ICT 106 1.5
Fundamentals of Computer Programming

M.D.R. PERERA

What is a Computer program

- A **computer program** is a list of instructions that tell a computer what to do.
- It is like a recipe.
- Everything a computer does is done by using a computer program.
  - Operating system
  - An office packagers…
  - Video games etc…

Programming language

- A programming language is a vocabulary and set of grammatical rules for instructing a computer
- It is used to create computer programs.
- It usually refers to high-level languages, such as BASIC, C, C++,Pascal, Python etc..

Programming language

- Need to convert your program in to machine language
  - Compile the program
  - Interpret the program

Programming language

- **Compile is to transform a program written in a high level programming language from source code into object code.**
- It is done by the **compiler.**
- A compiler reads the whole source code and translates it into a complete machine code

Programming language

- **Interpreter is a program that executes instructions written in a high-level language.**
- It reads the source code one instruction or line at a time, converts this line into machine code and executes it.

Programming language

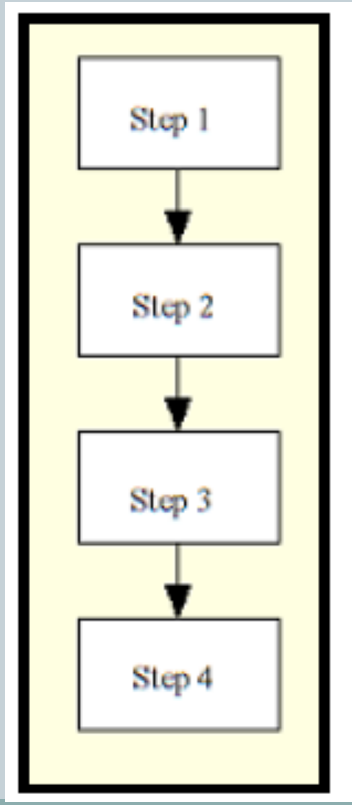
- Computer programming is the process of
  - writing,
  - testing,
  - debugging/trouble shooting,
  - maintaining the source code of computer programs.

Common Features of All Program

- All programs could be structured in the following four ways:
  - Sequences of instructions
  - Branches
  - Loops
  - Modules

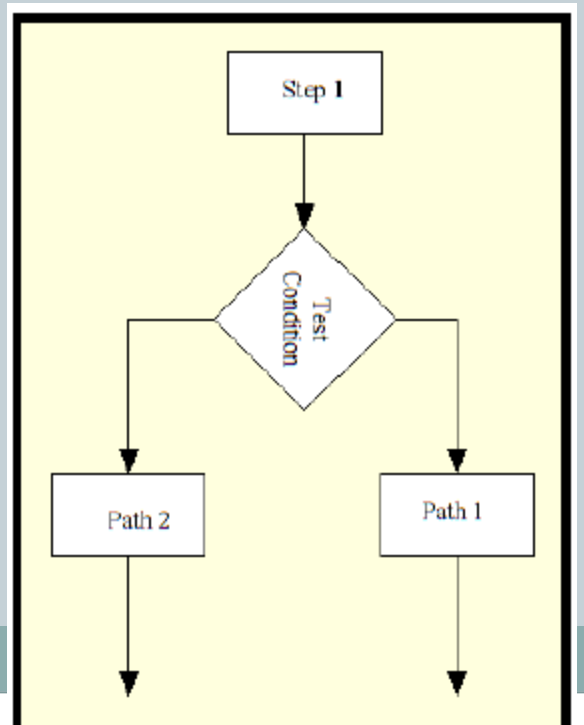
Sequences of instructions

- The program flows from one step to the next in strict sequence.



