(L);   % compute number of elements in the list L

% Initialization
Flag = 0;  % to denote X in L or not (0 = X is not in L, 1 = X in L)

% Input the element X that will be checked for the occurences in L
X = input('Masukkan sembarang nilai integer: ')

% Searching element X in L
for i=1:n
    if (X == L(i))
        Flag = 1;
    end;
end;

% Display the occurances of X in L
if (Flag == 0)
    disp('Elemen tidak ada pada List')
else
    disp('Elemen ada pada List')
end;
```

Examples (cont.)

```
% Create an upper triangular matrix containing ascending values in rows
%
% Determine the size of matrix
n = input('ukuran matriks: ')

% Initialization
A = zeros(n,n);    % create n x n matrix with zeros elements

% Input elements of the matrix in ascending order
for i=1:n
    for j=i:n
        A(i,j)=input('Elemen of A: ');
    end
end

% Display matrix A
A
```

```
A =

     1     2     3     4     5
     0     6     7     8     9
     0     0    10    11    12
     0     0     0    13    14
     0     0     0     0    15
```

Examples (cont.)

```
% Compute matrix multiplication C = A x B
%
% Determine matrices A(3 x 4) & B(4 x 2)
A = [3, -10, 5, -2; 1, 4, -2, 10; 3, 2, 9, -1];
B = [-25, 2; 7, 1; 2, 0; -4, -1];

% initialize matrix sized of A & B
rowA = size(A,1); colA = size(A,2);
rowB = size(B,1); colB = size(B,2);

% Compute C = A x B
% Check the size of matrices
if (colA == rowB)
   for i=1:rowA
      for j=1:colB
        C(i,j) = 0;
        for k=1:colA
            C(i,j) = C(i,j) + (A(i,k)*B(k,j));
        end;
      end;
   end;
   % display A, B, C
      A,B,C
else
    display('matrix sizes does not matched')
    A,B
end;
```

```
A =

   3  -10   5   -2
   1   4   -2   10
   3   2   9   -1

B =

  -25   2
   7    1
   2    0
  -4   -1

C =

 -127   -2
  -41   -4
  -39    9
```

Exercises

Bila diberikan suatu matrix *A* berukuran (10 x 10), buatlah algoritma/program untuk:

1. Membentuk matrik *lower triangular L* dan *upper triangular U* yang dibentuk dari matrik *A*, sedemikian sehingga: *A = L + U*

Contoh:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} + \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}.$$

2. Membentuk matrik tri-diagonal *TD* yang dibentuk dari matrik *A*

TD =

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{bmatrix}$$