CSC101
Introduction to ICT

Muhammad Sharjeel
muhammadsharjeel@cuilahore.edu.pk

Lecture - 8

Problem Solving Skills

How to Solve Problems

- *Problem: Your hair are dirty, and you want to clean your hair*

- *What steps would you propose to solve the above problem?*

How to Solve Problems

- Problem: Hair are dirty
- Solution: Clean hair
- How: "Wash the hair" algorithm

- Algorithm;
  - Turn on water tab
  - Wet your hair
  - Apply shampoo
  - Rinse
  - Dry off

How to Solve Problems

- Problem: How to make a brownie?
- Solution: ??

- Problem: How to make a cup of tea?
- Solution: ??

- Problem: How to get good marks?
- Solution: ??

Algorithms

- We are given a problem to solve:
  - Understand and analyze the problem and its requirements
    - Like in the case of dirty hair, the requirement is the clean hair
  - Devise steps to solve a problem

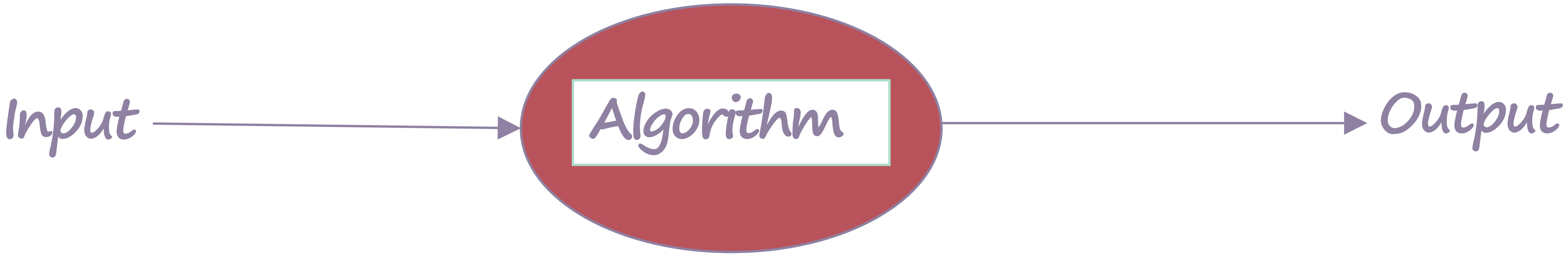
- The ordered collection of these steps is called an <u>algorithm</u>
- When writing an algorithm for the computer, the "order of operations" is a must

- The formal definition of algorithm;

  "a step-by-step method for solving a problem or doing a task"

Algorithms

- Characteristics of algorithms;
  - Definite and having input and output
  - Well-ordered, the steps are in a clear order
  - Unambiguous, the operations described are understood by a computing agent without further simplification
  - Effectively computable, the computing agent can carry out the operation



Input

Algorithm

Output

Algorithms vs Pseudocode

- Pseudocode is an artificial and informal language that helps in developing algorithms
- It is a method of writing an algorithm that may be in informal English, combinations of computer and/or spoken languages (whatever works for you)

- Algorithm is a systematic logical approach used to solve problems while Pseudocode are statements in natural language (syntax of communication) about solving a problem

- Two important concepts, when writing algorithms;
  - Variable
    - the identifier associated with a memory location used to store data
    - Sometimes called a named memory location
  - Constant
    - a data item with a fixed value

Algorithms

- Example: Write an algorithm to determine a student's final score* and then print it to the output screen

  * the final score is calculated as the average of the four subjects marks

- Pseudocode

```
Input a set of 4 marks (numbers)
Calculate their average, add all the numbers and divide by 4
Print average
```

- Algorithm

```
START
Step 1: INPUT n1, n2, n3, n4
Step 2: score ← (n1+n2+n3+n4)/4
Step 3: PRINT score
END
```

Rules of Algorithms

- Input:
  - use keyword "INPUT" or "GET" followed by a list of variables separated by a single comma

