

1

ALGORITHMS AND FLOW CHARTS

by
Dr. Sumit Srivastava
Dept. of Computer Science & Engineering

Unit -I CS101 PPS @Sumit

1

Introduction

2

- Intelligence is one of the key characteristics which differentiate a human being from other living creatures on the earth. Basic intelligence covers day to day problem solving and making strategies to handle different situations which keep arising in day to day life.

CS101 PPS @Sumit

2

Introduction

3

- One person goes Bank to withdraw money. After knowing the balance in his account, he/she decides to withdraw the entire amount from his account, but he/she has to leave minimum balance in his account.

CS101 PPS @Sumit

3

Introduction

4

- Here deciding about how much amount he/she may withdraw from the account is one of the example of the basic intelligence. During the process of solving any problem, one tries to find the necessary steps to be taken in a sequence.

CS101 PPS @Sumit

4

Problem Solving Approach

5

- Identify the problem.
- Understand the problem.
- Identify alternative ways to solve problem.
- Select best alternative.
- List solution steps for alternative chosen.
- Evaluate solution.

CS101 PPS @Sumit

5

Program Development Life Cycle (PDLC)

6

- When we want to develop a program by using any programming language, we have to **follow a sequence of steps**. These steps are called phases in program development.

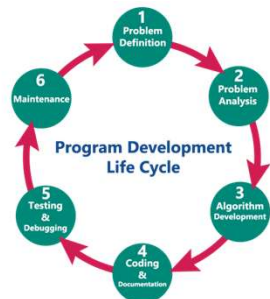
The program development life cycle is a **set of steps or phases** which are used to develop a program in any programming language.

CS101 PPS @Sumit

6

Phases of Program Development

7



CS101 PPS @Sumit

7

Phases of Program Development

8

1. Problem Definition

In this phase, we **define the problem statement**, and we decide the boundaries of the problem. In this phase we need to understand the problem statement, what is our requirement, what should be the output of the problem solution. These are defined in this first phase of the program development life cycle.

2. Problem Analysis

In phase 2, we determine the requirements like variables, functions, etc. to solve the problem. That means we **gather the required resources** to solve the problem defined in the problem definition phase. We also determine the bounds of the solution.

3. Algorithm Development

During this phase, we **develop a step-by-step procedure** to solve the problem using the specification given in the previous phase. This phase is very important for program development. That means we write the solution in step-by-step statements.

CS101 PPS @Sumit

8

Phases of Program Development

9

4. Coding & Documentation

This phase uses a programming language to write or **implement the actual programming instructions** for the steps defined in the previous phase. In this phase, we construct the actual program.

5. Testing & Debugging

During this phase, we check whether the code written in the previous step is solving the specified problem or not. That means we **test the program** whether it is solving the problem for various input data values or not. We also test whether it is providing the desired output or not.

6. Maintenance

During this phase, the **program is actively used by the users**. If any enhancements found in this phase, all the phases are to be repeated to make the enhancements. That means in this phase, the solution (program) is used by the end-user. If the user encounters any problem or wants any enhancement, then we need to repeat all the phases from the starting, so that the encountered problem is solved or enhancement is added.

CS101 PPS @Sumit

9

Steps for Writing a Program

10

□ Algorithm

□ Pseudo code

□ Flowchart

□ Program

CS101 PPS @Sumit

10

ALGORITHM

CS101 PPS @Sumit

11

What is Algorithm?

12

- Algorithm can be defined as: "A sequence of activities to be processed for getting desired output from a given input."
- "A formula or set of steps for solving a particular problem. To be an algorithm, a set of rules must be unambiguous and a clear stopping point".

CS101 PPS @Sumit

12

Algorithm

13

- First Algorithm was written by “Ada Lovelace” in 1843.
- It is a step procedures to solve logical and mathematical problems.
- It is building blocks for programming.

