Conditional Algorithms

Lecture 4 – Sunday November 6, 2016

Outline

- Conditional Algorithms
- Logical Data Type
- Branching Constructs
- Exercises

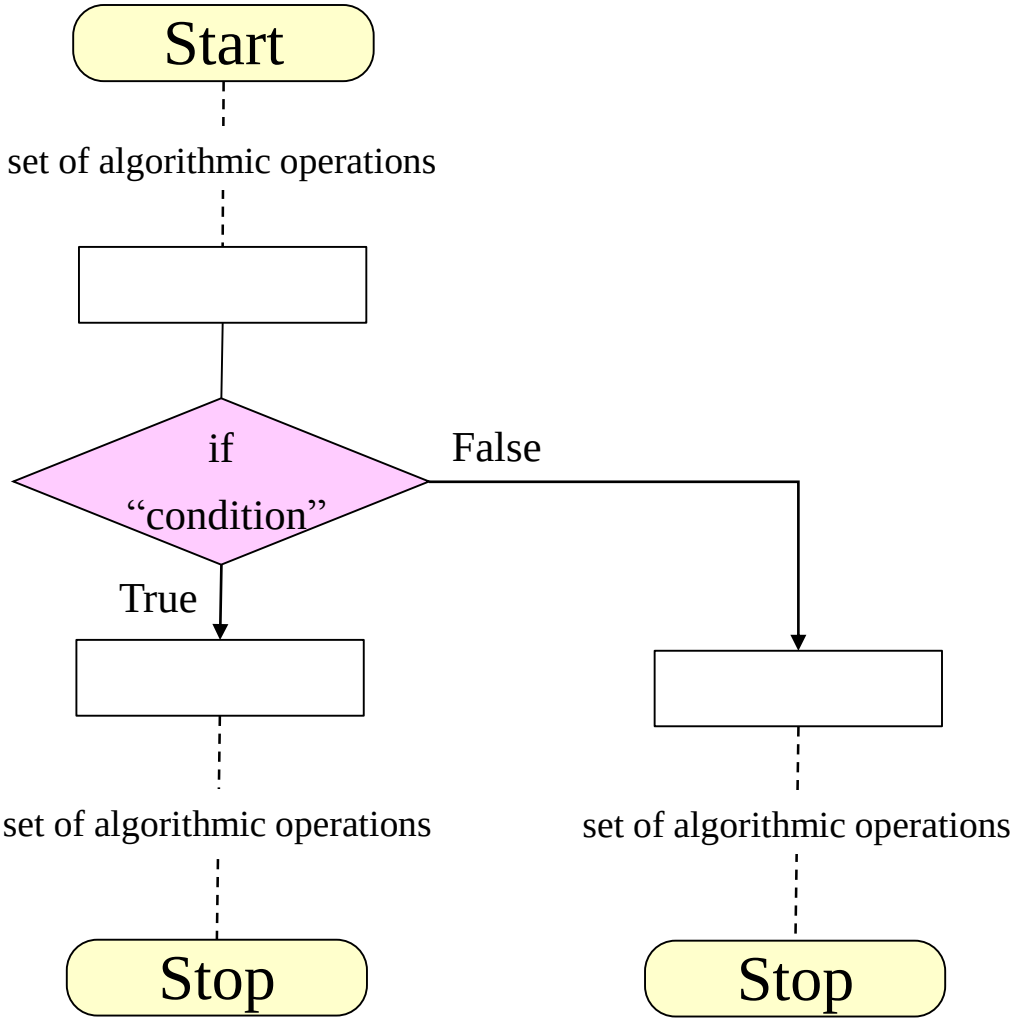
Outline

- **Conditional Algorithms**
- Logical Data Type
- Branching Constructs
- Exercises

Conditional Algorithms

Conditional operation is a control operation that allow us to alter the normal sequential flow of control in an algorithm.

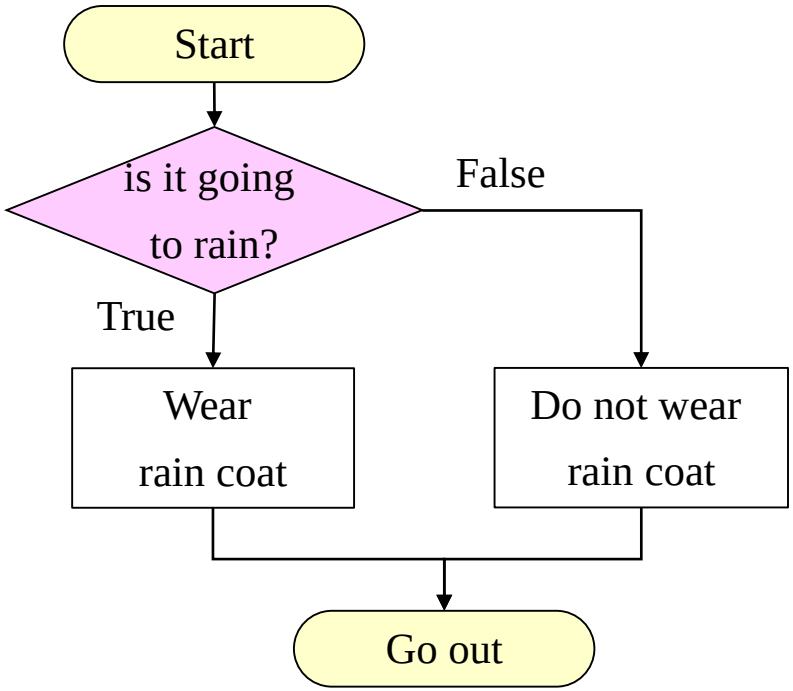
Conditional statements are the "**question-asking**" operations of an algorithm.



Conditional Algorithms

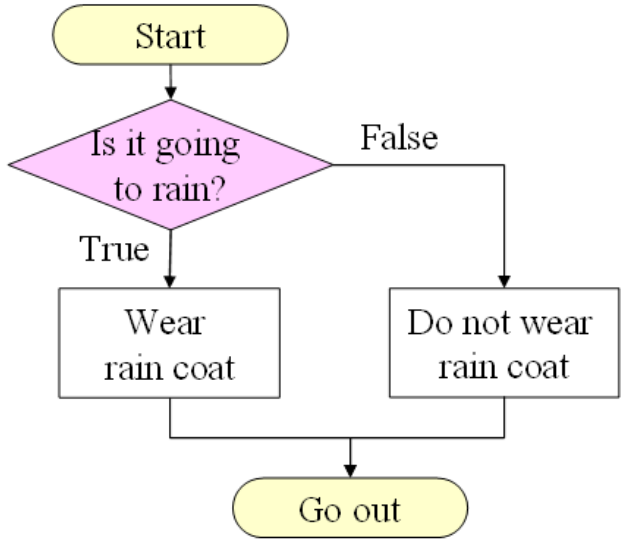
Let's say you are going out tomorrow. You need to make a decision on what to wear depending on the **weather forecast**. If the weather is rainy, then you will want to wear your rain-coat. Otherwise, you will not bother. Therefore, your decision will be based on a certain condition

- Is the weather going to be rainy?
- If the answer is yes then, wear the raincoat.
- If the answer is no then no raincoat is needed.