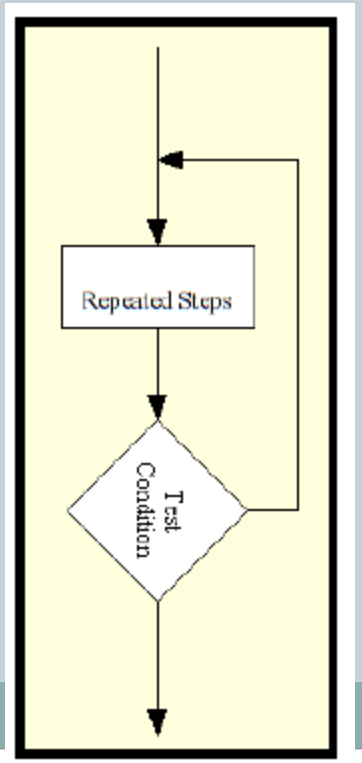
Branches

- The program reaches a decision point and if the result of the test condition is true then the program performs the instructions in Path 1, and if false it performs the actions in Path 2



Loops

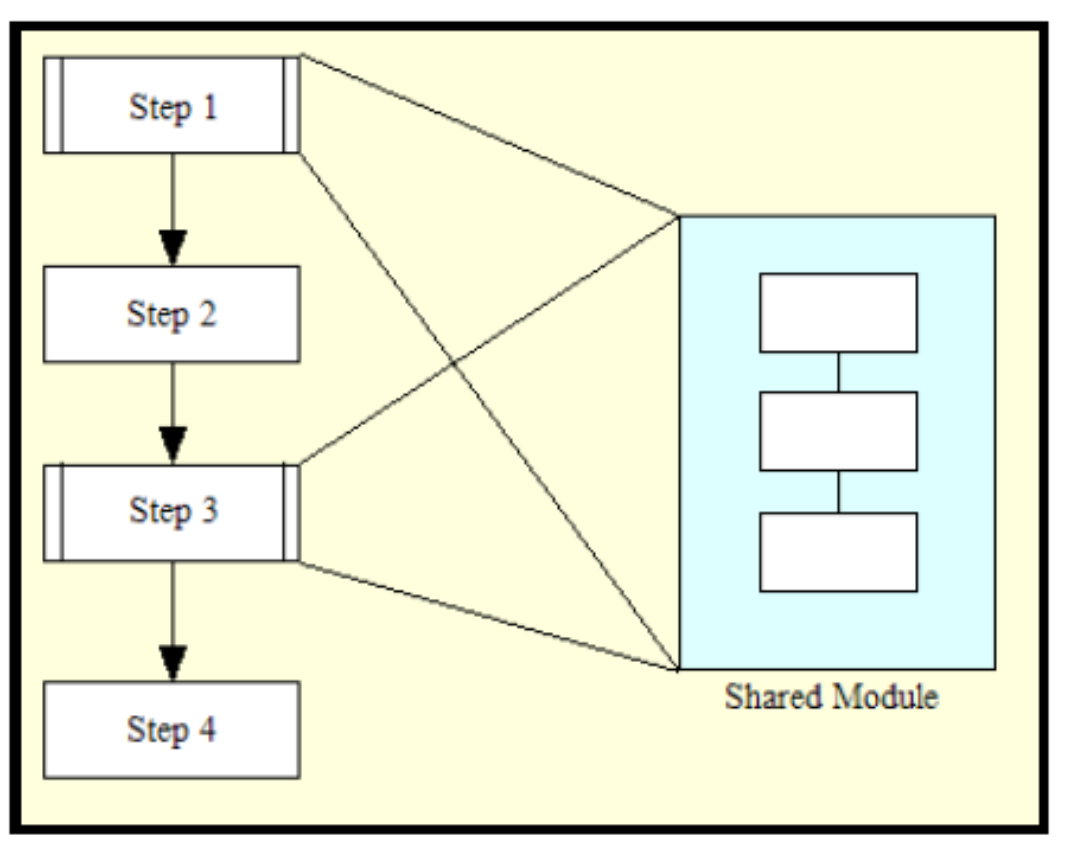
- The program steps are repeated continuously until test condition is satisfied
- After that it moves to the next piece of program block



Modules

- The program performs an identical sequence of actions several times.
- common actions are placed in a module (mini-program)
- main program call and execute it.