CS101 PPS @Sumit

13

Constructions of Algorithm

14

1. **Sequence (Linear)**
Flow of task one after another
2. **Selection (Conditional)**
If-Then-Else Decision
3. **Repetition (Loop)**
While & For

CS101 PPS @Sumit

14

Properties of Algorithm

15

- Input Specified
- Output Specified
- Definite
- Effective
- Finite

CS101 PPS @Sumit

15

Properties of Algorithm

16

Donald Ervin Knuth has given a list of five properties for an algorithm, these properties are:

- FINITENESS
- DEFINITENESS
- INPUT
- OUTPUT
- EFFECTIVENESS

CS101 PPS @Sumit

16

Properties of Algorithm

17

1. FINITENESS:

An algorithm must always terminate after a finite number of steps. It means after every step one reach closer to solution of the problem and after a finite number of steps algorithm reaches to an end point.

CS101 PPS @Sumit

17

Properties of Algorithm

18

2. DEFINITENESS

Each step of an algorithm must be precisely defined. It is done by well thought actions to be performed at each step of the algorithm. Also the actions are defined unambiguously for each activity in the algorithm.

CS101 PPS @Sumit

18

Properties of Algorithm

19

3. INPUT:

Any operation you perform need some beginning value/quantities associated with different activities in the operation. So the value/quantities are given to the algorithm before it begins.

CS101 PPS @Sumit

19

Properties of Algorithm

20

4. OUTPUT:

One always expects output/result (expected value/quantities) in terms of output from an algorithm. The result may be obtained at different stages of the algorithm. If some result is from the intermediate stage of the operation, then it is known as intermediate result and result obtained at the end of algorithm is known as end result. The output is expected value/quantities always have a specified relation to the inputs.

CS101 PPS @Sumit

20

Properties of Algorithm

21

5. EFFECTIVENESS:

Algorithms to be developed/written using basic operations. Actually, operations should be basic, so that even they can in principle be done exactly and in a finite amount of time by a person, by using paper and pencil only.

CS101 PPS @Sumit

21

Properties of Algorithm

22

- Any algorithm should have **all these five properties** otherwise it will not fulfil the basic objective of solving a problem in finite time.
- Every step of an algorithm puts you closer to the solution

CS101 PPS @Sumit

22

Algorithm Examples

23

- Let us take one simple day-to-day example by writing algorithm for making, "Maggi Noodles" as a food.

- | | |
|----------|--|
| Step 1: | Start |
| Step 2: | Take pan with water |
| Step 3: | Put pan on the burner |
| Step 4: | Switch on the gas/burner |
| Step 5: | Put maggi and masala |
| Step 6: | Give two minutes to boil |
| Step 7: | Take off the pan |
| Step 8: | Take out the maggi with the help of fork/spoon |
| Step 9: | Put the maggi on the plate and serve it |
| Step 10: | Stop. |

CS101 PPS @Sumit

23

Algorithm Examples

24

- Write an algorithm to print "Good Morning"

- | | |
|---------|----------------------|
| Step 1: | Start |
| Step 2: | Print "Good Morning" |
| Step 3: | Stop |

CS101 PPS @Sumit

24

Algorithm Examples

25

- Write an algorithm to find area of a rectangle.

Step 1: Start
 Step 2: Take length and breadth and store them as L and B?
 Step 3: Multiply by L and B and store it in area
 Step 4: Print area
 Step 5: Stop

CS101 PPS @Sumit

25

Algorithm Examples

26

- Write an algorithm to check whether he is eligible to vote? (more than or equal to 18 years old).

Step 1: Start
 Step 2: Take age and store it in age
 Step 3: Check age value, if age \geq 18 then go to step 4 else step 5
 Step 4: Print "Eligible to vote" and go to step 6
 Step 5: Print "Not eligible to vote"
 Step 6: Stop

CS101 PPS @Sumit

26

Algorithm Examples

27

- Write an algorithm to check whether given number is +ve, -ve or zero.

