Computer Programming: Creating Arrays

Asst. Prof. Dr. Yalçın İşler
Izmir Katip Celebi University

Outline

- Creating a one-dimensional array (Vector)
- Creating a two-dimensional array (Matrix)
- Variables – revisited
- Transpose
- Array addressing
- Adding and deleting elements from arrays
- Array functions
- Strings

Creating vectors

- Vector: One-dimensional array in either rows or columns.

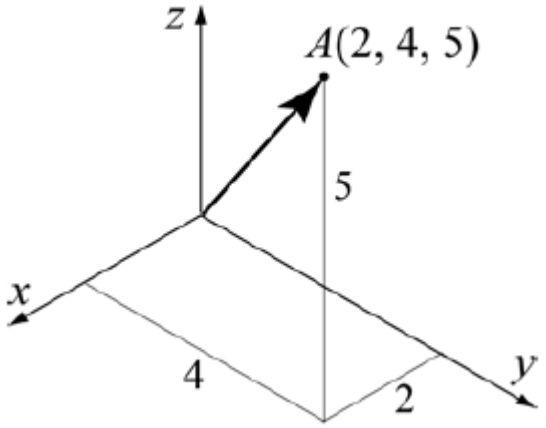
```
>> yr=[1984 1986 1988 1990 1992 1994 1996]

yr =
     1984     1986     1988     1990     1992     1994     1996

>> pop=[127;  130;  136;  145;  158;  178;  211]

pop =
   127
   130
   136
   145
   158
   178
   211

>>
```

The list of years is assigned to a row vector named `yr`.

The population data is assigned to a column vector named `pop`.

Year	1984	1986	1988	1990	1992	1994	1996
Population (Millions)	127	130	136	145	158	178	211

Creating vectors (cont'd)




```
>> pntAH=[2,   4,   5]
pntAH =
     2     4     5
>> pntAV=[2
4
5]
pntAV =
     2
     4
     5
>>
```

The coordinates of point *A* are assigned to a row vector called `pntAH`.

The coordinates of point *A* are assigned to a column vector called `pntAV`. (The **Enter** key is pressed after each element is typed.)

Creating a vector by constant increment

`variable_name = [m:q:n]` or `variable_name = m:q:n`

(The brackets are optional.)

```
>> x=[1:2:13]
x =
     1     3     5     7     9    11    13
```

First element 1, spacing 2, last element 13.

```
>> y=[1.5:0.1:2.1]
y =
   1.5000   1.6000   1.7000   1.8000   1.9000   2.0000   2.1000
```

First element 1.5, spacing 0.1, last element 2.1.

```
>> z=[-3:7]
z =
    -3    -2    -1     0     1     2     3     4     5     6
7
```

First element –3, last term 7.
If spacing is omitted, the default is 1.

```
>> xa=[21:-3:6]
xa =
    21    18    15    12     9     6
>>
```

First element 21, spacing –3, last term 6.

Creating a vector by constant increment

`variable_name = linspace(xi,xf,n)`

```
>> va=linspace(0,8,6)
va =
       0    1.6000    3.2000    4.8000    6.4000    8.0000
>> vb=linspace(30,10,11)
vb =
   30    28    26    24    22    20    18    16    14    12
10
>> u=linspace(49.5,0.5)

u =
  Columns 1 through 10
   49.5000   49.0051   48.5101   48.0152   47.5202   47.0253
46.5303   46.0354   45.5404   45.0455
.............
Columns 91 through 100
    4.9545    4.4596    3.9646    3.4697    2.9747    2.4798
1.9848    1.4899    0.9949    0.5000
>>
```

6 elements, first element 0, last element 8.

11 elements, first element 30, last element 10.

First element 49.5, last element 0.5.

When the number of elements is omitted, the default is 100.

100 elements are displayed.

Creating matrices

- Matrix (table): Two-dimensional array in both rows and columns.

```
>> a=[5   35   43;   4   76   81;   21   32   40]
a =
      5     35     43
      4     76     81
     21     32     40
>> b = [7   2   76   33   8
1   98   6   25   6
5   54   68   9   0]
b =
      7      2     76     33      8
      1     98      6     25      6
      5     54     68      9      0
>>
```

A semicolon is typed before a new line is entered.

