

Contents

| Unit 2 | Introduction | Page No. |
|---------------|---|-----------------|
| 2.0 | Structure of the Unit | 1 |
| 2.1 | Unit Outcomes | 1 |
| 2.2 | Problem-Solving techniques | 1 |
| 2.3 | Algorithms | 4 |
| 2.4 | Flowcharts | 8 |
| 2.5 | Pseudocode | 12 |
| 2.6 | Programming Languages | 15 |
| 2.7 | Programming paradigms: Object-oriented and Procedure-oriented Programming | 16 |
| 2.8 | Self-Assessment Questions | 18 |
| 2.9 | Multiple Choice Questions (MCQs) | 19 |
| 2.10 | Keys to MCQs | 19 |
| 2.11 | Summary of the Unit | 22 |
| 2.12 | Keywords | 22 |
| 2.13 | Recommended Learning Resources | 22 |

Unit 2

Problem-Solving and Programming Concepts

Structure of the Unit

- 2.1. Unit Outcomes
- 2.2. Problem-Solving Techniques
- 2.3. Algorithms
- 2.4. Flowcharts
- 2.5. Pseudocode
- 2.6. Classification and Characteristics of Programming Language
- 2.7. Programming paradigms: Procedure-oriented programming and OOP
- 2.8. Self-Assessment Questions and Activities
- 2.9. Multiple-Choice Questions
- 2.10. Keys to Multiple Choice Questions
- 2.11. Summary of the Unit
- 2.12. Recommended Resources for Further Reading

2.1. Unit Outcomes

After the successful completion of this unit, the student will be able to:

- 1. Examine the problem and discuss problem-solving techniques.
- 2. Develop algorithms for solving a problem.
- 3. Design flowcharts to express the logic.
- 4. Discuss procedure-oriented and object-oriented programming (OOP) approaches.

2.2. Problem Solving Techniques

Computational thinking needs skills for problem-solving. Problem solving can be done in numerous ways which include methods inside as well as outside the world of computers. The most common strategies followed by computer scientists are given below.

- 1. Problem Solving
- 2. Logical Reasoning
- 3. Decomposition
- 4. Abstraction

2.2.1 Problem-Solving

At the beginning of the problem-solving stage, a problem definition is fixed. The definition serves as a goal of the software developer. A systematic approach to software engineering (SE) is followed. The major steps in SE are:

- Requirement Analysis
- Design
- Implementation
- Testing and Maintenance

In requirement analysis, the customer's needs are listed as functional and non-functional requirements. The functional requirements capture all the details of the tasks that are performed at the client's place. E.g., The user and system requirements for a pharmacy department in a Hospital Management System are outlined in Figure 2.1.

Generate the reports for the medicines such as tablets, syrups, ointments, etc. every month. List the details including its name, cost, company, quantity, etc.

Figure 2.1 User Requirements in an Abstract Form

The above-mentioned requirements can be explored in detail as given in Figure 2.2.

1. Generate a summary of the medicines prescribed by all the doctors in the hospital.
2. Generate a report on every working day of the month before 7 p.m.
3. Every medicine must be with its name, dosage, cost, name, details of the patient's purchase, doctor's name, etc.
4. If there are medicines available in different dosages such as 10mg, 20 mg, or 30 mg, then the relevant quantity and names should be listed.
5. Access to the software must be secure. It should be given to authorized persons with suitable login and password.
6. He/she should be on the access list.
7. Anonymous users should not be allowed to access the list.
8. The report should be printable, and it should be made available on the Internet.

Figure 2.2 System Requirements for the Hospital Management System

Functional Requirements: Functional requirements describe what are the functions of the software system. The functional requirements represent how the software system should react to a set of inputs and generate output. Also, what functions are not supported is expressed using functional requirements. A set of functions is used in the functional requirements as shown in Figure 2.3. The terms i1, i2, and i3 indicate the input, and o1, o2, and o3 represent the output of the system.

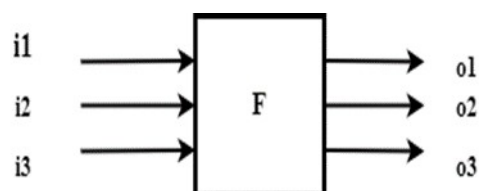


Figure 2.3 Functional Requirements View

The functional requirements represent user requirements. Functional requirements are expressed using an abstract way. The different functions of the system, error conditions, and handling problematic situations are expressed in the functional requirements.

Considering the software of the hospital management System, the different functional requirements are expressed as follows:

- List all the clinics in the city.
- Every clinic must prepare a list of appointments for patients.
- Every clinic must maintain a list of employees with a unique employee code number.
- If a user wants to search for a good clinic, he/she can enter the area, city, and specialty of the doctor such as dental, ortho, cardiac experts, etc. The output for this query will be the details of the clinic with the necessary information related to the user's question.

Non-functional Requirement: Non-functional requirements are not directly connected to the functionality of the system, but they are related to other parameters such as reliability, efficiency, delay, performance, security, or availability. The non-functional requirements commonly refer to features of the system. These requirements are critical. Non-functional requirements are efficiency, security, safety, availability, portability, etc.

Examples: In a chemical plant or aircraft system design, reliability is very important. One non-functional requirement can lead to many functional requirements and can affect the entire architecture of the system. In an ATM, cash withdrawal must be extremely secure and there should not be any unauthorized access to the bank accounts. So, if the security non-functional requirement is not achieved, then the entire ATM software is not valid and useless.