Modules



Common Features of All Program

- Along with these structures programs also need a few more features to make them useful:
  - Data
  - Operations (add, subtract, compare etc.)
  - Input / Output capability (e.g. to display results)

The Programming Process

- **Step 1:** Defining the problem.
- **Step 2:** Planning the solution.
- **Step 3:** Code the program.
- **Step 4:** Test the program.
- **Step 5:** Documentation.

Step 1: Defining the problem

- Identify tasks of the problem
- specified the kind of input, processing, and output required.

Step 2: Planning the solution

- The ways of planning the solution
  - draw a flow chart or write pseudocode
  - possibly both.

Step 2: Planning the solution

- A flowchart is a pictorial representation of a step-by-step solution to a problem.
- It is a map of what your program is going to do and how it is going to do.

Step 2: Planning the solution

- Pseudocodeis:
  - English-like nonstandard language
  - It focus on the program logic

Step 3: Code the program

- Translate the program logic in to programming language

Step 4: Test the program

- Almost all programs may contain a few errors, or bugs.
- Testing is necessary to find out if the program produces a correct result.
- Usually it is performed with sample data
- Debugging is the process of locating and removing errors

Step 4: Test the program

- Types of Error
  - Syntax Errors
  - Semantic Errors
  - Run-time Errors

Step 4: Test the program

- **Syntax Errors:**
  - Violation of syntactic rules in a Programming Language
  - Interpreter or Compiler can finds it at the syntax check phase.
- **Semantic Errors:**
  - Doing logical mistakes in the source code.
  - Interpreters and Compilers can not notice them
  - but on run time, it causes unexpected results.

Test the program

- **Run-time Errors:**
  - Occur on program execution.
  - Mostly caused by invalid data entry or tries to access unavailable resources.
  - It occurs on run time and may crash the program execution

Step 5: Documentation

- Documentation is a written detailed description of the programming cycle
- it specific the facts about the program.
- materials include the origin and nature of the problem
- brief narrative description of the program using the logic tools
  - flowcharts and pseudocode,
  - data-record descriptions,
  - program listings
  - testing results. etc
- Comments in the program itself are also considered an essential part of documentation.

Computer Programmer

- A programmer is person who writes computer program.
- Computer programmers write, test, and maintain programs or software

Required Skills to Become a Programmer

- **Programming :**
  - Writing computer programs for various purposes.
- **Writing:**
  - Communicating effectively with others in writing as indicated by the needs of the audience.
- **Reading Comprehension:**
  - Understanding written sentences and paragraphs in work-related documents.
- **Critical Thinking:**
  - Using logic and analysis to identify the strengths and weaknesses of different approaches.

Required Skills to Become a Programmer

- **Computers and Electronics**
  - Knowledge of electric circuit boards, processors, chips, and computer hardware and software, including applications and programming.
- **Mathematics:**
  - Knowledge of numbers, their operations, and interrelationships including arithmetic, algebra, geometry, calculus, statistics, and their applications.
- **Oral Expression:**
  - The ability to communicate information and ideas with others

Required Skills to Become a Programmer

- **Oral Comprehension:**
  - The ability to listen to and understand information and ideas presented through spoken words and sentences.
- **Written Expression**
  - The ability to communicate information and ideas in-writing with others.
- **Written Comprehension:**
  - The ability to read and understanding the problem which presented in writing

Required Skills to Become a Programmer

- **Deductive Reasoning:**
  - The ability to apply general rules to specific problems to come up with logical answers.
- **Information Organization:**
  - Finding ways to structure or classify multiple pieces of information.

Generations of Programming Language

- The first generation : low-level languages that are machine language.
- The second generation: low-level languages that generally consist of assembly languages.
- The third generation: high-level languages such as C.