The **Enter** key is pressed before a new line is entered.

Creating matrices (cont'd)

```
>> A=[1:2:11; 0:5:25; linspace(10,60,6); 67 2 43 68 4 13]
A =
     1     3     5     7     9    11
     0     5    10    15    20    25
    10    20    30    40    50    60
    67     2    43    68     4    13
>>
```

Some special matrices

```
>> zr=zeros(3,4)
zr =
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> ne=ones(4,3)
ne =
     1     1     1
     1     1     1
     1     1     1
     1     1     1
>> idn=eye(5)
idn =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
>>
```

Variables – revisited

- All variables in MATLAB are arrays. A scalar is an array with one element, a vector is an array with one row, or one column, of elements, and a matrix is an array with elements in rows and columns.
- The variable (scalar, vector, or matrix) is defined by the input when the variable is assigned. There is no need to define the size of the array before the elements are assigned.
- Once a variable exists, as a scalar, vector, or a matrix, it can be changed to be any other size, or type, of variable.

Transpose operator

```
>> aa=[3   8   1]
aa =
     3      8      1
>> bb=aa'
bb =
     3
     8
     1
>> C=[2 55 14 8; 21 5 32 11; 41 64 9 1]
C =
     2    55    14     8
    21     5    32    11
    41    64     9     1
>> D=C'
D =
     2    21    41
    55     5    64
    14    32     9
     8    11     1
>>
```

Define a row vector aa.

Define a column vector bb as the transpose of vector aa.

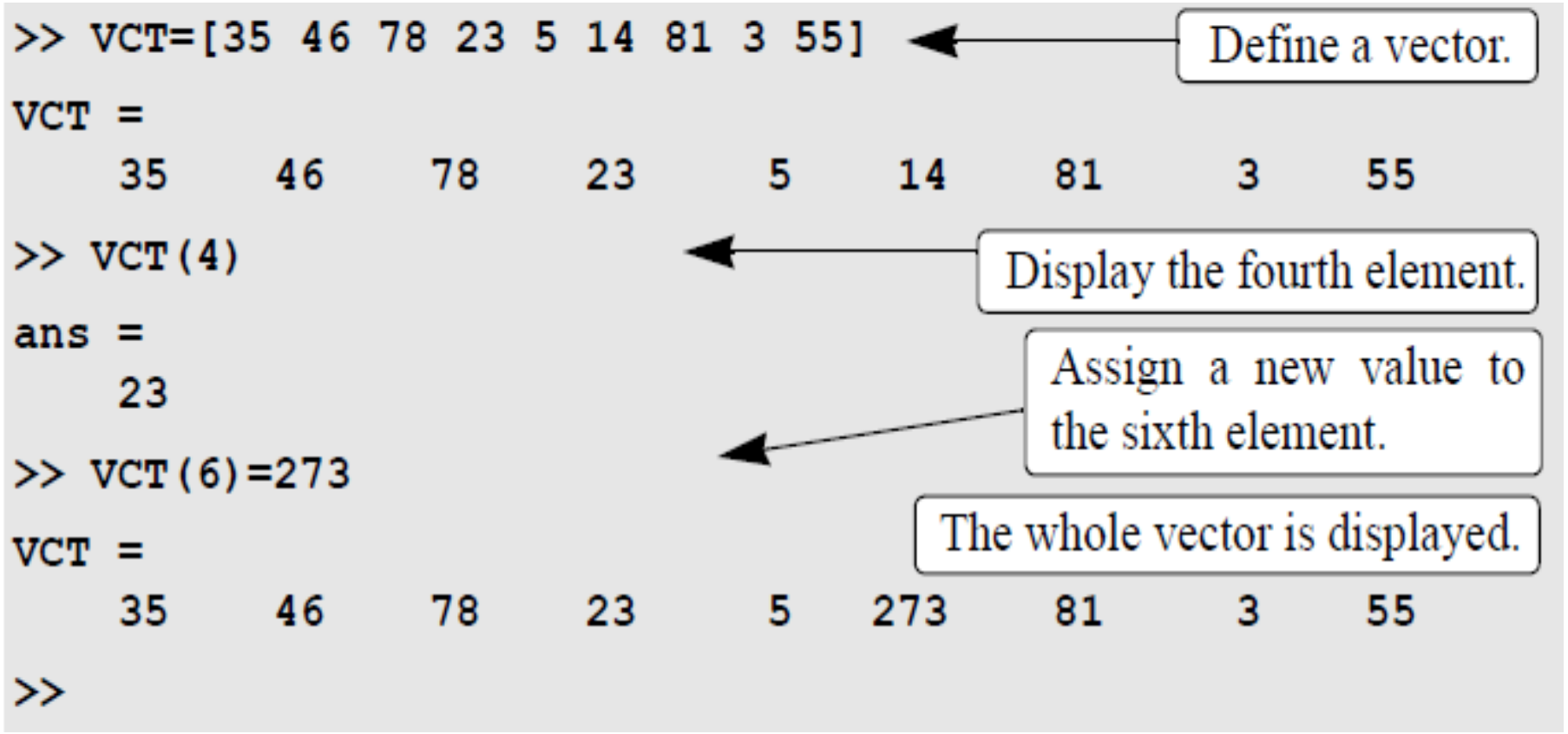
Define a matrix C with 3 rows and 4 columns.

