Control Structure of Algorithm

Control Structures

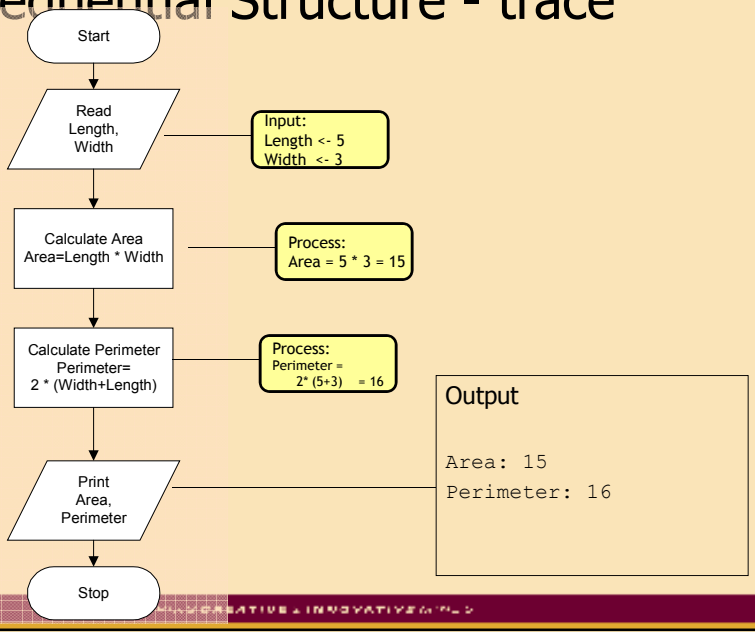
- Describe the flow of execution
- Basic types of control structure:
  1. Sequential
  2. Selection
  3. Repetition

Sequential Structure

- A series of steps or statements that are executed in the order they are written in an algorithm.
- Pseudo code - Mark the beginning & end of a block of statements.

```
1.  Start
2.  Statement_1
3.  Statement_2
4.  Statement_3
n.   Statement_n+1
N+1.End
```

Sequential Structure - trace



Trace Table

- Trace tables allow developers to test their algorithms in order to make sure there are no logic errors.
- Within the trace table, each variable, conditional test and output must be listed.

Example

Length	Width	Area	Perimeter	Output
10	15	150	50	
20	20	400	80	
30	15	450	90	

In-Class Exercise

- Execute and Trace Figure 2.8 (pg. 24).
- Execute Figure 2.9 (pg. 25) with the following input values. Write the respective output for each input values.

Input Values			
Book	2.50	20	100.00
Pencil	1.00	100	100.00
Table	350.00	5	2000.00

Selection Structure

• Selection allows you to choose between two or more alternatives; that is it allows you to make decision.

• Decisions made by a computer must be very simple since everything in the computer ultimately reduces to either true (1) or false (0).

• If complex decisions are required, it is the programmer's job to reduce them to a series of simple decisions that the computer can handle.

Selection Structure – Problem Examples

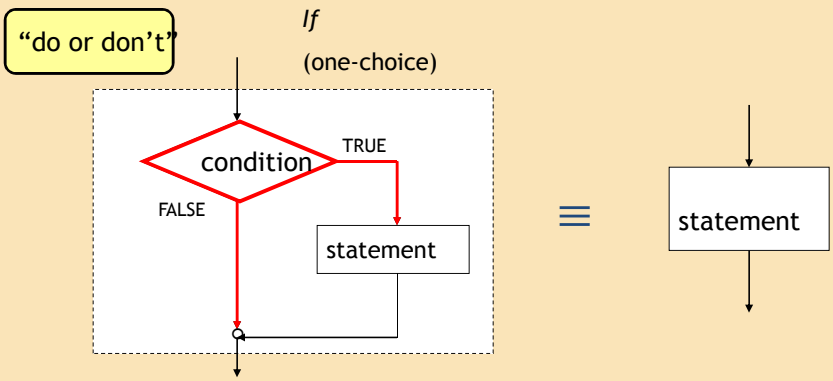
- Problem 1: Determine whether profit, return capital or loss.
- Problem 2: Determine whether a number is even or odd.
- Problem 3: Determine whether the marks is less than 60%. If it is less than 60, then print “fail”, otherwise print “pass”.
- Problem 4: Determine whether the speed limit exceeds 110 km per hour. If the speed exceeds 110, then fine = 300, otherwise fine = 0. Display fine.
- Problem 5: Determine whether the age is above 12 years old. If the age is above 12, then ticket = 20, otherwise ticket = 10. Display ticket.

Selection Structure (*One Alternative if Statement*)

- Pseudo code – requires the use of the keywords `if`.

```
Algorithm: one choice selection
:
n.   if condition
          n.1 statement
n+1. end_if
:
```

Selection Structure (*cont*..)



If set condition is true, execute the statement, else do nothing

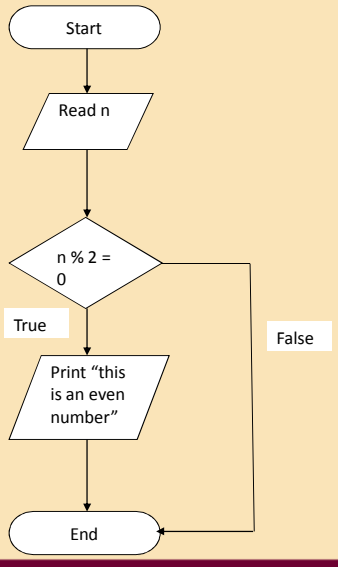
Example

- Determine whether an input number is even. If the number is even, print "This is even number".