Conditional Algorithms

We can express this situation, using the **if-statement** in the following manner:



Flow chart

```
if (weather = “rainy”) then
    print “Wear rain coat”
else
    print “Do not wear rain coat”
endif
```

Pseudo-code

```
if weather == ‘rainy’
   display (‘Wear rain coat ‘);
else
   display (‘Do not wear rain coat‘);
end
```

Matlab code

Outline

- Conditional Algorithms
- **Logical Data Type**
- Branching Constructs
- Exercises

Logical Data Type

The **logical** data type is a special type of data that can have once of only two possible values: **true** or **false**.

To create a logical variable **a1** containing the logical value **true**:

```
a1=true;
```

```
>>whos a1
```

Name	Size	Bytes	Class
a1	1x1	1	logical array

Logical Data Type

- **Relational Operators**

a1 op a2

Operator	Operation
==	Equal to
~=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Operation	Result
3<4	true (1)
3<=4	true (1)
3==4	false (0)
3>4	false (0)
4<=4	true (1)
'A'<'B'	true (1)

Logical Data Type

- **Logical Operators**

l1 op l2

Operator	Operation
&	Logical AND
&&	Logical AND with shortcut evaluation
|	Logical OR
||	Logical OR with shortcut evaluation
xor	Logical Exclusive OR
~	Logical NOT

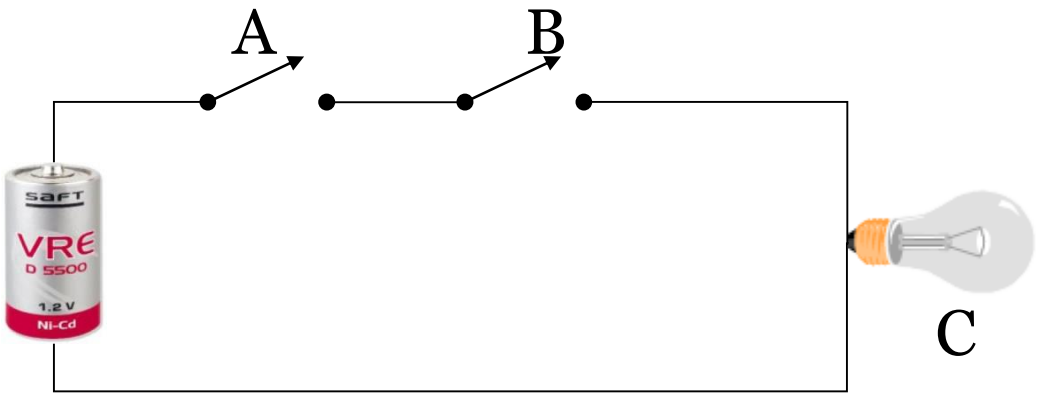
Returns ones where either A or B is True (nonzero); returns False (zero) where both A and B are False (zero) or both are True (nonzero).

Logical Data Type

- **Logical Operators**
  - ◇ Logical AND
  - ◇ Logical OR
  - ◇ Logical Negation (NOT).

Binary Operators: Logical AND

- Consider two logic variables A and B and the result is C.
- C is true if and only if A is true AND B is true



In order for current to flow, both switches must be closed

C = A.B

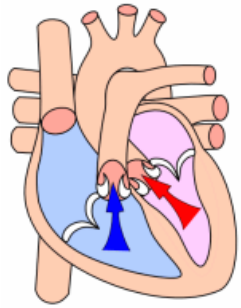
Truth Table

Inputs		Output
A	B	C=A && B
0	0	0
0	1	0
1	0	0
1	1	1

Binary Operators: Logical AND

- (A AND B) yields true only if both A and B are true.

Example: Adult blood pressure is considered normal at 120/80 where the first number is the systolic pressure and the second is the diastolic pressure.