Define a matrix D as the transpose of matrix C. (D has 4 rows and 3 columns.)

Array addressing

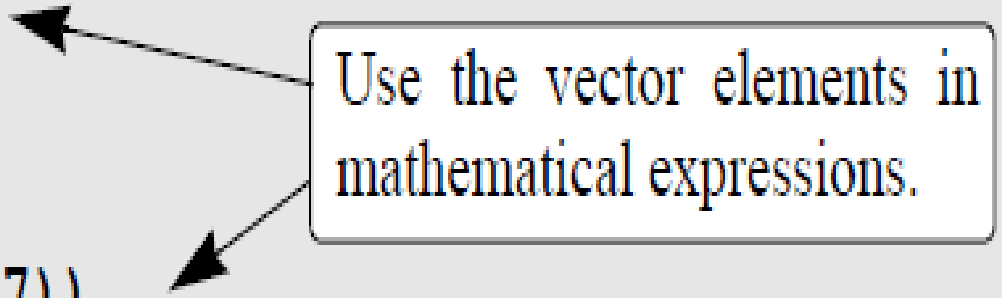
- The address of an element in a vector is its position in the row (or column). For a vector named v, v(*k) refers to the element in position k where the first position is* 1.
- For example, if the vector ***v*** *has nine elements: v = 35 46 78 23 5 14 81 3 55;* then *v(4) = 23, v(7) = 81, and v(1) = 35.*

Array addressing (cont'd)



Array addressing (cont'd)

```
>> VCT(2)+VCT(8)
ans =
     49

>> VCT(5)^VCT(8)+sqrt(VCT(7))

ans =
    134
>>
```



Array addressing (cont'd)

- The address of an element in a matrix is its position, defined by the row number and the column number where it is located. For a matrix assigned to a variable *m, m(k,p) refers to the element in row k and column p.*
- For example, if the matrix is ***m*** as seen in below; then, *m(1,1) = 3, and m(2,3) = 10.*

$$m = \begin{bmatrix} 3 & 11 & 6 & 5 \\ 4 & 7 & 10 & 2 \\ 13 & 9 & 0 & 8 \end{bmatrix}$$

Array addressing (cont'd)

```
>> MAT=[3 11 6 5; 4 7 10 2; 13 9 0 8]
MAT =
     3    11     6     5
     4     7    10     2
    13     9     0     8
>> MAT(3,1)=20
MAT =
     3    11     6     5
     4     7    10     2
    20     9     0     8
>> MAT(2,4)-MAT(1,2)
ans =
    -9
```

Create a 3×4 matrix.

Assign a new value to the (3,1) element.

Use elements in a mathematical expression.