- Examples:

```
INPUT a
INPUT a, b
GET a
GET a, b
```

Rules of Algorithms

- Output:
    - use keyword “OUTPUT”, “DISPLAY”, “WRITE”, or “PRINT” followed by a variable name or text
    - enclose “text/message” in inverted commas
    - do not enclose variable name in inverted commas

- Examples:

```
OUTPUT  “Enter a number”
DISPLAY  “Your number is ”  num
```

Rules of Algorithms

- Storage/Assignment:
  - to give an initial value, use 'INITIALIZE' or 'SET' in combination with "=", ":=" or use keyword "=", ":=", "<-"
  - to keep a variable for later use, use 'SAVE' or 'STORE'

- Examples:

```
INITIALIZE  x
SET  x = 8
x = 8
SET  x := 8
 x := 8
 x <- 8
```

Rules of Algorithms

- Arithmetic operations:
  - use different mathematics symbols or expressions
- Logical (or comparison) operations:
  - use logical and comparison operators

- Examples:

  $x = 5, y = 7$

  $x > y$ or $x < y$

  $z = x + y$

  $z = 12$

Structure Theorem

- A structure theorem states that it is possible to write any algorithm by using only three basic control structures;

1. Sequence
   - I must study classes from grade 1 to grade 10
   - I cannot skip any class in order to reach in grade 10
2. Repetition
   - If I am failed in a grade, I must repeat it until pass
3. Selection
   - I have passed my 10th grade, now I must select between Science and Arts groups

Structure Theorem

- Sequence is an ordered list of steps to be executed
- It determines an order in which one step follows the other
- For example, you can not calculate average if you don't have a series of numbers (at least 2) as input etc.

- A loop is a repetition of all or some part(s) of the commands (steps)
- A loop often has a counter (a variable) and continues to repeat a specified number of times
- A loop may also continue, till a condition is true, or until a certain condition is met (e.g., until the end of a file or until a number reaches a set limit)

Structure Theorem (Repetition)

- FOR is a loop that allows steps to be repeatedly executed
    - it is typically used when the number of iterations are known beforehand

- Example:

```
FOR (number=1; number<10; number++)
    DISPLAY number
END FOR
```

Structure Theorem (Repetition)

- WHILE is a control flow statement that allows steps to be executed repeatedly based on a given condition

- Example:

```
number = 1
WHILE (number<10)
    DISPLAY number
    number = number + 1
END WHILE
```

Structure Theorem (Repetition)

- DO WHILE is a statement that performs the action(s) at least once

- Example:

```
number = 1
DO
    DISPLAY number
    number = number + 1
WHILE (number<10)
```

Structure Theorem (Repetition)

- Example: Write an algorithm to read and print 10 records using the while loop

```
START
SET total = 0
WHILE (total < 10)
    READ record
    PRINT record
    total = total + 1
END WHILE
END
```

- The variable 'total' is initialized before the loop condition is executed
- It is then incremented within the body of the loop, so the loop will eventually stop

Structure Theorem (Selection)

- Sometimes we need to put certain condition(s) before performing an action
- **If** the condition(s) is ‘true’, **then** action will be executed, **else** not

- **Selection** compares two pieces of information and select one of the two alternatives
- It is represented by the IF statement and keywords IF, THEN, ELSE and ENDIF

- IF statement always has a condition to check, often a comparison between a variable and a number
- The IF statement also must specify what to do if the condition/comparison is true
- These instructions (for “true”) comes after the word THEN

Structure Theorem (Selection)

- IF – THEN – END IF (Single IF)
  - Single IF selection statement either performs an action if a condition is true or skips the action if the condition is false
- IF – ELSE – END IF (Double IF)
  - Double IF selection statement performs an action if a condition is true and performs a different action if the condition is false
- IF – ELSE IF – ELSE – END IF (Multiple IF)
  - Multiple IF selection statement performs one of the many different actions, depending on the value of the expression
- IF – IF – ELSE – END IF – ELSE – END IF (Nested IF)
  - Nested IF selection statement means an if statement inside another if statement, it is an if statement that is the target of another if statement
- Switch (Alternate to Multiple IF)
  - The SWITCH selection statement is an alternative to multiple IF statement