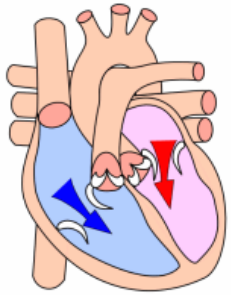
Systolic Pressure

A = True if Systolic Pressure = 120

B = True if Diastolic Pressure = 80

C = True $\Rightarrow$ Blood pressure is normal

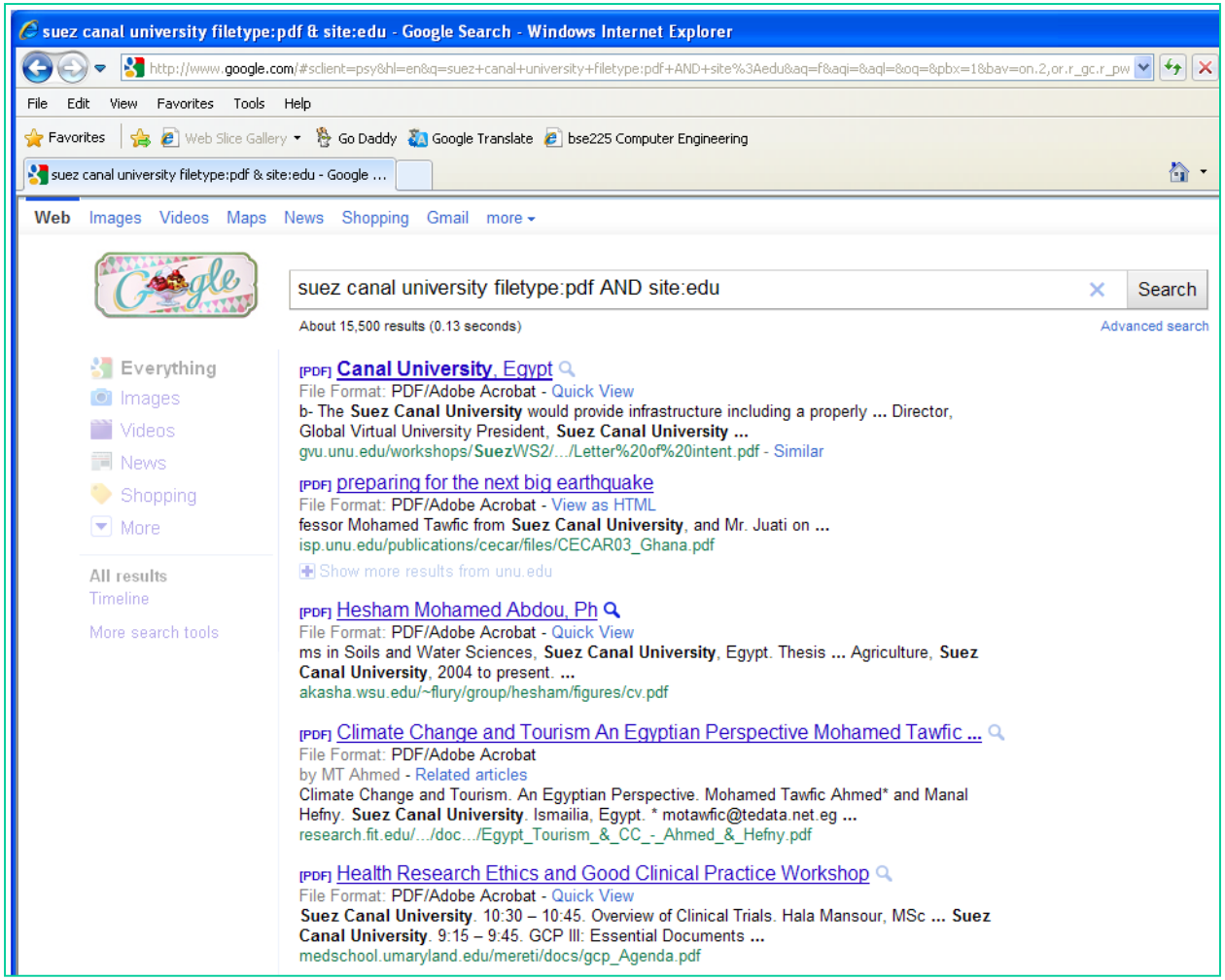
C = A && B



Diastolic Pressure

Binary Operators: Logical AND

- Example-2: Google Search



operator:parameter

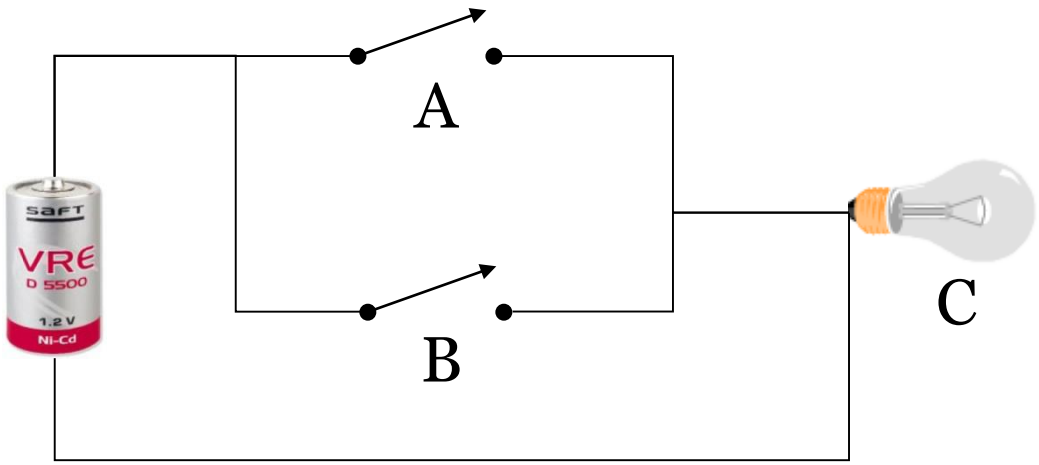
filetype:pdf will search for pdf only.

site:edu will search all site in edu top domain.

suez canal university filetype:pdf AND site:edu will search for pdf and all site in edu top domain.

Binary Operators: Logical OR

- Consider two logic variables A and B and the result is C.
- C is true if A is true OR B is true



Current flows if either switch is closed

C = A || B

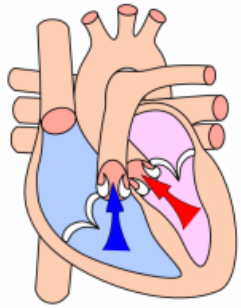
Truth Table

Inputs		Output
A	B	C = A || B
0	0	0
0	1	1
1	0	1
1	1	1

Binary Operators: Logical OR

- (A OR B) yields true only if either A or B, or both are true.

Example: Adult blood pressure is considered normal at 120/80 where the first number is the systolic pressure and the second is the diastolic pressure.