Generations of Programming Language

- The fourth generation: it consist of statements similar to statements in a human language
  - commonly used in database programming and scripts).
  - It contain visual tools to help develop a program. (Ex: Visual Basic)

Generations of Programming Language

- There are three types of programming language:
  - Machine language(Low-level language)
  - Assembly language(Low-level language)
  - High-level language

- Low-level languages : closer to the language used by a computer
- high-level languages : closer to human languages.

High-Level Language

- Examples of High-level Language:
  - BASIC
  - PASCAL
  - C
  - C++
  - JAVA
  - COBOL
  - PHYTON

Choosing a Programming Language

- Before you decide on what language to use, you should consider the following:
  - server platform
  - server software you run
  - budget
  - previous experience in programming

writing a program

- Before writing a program:
  - Have a thorough understanding of the problem
  - Carefully plan an approach for solving it
- While writing a program:
  - Use good programming principles
- Computing problems
  - solved by executing a series of actions in a specific order

Introduction to Algorithms

- Algorithm: procedure in terms of
  - A sequence of instructions.
  - A procedure or formula for solving a problem.
  - Often used for calculation, data processing and programming.
  - Algorithms can be expressed in any language
- Program control
  - Specify order in which statements are to be executed
- Decide the steps to be followed for solving the problem in question.

Introduction to Algorithms

- Description of the algorithm in a standard form (easy to understand for anyone)
- Language independent
- Algorithms for making things will often be divided into sections
  - parts/ components/ inputs required to accomplish the task
  - actions/steps/methods(processing) to produce the required out come (output).

Introduction to Algorithms

- Two forms of Algorithm:
  - Pseudocode (mix. of English & programming language)
  - Flowchart(using charts)

Pseudocode

- It is fake code
- It specifies the steps required to accomplish the task.
- It is a type of structured English that is used to specify an algorithm.
- It cannot be compiled nor executed
- there are no real formatting or syntax rules.

Pseudocode

```
If the student's mark is greater than or equal to
  40
    Print "passed"
else
    Print "failed"
```

Advantages of Pseudocode

- Reduced complexity.
- Increased flexibility.
- Ease of understanding.

Why is Pseudocode Necessary?

- The programming process is a complicated one.
- Need to understand the program specifications.
- Need to organize your thoughts and create the program.
- To break the main tasks that must be accomplished into smaller ones
- It will save time at the later during (in Construction & testing phase).

How to Write Pseudocode Statements?

- There are six basic computer operations

1. receive information
   - Read (information from a file)
   - Get (information from the keyboard)

2. put out information
   - Write (information to a file)
   - Display (information to the screen)

How to Write Pseudocode Statements?

3. perform arithmetic operation, use actual mathematical symbols or the words for the symbols

Ex: Add number to total

- Total = total + number
- +, -, *, / or Calculate, Compute words are also used

How to Write Pseudocode Statements?

- A computer can assign a value to a piece of data

1. to give data an initial value use
   - “Initialize”, “Set” words

2. to assign a value as a result of some processing
   - use “=“ symbol
   - x=5+y

3. to keep a piece of information for later use
   - “Save”, “Store” words

How to Write Pseudocode Statements?

5. compare two piece of information and select one of two alternative actions

```
If the student's mark is greater than or equal to
  40
    Print "passed"
else
    Print "failed"
```

How to Write Pseudocode Statements?

6. repeat a group of actions

```
While student counter is less than or equal to ten
    Input the next exam result
    Add one to student counter

FOR a number of times
    some action
ENDFOR
```

Data Dictionaries

- The pseudo code by itself doesn't provide enough information to be able to write program code.
- Data Dictionaries are used to describe the data used in the Pseudo Code.
- The standard data types used in Pseudo Code are Integer, Double, String, Char and Boolean.

Data Dictionaries

Name	Data Type	Description
intNumber1	Integer	The first number to be added
intNumber2	Integer	The second number to be added
intTotal	Integer	The total of number1 and number2 added together

Program Specification

Example 1:

Write a program that obtains three integer numbers from the user and prints the average of those numbers.

