

Computer Fundamentals

Flowchart

Professor Dr. M. Ismail Jabiullah

Professor

Department of CSE

Daffodil International University

Bangladesh

Flowchart

Topics

- Problem Analysis
- Algorithm
- Characteristics of an Algorithm
- Algorithm Build-up Process
- Some Algorithms
- Flowcharts
- Needs for drawing Flowcharts
- Rules for drawing flowcharts
- Symbols of Flowcharts
- Examples of Flowcharts
- Advantages of drawing Flowcharts
- Limitations of drawing Flowcharts
- Pseudocode
- Examples
- Advantages of pseudocode
- Limitations of pseudocode
- Four Steps for Programming Process
- Source Program, Object Program and Executable Program

Problem Analysis

- A program is a set of instructions or command written in a computer programming language understandable by the computer system.
- To prepare a program, one has to prepare an algorithm, then to prepare a flowchart and finally, to prepare pseudo code of that program.

- The steps of the work are given below.
 - ❑ Write the sequence of tasks
 - ❑ Prepare the algorithm
 - ❑ Prepare the flowchart
 - ❑ Prepare the pseudocode
 - ❑ Write the program code

Algorithm

- To prepare an algorithm, one have to first prepare a program logic that is the correct sequence of instructions or commands needed to solve the problem at hand.
- The term algorithm is often used to refer to the logic of a program.
- It is a step-by-step instructions or commands of the problem description of how to arrive at the solution of the given computer problem.
- So, an algorithm is a sequence of instructions designed in a manner that if the instructions are executed in the specified sequence, the desired results will be obtained.

Characteristics of an Algorithm

- **Finiteness:** It must be finite.
- **Definiteness:** It must be unambiguous.
- **Non-intuitiveness:** It must be executable.
- **Input:** It must have 0 or more input.
- **Output:** It must have 1 or more output.
- **Completeness/generalality:** It should be general so that it can solve any problem of a particular type for which it is constructed.

Algorithm Build-up Process

There are various ways in which an algorithm can be represented. Programmers normally use one or more of the following ways to build-up an algorithm:

- As programs
- As flowcharts
- As pseudo codes

Examples

Problem 1: Write an algorithm that finds the sum and the subtraction of two numbers that are given.

Algorithm

Part A:

To find the sum and subtract of two given numbers. Here, SUM and SUB are the two variables where intermediary values are put.

Part B:

Step 1. Set the values $X = 20$, and $Y = 15$.

Step 2. Compute $X + Y$ and put it to the variable SUM.

Step 3. Compute $X - Y$ and put it to the variable SUB.

Step 4. Write the values of SUM and SUB.

Step 5. Stop.

Problem 2: Write an algorithm that finds the sum, subtract and product of two numbers where numbers are taken from the keyboard as input.

Algorithm

Part A:

To find the sum, subtract and product of two numbers X, Y taken from the keyboard as input. Take SUM, SUB and PRODUCT to store the intermediary values.

Part B:

Step 1. Input two numbers X, Y.

Step 2. Compute $X + Y$ and put it to the variable SUM.

Step 3. Compute $X - Y$ and put it to the variable SUB.

Step 4. Compute $X * Y$ and put it to the variable PRODUCT.

Step 5. Write the value of SUM, SUB and PRODUCT.

Step 6. Stop.

Flowcharts

- A flowchart is a pictorial representation of an algorithm.
- It is often used by the programmers as program-planning tool for organizing a sequence of steps necessary to solve a computer problem by a computer system.
- It uses boxes of different shapes to denote different types of instructions.
- The actual instructions are written within these boxes using clear and concise statements.
- These boxes are connected by solid lines having arrow marks to indicate the flow of operations, that is, the exact sequences in the instructions are to be executed.
- The process of drawing a flowchart for an algorithm is often referred to as flowcharting.

Needs for Drawing Flowcharts

Flowchart is drawn, because

- It is easier to understand at a glance than a narrative description.
- The flowchart assists the programmer when he actually starts writing the program.
- If the programmer himself or someone else wishes to correct or modify the program after sometime, the flowchart may be more clear and easy to understand than the actual program.
- Provide effective program documentation.