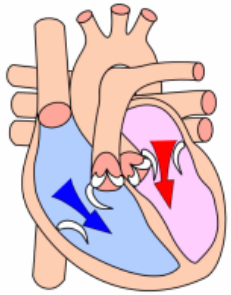
Systolic Pressure

A = True if Systolic Pressure ≠ 120

B = True if Diastolic Pressure ≠ 80

C = True ⇒ Blood pressure is abnormal

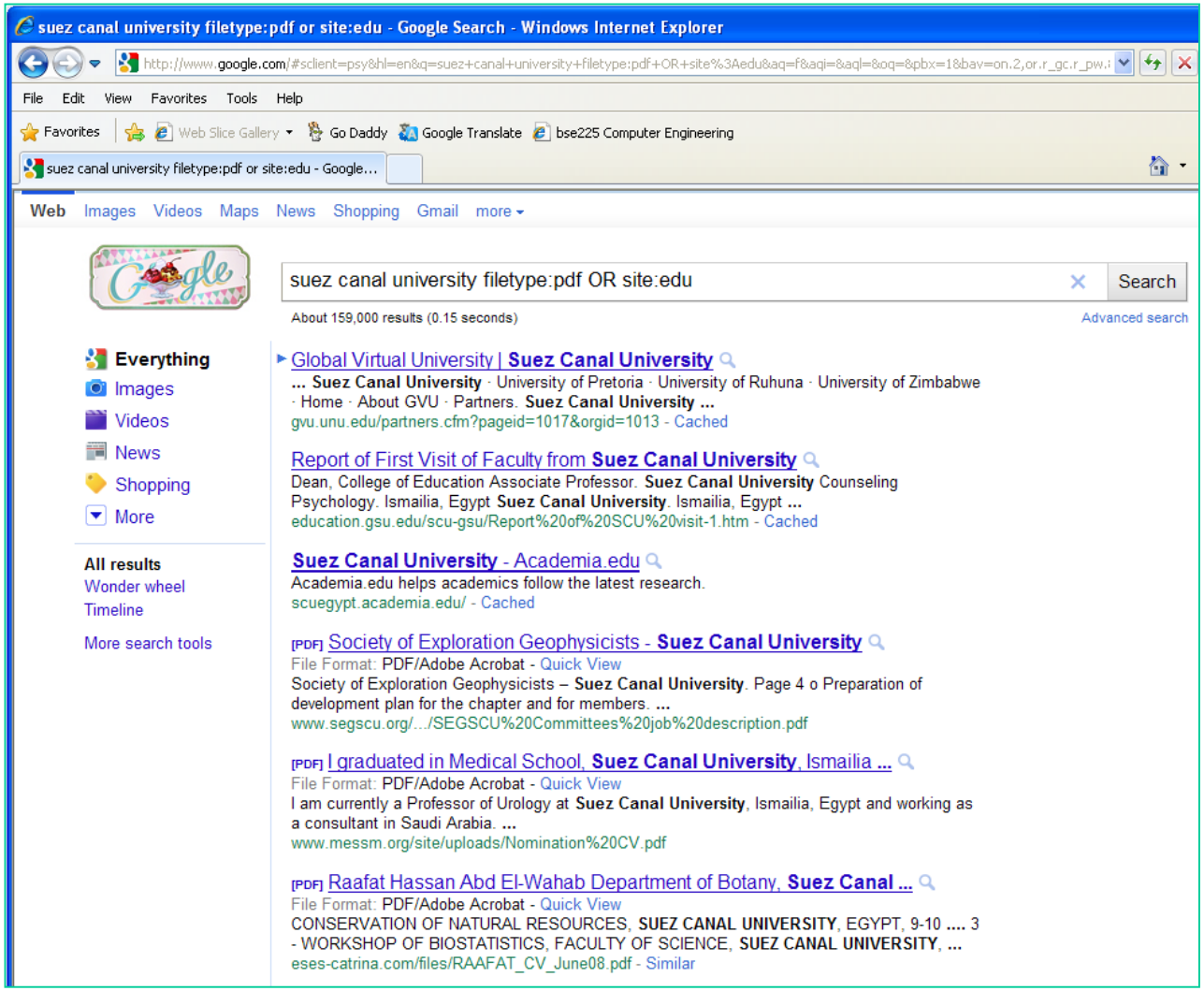
C = A || B



Diastolic Pressure

Binary Operators: Logical OR

- Example-2: Google Search



operator:parameter

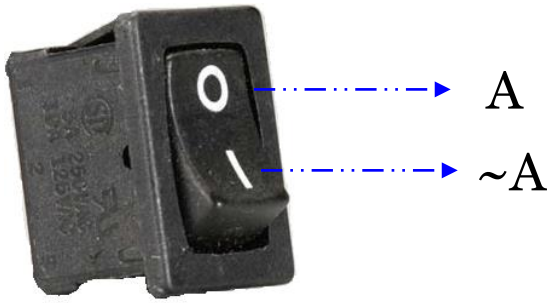
filetype:pdf will search for pdf only.

site:edu will search all site in edu top domain.

suez canal university filetype:pdf OR site:edu will search for pdf or all site in edu top domain.

Binary Operators: Logical Negation

- NOT is denoted by a bar (~) before the variable.
- Consider a logic variable A and the result is C.
- C is true if A is false and vice versa.
- C = ~ A



Truth Table

Input	Output
A	C = ~A
0	1
1	0

Binary Operators: Logical Negation

- Inverts its operand.

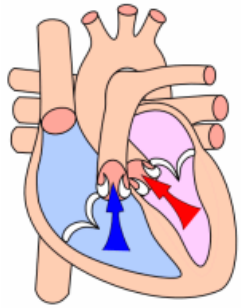
Example: Adult blood pressure is considered normal at 120/80 where the first number is the systolic pressure and the second is the diastolic pressure.

A = True if Systolic Pressure = 120

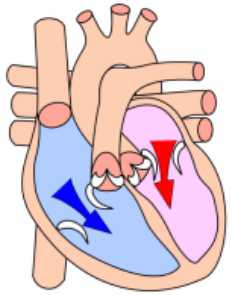
~A = False $\Rightarrow$ Systolic Pressure $\neq$ 120

B = True if Diastolic Pressure = 80

~B = False $\Rightarrow$ Diastolic Pressure $\neq$ 80



Systolic Pressure



Diastolic Pressure

Logical Data Type

- Boolean Expressions:

  C = A && B is read "C is equal to A and B."

  z = x || y is read "z is equal to x OR y."

  D = ~A is read "D is equal to NOT A."

- Using the basic operations, we can form more complex expressions:

$$Z = (A \&\& B \,||\, \sim C) \,||\, X \;\&\& Y$$

If A=True, B=False, C=True, X=True, Y=False.

Z=(True && False ||True) || False && False

=(False || True) || False=True || False=True

Logical Data Type

- **Example**

Assuming that x=-10, y=50, and z=60 determine the value of the following Boolean expression:

(0<x<50)AND(50<y<100)OR([y-x]=z)

(0<x<50) ⇒ (0<[-10]<50) ⇒ FALSE

(50<y<100) ⇒ (50<50<100) ⇒ FALSE

([50-(-10)]=60) ⇒ (60=60) ⇒ TRUE

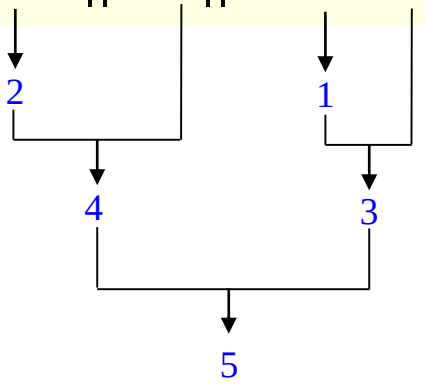
FALSE AND FALSE ⇒ FALSE

FALSE OR TRUE ⇒ TRUE

Logical Data Type

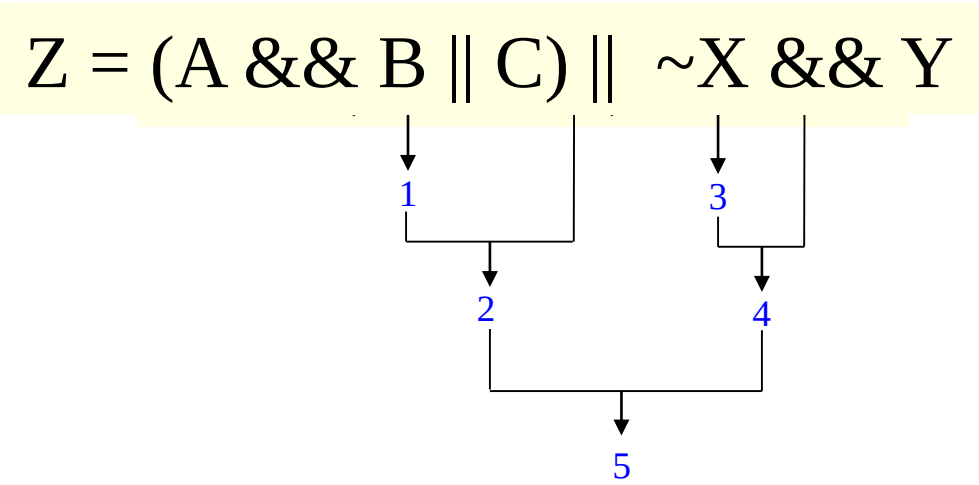
- **Operator Precedence**
  - ◇ NOT has the highest precedence, followed by AND, and then OR.
  - ◇ All higher-precedence operators are evaluated before any lower-precedence operators.
  - ◇ Operators at the same precedence are evaluated left-to-right.