Pseudocode:

- Prompt the user to enter the first integer
- Prompt the user to enter a second integer
- Prompt the user to enter a third integer
- Compute the average of the three user inputs
- Display an output prompt that explains the answer as the "Average of three numbers"
- Display the result

Program Specification

- We might usually specify the procedure of solving this problem as
- Compute the average by **adding the three numbers and divide by three.**
- Here, Read(or Ask) and Write(or Say) are implied.
- However in an algorithm, these steps have to be made explicit.

Programming algorithm

Thus a possible algorithm is:

Step1: Start

Step2: Read values of **Number1 , Number12, Number13**

Step3: **Summation = Number1 + Number2+ Number3**

Step4: **Average = Summation / 3**

Step5: Print the value of " **Average** "

Step6: End

Programming algorithm

- Finding biggest of two numbers.

Step1: Start

Step2: Read **Number1, Number2**

Step3: If "**Number1**" **is grater than "Number2"**,then

BiggestNo = Number1

otherwise **BiggestNo = Number2**,

Step4: Print **BiggestNo**

Step5: End

πr^2

Programming algorithm

- Write a Pseudocode for these problems.
  1. Area of a rectangle
  2. Area of a circle
  3. Convert from Celsius to Fahrenheit. (Multiply by 9, then divide by 5, then add 32)
  4. Volume of a cube
  5. Volume of cylinder
  6. Average speed = Distance / traveled Time
  7. Average value of the three numbers

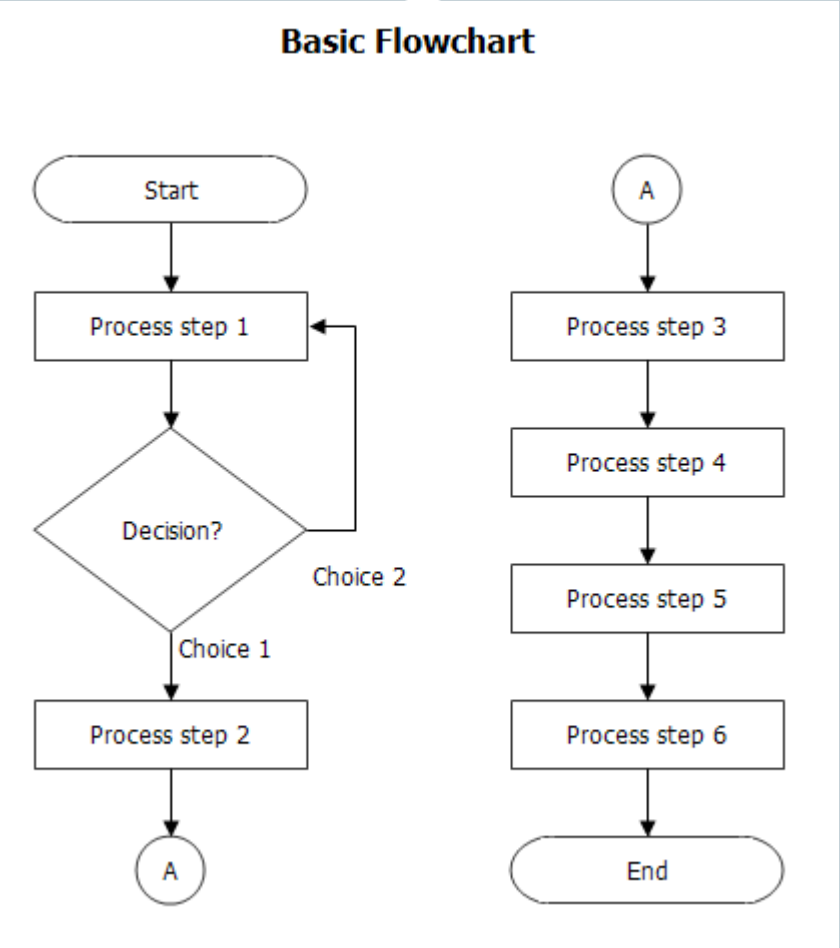
Flowchart

- Graphical representation of an algorithm.
- playa vital role in the programming of a problem and are quite helpful in understanding the logic of complicated and lengthy problems.