Step 1: Start
 Step 2: Take any number and store it in n.
 Step 3: Check n value, if $n > 0$ then go to step 5 else go to step 4
 Step 4: Check n value, if $n < 0$ then go to step 6 else go to step 7
 Step 5: Print "Given number is +ve" and go to step 8
 Step 6: Print "Given number is -ve" and go to step 8
 Step 7: Print "Given number is zero"
 Step 8: Stop

CS101 PPS @Sumit

27

Algorithm Examples

28

- Write an algorithm to check whether given number is +ve, -ve or zero.

Step 1: Start
 Step 2: Take any number and store it in n.
 Step 3: Check n value, if $n > 0$ then go to step 5 else go to step 4
 Step 4: Check n value, if $n < 0$ then go to step 6 else go to step 7
 Step 5: Print "Given number is +ve" and go to step 8
 Step 6: Print "Given number is -ve" and go to step 8
 Step 7: Print "Given number is zero"
 Step 8: Stop

CS101 PPS @Sumit

28

Pseudo Code

CS101 PPS @Sumit

29

Pseudo Code

30

- Used to describe How Algorithm should work.
- It is an informal way of programming
- It summarizes how program will flow.

CS101 PPS @Sumit

30

Example

31

□ Write a pseudo code for addition of two numbers.

1. Start Program
2. Enter two numbers A, B
3. Add the numbers together
4. Print Sum
5. End Program

CS101 PPS @Sumit

31

FLOW CHART

CS101 PPS @Sumit

32

Flowchart

33

- The flowchart is a diagram which visually presents the flow of data through processing systems.
- This means by seeing a flow chart one can know the operations performed and the sequence of these operations in a system.
- Algorithms are nothing but sequence of steps for solving problems. So flow chart can be used for representing an algorithm.
- A flowchart, will describe the operations (and in what sequence) are required to solve a given problem.

CS101 PPS @Sumit

33

Flowchart

34

- A flowchart is a type of diagram that represents an algorithm, workflow or process.
- The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.
- This diagrammatic representation illustrates a solution model to a given problem.
- Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

CS101 PPS @Sumit

34

Flowchart

35

- Frank Gilbert is father of Flowchart.
- It is simply a graphical representation of steps.
- Used in presenting the flow of algorithm.
- 3 types of Flowchart
 1. Process Flowchart
 2. Data Flowchart
 3. Business Process Modelling

CS101 PPS @Sumit

35

BUILDING BLOCKS OF FLOW CHART OR COMMON SYMBOLS OF FLOW CHART

CS101 PPS @Sumit

36

Building Blocks Of Flowchart

37


- ❑ The American National Standards Institute (ANSI) set standards for flowcharts and their symbols in the 1960s.
- ❑ The International Organization for Standardization (ISO) adopted the ANSI symbols in 1970.
- ❑ The current standard was revised in 1985. Generally, flowcharts flow from top to bottom and left to right.

CS101 PPS @Sumit

37

Building Blocks Of Flowchart

38

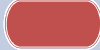
ANSI/ISO Shape	Name	Description
	Flowline (Arrowhead)	Shows the process's order of operation. A line coming from one symbol and pointing at another. Arrowheads are added if the flow is not the standard top-to-bottom, left-to-right.

CS101 PPS @Sumit

38

Building Blocks Of Flowchart

39


ANSI/ISO Shape	Name	Description
	Terminal	Indicates the beginning and ending of a program or sub-process. Represented as a stadium, oval or rounded (fillet) rectangle. They usually contain the word "Start" or "End", or another phrase signaling the start or end of a process, such as "submit inquiry" or "receive product".