$$Z = A \;\&\&\; B \;||\; C \;||\; \sim\overline{X} \;\&\&\; Y$$



Logical Data Type

- **Operator Precedence**
  - ◇ Parentheses can be used to override operator precedence.



Logical Data Type

- **Operator Precedence**

1. All arithmetic operators are evaluated first.
2. All relational operators (==, ~=, >, >=, <, <=) are evaluated, working from left to right.
3. All ~ operators are evaluated.
4. All & and && operators are evaluated, working from left to right.
5. All | , | | , and xor operators are evaluated, working from left to right.

Logical Data Type

- **Operator Precedence: Example**

  Assume that the following variables are initialized with the values shown, and calculate the result of the specified expressions:

  valuel = true

  value2 = false

  value3 = 1

  value4 = -10

  value5 = 0

  va1ue6 = [1 2; 0 1 l

Logical Data Type

- **Operator Precedence: Example**

	Expression	Result	Comment
(*a*)	`~value1`	`false`	
(*b*)	`~value3`	`false`	The number 1 is converted to `true` before operation is applied.
(*c*)	`value1 | value2`	`true`	
(*d*)	`value1 & value2`	`false`	
(*e*)	`value4 & value5`	`false`	−10 is converted to `true` and 0 is converted to `false` before the operation is applied.
(*f*)	`~(value4 & value5)`	`true`	−10 is converted to `true` and 0 is converted to `false` before the operation is applied.
(*g*)	`value1 + value4`	−9	`value1` is converted to the number 1 before the addition is performed.
(*h*)	`value1 + (~value4)`	1	The logical `value1` is converted to the number 1 before the addition is performed. The number `value4` is converted to `true` before the NOT is performed. Then `~value4` is evaluated to be `false`. This `false` value is converted to 0 before the addition, so the final result is 1 + 0 = 1.
(*i*)	`value3 && value6`	Illegal	The `&&` operator must be used with scalar operands.
(*j*)	`value3 & value6`	$\begin{bmatrix} \texttt{true} & \texttt{true} \\ \texttt{false} & \texttt{true} \end{bmatrix}$	AND between a scalar and an array operand.

valuel = true

value2 = false

value3 = 1

value4 = -10

value5 = 0

va1ue6 = [1 2; 0 1]

Logical Data Type

- **Logical Functions**

Function	Purpose
ischar(a)	Returns true if a is a character array and false otherwise.
isempty(a)	Returns true if a is an empty array and false otherwise.
isinf(a)	Returns true if the value of a is infinite (Inf) and false otherwise.
isnan(a)	Returns true if the value of a is NAN (not a number) and false otherwise.
isnumeric(a)	Returns true if a is a numeric array and false otherwise.
logical	Converts numerical values to logical values: if a value is non-zero, it is converted to true. If it is zero, it is converted to false.

Logical Data Type

- **Exercise**

  Assume that a, b,c, and d are defined, and evaluate the following expressions:

  a=20; b=-2;

  C=0; d=1;

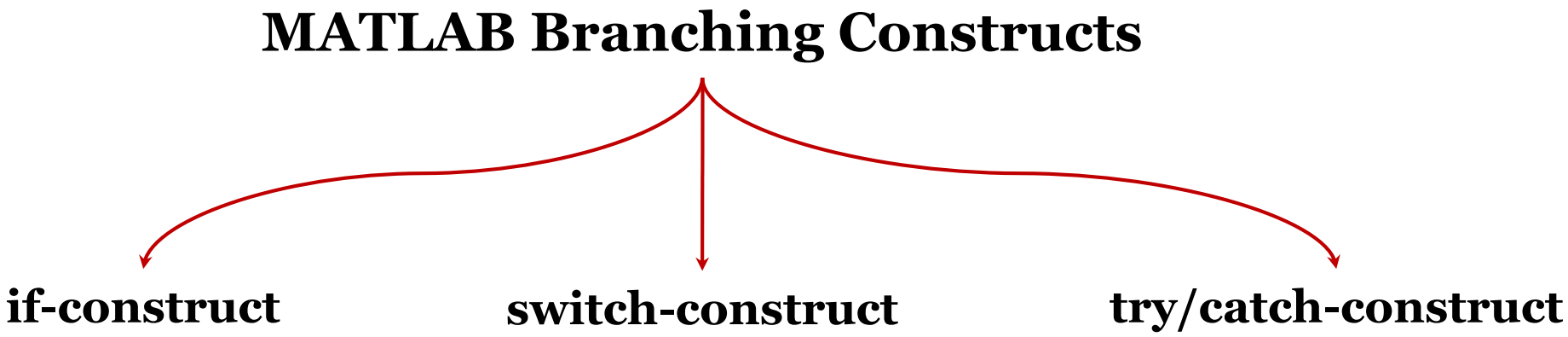
  1. a>b && c>d
  2. d || b > a
  3. a+b^2 > a*c

Outline

- Conditional Algorithms
- Logical Data Type
- **Branching Constructs**
- Exercises

Branching Constructs

- Branches are MATLAB statements that permit us to select and execute specific sections of code (called *blocks*) while skipping other sections of code.
- They are variations of the **if** construct, the **switch** construct, and the **try/catch** construct.



Branching Constructs

- **The if Construct**

```
if control_expr_1
    Statement 1
    Statement 2      } Block 1
    ....
elseif control_expr_2
    Statement 1
    Statement 2      } Block 2
    ....
else
    Statement 1
    Statement 2      } Block 3
    ....
end
```

Branching Constructs

- **The if Construct: Example-0**

What will the following MATLAB code print?

```
a = 10;
if a ~= 0
    disp('a is not equal to 0')
end
```

Solution

a is not equal to 0

Branching Constructs

- **The if Construct: Example-0**

What will the following MATLAB code print?

```
a = 10;
if a > 0
    disp('a is positive')
else
    disp('a is not positive')
end
```

Solution

a is positive

Branching Constructs

- **The if Construct: Example-0**

What will the following MATLAB code print?

```
a = 10;
if (a > 0)
   disp('a is positive')
else
   disp('a is not positive')
end
```

Solution

The parentheses around the relational expression a > 0 will not change its validity, so this code will print 'a is positive'.