2.2.2 Logical Reasoning

Logic is a critical factor in problem-solving. The functional requirements can be solved using logic at the start. Computer software involves a set of instructions based on some logic so that the program makes correct choices. As in online purchases, the flow of logic used in a program can be summarized as shown in Figure 2.4.

| |
|---|
| Allow the user to explore the products in different categories designed for men and women. Develop logic for the selection of the product. Ensure you mention the return policy. After selection, prepare an order summary. Do the billing with online transactions including bank and all the security features. Generate the acknowledgement including address, tracking number, etc. |
|---|

Figure 2.4 Logical Reasoning Approach for Online Purchase

To make different choices/options, most programming languages make use of IF instructions. If the user has selected Net banking, THEN list the banks and initiate transactions by making the bank login page available, manage One Time Password (OTP), and so on. Some more examples of IF-THEN are as follows:

- IF one song is finished on an audio player, THEN begin playing the next.
- IF the user clicks on confirm order, THEN proceed to purchase.

- IF the user has entered the login name and password, THEN verify the details and proceed to the email inbox.

A computer program includes such IF-THEN statements, it is saved in a set of cause-effect relationships.

A cause-effect relationship is translated into instructions as given in Figure 2.5.

| Cause | Effect |
|---|---|
| Username and Password are entered | Verify the username and password |
| The user clicks on the close button on a window | Remove the window from the display |
| Battery charge in a laptop is below 5% | Pop the window for charging the battery |
| A new email is received | Pop the message in bold letters or an alarming tone |

Figure 2.5 Cause and Effect in Logical Reasoning

The reasoning is classified as follows:

Deductive: This method uses mathematical proofs to arrive at a conclusion using the available proofs.

Example: A Pythagorean Theorem states that $a^2 + b^2 = c^2$. Here a and b are measurements of sides of a triangle. The term c represents the length of the hypotenuse. The deductive reasoning from this equation is the distance from opposing corners of a rectangular room that is 12 feet wide, and 16 feet long is 20 feet. This is given as $12^2 + 16^2 = 400$, which means the square root of 400 is 20.

Inductive: Here the start is with a set of data to infer a rule or principle. Inductive reasoning helps recognize the general rules. occurs during the process of recognizing the generalities. Algorithms are used to describe the logic that includes various steps to solve a problem. The algorithms include three components as given below.

1. Software for communicating with the user
2. Software for retrieving and storing data
3. Software for calculating results based on user input and/or retrieved data

This pattern is called 3-tier software architecture. User communication with his/her personal computer is tier 1. Computers using storage or databases are tier 2. The computers owned by social network companies are known as tier 3. The performance of computer applications, confidentiality uses, and working can be understood using 3 tier software architecture.

Abductive: This method differs from inductive reasoning. This method uses observations and facts to conclude.

2.3. Algorithm

An algorithm is a sequence of simple steps or discrete actions to solve a problem. An algorithm is a stepwise procedure used for computations that are performed on a computer. Algorithms are written in natural language with an initial set of inputs with a desired output. Algorithms are not executed on a computer. Algorithms result in achieving some goal. For example, a cooking recipe is an algorithm to prepare a dish. The recipes include ingredients and directions. Algorithms play an important role in the IT industry, solving problems in mathematical and science problems, data science, AI, several engineering applications, etc. The features of the algorithm are:

- A set of steps with start and stop as delimiters.
- Names are used in the algorithm relevant to the problem. For example, RollNo, Name, Salary, a, b, c, num1, num2, etc. The names are known as identifiers. The rule of declaring identifiers is to begin with an alphabet and it can be a combination of alphabets, digits, and underscore.
- Binding: The value on the right side of the arrow is bound to the identifier on the left of the arrow.

This format is known as assigning expressions to the variable. The general syntax for binding is given as follows.

- identifier \leftarrow expression
- fact \leftarrow 1
- x \leftarrow y
- Algorithms include if, if-then-else, repeat until, read, print, input, output, for, while, do-while, and display verbs.
- The three patterns of control in algorithms are sequential execution, selection, and repetition.

```

Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
        Sum  $\leftarrow$  num1 - num2
Step 5: Display sum
Step 6: Stop

```

Algorithm to add two numbers.

A good algorithm is expressed by the following characteristics.

- Precise input and output.
- Each step in the algorithm must be precise and unambiguous.
- Algorithms must include effective logic out of different methods used to solve a problem.
- There should not be programming jargon or code in algorithms. Algorithms should help develop a software program.

Another example of an algorithm to find the largest of three numbers is shown below.

```

Step 1: Start
Step 2: Declare variables x,y, and z.
Step 3: Read variables x,y, and z.
Step 4: If x > y
        If x > z
            Display x is the largest number.
        Else
            Display z is the largest number.

```

```

Else
    If y > z
        Display y is the largest number.
    Else
        Display z is the greatest number.

```

Step 5: Stop

Algorithm to find the largest of 3 numbers.

In the above algorithm, there are statements for initializing values, conditions, and progress. Progress includes different sets of actions. Algorithms express the logic to be written in the program. An algorithm to find the factorial of a number is based on the formulas given below.

$$4! = 4 \times 3 \times 2 \times 1$$

24

The algorithm is given below.

```

Step 1: Start
Step 2: Declare variables n, fact, and i.
Step 3: Initialize variables
    fact ← 1
    i ← 1
Step 4: Read the value of n
Step 5: Repeat the steps until i = n
    5.1: fact ← fact × i
    5.2: i ← i + 1
Step 6: Display fact
Step 7: Stop

```

Algorithm to find the factorial of a number

A prime number is a number divisible by itself and it is not divisible by any other number.

