Algorithms I

CEN111 Algorithms and Programming I

Introduction

- Problem

- Problem Solving

Algorithms

- Problem solving phase
  - produce an ordered sequence of steps that describe solution of problem

- Implementation phase
  - implement the program in some programming language

Algorithms

- The word is derived from the phonetic pronunciation of the last name of Abu Ja'far Mohammed ibn Musa al-Khowarizmi.
- An algorithm is a sequence of steps to solve a particular problem
- An algorithm is an ordered set of unambiguous steps that produces a result and terminates in a finite time

Algorithms

- It is a step-by-step representation of a solution to a given problem
- An algorithm uses a definite procedure.
- It is not dependent on any programming language.
- Every step in an algorithm has its own logical sequence.

Properties of Algorithms

- **Finiteness:** An algorithm must always terminate after a finite number of steps.
- **Definiteness:** Each step of an algorithm must be precisely defined. Also the actions are defined unambiguously for each activity in the algorithm.
- **Input:** Any operation you perform need some beginning value/quantities associated with different activities in the operation.
- **Output:** One always expects output/result (expected value/quantities) in terms of output from an algorithm.
- **Effectiveness:** Algorithms to be developed/written using basic operations.

Algorithms

- Define your algorithms input
- Define the variables
- Outline the algorithm's operations
- Output the results of your algorithm's operations

Algorithms

- Unambiguous
- Executable
- Ordered

Algorithms

- Find the area of a circle of radius r.

Inputs to the algorithm: Radius of the circle.

Expected output: Area of the circle

Algorithm:

1. Start
2. Get radius r
3. area=PI*r*r
4. Print area
5. End

Pseudocode

- **Pseudocode** is an artificial and informal language that helps programmers develop algorithms. Pseudocode is very similar to everyday English.
- **pseudo - code** it cannot be executed on a real computer, but it models and resembles real programming code, and is written at roughly the same level of detail.

Some Keywords

- **Start** - **End**
- **Goto**
- Set, Initialize
- Read, **Get**
- **Print**, Write, Display

Assignment

- ← or = is used to assign value to a variable
- to assign value 3 to the variable radius, the statement is

  radius=3 or radius ←3

  - C=A+B

  - R=R+1

Mathematical Operators

Operator	Meaning	Example
+	Addition	A + B
-	Substraction	A - B
*	Multiplication	A * B
/	Divison	A / B
^	Power	A^B
%	Reminder	A % B

Algorithms

- Find the sum of two numbers

```
var: number1, number2, sum
1. Start
2. Get number1
3. Get number2
4. sum=number1+number2
5. Print sum
6. End
```

Algorithms

- Find the avarege of two numbers

```
var: number1, number2, sum, avg
1. Start
2. Get number1
3. Get number2
4. sum = number1 + number2
5. avg = sum / 2
6. Print avg
7. End
```

Structures

- Sequence
- Branching (Selection)
- Loop (Repetition)

Relational Operators

Operator	**Meaning**	**Example**
<	Less than	A < B
<=	Less than or equal to	A <=B
==	Equal to	A == B
!=	Not equal to	A != B
>	Greater than	A > B
>=	Greater than or equal to	A >=B

Algorithms

- Determine wheter the the student passed or failed according the entered GPA. To pass, GPA must be greater than or equal to 60.

```
var: gpa
1. Start
2. Get gpa
3. If gpa >= 60
         print  "Passed"
     else
         print   "Failed"
4. End
```

Algorithms

- Find the greater of two numbers

```
var: number1, number2, max
1.  Start
2.  Get number1
3.  Get number2
4.  If number1 > number2
            max = number1
      else
            max = number2
5. Print max
6. End
```

Logical Operators

Operator	**Example**	**Meaning**
AND	A<B AND B<C	Result is true if both conditions are true else false
OR	A<B OR B<C	Result is true if either A<B or B<C are true else false
NOT	NOT (A<B)	Result is true if A<B is false else true

Algorithms

- Find the greatest of three numbers

```
var: a, b, c, max

1. Start
2. Get a, b, c
3. If a >= b and a >= c then max = a
4. If b >= a and b >= c then max = b
5. If c >= a and c >= b then max = c
6. Print max
7. End
```

Structures

- Sequence
- Branching (Selection)
- Loop (Repetition)