Branching Constructs

- **The if Construct: Example-0**

What will the following MATLAB code print?

```
a = 5;
b = 3;
c = 2;
if a < b*c
    disp('Hello world')
else
    disp('Goodbye world')
end
```

Solution

b*c gives a value of 6, and 5 < 6, so this code will print 'Hello world'.

Branching Constructs

- **The if Construct: Example-o**

What will the following MATLAB code print?

```
a = 5;
b = 3;
c = 2;
if (a < b)*c
    disp('Hello world')
else
    disp('Goodbye world')
end
```

Solution

The parentheses in this expression change its meaning completely. First, a < b is evaluated, and since it is false for the given values of a and b, it evaluates to zero. The zero is than multiplied by c, giving a value of zero which is interpreted by MATLAB as false. So this code prints 'Goodbye world‘.

Branching Constructs

- **The if Construct: Example-0**

What will the following MATLAB code print?

```
p1 = 3.14;
p2 = 3.14159;
if p1 == p2
    disp('p1 and p2 are equal')
else
    disp('p1 and p2 are not equal')
end
```

Solution

p1 and p2 are not equal

Branching Constructs

- **The if Construct: Example-0**

What will the following MATLAB code print?

```
a = 5;
b = 10;
if a = b
   disp('a and b are equal')
else
   disp('a and b are not equal')
end
```

Solution

This code will generate an error message, since a = b assigns the value of b to a. To check if a and b are equal, use a == b

Branching Constructs

- **The if Construct: Example-0**

For what values of the variable a will the following MATLAB code print 'Hello world'?

```
if ~ a == 0
    disp('Hello world')
else
    disp('Goodbye world')
end
```

Solution

Any value that is not zero.

Branching Constructs

- **The if Construct: Example-o**

For what values of the variable a will the following MATLAB code print 'Hello world'?

```
if a < 7 || a >= 3
    disp('Hello world')
else
    disp('Goodbye world')
end
```

Solution

Every value of a will print 'Hello world'.

Branching Constructs

- **The if Construct: Example-0**

  Write an if statement that will print 'a is very close to zero' if the value of the variable a is between -0.01 and 0.01.

```
if a >= -0.01 && a <= 0.01
    disp('a is very close to zero')
end
```

Branching Constructs

- **The if Construct: Example-1**

Write an algorithm in pseudo-code to evaluate a function *f(x,y)* for any two user-specified values *x* and *y*. The function *f(x,y)* is defined as follows:

$$f(x,y)=\begin{cases} x+y & x\geq 0 \text{ and } y\geq 0 \\ x+y^2 & x\geq 0 \text{ and } y<0 \\ x^2+y & x<0 \text{ and } y\geq 0 \\ x^2+y^2 & x<0 \text{ and } y<0 \end{cases}$$

Implement this algorithm using MATLAB.

1. Prompting for input of x and y;
2. Format the display of the computed *f(x,y)*;

Branching Constructs

- **The if Construct: Example-1**

Inputs: x, y

Output: $f(x,y) \Rightarrow fun$

Expression:

$$fun = \begin{cases} x + y & x \geq 0 \text{ and } y \geq 0 \\ x + y^2 & x \geq 0 \text{ and } y < 0 \\ x^2 + y & x < 0 \text{ and } y \geq 0 \\ x^2 + y^2 & x < 0 \text{ and } y < 0 \end{cases}$$

Branching Constructs

- **The if Construct: Example-1**

Algorithm in Pseudo-code

BEGIN

get x and y

if x≥0 and y≥0 then

set *fun* to $x+y$

elseif x≥0 and y<0 then

set *fun* to $x+y^2$

elseif x<0 and y ≥ 0 then

set *fun* to x^2+y

elseif x<0 and y<0 then

set *fun* to x^2+y^2

endif

print *fun*

END

Branching Constructs

- **The if Construct: Example-1**

```
%Script file: funxy.m
%
% Purpose:
% This program solves the function f(x,y) for a
% user-specified x and y, where f(x,y) is defined as:
% f(x,y)=x+y    x>=0 and y>=o
% f(x,y)=x+y^2  x>=0 and y<o
% f(x,y)=x^2+y  x<0 and y>=o
% f(x,y)=x^+y^  x<0 and y<o

%Record of revisions:
%Date       Programmer      Description of change
%=====      ==========      =====================
%01/03/2004 S. Chapman     Orignial Code
%
%Define variables:
% x:    First independent variable
% y:    Second independent variable
% fun:  Resulting function

%Prompt the user for the values of x and y
x=input('Enter the x coefficient:');
y=input('Enter the y coefficient:');

%Calculate the function f(x,y) based upon the signs of x and y
if x>=0 && y >=0
    fun=x+y;
elseif x >= 0 && y < 0
    fun=x+y^2;
elseif x < 0 && y >= 0
    fun=x^2+y;
else   %x< 0 and y <0
    fun=x^2+y^2;
end

%Write the results
display(['The value of the function is ' num2str(fun)])
```

```
>>
>> funxy
Enter the x coefficient:2
Enter the y coefficient:3
The value of the function is 5
>> funxy
Enter the x coefficient:2
Enter the y coefficient:-3
The value of the function is 11
>> funxy
Enter the x coefficient:-2
Enter the y coefficient:3
The value of the function is 7
>> funxy
Enter the x coefficient:-2
Enter the y coefficient:-3
The value of the function is 13
```

Branching Constructs

- **The if Construct: Example-2**

Write an algorithm in pseudo-code that outputs the grade of the student according to the following rules:

- Grade “A” if grade>95

- Grade “B” if 86< grade ≤ 95

- Grade “C” if 76< grade ≤ 86

- Grade “D” if 66< grade ≤ 76

- Grade “F” if grade ≤ 66

Write a Matlab program to implement the proposed algorithm.

