Pseudocodes And Flowcharts

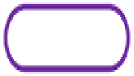
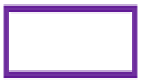
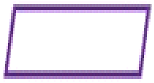
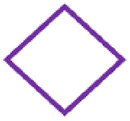
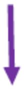
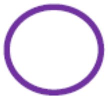
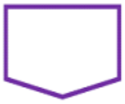
FLOWCHARTS

Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols to depict processes which are connected among them to indicate the flow of information and processing.
The process of drawing a flowchart for an algorithm is known as "flowcharting".

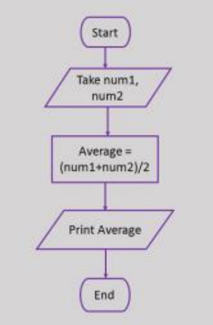
These are some points to keep in mind while developing a flowchart

- Flowchart can have only one start and one stop symbol
- General flow of processes is top to bottom or left to right
- Arrows should not cross each other

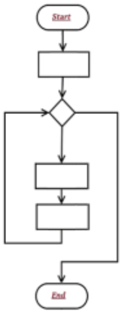
Basic Symbols used in Flowchart Designs:

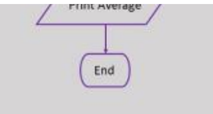
Symbol	Symbol Name	Purpose
	Start/Stop	Used at the beginning and end of the algorithm to show start and end of the program. The oval symbol indicates Start, Stop and Halt in a program's logic flow. A pause/halt is generally used in a program logic under some error conditions. Terminal is the first and last symbols in the flowchart
	Process	Indicates processes like mathematical operations A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.
	Input/ Output	Used for denoting program inputs and outputs. A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.
	Decision	Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.
	Arrow	Shows relationships between different shapes. Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.
	On-page Connector	Connects two or more parts of a flowchart, which are on the same page. Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.
	Off-page Connector	Connects two parts of a flowchart which are spread over different pages.

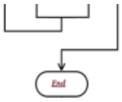
Example Of Flow Chart



Sample Of A Flow Chart





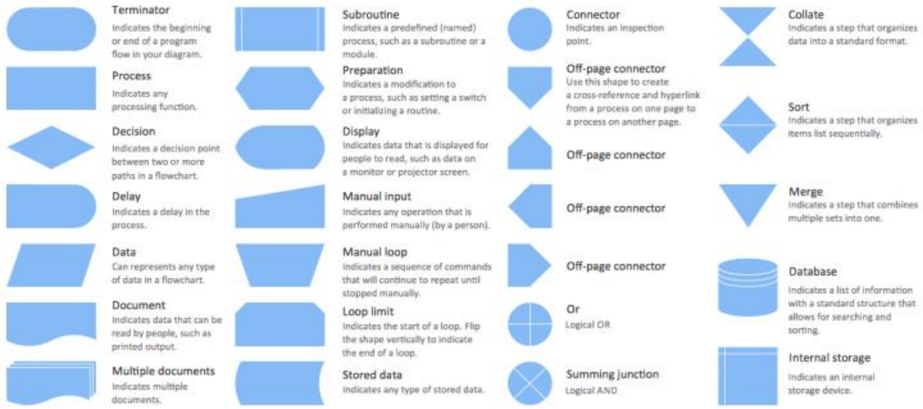


Advantages of Flowchart:

- Flowcharts are better way of communicating the logic of system.
  As it provides the pictorial representation of the steps; therefore, it simplifies the logic and subsequent steps.
- Makes Communication Better
  Because of having easily understandable pictorial logic and steps, it is a better and simple way of representation.
- Flowcharts act as a guide for blueprint during program designed.
- Useful in Coding
  The flow-chart also helps in coding process efficiently, as it gives directions on what to do, when to do, and where to do. It makes the work easier
- Flowcharts helps in debugging process.
  flowchart also helps in finding the error (if any) in program
- With the help of flowcharts programs can be easily analyzed.
  Once the flow-chart is prepared, it becomes very simple to analyze the problem in an effective way.
- It provides better documentation.
  a flowchart also helps in preparing the proper document
- Flowcharts serve as a good proper documentation.