Examples of prime numbers are 3, 7, 11, 13, 17, 23, 31, ...

Examples of non-prime numbers are 2, 4, 6, 12, 14, 25, 34, ...

The most interesting part of prime numbers is only 2 is the smallest even prime number. For any number to check whether it is prime, it is divided by all the numbers from 2 to the number itself. If the number is divisible that means its remainder is zero, then the loop in which it is checked is stopped. The modulus (%) operator returns the remainder. Before exiting a separate indicating variable flag is set to 1. Outside the loop, if the flag is 0 which was set at the beginning, then the number is not prime else it is a prime number. A stepwise description to solve the prime number is depicted below.

```

Step 1: Start
Step 2: Declare variables n, i, flag.
Step 3: Initialize variables.
    flag ← 1
    i ← 2

```

```

Step 4: Read n from the user.
Step 5: Repeat the steps until i=(n/2)
    5.1 If the remainder of n÷i equals 0
        flag ← 0
        Go to step 6
    5.2 i ← i+1
Step 6: If flag = 0
    Display n is not prime.
else
    Display n is prime.
Step 7: Stop

```

Algorithm to check whether the number is prime or not.

The next example is about finding the roots of quadratic equations. For an equation $ax^2+bx+c=0$, the roots are the values that satisfy this equation. The formula for finding roots of quadratic equations is as follows.

$$r_1 = -b + \sqrt{\frac{b^2 - 4ac}{2a}}$$

$$r_2 = -b - \sqrt{\frac{b^2 - 4ac}{2a}}$$

An algorithm for calculating roots of quadratic equations includes reading the variables a, b, and c. The description of the steps is given below.

```

Step 1: Start
Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;
Step 3: Calculate D
    D ← b2-4ac
Step 4: If D ≥ 0
    r1 ← (-b+√D)/2a
    r2 ← (-b-√D)/2a
    Display r1 and r2 as roots.
Else
    Calculate the real part and imaginary part.
    r_p ← -b/2a
    i_p ← √(-D)/2a
    Display r_p+j(i_p) and r_p-j(i_p) as roots
Step 5: Stop

```

Algorithm for calculating roots of quadratic equations.

Fibonacci Series: The Fibonacci series is the sequence of numbers (also called Fibonacci numbers), where every number is the sum of the preceding two numbers. The first two numbers in the series are 0 and 1. In some versions, the first term '0' is omitted. In the Fibonacci series, the steps in generating

numbers are as follows. These steps are repeated for n number of times or a target number n is reached. The main steps in the Fibonacci series are given as follows.

$$\begin{aligned}f_1 &= 0 \\f_2 &= 1 \\f_3 &= f_1 + f_2 \\f_1 &= f_2 \\f_2 &= f_3\end{aligned}$$

Step 1: Start
Step 2: Declare variables first, second, third, max, count
Step 3: Read a maximum number of variables as max
Step 3: Initialize variables first \leftarrow 0 second \leftarrow 1 and count \leftarrow 1
Step 4: Display first and second
Step 5: Repeat the steps until count \leq max
 5.1: third \leftarrow first + second
 5.3: first \leftarrow second
 5.3: second \leftarrow third
 5.4: Display third
 5.3: count \leftarrow count + 1
Step 6: Stop

Algorithm for generating 'n' number of Fibonacci numbers

A Fibonacci series can thus be given as, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, . . . The steps are given in Figure 2.10. The first two numbers are 0 and 1.

The Fibonacci number series is applied in a grouping of numbers used in mathematical computing, coding, and cryptography for giving information security, finance market trading, etc.

Algorithms are useful for consistent logic and a simple way of understanding the programming steps. Algorithms include reusability. But the drawbacks of algorithms are lack of creativity, consume more time to create, and may not provide a good solution every time. Another way of giving logical reasoning is given using a flowchart.

2.4. Flowchart

A flowchart is defined as a graphical representation of the operations involved in a data processing system. A complex algorithm can be expressed using a flowchart. A flowchart is also known as an activity diagram. The flowchart contains pictures along with the text. Flowcharts can be used as a model to express the steps to build logic. Flowcharts include symbols and the lines connecting symbols. The symbols used in the flowchart are given in Figure 2.6.






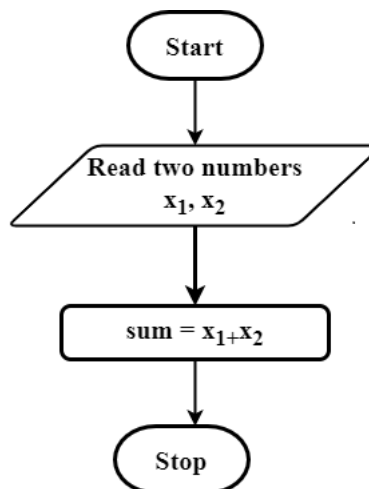
| Symbol | Name | Function |
|---|--------------|---|
|  | Start/end | An oval represents a start or end point. |
|  | Arrows | A line is a connector that shows relationships between the representative shapes. |
|  | Input/Output | A parallelogram represents input or output. |
|  | Process | A rectangle represents a process. |
|  | Decision | A diamond indicates a decision. |