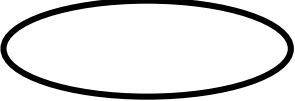




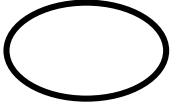


Rules for Drawing Flowcharts

There are some rules to draw flowcharts.

To deal a good flowchart, programmers need to maintain a number of general rules and guidelines and they are given below:

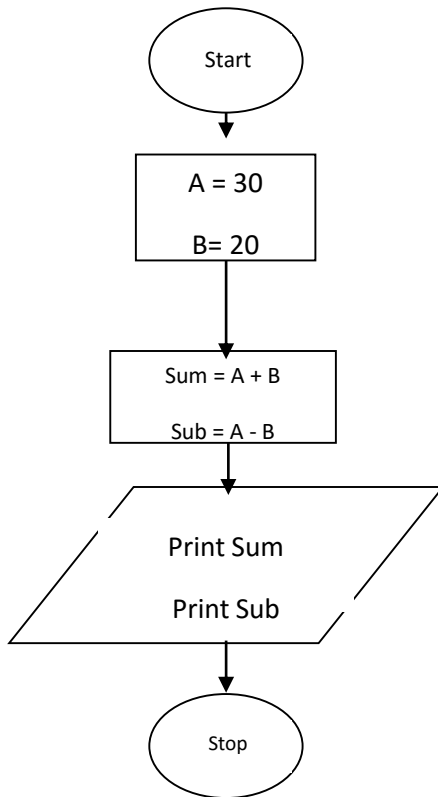
- First chart the main line of logic, then incorporate detail.
- Maintain a consistent level of detail for a given flowchart.
- Do not chart every detail, otherwise, the flowchart will only be a step-by-step graphic representation of the program.
- Words in the flowchart symbols should be common statements, which are easy to understand.
- It is recommended to use descriptive titles written in designer's own language, rather than in machine-oriented language.
- Be consistent in using names and variables in the flowchart.
- Go from left to right, and top to bottom in constructing flowcharts.
- Keep the flowchart as simple as possible.
- The crossing of flow lines should be avoided, as far as practicable.

Symbols of Flowcharts

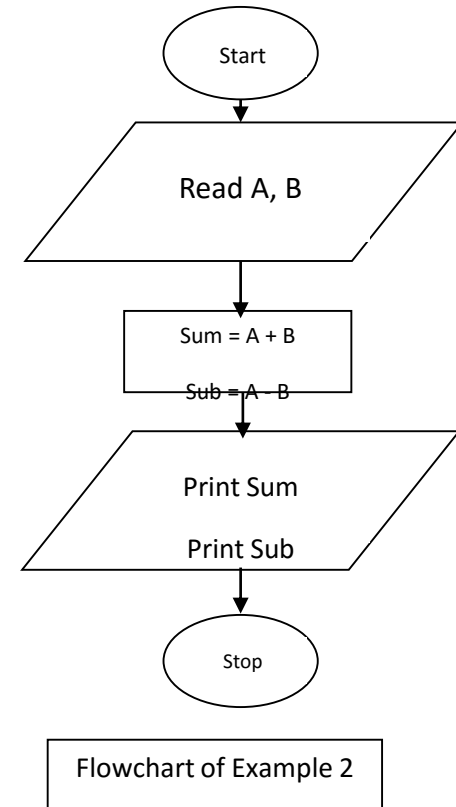
	Shape	Name
	Oval	Terminal
	Rectangle	Assertion or Process
	Parallelogram	Input or output
	Diamond	Decision
	Hexagon	Preparation instruction or Group
	Circle	Connector
	Capsule	Start or stop
	Line	Flow

Examples of Flowchart

Example: Draw a flowchart that finds the sum and the subtraction of two numbers that are given.



Flowchart of Example 1



Flowchart of Example 2

Example: Draw a flowchart that finds the sum and the subtraction of two numbers that are taken from the keyboard as input.

Advantages of Drawing Flowcharts

Drawing flowcharts have some benefits and they are described in the following:

Better Communication: A flowchart is a pictorial representation of a program. Hence, it is easier for a programmer to explain the logic of a program to some other programmer, or his/her boss through a flowchart, rather than the program itself.

Proper Program Documentation: Program documentation involves collecting, organizing, storing, and otherwise maintaining a complete historical record of programs, and the other documents associated with a system.