Algorithms

- Find odd numbers between 1 to 10

var: counter

```
1. Start
2. counter = 1
3. Print counter
4. counter = counter + 2
5. If counter <= 10 then Goto Step 3
6. End
```

Algorithms

- Find odd numbers between 1 to 10

var: counter

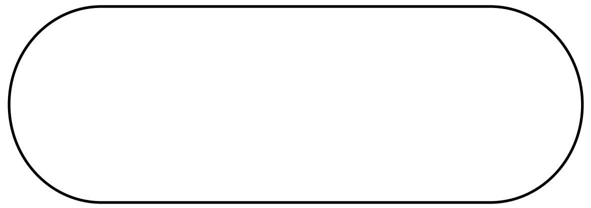
```
1. Start
2. counter = 1
3. If counter % 2 ==1 then print counter
4. counter = counter + 1
5. If counter <= 10 then Goto Step 3
6. End
```

Flowchart

- Flowchart is a diagram which visually presents the flow of data through processing systems.
- A flowchart is a graphical representation of an algorithm.
- Once the flowchart is drawn, it becomes easy to write the program in any high level language.

Flowchart Symbols

- Terminal



- Indicates the starting or ending of the process.

Flowchart Symbols

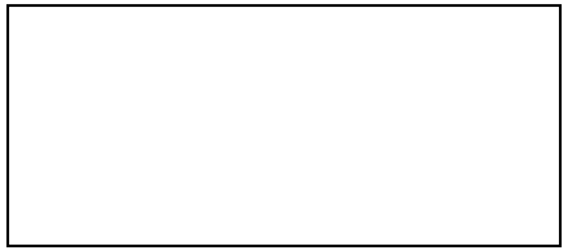
- Flow lines

$$\longrightarrow$$

- An arrow coming from one symbol and ending at another symbol.
- Shows direction of flow.

Flowchart Symbols

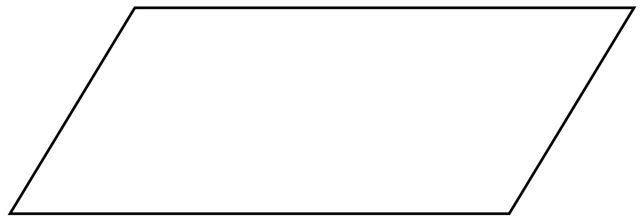
- Process



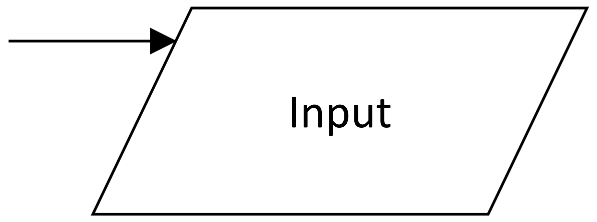
- Indicates any type of internal operation inside the processor or memory

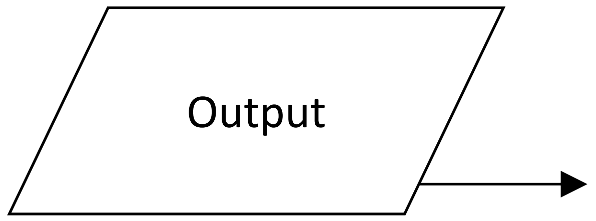
Flowchart Symbols

- Input/output



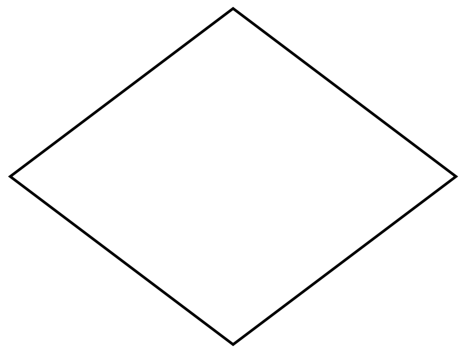
- Used for any I/O operation. Indicates that the computer is to obtain data or output results.





Flowchart Symbols

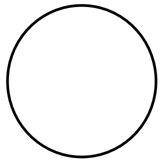
- Decision



- Used to ask a question that can be answered in a binary format.

