

CS101: Problem Solving through C Programming

Sachchida Nand Chaurasia
Assistant Professor

Department of Computer Science
Banaras Hindu University
Varanasi

Email id: snachaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com



December 30, 2020

Topics

- Introduction to Computer Science in the context of Programming Languages
- Program development phases
- Algorithms
- Flowcharts
- Types of Programming Languages
- Compiler and Interpreter
- Linker

Introduction to Computer Science in the context of Programming Languages I

Computer System Contains 2-Basic Parts

- i. Hardware (H/w)
- ii. Software (S/w)

With S/w computer can do:

1. Store Data.
2. Retrieve Data.
3. Solve Different type of problem.
4. Create friendly environment for S/w development.

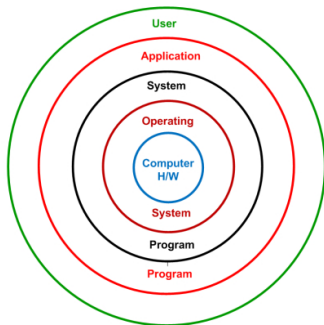


Figure 1: Overview of Computer System

Introduction to Computer Science in the context of Programming Languages II

- Computer science is the study of algorithmic processes and computational machines.
- As a discipline, computer science spans a range of topics from theoretical studies of algorithms, computation and information to the practical issues of implementing computing systems in hardware and software.
- Computer science addresses any computational problems, especially information processes, such as control, communication, perception, learning, and intelligence.

Program development phases I

- Programming language theory is a branch of computer science that deals with the design, implementation, analysis, characterization, and classification of programming languages and their individual features.
- It falls within the discipline of computer science, both depending on and affecting mathematics, software engineering, and linguistics.

Program development phases II

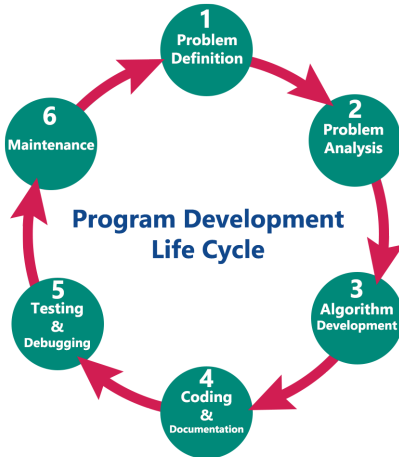


Figure 2: Basic program development phases

Program development phases III

► The following are six steps in the Program Development Life Cycle (PDLC):

- ❶ Analyze the problem: The computer user must figure out the problem, then decide how to resolve the problem - choose a program.
- ❷ Design the program: First design an algorithm for your problem. A flow chart is important to use during this step of the PDLC. This is a visual diagram of the flow containing the program. This step will help you break down the problem.

Program development phases IV

- ③ Code the program: This is using the language of programming to write the lines of code. The code is called the listing or the source code. The computer user will run an object code for this step.
- ④ Debug the program: The computer user must debug. This is the process of finding the “**bugs**” on the computer. The bugs are important to find because this is known as errors in a program.

Program development phases V

- ⑤ Formalize the solution: One must run the program to make sure there are no syntax and logic errors. Syntax are grammatical errors and logic errors are incorrect results.
- ⑥ Document and maintain the program: This step is the final step of gathering everything together. Internal documentation is involved in this step because it explains the reason one might have made a change in the program or how to write a program.

Algorithms I

Definition: An algorithm is a set of well-defined instructions in sequence to solve a problem.

OR

A finite set of steps that must be followed to solve any problem is called an algorithm.

► Algorithm is generally developed before the actual coding is done. It is written using English like language so that it is easily understandable even by non-programmers.

► **Some good qualities of an algorithm are:**

- Has finite number of steps.

Algorithms II

- Input and output should be defined precisely.
- Each step in the algorithm should be clear, unique and unambiguous.
- An algorithm should not include computer code.
Instead, the algorithm should be written in such a way that it can be used in different programming languages.
- Produces desired output.