Branching Constructs

- **The if Construct: Example-2**

  Inputs: numerical grade

  Output: letter grade

```
BEGIN
get grade
 if (grade>95) then
    print A
else if (grade>86) then
    print B
else if (grade>76) then
    print C
else if (grade>66) then
    print D
else
    print F
endif
END
```

Branching Constructs

- **The if Construct: Example-2**

```
%Script file: letter_grade.m
%
% Purpose:
% This program reads in a numerical grade and assigns
% a letter grade to it according to the following table:
% Grade "A" if grade>95
% Grade "B" if 86< grade <= 95
% Grade "C" if 76< grade <= 86
% Grade "D" if 66< grade <= 76
% Grade "F" if grade <= 66

%Record of revisions:
%Date       Programmer      Description of change
%=====      ==========      =====================
%16/05/2011 Alaa Khamis     Orignial Code
%
%Define variables:
% grade:  Numerical Grade

%Prompt the user for the numericla grade
grade=input('Enter the numerical grade:');

%Print the corresponding letter grade
if grade > 95.0
    display('The garde is A.');

elseif grade > 86.0
    display('The garde is B.');

elseif grade > 76.0
    display('The garde is C.');

elseif grade > 66.0
    display('The garde is D.');

else   %grade <= 66
    display('The garde is F.');
end
```

```
Enter the numerical grade:55
The garde is F.
>> letter_grade
Enter the numerical grade:99
The garde is A.
>> letter_grade
Enter the numerical grade:77
The garde is C.
>> letter_grade
Enter the numerical grade:89
The garde is B.
>> letter_grade
Enter the numerical grade:70
The garde is D.
```

Branching Constructs

- **The if Construct: Example-3**

Both color monitor and your eyes use just three colors-red, blue and green- to create all other colors. In particular, yellow is made by combining red and green, magenta (a shade of purple) by combing red and blue, and cyan by combing green and blue. Write a Matlab program that asks the user which of the three colors- yellow, magenta, or cyan- to break down into its two components.

If the user enters the letter 'Y' (or 'y') for "yellow", the following message is displayed:

Yellow is made by combining red and green.

Similarly, if the user enters the letter 'M' (or 'm') for "magenta", the following message is displayed:

Magenta is made by combining red and blue.

and if the letter 'C' (or 'c') for "cyan" is entered, the following message is displayed:

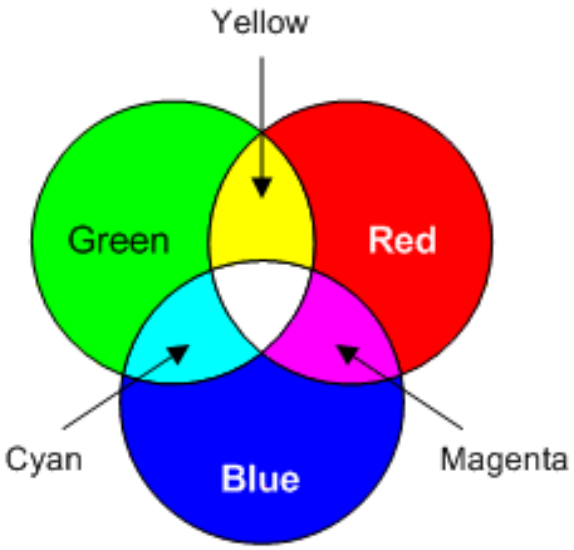
Cyan is made by combining green and blue.

Allow either uppercase or lower case, but if anything other than 'Y', 'y', 'M', 'm', 'C', or 'c' is entered, print an error message.

Branching Constructs

- **The if Construct: Example-3**

Both color monitor and your eyes use just three colors-red, blue and green- to create all other colors. In particular, yellow is made by combining red and green, magenta (a shade of purple) by combing red and blue, and cyan by combing green and blue. Write a Matlab program that asks the user which of the three colors- yellow, magenta, or cyan- to break down into its two components.



If the user enters the letter 'Y' (or 'y') for "yellow", the following message is displayed:

Yellow is made by combining red and green.

Similarly, if the user enters the letter 'M' (or 'm') for "magenta", the following message is displayed:

Magenta is made by combining red and blue.

and if the letter 'C' (or 'c') for "cyan" is entered, the following message is displayed:

Cyan is made by combining green and blue.

Allow either uppercase or lower case, but if anything other than 'Y', 'y', 'M', 'm', 'C', or 'c' is entered, print an error message.

Branching Constructs

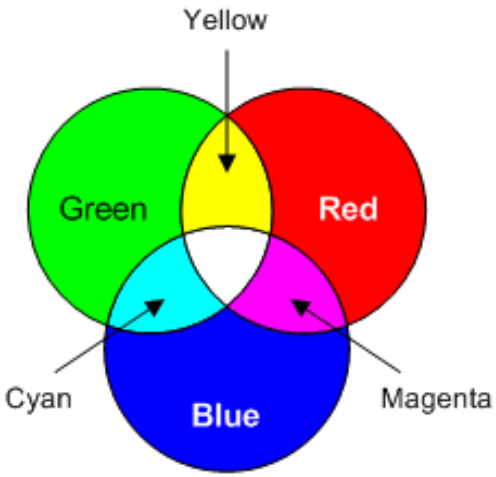
- **The if Construct: Example-3**

Inputs:

color_letter

Output:

message




```
BEGIN
get color_letter
if (color_letter =Y or y)  then
  print "Yellow is made by combining red and green"
  else if (color_letter =M or m)  then
     print "Magenta is made by combining red and blue"
        else if (color_letter =C or c)  then
          print "Cyan is made by combining green and blue"
        else
          print "ERROR!"
        endif
    endif
endif
END
```

Branching Constructs

- **The if Construct: Example-3**

```
% Script file: color.m
%
% Purpose:
% This program asks the user which of the three colors- yellow,
% magenta, or cyan- to break down into its two components.
% Yellow is made by combining red and green.
% Magenta is made by combining red and blue.
% Cyan is made by combining green and blue.

%Record of revisions:
%Date       Programmer      Description of change
%=====      ==========      =====================
%16/05/2011 Alaa Khamis     Orignial Code
%
%Define variables:
% color_letter:    Color to be decomposed into primary components

%Prompt the user for the color (read it as a string)
color_letter=input('Enter the color:', 's');

%Calculate the function f(x,y) based upon the signs of x and y

if color_letter== 'Y' || color_letter== 'y'
% strcmp can alsio be used to compare stings as follows
% if strcmp(color_letter, 'Y') || strcmp(color_letter, 'y')
    display('Yellow is made by combining red and green.');

elseif color_letter=='M' || color_letter=='m'
    display('Magenta is made by combining red and blue.');

elseif color_letter=='C' || color_letter=='c'
    display('Cyan is made by combining green and blue.');

else
    display('Invalid color.');
end
```

```
>>
>> color
Enter the color:y
Yellow is made by combining red and green.
>> color
Enter the color:Y
Yellow is made by combining red and green.
>> color
Enter the color:m
Magenta is made by combining red and blue.
>> color
Enter the color:M
Magenta is made by combining red and blue.
>> color
Enter the color:c
Cyan is made by combining green and blue.
>> color
Enter the color:C
Cyan is made by combining green and blue.
>> color
Enter the color:b
Invalid color.
>>
```