Efficient Coding: Once a flowchart is ready, programmers find it very easy to write the corresponding program, because the flowchart acts as a road map for them. It guides them to go from the starting point of the program to the final point, ensuring that no steps are omitted. The ultimate result is an error-free program, developed at a faster rate.

Systematic Debugging: A flowchart is very helpful in detecting, locating, and removing mistakes or bugs in a program in a systematic manner, because programmers find it easier to follow the logic of the program in flowchart form.

Systematic Testing: Testing is the process of confirming whether a program will successfully do all the jobs for which it has been designed under the specified constraints. A flowchart proves to be very helpful in designing the test data for systematic testing of programs.

Limitations of Drawing Flowcharts

- Drawing flowcharts have some limitations and they are described in the following:
- Flowcharts are very time consuming, and laborious to draw with proper symbols and spacing, especially for large complex programs.
- Owing to the symbol-string nature of flowcharting, any changes or modifications in the program logic will usually require a completely new flowchart. Redrawing a flowchart being a tedious task, many programmers do not redraw or modify the corresponding flowchart when they modify the programs.
- There are no standards determining the amount of detail that should be included in a flowchart.

Pseudocode

- Pseudocode is the programming analysis tool which is used for planning program logic.
- “Pseudo” means imitation or false, and “Code” refers to the instructions written in a programming language.
- Pseudocode, therefore, is an imitation of actual computer instructions.
- These pseudo-instructions are phrases written in ordinary natural language, which cannot be understood by the computer.

Example of Pseudocode

Write an algorithm that finds the sum, subtract and product of two numbers where numbers are taken from the keyboard as input.

Pseudocode

Input two numbers X, Y.

Compute $X + Y$ and put it to the variable SUM.

Compute $X - Y$ and put it to the variable SUB.

Compute $X * Y$ and put it to the variable PRODUCT.

Write the value of SUM, SUB and PRODUCT.

Advantages of Pseudocode

The identified advantages of pseudocodes are given below:

- Converting a Pseudocode to a programming language is much more easier than converting a flowchart to a programming language.
- As compared to a flowchart, it is easier to modify the pseudocode of a program logic, when program modifications are necessary.
- Writing of pseudocode involves much less time and effort than drawing an equivalent flowchart.
- Pseudocode is easier to write than an actual programming language, because it has only a few rules to follow, allowing the programmer to concentrate on the logic of the program.

Limitations of Pseudocode

The identified limitations of pseudocodes are given below:

- In case of pseudocode, a graphic representation of program logic is not available.
- There are no standard rules to follow in using pseudocode.
- Different programs use their own style of writing pseudocode. Hence, communication problem occurs due to lack of standardization.
- For a beginner, it is more difficult.

Four Steps for Programming Process

The process of program development falls within the implementation steps of the systems development life cycle.

Parallel to the purchase, testing, and installation of hardware is the development of software.

The programming process follows a four-step structure of its own, producing application programs that implement the processing portion of a computer system.

The four-steps are:

- Designing the problem
- Writing program code
- Testing and debugging
- Documentation and training

Some Terms

Source Program

A program written in assembly language or high level programming language is known as source program.

Object Program

Any program not written in machine language has to be translated before it is executed by the computer. Object program is a translation of source program into machine language program.

Executable Program

Any program that is to be run by the compiler or editor of the programming language is converted into an executable program that can be run in any computer to perform the desired tasks.

Differences between Testing and Debugging

Testing	Debugging
(1) Testing is the process of validating the correctness of a program.	(1) Debugging is the process of eliminating errors in a program.
(2) Its objective is to demonstrate that the program meets its design specifications.	(2) Its objective is to detect the exact cause of, and remove known errors in the program
(3) Testing is complete when all desired verifications against specifications have been performed.	(3) Debugging is complete when all known errors in the program have been fixed. Debugging process ends only temporarily, because it must be restarted whenever a new error is found in the program.
(4) Testing is a definable process, which can and should be planned and scheduled properly.	(4) Debugging, being a reactive process, cannot be planned ahead of time. It must be carried out as and when errors are found in a program.
(5) Testing can begin in the early stages of software development.	(5) Debugging can begin only after the program is coded.

Thanks