► **Writing algorithm for solving a problem offers these advantages -**

Algorithms III

- Promotes effective communication between team members
- Enables analysis of problem at hand
- Acts as blueprint for coding
- Assists in debugging
- Becomes part of software documentation for future reference during maintenance phase

Algorithms IV

► General flow of an algorithm is

- Start
- Declare variables
- Initialize variables
- Perform operation
- Display output
- Stop

Algorithms V

Example: algorithm for sum of two numbers is as follows:

Step 1: Start

Step 2: Declare variables **Num1**, **Num2** and **Sum**.

Step 3: Read values for **Num1** and **Num2**.

Step 4: **Sum = Num1 + Num2** \rightarrow Add **Num1** and **Num2** and assign the result to **Sum**.

Step 5: Display **Sum**

Step 6: Stop

Flowcharts I

- A flowchart is a graphical representations of steps.
- It was originated from computer science as a tool for representing algorithms and programming logic but had extended to use in all other kinds of processes.
- Nowadays, flowcharts play an extremely important role in displaying information and assisting reasoning.
- They help us visualize complex processes, or make explicit the structure of problems and tasks.
- A flowchart can also be used to define a process or project to be implemented.








Symbol	Symbol Name	Purpose
	Start/Stop	Used at the beginning and end of the algorithm to show start and end of the program.
	Process	Indicates processes like mathematical operations.
	Input/ Output	Used for denoting program inputs and outputs.
	Decision	Stands for decision statements in a program, where answer is usually Yes or No.
	Arrow	Shows relationships between different shapes.
	On-page Connector	Connects two or more parts of a flowchart, which are on the same page.
	Off-page Connector	Connects two parts of a flowchart which are spread over different pages.

Figure 3: Basic flowchart symbols

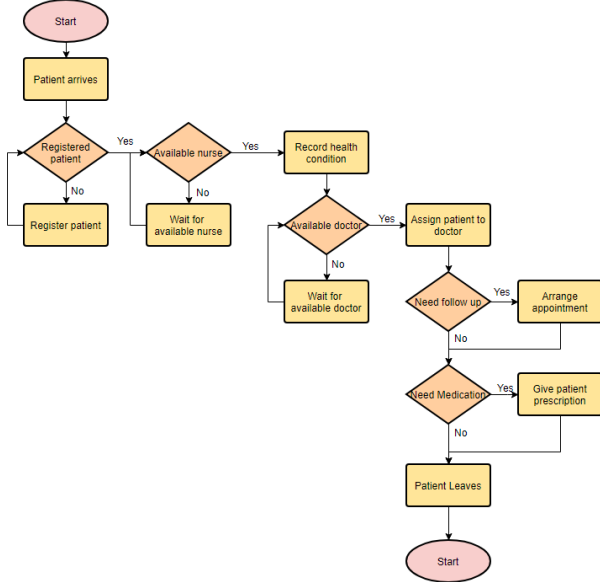


Figure 4: Medical service

C : temperature in Celsius
F : temperature Fahrenheit

Algorithm

Step-1 Start

Step-2 Input temperature in Celsius say C

Step-3 $F = (9.0/5.0 \times C) + 32$

Step-4 Display Temperature in Fahrenheit F

Step-5 Stop

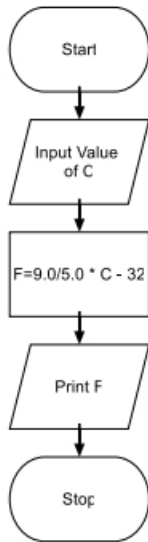


Figure 5: Algorithm and flowchart to convert temperature from Celsius to Fahrenheit