Flowchart

- A flow chart can used to:
  - Define and analyze processes
  - Build a step-by-step picture of the process for analysis, discussion, or communication
  - Define, standardize or find areas for improvement in a process

Flowchart


Basic Flowchart
Start
Process step 1
Decision?
Choice 2
Choice 1
Process step 2
A
A
Process step 3
Process step 4
Process step 5
Process step 6
End

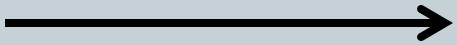
Flowchart Symbols

- **Start and end symbols**

  - Represented as lozenges, ovals or rounded rectangles
  - Usually containing the word "Start“ or“ End"

Flowchart Symbols

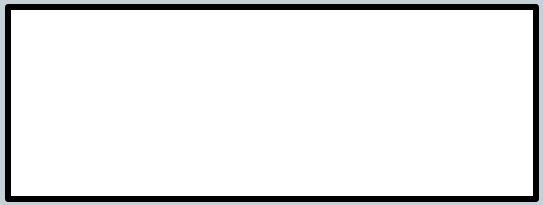
- **Arrows**



- Used to define the "flow of control“.
- An arrow coming from one symbol and ending at another symbol.
- It represents that control is passes to one process to another process.

Flowchart Symbols

- **Processing steps**



- Represented as rectangles.

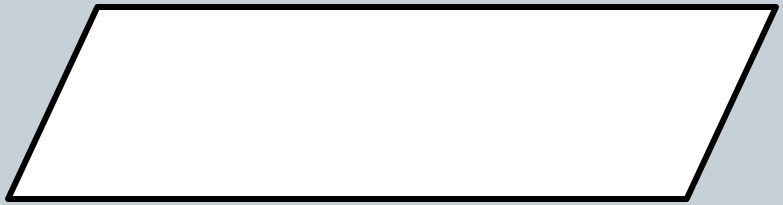
  Examples:

  “counter is incremented by 1 ";

  "save changes“ etc..

Flowchart Symbols

- **Input / Output**



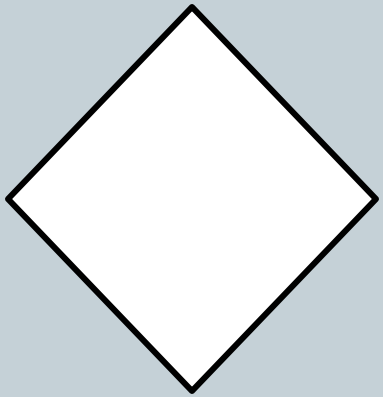
- Represented as a parallelogram.
- Examples:

  Read marks from the user;

  Display marks .

Flowchart Symbols

- **Conditional or decision**



- Represented as a diamond.
- These typically contain a Yes/No question or True/False test

Rules for Flowchart

1. Every flow chart has a START symbol and a STOP symbol.
2. The flow of sequence is generally from the top of the page to the bottom of the page.
3. Use arrow-heads defines the flow direction.
4. It is better to put one flowchart per page.
5. A page should have a page number and a title.
6. A It is better restrict around 15 symbols (not including START and STOP).

Advantages of Using Flowcharts

- Communication:
- Effective analysis:
- Proper documentation:

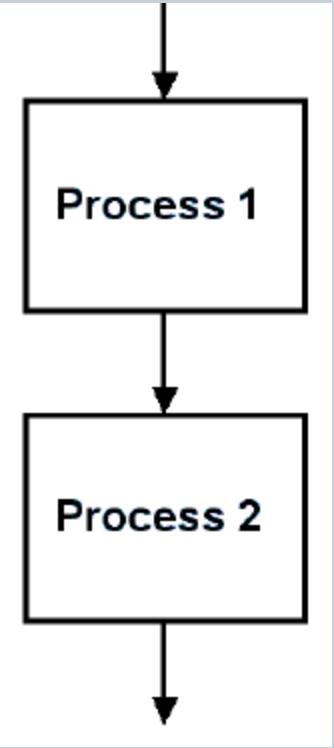
  Efficient Coding:
- Proper Debugging:
- Efficient Program Maintenance:

Basic Control Structures

- Sequence
- Selection
- Loop

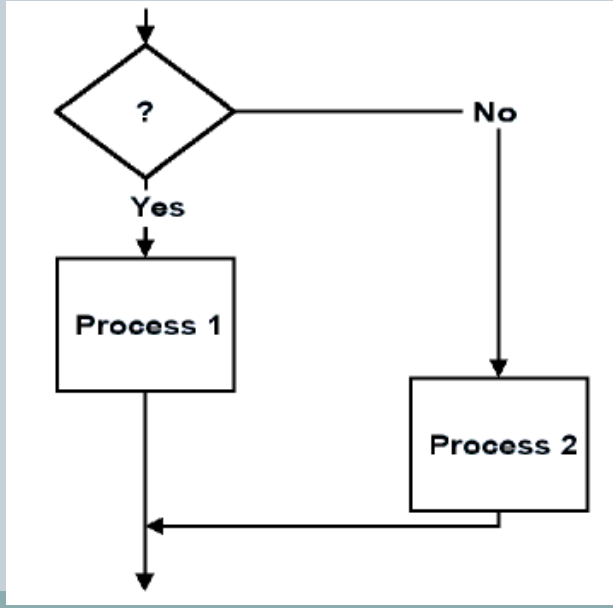
Sequence

- Steps that execute in sequence are represented by symbols that follow each other top to bottom or left to right.
- Top to bottom is the standard.



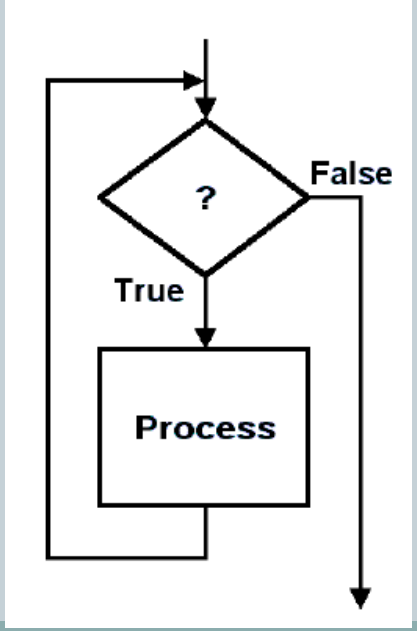
Selection

- Once the condition is evaluated, the control flows in to one of two paths.
- Once the conditional execution is finished, the flows rejoin before leaving the structure.



Loop

- Either the processing repeats or the control leaves the structure.



Algorithm

- **Problem :** Get the average value of two numbers
- **Algorithm:**

- Output: Ask user to enter two numbers
- Input: Read two numbers (Number1, Number2)
- Process: calculate average
- Output: the average value of the two numbers

Algorithm

- Process: calculate average
- Mathematical formula:

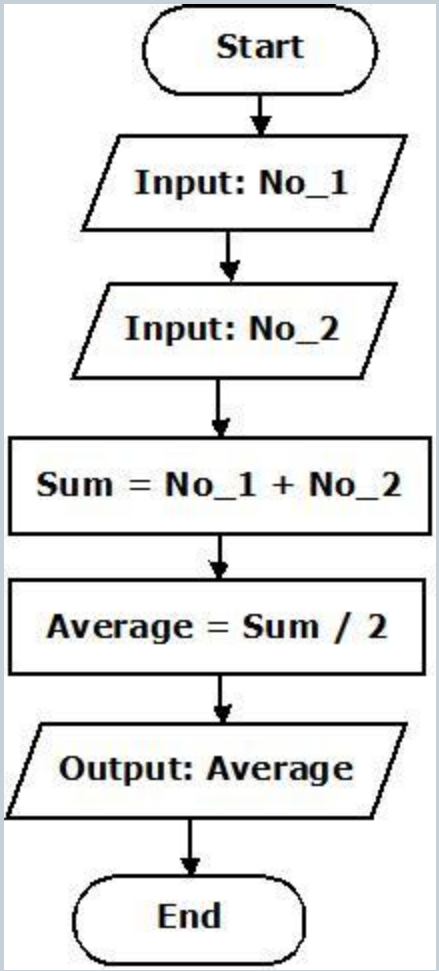
$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Algorithm