Flowchart Symbols

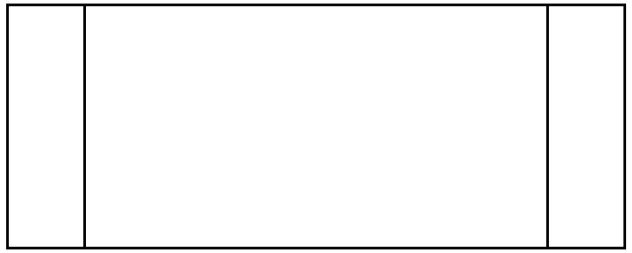
- Connector



- Allows the flowchart to be drawn without intersecting lines or without a reverse flow.

Flowchart Symbols

- Predefined process



- Used to invoke asubroutine or an interrupt program..

Flowcharts

- All boxes of the flowchart are connected with arrows.
- Flowchart symbols have an entry point on the top of the symbol with no other entry points. The exit point for all flowchart symbols is on the bottom except for the Decision symbol.
- The Decision symbol has two exit points; these can be on the sides or the bottom and one side.
- Generally a flowchart will flow from top to bottom. However, an upward flow can be shown as long as it does not exceed 3 symbols.

Flowcharts

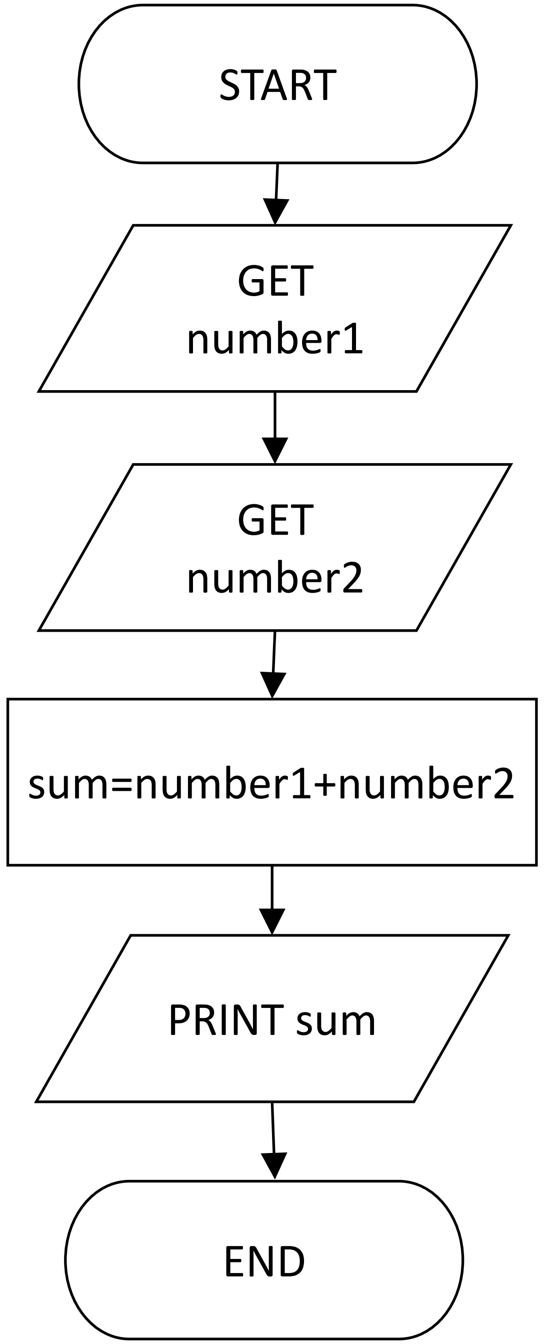
- Connectors are used to connect breaks in the flowchart.
  - From one page to another page.
  - From the bottom of the page to the top of the same page.
  - An upward flow of more then 3 symbols
- Subroutines and Interrupt programs have their own and independent flowcharts.
- All flow charts start with a Terminal or Predefined Process (for interrupt programs or subroutines) symbol.
- All flowcharts end with a terminal.

Problem

- Find the sum of two numbers

var: number1, number2, sum

1. Start
2. Get number1
3. Get number2
4. sum = number1 + number2
5. Print sum
6. End



Exercise

- Find the greater of two numbers

```
var: number1, number2, max
1.  Start
2.  Get number1
3.  Get number2
4.  If number1 > number2
            max = number1
      else
            max = number2
5. Print max
6. End
```

Homework

- Using flowcharts, write an algorithm to read three numbers then display the smallest.

CLASS
Boston Dynamics