Structure Theorem (Selection)

- Single IF

```
IF (condition) THEN
        <<steps>>
END IF
```

- Double IF

```
IF (condition) THEN
        <<steps>>
ELSE
        <<steps>>
END IF
```

Structure Theorem (Selection)

- Multiple IF

```
IF (condition) THEN
    <<steps>>
ELSE IF (condition) THEN
    <<steps>>
ELSE IF (condition) THEN
    <<steps>>
ELSE
    <<steps>>
END IF
```

- Nested IF

```
IF (condition) THEN
    IF (condition) THEN
        <<steps>>
    ELSE
        <<steps>>
    END IF
ELSE
    <<steps>>
END IF
```

Structure Theorem (Selection)

- Example: Write an algorithm to determine if a student has passed* a subject and print it to the output screen

  *the passing marks are 50 and above

- Pseudocode

```
Input marks (a number)
If the marks are 50 or above, print "PASS"
```

- Algorithm

```
START
Step 1:  INPUT marks
Step 2: IF (marks >= 50) THEN
              PRINT  "PASS"
          END IF
END
```

Structure Theorem (Selection)

- Example: Write an algorithm to determine a student's final score*, indicate whether the student is pass or fail**, and print it to the output screen

  *the final score is calculated as the average of the four subjects marks

  **the passing marks are 50 and above

- Pseudocode

```
    Input a set of 4 marks (numbers)
    Calculate their average, add all the numbers and divide by 4
    If average is below 50, print "FAIL", else, print "PASS"
```

Structure Theorem (Selection)

- Example: Write an algorithm to determine a student's final score*, indicate whether the student is pass or fail**, and print it to the output screen

  *the final score is calculated as the average of the four subjects marks

  **the passing marks are 50 and above

- Algorithm

```
START
Step 1:  INPUT n1, n2, n3, n4
Step 2: score ← (n1+n2+n3+n4)/4
Step 3: IF (score < 50) THEN
                PRINT  "FAIL"
            ELSE
                PRINT  "PASS"
            END IF
END
```

Structure Theorem (Selection)

- Example: Write an algorithm to determine a student's final score*, determine whether the student is pass or fail**, if the student is pass, calculate the student's grade (based on the criterion given below), and print the grade to the output screen

  *the final score is calculated as the average of the four subjects marks

  **the passing marks are 50 and above

  - If marks are between 50 and 59, grade is E
  - If marks are between 60 and 69, grade is D
  - If marks are between 70 and 79, grade is C
  - If marks are between 80 and 89, grade is B
  - If marks are between 90 and 100, grade is A

Structure Theorem (Selection)

- Algorithm

```
START
Step 1:  INPUT n1, n2, n3, n4
Step 2: score ← (n1+n2+n3+n4)/4
Step 3: IF (score >= 50) THEN
          PRINT  “PASS”
            IF (score >= 50 && score <= 59)
                PRINT  “Grade is E”
            ELSE IF (score >= 60 && score <= 69)
                PRINT  “Grade is D”
            ELSE IF (score >= 70 && score <= 79)
                PRINT  “Grade is C”
            ELSE IF (score >= 80 && score <= 89)
                PRINT  “Grade is B”
            ELSE IF (score >= 90 && score <= 100)
                PRINT  “Grade is A”
            END IF
        ELSE
          PRINT  “FAIL”
        END IF
END
```

Structure Theorem (Repetition+Selection)

- Example: Write a WHILE loop structure that displays any number input by the user on the screen and only terminates when the user enters a sentinel value (assume the sentinel value to be -1)