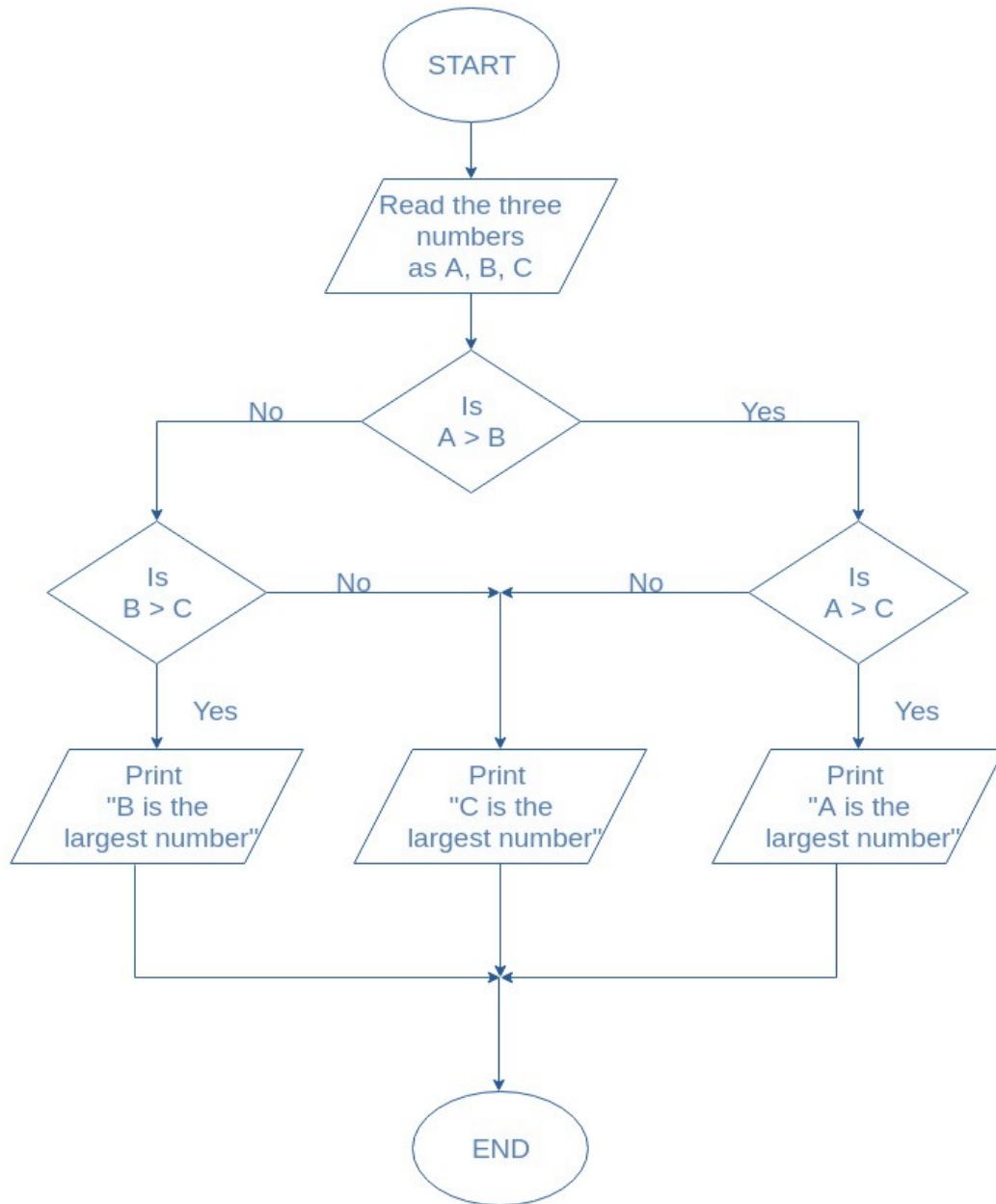
Figure 2.6 Symbols in Flowcharts

Advantages and drawbacks of flowcharts: Flowcharts are useful for making clear logic. The logic is simpler with pictorial representation. The analysis of the problem is more effective by representing it in the form of a flowchart. Flowcharts help in detecting errors. The programs can be analyzed or developed using a flowchart as a blueprint. However, there are some limitations to the flowchart: if the problem is bigger, then the flowchart can be complex and will be many pages. Such a flowchart is difficult to modify as the program is updated regularly. Some of the examples of the flowchart are given as follows.