Pseudocode

1. Start
2. Read n
3. If n modulus 2 = 0
   1. Print “This is an even number”
4. End if
5. End

Flowchart



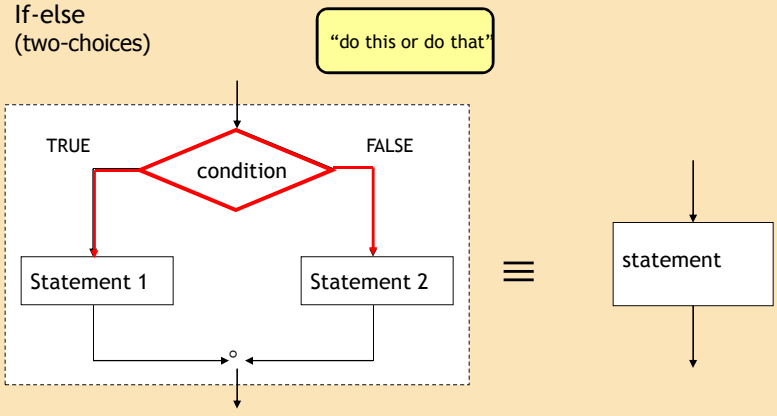
Selection Structure (*Two Alternatives if Statements*)

- Pseudo code – requires the use of the keywords `if` and `else`.

```
Algorithm: two choices selection
:
n.    if condition
            n.1 statement
            :
n+1.  else
      n+1.1 statement
      :
n+2.  end_if
:
```

Selection Structure (*cont..*)

If-else
(two-choices)



If set condition is true, execute the first statement, else execute second statement

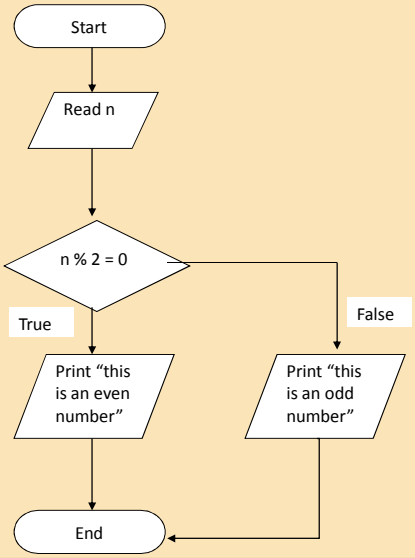
Example

- Determine whether an input number is even or odd. If the number is even, print “This is even number”. Else, print “This is odd number”.

Pseudocode

1. Start
2. Read n
3. If n modulus 2 = 0
   1. Print “This is an even number”
   2. Go step 6
4. Else
   1. Print “This is an odd number”
5. End if
6. End

Flowchart



Relational Operators

- Used to compare numbers to determine relative order
- Operators:

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

UTM

Relational Expressions

- Boolean expressions – `true` or `false`
- Examples:

  `12 > 5` is `true`

  `7 <= 5` is `false`

  if `x` is 10, then

  `x == 10` is `true`,

  `x != 8` is `true`, and

  `x == 8` is `false`

UTM

Logical Operators

- Used to create relational expressions from other relational expressions
- Operators, meaning, and explanation:

`&&`	AND	New relational expression is true if both expressions are true
`||`	OR	New relational expression is true if either expression is true
`!`	NOT	Reverses the value of an expression – true expression becomes false, and false becomes true

Truth Table

AND (&&)		
P	Q	P && Q
T	T	T
T	F	F
F	T	F
F	F	F

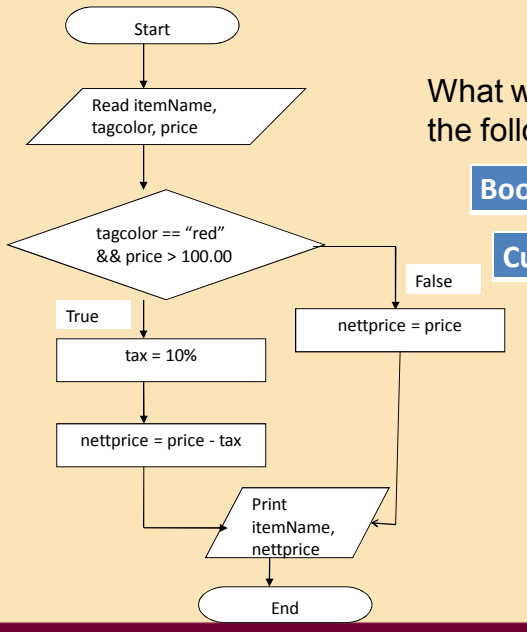
OR (||)		
P	Q	P||Q
T	T	T
T	F	T
F	T	T
F	F	F

Logical Operators - examples

```
int x = 12, y = 5, z = -4;
```

`(x > y) && (y > z)`	
`(x > y) && (z > y)`	
`(x <= z) || (y == z)`	
`(x <= z) || (y != z)`	
`!(x >= z)`	

In-Class Exercise



What will be the output for the following input?

Book	Green	350.00
Curtain	Red	500.00

In-Class Exercise 2

- Write a pseudo code for a program that will accept 2 numbers. If the first number is greater than the second number, find the difference between the numbers and print the numbers and difference value. If the second number is greater than the first number, find the sum of the two values and print the numbers and the sum.
- Draw the flowchart for the pseudo code.
- Trace the algorithm with the following input. Write the output:

40	50
70	30

In-Class Exercise 3

- Write down an algorithm (pseudo code) and draw a flowchart to read two numbers. If the first number is greater than the second number and it is larger than 50, find the sum and print the sum. Else, print the difference.
- Verify your result by a trace table. (Use 52, 30 as the numbers read)