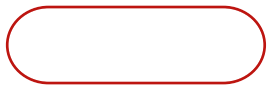
```
INPUT  a
WHILE (a != -1)
    DISPLAY a
    INPUT a
END WHILE
```

Flowchart

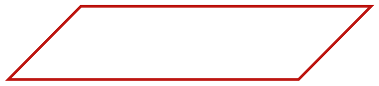
- Flowchart is a graphical representation that shows logic solution
- Emphasizes individual steps and their interconnections
- It must have a start and stop step
- All steps in a flowchart must connect, i.e., you can't leave a step "hanging" with no connection

Flowchart (Symbols)

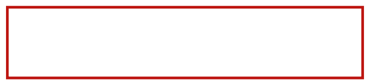
- Start/End
  - used at the beginning and end of each flowchart



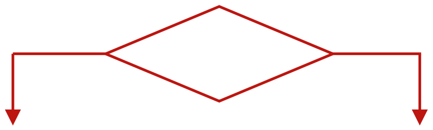
- Input/Output
  - shows when data comes in or information is printed out



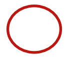
- Process
  - used to show calculations, storing of data in variables, and other "processes"



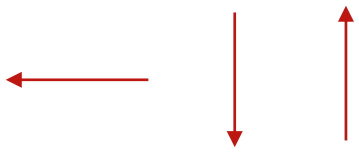
- Decision
  - used to show that the flow must decide whether
  - something (usually a comparison between numbers) is true or false



- Connector
  - used to show that flowchart continues another page



- Flow Direction
  - shows the direction of flow



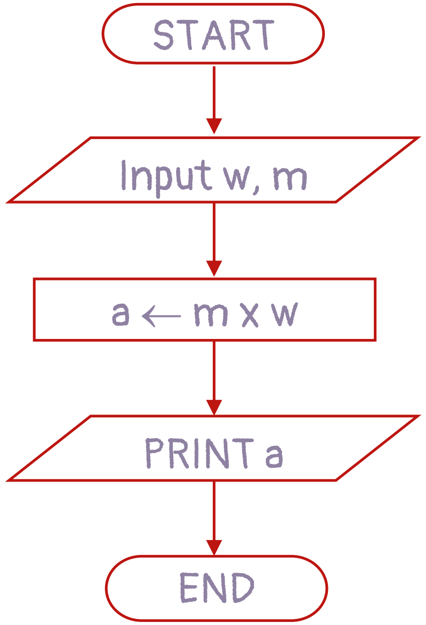
Flowchart

- *Example: Write an algorithm and draw flowchart that inputs two numbers from the user, multiply it, and print the answer to the screen*

Algorithm

```
START
Step 1: INPUT w, m
Step 2: a ← m x w
Step 3: PRINT a
END
```