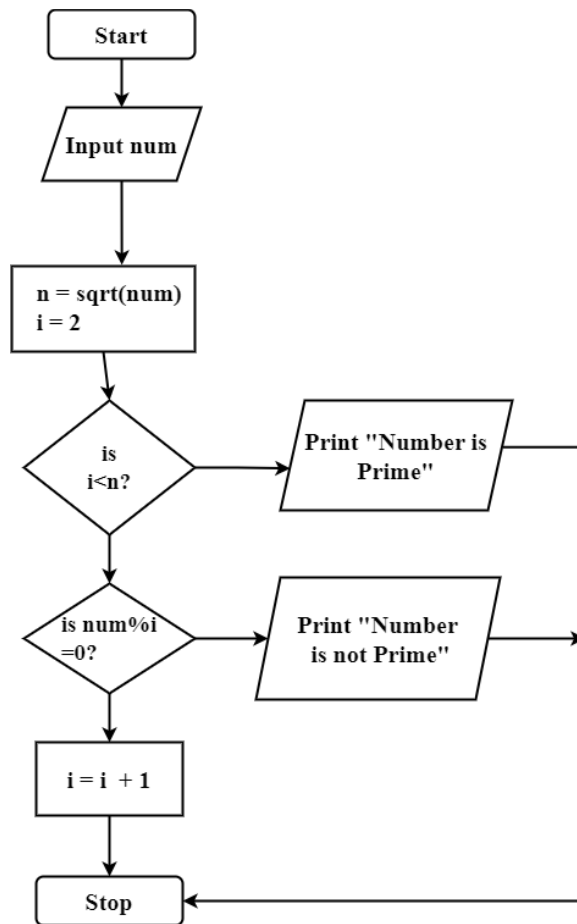
- 1) Flowchart to add two numbers.
- 2) Flowchart to find the largest of three numbers.
- 3) Flowchart to find whether the given number is prime or not.
- 4) Flowchart to check whether the number is even or odd number.
- 5) Flowchart to find the factorial of a number.



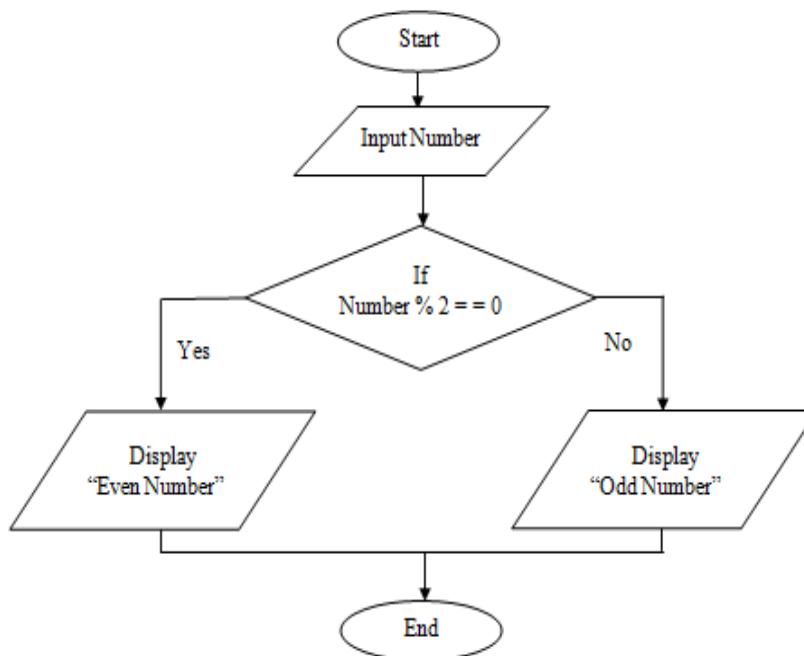
Flowchart to add two numbers.



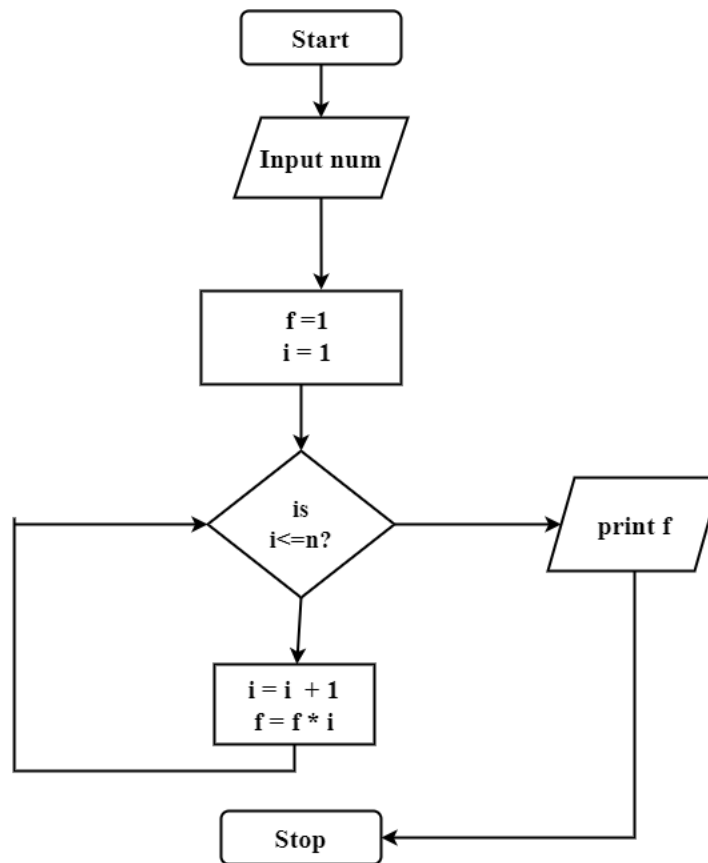
Flowchart to find the largest of 3 numbers.



Flowchart to find the prime number



Flowchart to check even/odd numbers.



Flowchart for finding the factorial of a number

2.5. Pseudocode

A computer program is informally expressed using pseudocode. A pseudocode gives a rough version of programming instructions and flow. It does not follow any strict programming guidelines or syntax. Pseudocode follows the different steps of the algorithm namely: Sequence, selection, iteration, case selection, etc. The common keywords used in the pseudocode are WHILE, DO-WHILE, FOR, IF, IF-ELSE, IF-THEN-ELSE, PRINT, READ, and REPEAT UNTIL. These statements show the dependency of the instructions. A pseudocode for checking whether a person can vote or not is given below.