L : Side Length of Square
AREA : Area of Square
PERIMETER : Perimeter of Square

Algorithm

Step-1 Start

Step-2 Input Side Length of Square say L

Step-3 Area = $L \times L$

Step-4 PERIMETER = $4 \times L$

Step-5 Display AREA, PERIMETER

Step-6 Stop

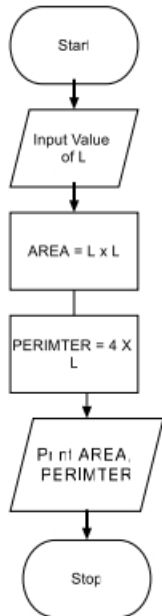


Figure 6: Algorithm and flowchart to find Area and Perimeter of Square

R : Radius of Circle
AREA : Area of Circle
PERIMETER : Perimeter of Circle

Algorithm

Step-1 Start

Step-2 Input Radius of Circle say R

Step-3 Area = $22.0/7.0 \times R \times R$

Step-4 PERIMETER = $2 \times 22.0/7.0 \times R$

Step-5 Display AREA, PERIMETER

Step-6 Stop

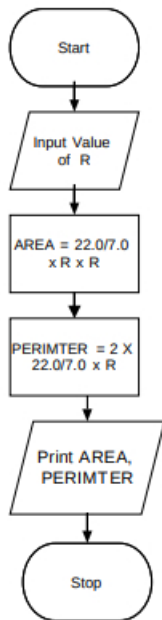


Figure 7: Algorithm and flowchart to find Area and Perimeter of Circle

Algorithm

Step-1 Start

Step-2 Input Two Numbers Say NUM1, NUM2

Step-3 Display Before Swap Values NUM1, NUM2

Step-4 TEMP = NUM1

Step-5 NUM1 = NUM2

Step-6 NUM2 = NUM1

Step-7 Display After Swap Values NUM1, NUM2

Step-8 Stop

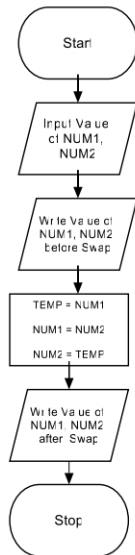


Figure 8: Algorithm and flowchart to Swap Two Numbers using Temporary Variable

Algorithm

Step-1 Start

Step-2 Input Two Numbers Say A,B

Step-3 Display Before Swap Values A, B

Step-4 $A = A + B$

Step-5 $B = A - B$

Step-6 $A = A - B$

Step-7 Display After Swap Values A, B

Step-8 Stop

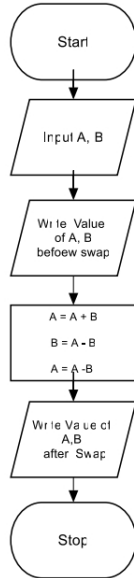


Figure 9: Algorithm and flowchart to Swap Two Numbers without using temporary variable

Algorithm

Step-1 Start

Step-2 Input two numbers say
NUM1, NUM2

Step-3 IF NUM1 < NUM2 THEN
 print smallest is NUM1
ELSE
 print smallest is NUM2
ENDIF

Step-4 Stop

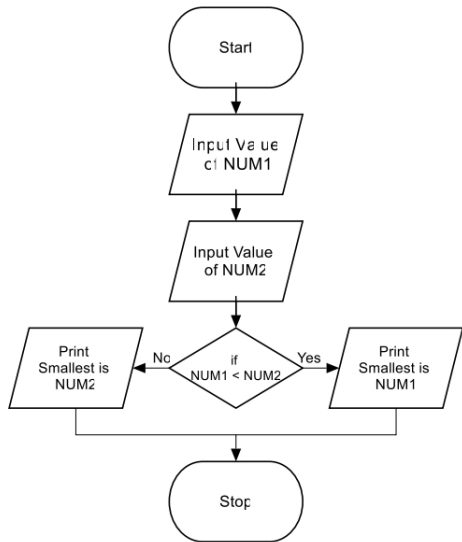


Figure 10: Algorithm and flowchart to find the smallest of two numbers

Algorithms and Flowcharts I

Problem1: write algorithm to find the greater number between two numbers.

Step1: Start

Step2: Read/input A and B

Step3: **if** A greater than B then C=A

Step4: **if** B greater than A then C=B

Step5: Print C

Step6: End

Algorithms and Flowcharts II

Problem2: write algorithm to find the result of equation:

$$f(x) = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Step1: Start

Step2: Read/input x

Step3: **if** X Less than zero then F=-x

Step4: **if** X greater than or equal zero then F=x

Step5: Print F

Step6: End

Algorithms and Flowcharts III

Problem3: A algorithm to find the largest value of any three numbers.