CS101 PPS @Sumit

39

Building Blocks Of Flowchart

40


ANSI/ISO Shape	Name	Description
	Process	Represents a set of operations that changes value, form, or location of data. Represented as a rectangle

CS101 PPS @Sumit

40

Building Blocks Of Flowchart

41


ANSI/ISO Shape	Name	Description
	Decision	Shows a conditional operation that determines which one of the two paths the program will take. The operation is commonly a yes/no question or true/false test. Represented as a diamond (rhombus).

CS101 PPS @Sumit

41

Building Blocks Of Flowchart

42

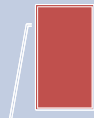
ANSI/ISO Shape	Name	Description
	Input / Output	Indicates the process of inputting and outputting data, as in entering data or displaying results. Represented as a parallelogram

CS101 PPS @Sumit

42

Building Blocks Of Flowchart

43


ANSI/ISO Shape	Name	Description
	Annotation (Comment)	Indicating additional information about a step the program. Represented as an open rectangle with a dashed or solid line connecting it to the corresponding symbol in the flowchart.

CS101 PPS @Sumit

43

Building Blocks Of Flowchart

44


ANSI/ISO Shape	Name	Description
	Predefined Process Functions	Shows named process /Function which is defined elsewhere. Represented as a rectangle with double-struck vertical edges.

CS101 PPS @Sumit

44

Building Blocks Of Flowchart

45


ANSI/ISO Shape	Name	Description
	On-page Connector	Shows named process /Function which is defined elsewhere. Represented as a rectangle with double-struck vertical edges.

CS101 PPS @Sumit

45

Building Blocks Of Flowchart

46


ANSI/ISO Shape	Name	Description
	Off-page Connector	A labelled connector for use when the target is on another page. Represented as a home plate-shaped pentagon.

CS101 PPS @Sumit

46

Building Blocks Of Flowchart

47


ANSI/ISO Shape	Name	Description
	Storage/ Backup	Magnetic Tape used for secondary storage/Backup

CS101 PPS @Sumit

47

Building Blocks Of Flowchart

48

ANSI/ISO Shape	Name	Description
	Storage/ Backup	Magnetic Disk used for secondary storage/Backup

CS101 PPS @Sumit

48

Advantages Of Using Flowcharts

49

□ Flow chart is used for representing algorithm in pictorial form.

This pictorial representation of a solution/system is having many advantages. These advantages are as follows:

1. Communication.
2. Effective analysis.
3. Documentation of program/system.
4. Efficient program maintenance.
5. Coding of the program.

CS101 PPS @Sumit

49

Advantages Of Using Flowcharts

50

1. COMMUNICATION:

A Flowchart can be used as a better way of communication of the logic of a system and steps involve in the solution, to all concerned particularly to the client of system.

CS101 PPS @Sumit

50

Advantages Of Using Flowcharts

51

2. EFFECTIVE ANALYSIS:

A flowchart of a problem can be used for effective analysis of the problem.

CS101 PPS @Sumit

51

Advantages Of Using Flowcharts

52

3. DOCUMENTATION OF PROGRAM/ SYSTEM:

Program flowcharts are a vital part of a good program documentation. Program document is used for various purposes like knowing the components in the program, complexity of the program etc.

CS101 PPS @Sumit

52

Advantages Of Using Flowcharts

53

4. EFFICIENT PROGRAM MAINTENANCE:

Once a program is developed and becomes operational it needs time to time maintenance. With help of flowchart maintenance become easier.

CS101 PPS @Sumit

53

Advantages Of Using Flowcharts

54

5. CODING OF THE PROGRAM:

Any design of solution of a problem is finally converted into computer program. Writing code referring the flowchart of the solution become easy.

CS101 PPS @Sumit

54



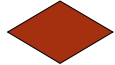

Limitations of Using Flowcharts

- COMPLEXITY OF LOGIC:** If program logic is complex then flowchart of the program becomes complicated.
- ALTERATIONS AND MODIFICATIONS IN LOGIC:** any alterations in the program logic may require redrawing of flowchart completely.
- REUSE IS NOT POSSIBLE:** As the flowchart symbols cannot be typed, always reproduction of flowchart symbols are required.