1. If a person's age is greater than or equal to 18

 Print "Eligible for voting"

else

 Print "Not eligible for voting"

Pseudocode 1: Check the Eligibility of Voting

Pseudocodes help understand the logic to be used in the program without bothering about syntax.

Pseudocodes are written in multiple ways, in some methods BEGIN and END keywords are used to start and stop a pseudocode. Pseudocode helps test the approach used in a program and the translation of algorithms into code. Some more examples are shown below.

```
BEGIN
  NUMBER a1, a2, sum
  OUTPUT("Enter number1:")
  INPUT a1
  OUTPUT("Enter number2:")
  INPUT a2
  sum=a1+a2
  OUTPUT sum
END
```

Pseudocode 2: Addition of Two Numbers

```
BEGIN
  NUMBER l, b, area, perimeter
  INPUT l
  INPUT b
  area=l*b
  perimeter=2*(l + b)
  OUTPUT area
  OUTPUT perimeter
END
```

Pseudocode 3 : Area and Parameter of Rectangle

```
BEGIN
  NUMBER x
  OUTPUT "Enter a Number"
  INPUT num
  IF num>0 THEN
    OUTPUT "Number is positive"
  ELSE IF num <0 THEN
    OUTPUT "Number is negative"
  ELSE
    OUTPUT "Number is zero"
  ENDIF
END
```

Pseudocode 4: check whether the number +ve/-ve

```
BEGIN
  NUMBER count, sum=0
```

```

FOR counter=1 TO 100 STEP 1 DO
    sum=sum + count
ENDFOR
OUTPUT sum
END

```

Pseudocode 5: Addition of First 100 Natural Numbers

In programming array is a list or structure holding multiple values of similar data type. The data types are integer, real number, float (with decimal point), character (for alphabets), string (for lines of characters), etc. In example 6 given below, the + symbol in the OUTPUT statement is used to append the value to the message.

```

BEGIN
NUMBER i=0, n=5, add=0
ARRAY numbers={69,4,1,17,225}
FOR i=0 TO n-1 STEP 1 DO
    add= add + numbers[i]
ENDFOR
OUTPUT "Sum of numbers in the array"+ sum
END

```

Pseudocode 6 : Elements of an array Addition

A pseudocode to interchange/swap the values of two variables is given in Pseudocode 8

```

BEGIN
NUMBER x, y
x=10, y=20
OUTPUT "Value of Number x :"+x
OUTPUT "Value of Number y :"+y
temp=x
x=y
y=temp
OUTPUT "Value of Number x :"+a
OUTPUT "Value of Number y :"+b
END

```

Pseudocode 8: Swap Algorithm

A similar pseudocode for swapping two variables without taking the third variable is given below.

```

BEGIN
NUMBER x, y
x=10, y=20
OUTPUT "Value of Number x :"+x
OUTPUT "Value of Number y :"+y
x=x + y

```

```
y = x - y
x = x - y
OUTPUT "Value of x:" + x
OUTPUT "Value of y :" + y
END
```

Pseudocode 7: Swapping without third variable

A large program is divided into smaller parts and each part is implemented separately. In such cases, the different keywords to be used are PROCEDURE, RETURN, SET, etc. An example of a procedure for a small subprogram is given below. The two identifiers inside the round brackets represent the parameters to be passed.

```
PROCEDURE Addition(x, y)
    SET result = 0
    result = x + y
    RETURN result
```

Pseudocode 8: Procedure for Addition

2.6. Programming Language

Software technology is continuously changing with the introduction of new concepts and methods. In today's rapidly changing software industry, there are many new approaches in every software development life cycle phase. The software crisis is a result of such incremental complexity of the IT industry. The challenges in software include:

- The representation of real-life entities.
- Development of a good software product in the available time.
- Reusing the software components in the design and implementation of software.
- Minimizing the cost of software development while meeting all the customer's requirements.
- Developing adaptable and maintainable software products so new features can be added in the future.

The delivery of the software product, meeting several challenges, is critical. The history of the US software projects in the 1970s shows that many of the US Défense projects were incomplete and undelivered, and only a few were used without changing requirements. The software demands many changes in the presentation and data. In the real world, the requirements change frequently, and developing quality software that adapts to the changes in the environment is difficult. The software should meet the client's functional requirements and other non-functional requirements such as space and time efficiency, adaptive and corrective maintenance, reusing of the software components, etc. Approximately 42 percent of the maintenance cost of the software is due to changes in user requirements. Over the past several years, the software has evolved. Software evolution is considered a layer of growth where each layer contributes additional features over the previous one. Each layer expresses the functionality of the software as depicted in Figure 2.8.