Colon operator for array addressing

- **For a vector:**
  - *v(:) Refers to all the elements of the vector v (either a row or a column vector).*
  - *v(m:n) Refers to elements m through n of the vector v.*
- **For a matrix:**
  - *A(:,n) Refers to the elements in all the rows of column n of the matrix A.*
  - *A(n,:) Refers to the elements in all the columns of row n of the matrix A.*
  - *A(:,m:n) Refers to the elements in all the rows between columns m and n of the* matrix *A.*
  - *A(m:n,:) Refers to the elements in all the columns between rows m and n of the* matrix *A.*
  - A(*m:n,p:q) Refers to the elements in rows m through n and columns p through q of the matrix A.*

Array addressing (cont'd)

Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> v=[4 15 8 12 34 2 50 23 11]

v =

     4    15     8    12    34     2    50    23    11

>> u=v(3:7)

u =

     8    12    34     2    50

>> u=v([3 6:8])

u =

     8     2    50    23

fx >>
```

Array addressing (cont'd)

```
>> v=4:3:34
v =
     4     7    10    13    16    19    22    25    28    31    34
>> u=v([3,  5,  7:10])
u =
    10    16    22    25    28    31
>> A=[10:-1:4; ones(1,7); 2:2:14; zeros(1,7)]
A =
    10     9     8     7     6     5     4
     1     1     1     1     1     1     1
     2     4     6     8    10    12    14
     0     0     0     0     0     0     0
>> B = A([1,3],[1,3,5:7])
B =
    10     8     6     5     4
     2     6    10    12    14
```

Create a vector v with 11 elements.

Create a vector u from the 3rd, the 5th, and 7th through 10th elements of v.

Create a 4×7 matrix A.

Create a matrix B from the 1st and 3rd rows, and 1st, 3rd, and 5th through 7th columns of A.

Adding elements

```
>> DF=1:4
DF =
     1     2     3     4
>> DF(5:10)=10:5:35
DF =
     1     2     3     4    10    15    20    25    30    35
>> AD=[5   7   2]
AD =
     5     7     2
>> AD(8)=4
AD =
     5     7     2     0     0     0     0     4
>> AR(5)=24
AR =
     0     0     0     0    24
>>
```

Define vector DF with 4 elements.

Adding 6 elements starting with the 5th.

Define vector AD with 3 elements.

Assign a value to the 8th element.

MATLAB assigns zeros to the 4th through 7th elements.

Assign a value to the 5th element of a new vector.

MATLAB assigns zeros to the 1st through 4th elements.

Appending existing vectors

```
>> RE=[3   8 1   24];
>> GT=4:3:16;
>> KNH=[RE   GT]
KNH =
     3     8     1    24     4     7    10    13    16
>> KNV=[RE'; GT']
KNV =
      3
      8
      1
     24
      4
      7
     10
     13
     16
```

Define vector RE with 4 elements.

Define vector GT with 5 elements.

Define a new vector KNH by appending RE and GT.

Create a new column vector KNV by appending RE′ and GT′.

Adding elements to matrix

```
>> E=[1 2 3 4; 5 6 7 8]
E =
     1     2     3     4
     5     6     7     8
>> E(3,:)=[10:4:22]
E =
     1     2     3     4
     5     6     7     8
    10    14    18    22
>> K=eye(3)
K =
     1     0     0
     0     1     0
     0     0     1
>> G=[E   K]
G =
     1     2     3     4     1     0     0
     5     6     7     8     0     1     0
    10    14    18    22     0     0     1
```

Define a 2×4 matrix E.

Add the vector 10 14 18 22 as the third row of E.

Define 3×3 matrix K.

Append the matrix K to matrix E. The number of rows in E and K must be the same.

Adding elements

```
>> AW=[3 6 9; 8 5 11]
AW =
     3     6     9
     8     5    11
>> AW(4,5)=17
AW =
     3     6     9     0     0
     8     5    11     0     0
     0     0     0     0     0
     0     0     0     0    17
>> BG(3,4)=15
BG =
     0     0     0     0
     0     0     0     0
     0     0     0    15
>>
```

Define a 2×3 matrix.

Assign a value to the (4,5) element.

MATLAB changes the matrix size to 4×5, and assigns zeros to the new elements.

Assign a value to the (3,4) element of a new matrix.

MATLAB creates a 3×4 matrix and assigns zeros to all the elements except BG(3,4).

Deleting elements