Step1: Start

Step2: Read/input A,B and C

Step3: **if** $(A \geq B)$ and $(A \geq C)$ then $Max=A$

Step4: **if** $(B \geq A)$ and $(B \geq C)$ then $Max=B$

Step5: **if** $(C \geq A)$ and $(C \geq B)$ then $Max=C$

Step6: Print Max

Step7: End

Algorithms and Flowcharts IV

Problem4: An algorithm to calculate even numbers between 0 and 99

Step1: Start

Step2: $I \leftarrow 0$

Step3: Write I in standard output

Step4: $I \leftarrow I+2$

Step5: If ($I \leq 98$) then go to line 3

Step6: End

Algorithms and Flowcharts V

Problem5: Design an algorithm which gets a natural value, n , as its input and calculates odd numbers equal or less than n . Then write them in the standard output:

Step1: Start

Step2: Read n

Step3: $I \leftarrow 1$

Step4: Write I

Step5: $I \leftarrow I + 2$

Step6: **if** ($I \leq n$) then go to line 4

Step7: End

Algorithms and Flowcharts VI

Problem6: Design an algorithm which generates even numbers between 1000 and 2000 and then prints them in the standard output. It should also print total sum:

```
Step1. Start
Step2. I ← 1000 and S ← 0
Step3. Write I
Step4. S ← S + I
Step5. I ← I + 2
Step6. if (I ≤ 2000) then go to line 3
Step7 else go to line 7
Step8. Write S
Step9: End
```

Algorithms and Flowcharts VII

Problem7: Design an algorithm with a natural number, n , as its input which calculates the following formula and writes the result in the standard output: $S = \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{n}$

Step1. Start

Step2. Read n

Step3. $I \leftarrow 2$ and $S \leftarrow 0$

Step4. $S = S + 1/I$

Step5. $I \leftarrow I + 2$

Step6. **if** ($I \leq n$) then go to line 4

Step **else** write S in standard output

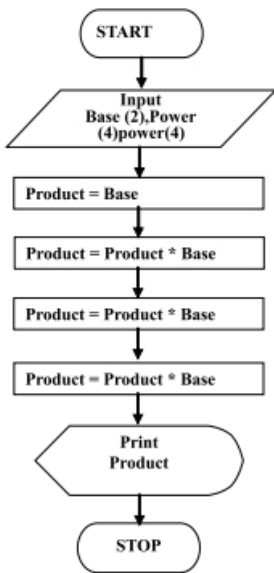
Step7. End

Algorithms and Flowcharts VIII

Problem8: Write an algorithm and draw a flow chart to calculate 2^4 :

```
Step1 . Start
Step2 . Input Base(2) , Power(4)
Step3 . Product= Base
Step4 . Product = Product * Base
Step5 . Product = Product * Base
Step6 . Product = Product * Base
Step7 . Print Product
Step8 . End
```

Algorithms and Flowcharts IX



Question: What happens if you want to calculate 2 to the power of 1000 (2^{1000})

Algorithms and Flowcharts XI

Step1. Start

Step2. Input Base (2) , Power (1000)

Step3. Product = Base

Step4. Counter = 1

Step5. **while** (Counter < Power)