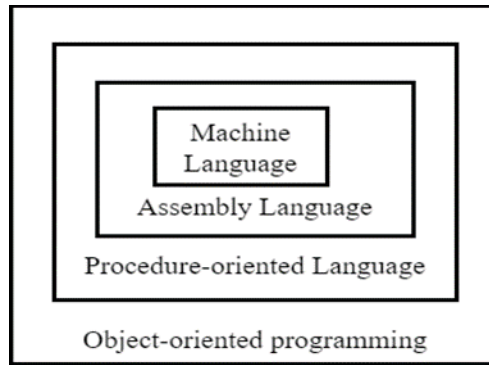


Figure 2.8. Layers Model of Evolution of Programming Languages

A computer program includes a set of instructions to do a well-defined task. Computer instructions are written using a programming language. Mainly A programming language specifies syntax, structure, and general format of the instructions and consists of a translator. The commonly used translators are interpreters and compilers. A ‘C’ programming language makes use of a compiler that converts programming language into binary form. Programming language is classified as:

- 1) **High-Level Language:** A high-level language is an English-like language that specifies the format and rules for writing computer programs. Software is developed using a high-level language and this language is easily understood by the persons, but it is difficult for a computer to understand. Popular high-level languages are Pascal, C, C++, Java, Python, SQL, etc.
- 2) **Low-Level Language:** The machine language or binary is a low-level language, and it consists of 0, and 1s. Another type of low-level language is assembly language. Low-level language is close to computer machine hardware (1 is considered as ON and 0 is considered as OFF signal). Assembly language is like machine language but uses symbols for machine language. The symbols are understandable and known as mnemonics. e.g., ADD, SUB, LOAD, etc. This is shown in Figure 2.8.

Assembly versus Machine Language

| Assembly Language | Machine Language |
|-------------------|------------------|
| ADD A, B | 11000111 |
| MOV C, B | 10000001 |
| MVI B, 0 | 10100011 |
| MOV AX, BX | 11111010 |

Figure 2.8 Unstructured or Low-level Language

2.7 Programming Paradigms

Programming paradigms are the approaches or styles for organizing the instructions in computer programs. There are many programming paradigms and not one is suitable for all types of problems to solve the problem. Programming paradigms are classified as follows.

2.7.1 Procedure-oriented Programming

Procedure-oriented programming (POP) is also known as the imperative programming approach that

depends on the procedures or modules that are designed using top-down design. In POP, the main task is divided into subtasks. Each subtask is achieved using a function. The main program with a set of functions is presented in Figure 2.9.

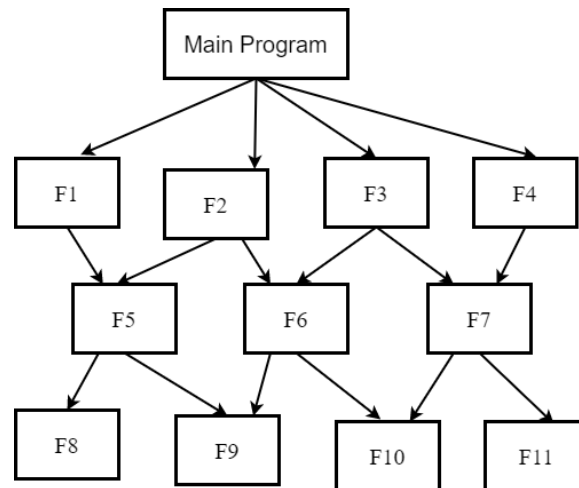


Figure 2.9 POP Model

The functions in the POP are represented as F1, F2, F3, The tasks in the POP are organized hierarchically. A top-down design such as a flowchart is used to organize the instructions for performing the tasks in POP. Here the attention is more on the actions but not on the data. Each function can use local data and the common important data are kept in a central repository and accessed by all the functions. This makes it challenging to identify which data is used by the function. If the data is to be shared amongst the functions, the data must be returned using suitable data structures. Each function may have its local data. Keeping consistency in data used in different functions is complex. Also, it may trigger some bugs if not handled properly. Large commonly shared data can be commonplace but may be vulnerable to security threats. The POP approach may not be suitable for some real-world problems because the emphasis is on functions and not on the data.

2.7.2 Object-oriented Programming (OOP)

The OOP approach overcomes the drawbacks of the POP. The free movement of the data is not permitted in OOP. The OOP treats the data with the highest importance and connects it with the functions. The data inside the functions are protected from any accidental modification. The given problem is divided into a set of entities known as objects. In OOP the problem is decomposed into several entities called objects. The object binds the data and functions together into a single unit. The functions of different objects are connected, and the data can be accessed only through the concerned functions. This is shown in Figure 2.10.

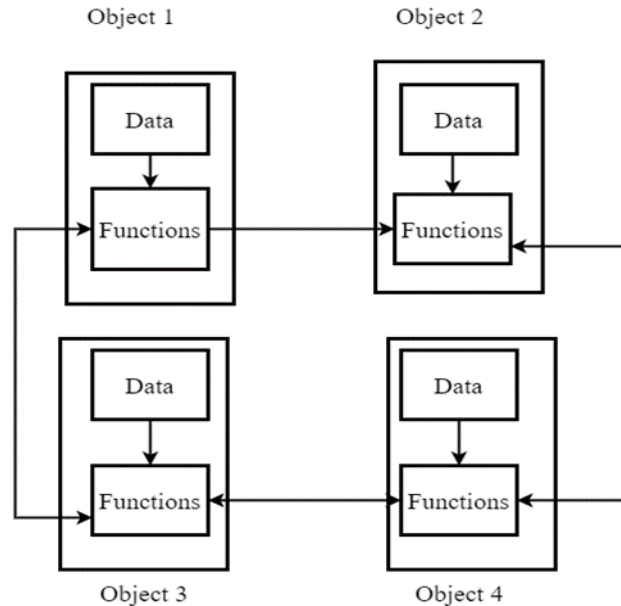


Figure 2.10 OOP Model

Examples of POP and OOP are shown in Figure 2.11.