Branching Constructs

- **The switch Construct**

```
switch (switch_expr)
case case_expr_1
    Statement 1
    ....                    } Block 1
case case_expr_2
    Statement 1
     ....                   } Block 2
otherwise
    Statement 1
    ....                    } Block 3
end
```

Branching Constructs

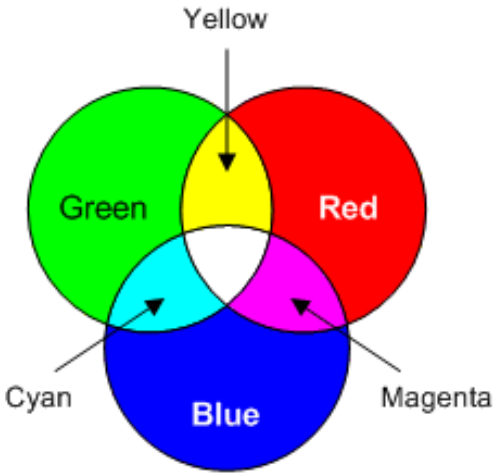
- **The switch Construct: Example-1**

Inputs:

color_letter

Output:

message




```
BEGIN
get color_letter
switch (color_letter)
case (Y or y)
   print "Yellow is made by combining red and green"
case (M or m)
    print "Magenta is made by combining red and blue"
case (C or c)
    print "Cyan is made by combining green and blue"
otherwise
     print "ERROR!"
End
END
```

Branching Constructs

- **The switch Construct: Example-1**

```
% Script file: color2.m
%
% Purpose:
% This program asks the user which of the three colors- yellow,
% magenta, or cyan- to break down into its two components.
% Yellow is made by combining red and green.
% Magenta is made by combining red and blue.
% Cyan is made by combining green and blue.

% In this program, we use switch construct.

%Record of revisions:
%Date       Programmer      Description of change
%=====      ==========      =====================
%16/05/2011 Alaa Khamis     Orignial Code
%
%Define variables:
% color_letter:    Color to be decomposed into primary components

%Prompt the user for the color (read it as a string)
color_letter=input('Enter the color:', 's');

%Calculate the function f(x,y) based upon the signs of x and y

switch(color_letter)
    case ('Y', 'y')
        display('Yellow is made by combining red and green.');

    case ('M' , 'm')
        display('Magenta is made by combining red and blue.');

    case ('C' , 'c')
        display('Cyan is made by combining green and blue.');
    otherwise
      display('Invalid color.');
end
```

```
>>
>> color
Enter the color:y
Yellow is made by combining red and green.
>> color
Enter the color:Y
Yellow is made by combining red and green.
>> color
Enter the color:m
Magenta is made by combining red and blue.
>> color
Enter the color:M
Magenta is made by combining red and blue.
>> color
Enter the color:c
Cyan is made by combining green and blue.
>> color
Enter the color:C
Cyan is made by combining green and blue.
>> color
Enter the color:b
Invalid color.
>>
```

Branching Constructs

- **The try/catch Construct**

The try/catch construct is a special form branching construct designed to trap errors.

```
try
   Statement 1
   Statement 2          } Block 1
   ....
catch
   Statement 1
   Statement 2          } Block 2
   ....
end
```

Branching Constructs

- **The try/catch Construct: Example-1**

This program creates an array and asks the user to specify an element of the array to display.

The user will supply a subscript number, and the program displays the corresponding array element.

The statements in the **try** block will always be executed in this program, while the statements in the **catch** block will be executed only if an error occurs in the **try** block.

Branching Constructs

- **The try/catch Construct: Example-1**

This program creates an array and asks the user to specify an element of the array to display.

The user will supply a subscript number, and the program displays the corresponding array element.

The statements in the **try** block will always be executed in this program, while the statements in the **catch** block will be executed only if an error occurs in the **try** block.

Branching Constructs

- **The try/catch Construct: Example-1**

```
% Script file: try_catch.m
%
% Purpose:
% Show how try/catch construct works,

%Initialize array
a=[1 -3 2 5]
try
    % Try to display an element
    index=input('Enter subscript of element to display: ');
    disp(['a('  int2str(index)  ') = ' num2str(a(index))]);
catch
    % If we get here an error occured
    disp(['Illegal subscript: '  int2str(index)]);
end
```

```
>> try_catch

a =

     1    -3     2     5

Enter subscript of element to display: 3
a(3) = 2
>> try_catch

a =

     1    -3     2     5

Enter subscript of element to display: 8
Illegal subscript: 8
>>
```

Outline

- Conditional Algorithms
- Logical Data Type
- Branching Constructs
- **Exercises**

Exercises

- **Exercise-1**

Write an algorithm in pseudo-code that inputs two numbers x and y, and computes and displays the value x/y if the value of y is not zero.

If y does have the value 0, then display the message "**Unable to perform division**". Implement the proposed algorithm using Matlab.

Exercises

- **Exercise-2**

Adult blood pressure is considered normal at 120/80 where the first number is the systolic pressure and the second is the diastolic pressure. Write a Matlab program that gets systolic and diastolic pressures as inputs and decides whether the pressure is normal or not.

Exercises

- **Exercise-3**

Write an algorithm in pseudo-code that gets the ambient temperature as input and decide whether the weather is cold, hot or nice based on the following criteria:

Temperature > 30 $\Rightarrow$ print “Hot weather”

Temperature < 18 $\Rightarrow$ print “Cold weather”

Otherwise $\Rightarrow$ print “Nice weather”.

Write a Matlab program to implement this algorithm.

Exercises

- **Exercise-4**

Write an algorithm in pseudo-code that gets the values of starting account balance, annually compounded rate and annual service charge. The algorithm includes the annual service charge only if the starting account balance at the beginning of the year is less than 1,000 pounds. If it is greater than or equal to 1,000 pounds, then no annual service charge is included. The algorithm should compute and display your balance after one year. Write a Matlab program that calculates the final balance.

Exercises

- **Exercise-5**

Australia is a great place to live, but it is also a land of high taxes. In 2002, individual citizens and residents of Australia paid the following income taxes:

Taxable Income (in A$)	Tax on this income
0-6,000	None
6,001-20,000	17 cents for each $1 over $6,000
20,001-50,000	$2,380 plus 30 cents for each $1 over $20,000
50,001-60,000	$11,380 plus 42 cents for each $1 over $50,000
Over 60,000	$15,380 plus 47 cents for each $1 over $60,000

In addition, a flat l.5% Medicare levy is charged on all income. Write a program to calculate how much income tax a person will owe based on this information.

Exercises

- **Exercise-5 (cont'd)**

  The program should accept a total income figure from the user and calculate the income tax, Medicare levy, and total tax payable by the individual.