Flowchart



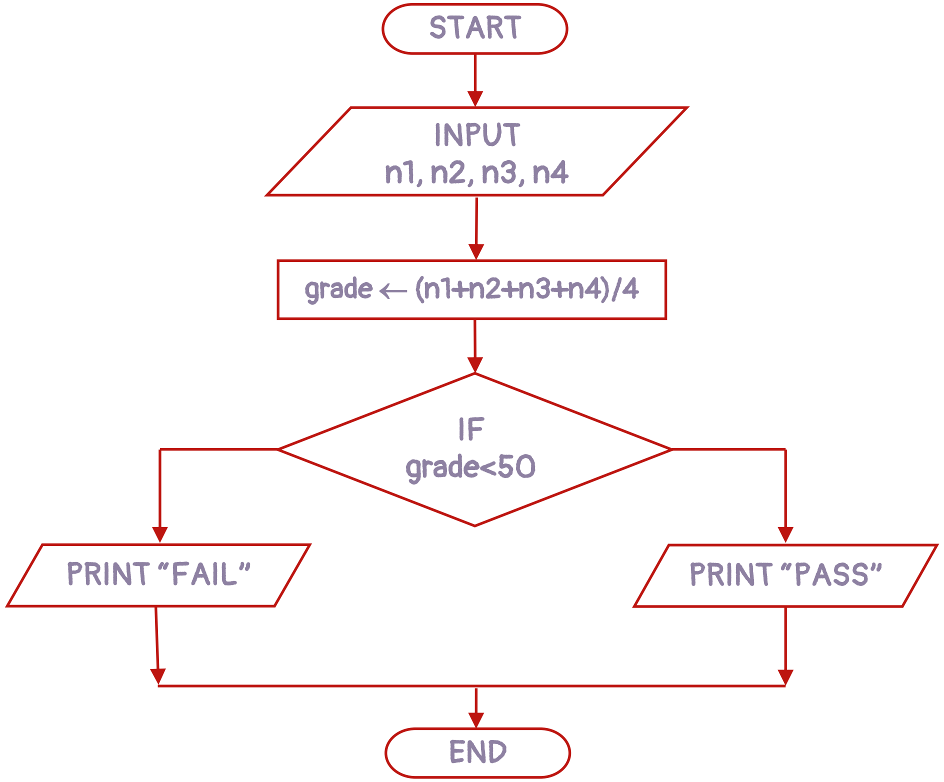
Flowchart

- Example: Write an algorithm and draw flowchart to determine a student's final score and indicate whether the student is pass or fail, and print it to the output screen

Algorithm

```
START
Step 1:  INPUT n1, n2, n3, n4
Step 2: grade ← (n1+n2+n3+n4)/4
Step 3: IF (grade < 50) then
            PRINT  "FAIL"
    ELSE
            PRINT  "PASS"
    END IF
END
```

Flowchart



Trace Tables & Dry Run

- Dry run
  - working through a section of an algorithm (program) manually
- Trace table
  - a technique used to test algorithms to make sure they work without any error

Read more: http://www.101computing.net/trace-table/

Trace Tables & Dry Run

- Example: Following is an algorithm that takes a number as an input, multiplies it by 2, and prints the output to the screen. Make trace table for the algorithm.

Algorithm

```
START
Step 1: SET x := 0
Step 2: INPUT y
Step 3: x := y * 2
Step 4: OUTPUT x
END
```

Trace Table

Step	Algorithm Lines	X	Y	Output
	START	–	–	–
1	SET x := 0	0	–	–
2	INPUT y	0	5	–
3	x := y * 2	10	5	–
4	OUTPUT x	10	5	10
	END	–	–	–

Trace Tables & Dry Run

- Example: Following is an algorithm that take five numbers as input from the user, calculates and display their sum and average. Make trace table for the algorithm

Algorithm

```
    START
    Step 1: SET sum := 0, average := 0, totalNumbers :=5
    Step 2: INPUT n1, n2, n3, n4, n5
    Step 3: sum := n1 + n2 + n3 + n4 + n5
    Step 4: average := sum / totalNumbers
    Step 5: PRINT "Sum is " sum
    Step 6: PRINT "Average is " average
    END
```

Trace Tables & Dry Run

Trace Table

Step	Algorithm Lines	n1	n2	n3	n4	n5	sum	average	totalNumbers	Output
	START	–	–	–	–	–	–	–	–	–
1	SET sum := 0, average := 0, totalNumbers :=5	–	–	–	–	–	0	0	5	–
2	INPUT n1, n2, n3, n4, n5	25	17	34	9	75	0	0	5	–
3	sum := n1 + n2 + n3 + n4 + n5	25	17	34	9	75	160	0	5	–
4	average := sum / totalNumbers	25	17	34	9	75	160	32	5	–
5	PRINT "Sum is " sum	25	17	34	9	75	160	32	5	Sum is 160
6	PRINT "Average is " average	25	17	34	9	75	160	32	5	Sum is 160 Average is 32
	END	–	–	–	–	–	–	–	–	–

THANK YOU