Consider a scenario of designing an inventory for vehicles that includes all the details of the cars and trucks. For the cars, the necessary information is color, engine name, model, number of wheels, mileage, etc. For the trucks, all these details are the same with some additional information such as towing capacity, cab size, and transmission type. Similarly, for the bus, all the details are necessary with additional information of say number of passengers. If it is procedure-oriented programming, for every new vehicle, a new form must be re-created, whereas, in OOP, an existing base class of vehicles can be extended to design any number of vehicle types.

Suppose there is a defect or error in the transmission type, in a procedure-oriented approach, every vehicle type form must be opened and modified, whereas, in OOP, only one vehicle class is modified, and using the inheritance feature, all the other classes of vehicles get corrected.

Figure 2.11 POP and OOP

2.8 Self-Assessment Questions

- Q1. Explain logical reasoning concepts with an example. [6 marks, L4]
- Q2. Discuss concepts in problem-solving methods. [5 marks, L3]
- Q3. Discuss the characteristics of the algorithm. [3 marks, L1]
- Q4. Write an algorithm to find the sum and average of 3 numbers. [5 marks, L2]
- Q5. Write an algorithm to print numbers divisible by 7 from 1 to 100 [5 marks, L3]
- Q6. Explain the advantages and drawbacks of a flowchart. [4 marks, L2]
- Q7. Discuss symbols used in the flowchart. [5 marks, L3]
- Q8. Draw a flowchart for printing odd numbers between 1 and 'n'. Read the value of 'n'. [6 marks,

L3]

- Q9. Draw a flowchart for reversing a number and check whether it is palindrome. [8 marks, L2]
- Q10. Write a pseudocode to declare the results of the student. Consider marks<50 as Fail, marks>=50, marks<60 as Second class, marks>60, and marks<70 as First class. Marks>70 is the distinction. [5 marks, L4]
- Q11. Discuss the concept of decomposition and abstraction in software. [5 marks, L4]
- Q12. Classify programming languages with suitable examples [6 marks, L1]
- Q13. Compare procedure-oriented programming and OOP. [6 marks, L2]
- Q14. Discuss abstraction with a suitable example. [5 marks, L1]

2.9 Self-Assessment Activities

- A1. Develop an algorithm to multiply two matrices.
- A2. Draw a flowchart to show the activities performed in an ATM.
- A3. Write an algorithm to find a textbook in the library.
- A4. Justify the procedure-oriented programming paradigm by giving suitable scenarios.
- A5. How does a pseudocode work? Explain using an application to search a number in a list of records.

2.10 Multiple-Choice Questions

- Q1. The stepwise description of steps to go from initial state to final is _____. [1 mark, L1]
- A. Program
 - B. Flowchart
 - C. Algorithm
 - D. Functions
- Q2. An algorithm represented in the form of programming language is _____. [1 mark, L1]
- A. Pseudocode
 - B. C Program
 - C. None of the options
 - D. Procedure
- Q3. The _____ provides a pictorial representation of a given problem. [1 mark, L1]
- A. Flowchart
 - B. Algorithm
 - C. Subroutine
 - D. Pseudocode
- Q4. The decision is represented using the _____ symbol. [1 mark, L1]
- A. Circle
 - B. Rectangle
 - C. Diamond
 - D. None of these

- Q5. Input/Output operation is shown using _____. [1 mark, L1]
A. Rectangle
B. Triangle
C. Parallelogram
D. Circle
- Q6. Part of the algorithm which is repeated for a fixed number of times is classified as _____. [1 mark, L3]
A. Iteration
B. Sequence
C. Selection
D. Condition
- Q7. What are the disadvantages of Algorithms? [1 mark, L1]
A. It takes a long time to write an algorithm.
B. Hard to demonstrate.
C. Both A and B
D. None of the above
- Q8. _____ symbol used to calculate or process any data in the flowchart [L1 mark, L3]
A. Connector
B. Terminal Box
C. Input/Output
D. Process
- Q9. _____ is a method of hiding unimportant details and highlighting major features. [1 mark, L1]
A. Procedure-oriented Programming
B. Abstraction
C. Divide and Conquer
D. Decomposition
- Q10. Dividing a main problem into subproblems and handling them is known as _____. [1 mark, L1]
A. Modularity
B. Abstraction
C. Logical Reasoning
D. Top-down Design

2.11 Keys to Multiple-Choice Questions

- Q1. Algorithm (C).
Q2. Pseudocode (A).
Q3. Flowchart (A).
Q4. Diamond (C).
Q5. Parallelogram (C).
Q6. Iteration (A).
Q7. Both A and B (C).
Q8. Process (D).
Q9. Abstraction (B)
Q10. Modularity (A).

2.12 Summary of the Unit

Problem solving is done using the four methods namely defining a problem, logical reasoning, divide and conquer, and abstraction. These methods are interconnected in the computational thinking approach. The problem definition is based on functional and non-functional requirements. Logical reasoning helps predict the course of actions and the results. Algorithms, flowcharts, and pseudocode are the tools used for building logic. Algorithms include sequential, selection, repetition, control abstraction, and concurrency patterns. Flowcharts are the models representing logic.

2.13 Keywords:

- Problem Solving
- Logical Reasoning
- Decomposition
- Abstraction
- Algorithm
- Flowchart
- Pseudocode

2.14 Recommended Learning Resources

[1] Herbert Schildt. (2017). *C Programming: The Complete Reference*. 4th ed. USA: McGraw-Hill.