Steps:

1. Input Number1
2. Input Number1
3. Summation = Number1 + Number2
4. Average = Summation /2
5. Output Average

Algorithm


Start
Input: No_1
Input: No_2
Sum = No_1 + No_2
Average = Sum / 2
Output: Average
End

Exercise

- Draw a flowchart to find the largest of three numbers

C language

C language

- It is developed for creating system applications
- It is direct interacts to the hardware devices such as drivers, kernals etc.
- It is considered as the base for other programming languages (it is known as mother language).

C language

- It can be defined by following ways:

  Mother language

  - It considered as the mother language of all the modern languages
  - most of the compilers,(JVMs, Kernals etc.) are written in C
  - most of languages follows c syntax (e.g. C++, Java etc.)

  System programming language

  used to do low level programming (e.g. driver and kernel).

  used to create hardware devices, OS, drivers, kernels etc.

  example, linux kernel is written in C.

C language

Procedure-oriented programming language

- specifies a series of steps or procedures for the program to solve the problem.
- A procedural language breaks the program into functions, data structures etc.

Structured programming language

- Structure means to break a program into parts or blocks
- break the program into parts using functions

Mid level programming language

1. it supports the feature of both low-level and high level language.

A Simple C Program: Printing a Line of Text

```
#include <stdio.h>


int main(void)
{
printf("hello, world\n");
return 0;
}
```

A Simple C Program: Printing a Line of Text

- **#include <stdio.h>:** includes the standard input output library functions. The “printf()” function is defined in stdio.h .
- **int main()**: The main() function is the entry point of every program in c language.
- **printf() :** The printf() function is used to print data on the console.
- **return 0** : The return 0 statement, returns execution status to the OS.
  - “0” value is used for successful execution
  - “1” for unsuccessful execution.

comments

- single-line comments : //
- Block comment : begin with /* and end with */

A Simple C Program: Printing a Line of Text

```
/*------------------------------------------------------------
Title:  A first program in C
Author : AS XXXX
Date: 2017-12-20
Version: 1.0
Description: prints a line of text in consol
------------------------------------------------------------*/

#include <stdio.h>

// function main begins program execution
int main(void)
{
printf("hello, world\n");
return 0; // indicate that program ended successfully
} // end function main
```

common escape sequences

Escape sequence	Description
\n	Newline. Position the cursor at the beginning of the next line.
\t	Horizontal tab. Move the cursor to the next tab stop
\a	Alert. Sound the system bell.
\\	Backslash. Insert a backslash character in a string.
\”	Double quote. Insert a double-quote character in a string.

A Simple C Program: Printing a Line of Text

```
/*-------------------------------------------------------------
Title:  A first program in C
Author : AS XXXX
Date: 2017-12-20
Version: 1.0
Description: prints a line of text in consol
-------------------------------------------------------------*/

#include <stdio.h>

// function main begins program execution
int main(void)
{
printf( "Welcome " );
printf( "to C!\n" );
return 0; // indicate that program ended successfully
} // end function main
```

A Simple C Program: Printing a Line of Text

```
/*-------------------------------------------------------------
Title:  A first program in C
Author : AS XXXX
Date: 2017-12-20
Version: 1.0
Description: prints a line of text in consol
-------------------------------------------------------------*/

#include <stdio.h>

// function main begins program execution
int main(void)
{
printf( "Welcome\nto\n C!\n" );
return 0; // indicate that program ended successfully
} // end function main
```