```
>> kt=[2 8 40 65 3 55 23 15 75 80]
kt =
    2     8    40    65     3    55    23    15    75    80
>> kt(6)=[]
kt =
    2    8   40   65    3   23   15   75   80
>> kt(3:6)=[]
kt =
    2     8    15    75    80
```

Define a vector with 10 elements.

Eliminate the sixth element.

The vector now has 9 elements.

Eliminate elements 3 through 6.

The vector now has 5 elements.

Deleting elements (cont'd)

```
>> mtr=[5 78 4 24 9; 4 0 36 60 12; 56 13 5 89 3]

mtr =

     5    78     4    24     9
     4     0    36    60    12
    56    13     5    89     3

>> mtr(:,2:4)=[]

mtr =

     5     9
     4    12
    56     3

>>
```

Define a 3×5 matrix.

Eliminate all the rows of columns 2 through 4.

Array functions

Function	Description	Example

`length(A)`	Returns the number of elements in the vector `A`.	`>> A=[5 9 2 4];`
		`>> length(A)`
		`ans =`
		`4`

`size(A)`	Returns a row vector `[m,n]`, where `m` and `n` are the size $m \times n$ of the array `A`.	`>> A=[6 1 4 0 12; 5 19 6 8 2]`
		`A =`
		`6 1 4 0 12`
		`5 19 6 8 2`
		`>> size(A)`
		`ans =`
		`2 5`

`reshape(A, m, n)`	Rearrange a matrix `A` that has `r` rows and `s` columns to have `m` rows and `n` columns. `r` times `s` must be equal to `m` times `n`.	`>> A=[5 1 6; 8 0 2]`
		`A =`
		`5 1 6`
		`8 0 2`
		`>> B = reshape(A,3,2)`
		`B =`
		`5 0`
		`8 6`
		`1 2`

Array functions (cont'd)

Function	Description	Example

`diag(v)`	When `v` is a vector, creates a square matrix with the elements of `v` in the diagonal.	`>> v=[7 4 2];`
		`>> A=diag(v)`
		`A =`
		`7 0 0`
		`0 4 0`
		`0 0 2`

`diag(A)`	When `A` is a matrix, creates a vector from the diagonal elements of `A`.	`>> A=[1 2 3; 4 5 6; 7 8 9]`
		`A =`
		`1 2 3`
		`4 5 6`
		`7 8 9`
		`>> vec=diag(A)`
		`vec =`
		`1`
		`5`
		`9`

Strings

- A string is an array of characters. It is created by typing the characters within single quotes.
- Strings can include letters, digits, other symbols, and spaces.
- Examples of strings: 'ad ef ', '3%fr2', '{edcba:21!', 'MATLAB'.
- A string that contains a single quote is created by typing two single quotes within the string.
- When a string is being typed in, the color of the text on the screen changes to maroon when the first single quote is typed. When the single quote at the end of the string is typed the color of the string changes to purple.

Strings (cont'd)

```
>> a='FRty 8'

a =

FRty 8

>> B='My name is John Smith'

B =

My name is John Smith

>> B(4)

ans =

n

>> B(12)

ans =

J

>>
```

Strings (cont'd)

```
variable_ name = char('string 1','string 2','string 3')
```

```
>> Info=char('Student Name:','John Smith','Grade:','A+')
Info =
Student Name:
John Smith
Grade:
A+
>>
```

A variable named `Info` is assigned four rows of strings, each with different length.

The function `char` creates an array with four rows with the same length as the longest row by adding empty spaces to the shorter lines.

String versus Number

Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> x=123

x =

   123

>> y='123'

y =

123

>> 2*x

ans =

   246

>> 2*y

ans =

    98   100   102

>>
```

Laboratory Session after

Do sample applications in Chapter 2 of the textbook.

Homework #4

Not later than the next week:

Solve problems 1, 2, 6, 7, 8, 12, 14, 18, and 21 from the Chapter 2 of the textbook using Matlab.