CS101 PPS @Sumit

55



Flowchart: basic symbols

-  → **Computation**
-  → **Input / Output**
-  → **Decision Box**
-  → **Start / Stop**

CS101 PPS @Sumit

56

Flowchart: basic symbols

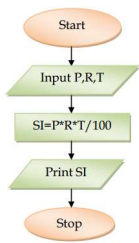
-  → **Flow of control**
-  → **Connector**

CS101 PPS @Sumit

57

Flowchart Example

□ Draw a flowchart to find the simple interest. (Sequence)

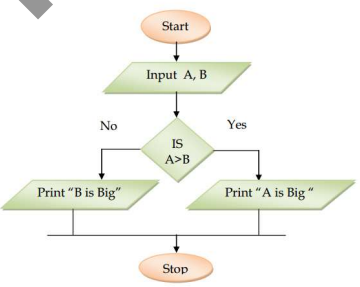


CS101 PPS @Sumit

58

Flowchart Example

□ Draw a flowchart to find bigger number among two numbers (selective)

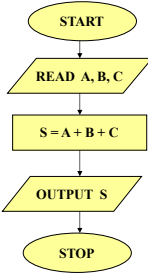


PS @Sumit

59

Flowchart Example

□ Draw a flowchart to add three numbers



CS101 PPS @Sumit

60

Flowchart Example

61

□ Draw a flowchart to find Larger of two numbers

```

    graph TD
      Start([START]) --> Read[/READ X, Y/]
      Read --> IsXgtY{IS X > Y?}
      IsXgtY -- YES --> OutputX[/OUTPUT X/]
      OutputX --> Stop1([STOP])
      IsXgtY -- NO --> OutputY[/OUTPUT Y/]
      OutputY --> Stop2([STOP])
  
```

CS101 PPS @Sumit

61

Flowchart Example

62

□ Draw a flowchart to find the Sum of first N natural numbers

```

    graph TD
      Start([START]) --> Read[/READ N/]
      Read --> Init[SUM = 0  
COUNT = 1]
      Init --> Loop[SUM = SUM + COUNT  
COUNT = COUNT + 1]
      Loop --> IsCountgtN{IS COUNT > N?}
      IsCountgtN -- NO --> Loop
      IsCountgtN -- YES --> OutputSum[/OUTPUT SUM/]
      OutputSum --> Stop([STOP])
  
```

CS101 PPS @Sumit

62

Flowchart Example

63

□ Draw a flowchart to find the $SUM = 1^2 + 2^2 + 3^2 + N^2$

```

    graph TD
      Start([START]) --> Read[/READ N/]
      Read --> Init[SUM = 0  
COUNT = 1]
      Init --> Loop[SUM = SUM + COUNT * COUNT  
COUNT = COUNT + 1]
      Loop --> IsCountgtN{IS COUNT > N?}
      IsCountgtN -- NO --> Loop
      IsCountgtN -- YES --> OutputSum[/OUTPUT SUM/]
      OutputSum --> Stop([STOP])
  
```

CS101 PPS @Sumit

63

Flowchart Example

64

□ Draw a flowchart to find the $SUM = 1.2 + 2.3 + 3.4 + \dots + N$ terms

```

    graph TD
      Start([START]) --> Read[/READ N/]
      Read --> Init[SUM = 0  
COUNT = 1]
      Init --> Loop[SUM = SUM + COUNT * (COUNT + 1)  
COUNT = COUNT + 1]
      Loop --> IsCountgtN{IS COUNT > N?}
      IsCountgtN -- NO --> Loop
      IsCountgtN -- YES --> OutputSum[/OUTPUT SUM/]
      OutputSum --> Stop([STOP])
  