Disadvantages of Flowchart:

- It is difficult to draw flowchart for large and complex programs.
- In this there is no standard to determine the amount of detail.
- Difficult to reproduce the flowcharts.
- It is very difficult to modify the Flowchart


Terminator
Indicates the beginning or end of a program flow in your diagram.
Process
Indicates any processing function.
Decision
Indicates a decision point between two or more paths in a flowchart.
Delay
Indicates a delay in the process.
Data
Can represents any type of data in a flowchart.
Document
Indicates data that can be read by people, such as printed output.
Multiple documents
Indicates multiple documents.
Subroutine
Indicates a predefined (named) process, such as a subroutine or a module.
Preparation
Indicates a modification to a process, such as setting a switch or initializing a routine.
Display
Indicates data that is displayed for people to read, such as data on a monitor or projector screen.
Manual input
Indicates any operation that is performed manually (by a person).
Manual loop
Indicates a sequence of commands that will continue to repeat until stopped manually.
Loop limit
Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.
Stored data
Indicates any type of stored data.
Connector
Indicates an inspection point.
Off-page connector
Use this shape to create a cross-reference and hyperlink from a process on one page to a process on another page.
Off-page connector
Off-page connector
Off-page connector
Or
Logical OR
Summing junction
Logical AND
Collate
Indicates a step that organizes data into a standard format.
Sort
Indicates a step that organizes items list sequentially.
Merge
Indicates a step that combines multiple sets into one.
Database
Indicates a list of information with a standard structure that allows for searching and sorting.
Internal storage
Indicates an internal storage device.

PSEUDOCODES

----> Plain english where you logically explain your algorithm

Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. Pseudocode summarizes a program's flow, but excludes underlying details. System designers write pseudocode to ensure that programmers understand a software project's requirements and align code accordingly.

Pseudo code is a term which is often used in programming and algorithm based fields. It is a methodology that allows the programmer to represent the implementation of an algorithm. Simply, we can say that it's the cooked up representation of an algorithm. Often at times, algorithms are represented with the help of pseudo codes as they can be interpreted by programmers no matter what their programming background or knowledge is

Pseudo code: It's simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.

Advantages of Pseudocode

- Improves the readability of any approach. It's one of the best approaches to start implementation of an algorithm.
- Acts as a bridge between the program and the algorithm or flowchart. Also works as a rough documentation, so the program of one developer can be understood easily when a pseudo code is written out. In industries, the approach of documentation is essential. And that's where a pseudo-code proves vital.
- The main goal of a pseudo code is to explain what exactly each line of a program should do, hence making the code construction phase easier for the

program.

- it enables the programmer to concentrate only on the algorithm part of the code development.

How to write a Pseudo-code?

1. Arrange the sequence of tasks and write the pseudocode accordingly.
2. Start with the statement of a pseudo code which establishes the main goal or the aim
3. The way the if-else, for, while loops are indented in a program, indent the statements likewise, as it helps to comprehend the decision control and execution mechanism. They also improve the readability to a great extent.
4. Use appropriate naming conventions. The human tendency follows the approach to follow what we see. If a programmer goes through a pseudo code, his approach will be the same as per it, so the naming must be simple and distinct.
5. Use appropriate sentence casings, such as CamelCase for methods, upper case for constants and lower case for variables.
6. Elaborate everything which is going to happen in the actual code. Don't make the pseudo code abstract.
7. Use standard programming structures such as 'if-then', 'for', 'while', 'cases' the way we use it in programming.
8. Check whether all the sections of a pseudo code is complete, finite and clear to understand and comprehend.
9. Don't write the pseudo code in a complete programmatic manner. It is necessary to be simple to understand even for a layman or client, hence don't incorporate too many technical terms.

Example Of Pseudocode:

```
// this program will compute area and perimeter of rectangle  ---> always tell at the start what the program is about.
Begin
* Display please provide length
input Length
* Display please provide breadth
input Breadth
// constant PI
variable Area
variable Perimeter

Area = Length * Breadth
Perimeter = 2*Length + 2*Breadth

Print Area
Print Perimeter

End
```

Variable ----> value that keeps on changing during the program

Constants -----> value that does not change during the program run

A pseudocode must be readable not only for programmers/coders but also for intended audience (includes clients and non technicals)

Remember it is best to keep a pseudocode simple and consized