Step6. Repeat steps 5 through 8

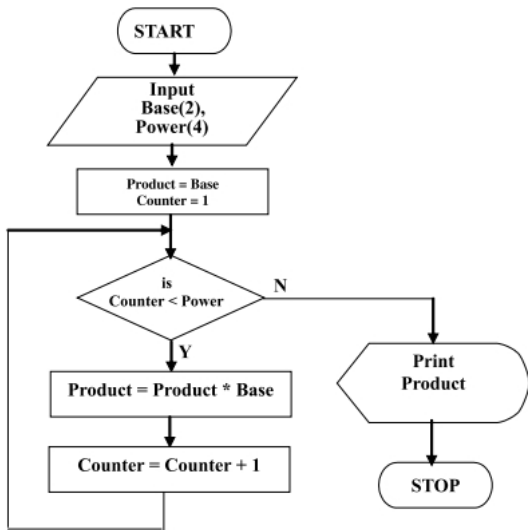
Step7: Product = Product * Base

Step8: Counter = Counter + 1

Step9: Print Product

Step10. End

Algorithms and Flowcharts XII



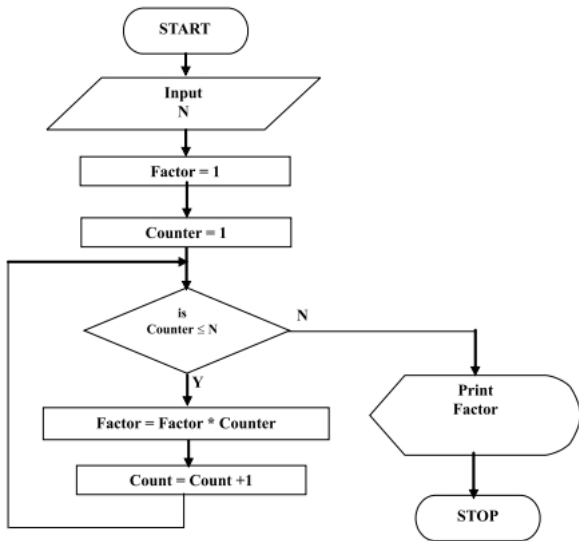
Algorithms and Flowcharts XIII

Problem9: Write an algorithm and draw a flow chart to calculate the factorial of a number (N). Verify your result by a trace table by assuming $N = 5$.

Algorithms and Flowcharts XIV

```
Step 1. Start
Step 2: Input N
Step 3: Factor = 1
Step 4: Counter = 1
Step 5: while (Counter <= N)
Step 6:     Repeat steps 4 through 6 \\Error in this line
Step 7:     Factor = Factor * Counter
Step 8:     Counter = Counter + 1
Step 9: Print (N, Factor)
Step10. End
```

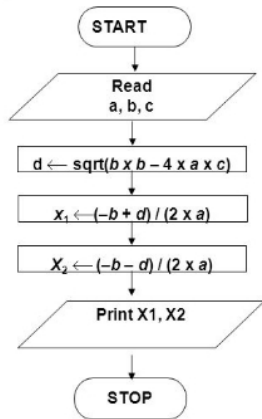
Algorithms and Flowcharts XV



Algorithms and Flowcharts XVI

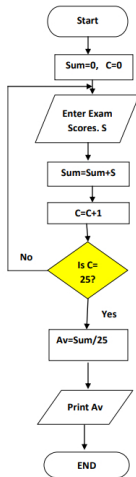
■ Algorithm:

- Step 1: Start
- Step 2: Read a, b, c
- Step 3: $d \leftarrow \text{sqrt}(b \times b - 4 \times a \times c)$
- Step 4: $x_1 \leftarrow (-b + d) / (2 \times a)$
- Step 5: $x_2 \leftarrow (-b - d) / (2 \times a)$
- Step 6: Print x1, x2
- Step 7: Stop



Algorithms and Flowcharts XVII

What will be the output of the flowchart



Summary I

- ✓ Algorithm is the sequence of steps to be performed in order to solve a problem by the computer.
- ✓ Three reasons for using algorithms are efficiency, abstraction and reusability.
- ✓ Algorithms can be expressed in many different notations, including natural languages, pseudocode, flowcharts and programming languages.

Summary II

- ✓ Analysis of algorithms is the theoretical study of computer program performance and resource usage, and is often practised abstractly without the use of specific programming language or implementation.
- ✓ The practical goal of algorithm analysis is to predict the performance of different algorithms in order to guide program design decisions.
- ✓ Most algorithms do not perform the same in all cases; normally an algorithm's performance varies with the data passed to it.

Summary III

- ✓ Typically, three cases are recognized: the best case, average case and worst case.
- ✓ Worst case analysis of algorithms is considered to be crucial to applications such as games, finance and robotics.
- ✓ O-notation, also known as Big O-notation, is the most common notation used to express an algorithm's performance in a formal manner.

Summary IV

- ✓ Flowchart is a graphical or symbolic representation of an algorithm. It is the diagrammatic representation of the step-bystep solution to a given problem.
- ✓ Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.
- ✓ Benefits of using flowcharts include ease of communication, effective and efficient analysis and coding, proper documentation and maintenance.

Summary V

- ✓ Limitations of using flowcharts include complex logic and multiple modifications.