```

CS101 PPS @Sumit

64

Flowchart Example

65

□ Draw a flowchart for Computing Factorial

```

    graph TD
      Start([START]) --> Read[/READ N/]
      Read --> Init[PROD = 1  
COUNT = 1]
      Init --> Loop[PROD = PROD * COUNT  
COUNT = COUNT + 1]
      Loop --> IsCountgtN{IS COUNT > N?}
      IsCountgtN -- NO --> Loop
      IsCountgtN -- YES --> OutputProd[/OUTPUT PROD/]
      OutputProd --> Stop([STOP])
  
```

CS101 PPS @Sumit

65

Flowchart Example

66

□ Draw a flowchart for Computing e^x series up to N terms

```

    graph TD
      Start([START]) --> Read[/READ X, N/]
      Read --> Init[TERM = 1  
SUM = 0  
COUNT = 1]
      Init --> Loop[SUM = SUM + TERM  
TERM = TERM * X / COUNT  
COUNT = COUNT + 1]
      Loop --> IsCountgtN{IS COUNT > N?}
      IsCountgtN -- NO --> Loop
      IsCountgtN -- YES --> OutputSum[/OUTPUT SUM/]
      OutputSum --> Stop([STOP])
  
```

CS101 PPS @Sumit

66

Flowchart Example

67

□ Draw a flowchart for Computing e^x series up to 4 decimal places

```

graph TD
    Start([START]) --> ReadX[/READ X/]
    ReadX --> Init[TERM = 1  
SUM = 0  
COUNT = 1]
    Init --> Loop[SUM = SUM + TERM  
TERM = TERM * X / COUNT  
COUNT = COUNT + 1]
    Loop --> Decision{IS  
TERM < 0.0001?}
    Decision -- NO --> Loop
    Decision -- YES --> Output[/OUTPUT SUM/]
    Output --> Stop([STOP])
    
```

CS101 PPS @Sumit

67

Flowchart Example

68

□ Draw a flowchart for Grade computation

MARKS \geq 90 → Ex
 89 \geq MARKS \geq 80 → A
 79 \geq MARKS \geq 70 → B
 69 \geq MARKS \geq 60 → C
 59 \geq MARKS \geq 50 → D
 49 \geq MARKS \geq 35 → P
 34 \geq MARKS → F

CS101 PPS @Sumit

68

Flowchart Example

69

□ Draw a flowchart for Grade computation

```

graph TD
    Start([START]) --> ReadMarks[/READ MARKS/]
    ReadMarks --> D1{MARKS >= 90?}
    D1 -- YES --> O1[/OUTPUT "Ex"/]
    O1 --> S1([STOP])
    D1 -- NO --> D2{MARKS >= 80?}
    D2 -- YES --> O2[/OUTPUT "A"/]
    O2 --> S2([STOP])
    D2 -- NO --> D3{MARKS >= 70?}
    D3 -- YES --> O3[/OUTPUT "B"/]
    O3 --> S3([STOP])
    D3 -- NO --> O4[/OUTPUT "A"/]
    O4 --> S4([STOP])
    
```

CS101 PPS @Sumit

69

Flowchart Example

70

□ Draw a flowchart for Grade computation

```

graph TD
    A((A)) --> D1{MARKS >= 60?}
    D1 -- YES --> O1[/OUTPUT "C"/]
    O1 --> S1([STOP])
    D1 -- NO --> D2{MARKS >= 50?}
    D2 -- YES --> O2[/OUTPUT "D"/]
    O2 --> S2([STOP])
    D2 -- NO --> D3{MARKS >= 35?}
    D3 -- YES --> O3[/OUTPUT "P"/]
    O3 --> S3([STOP])
    D3 -- NO --> O4[/OUTPUT "F"/]
    O4 --> S4([STOP])
    
```

CS101 PPS @Sumit

70

End of Today's Lecture

71

Doubts && Queries?

CS101 PPS @Sumit

71

72

THANK YOU

CS101 PPS @Sumit

72