Parul® University

Subject: Fundamental of Computer Science

Unit-3 COMPUTATIONAL PROBLEM SOLVING

Parul®
University

1. Program Development Cycle

It is the **step-by-step process of designing, writing, testing, and maintaining a program** to solve a computational problem.
It ensures that the solution is **systematic, correct, and efficient**.

1. **Problem Definition (Understanding the Problem)**

* Identify what the computational problem is asking.
* Define **input, processing, and output** clearly.
* Example: “Find the largest number in a list.”

  * Input → list of numbers
  * Processing → compare and find max
  * Output → largest number

1. Program Development Cycle

2. Problem Analysis

* Break the problem into smaller subproblems.
* Decide constraints, assumptions, and conditions.
* Example: Can the list be empty? Are numbers integers only?

3. Design the Algorithm / Flowchart

* Use **algorithms** (step-by-step instructions) or **flowcharts** (diagrammatic steps) to plan the solution.
* Example:

  * Step 1: Assume first element as max
  * Step 2: Compare with each element
  * Step 3: Update max if larger found
  * Step 4: Display result

Parul®
University

1. Program Development Cycle

4. Coding (Program Implementation)

***** Translate the algorithm into a programming language (Python, C, Java, etc.).
* Example in Python:

```
python
numbers = [12, 45, 7, 89, 23]
largest = max(numbers)
print("Largest number is:", largest)
```

5. Testing & Debugging

* Test the program with different inputs.
* Find and fix logical, runtime, or syntax errors.
* Example: What happens if the list has negative numbers? Or only one element?

1. Program Development Cycle

6. Documentation

* Write explanations (comments, user manuals, guides).
* Helps users and future developers understand the program.

7. Maintenance

* Update or modify the program when requirements change.
* Example: Extend program to find **both largest and smallest** numbers.

2. Pseudocode

Pseudocode is a step-by-step description of how an algorithm works.

It is written in simple English-like statements (not strict programming syntax).

Used for problem-solving and program

Structure of Pseudocode :-

1. **START / END – beginning and ending of the algorithm.**

Example :- START
INPUT number1
INPUT number2
sum ← number1 + number2
OUTPUT sum
END

2. Pseudocode

2. Input/Output – reading and displaying data.

Example :-
```
START
        INPUT A, B
        IF A > B THEN
        OUTPUT "A is largest“
        ELSE
        OUTPUT "B is largest"
        ENDIF
END
```

2. Pseudocode

3. Processing – calculations, conditions, loops.

Example :-
```
START
        INPUT N
        fact ← 1
        FOR i ← 1 TO N
        fact ← fact * i
        ENDFOR
        OUTPUT fact
END
```

4. Decision Making – IF…THEN…ELSE.

5. Repetition – WHILE, FOR, REPEAT loops.

Parul® University

3. Flowchart

In Fundamentals of Computer System, flowcharts are used to represent the logical flow of data and instructions in a process.

- Common Flowchart Symbols
- Oval → Start / End
- Parallelogram → Input / Output
- Rectangle → Process / Instruction
- Diamond → Decision (Yes/No)
- Arrows → Show flow direction

3. Flowchart

Example: Basic Computer System Flow (Input → Process → Output)
Description:-

Start

Input Data (from user, keyboard, mouse, etc.)

Process Data (CPU executes instructions)

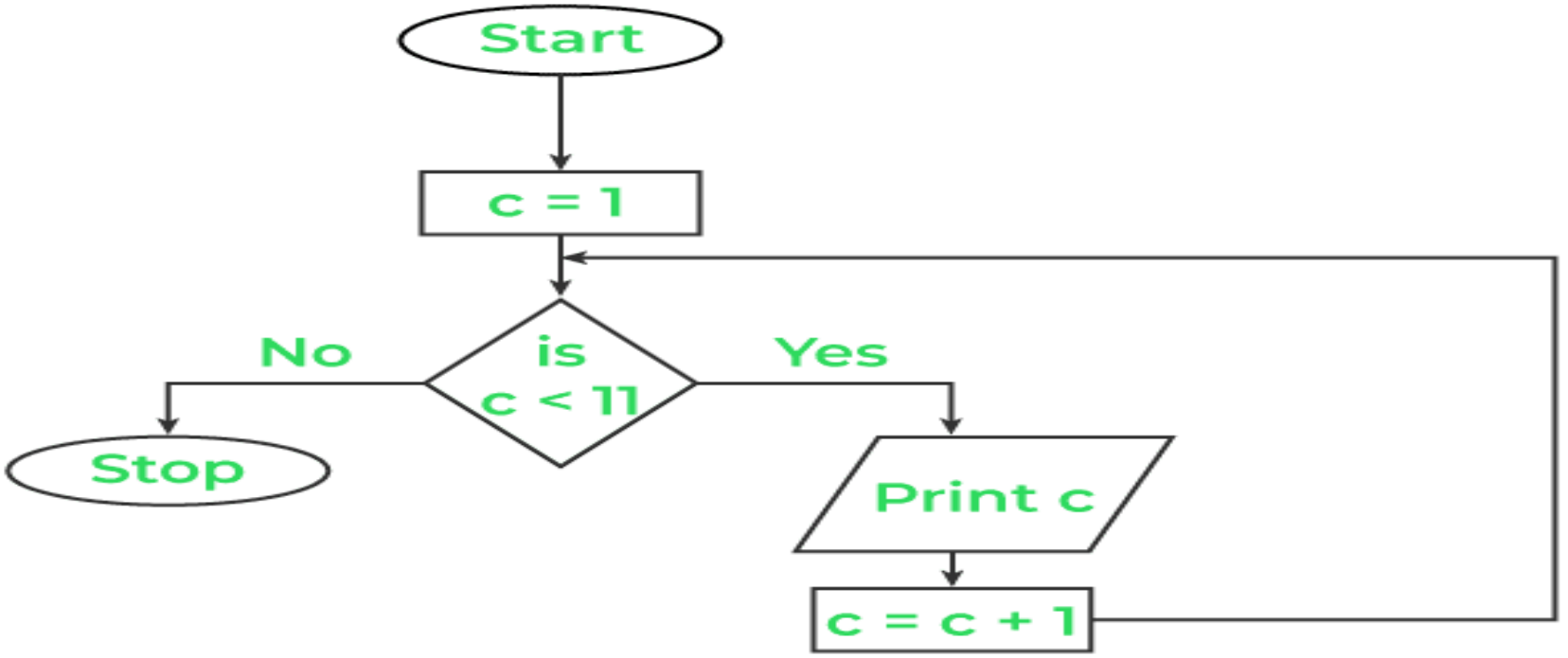
Decision → Is processing successful?

If Yes → Show Output

If No → Show Error Message

End

3. Flowchart



Parul® University

4. Representing Information As Bits

1. What is a Bit?

A bit (binary digit) is the smallest unit of information in a computer.

It can take only two values:

0 → OFF / False

1 → ON / True

Computers use binary (base-2) because electronic circuits easily represent two states: high voltage (1) and low voltage (0).

4. Representing Information As Bits

2. Grouping Bits

Nibble → 4 bits

Byte → 8 bits

Word → Typically 16, 32, or 64 bits (depends on CPU architecture)

3. Types of Information Representation

Computers represent all types of information (numbers, text, images, audio, video) using bits:

a) Numbers

Unsigned integers → Represented directly in binary (e.g., 13 = 1101)

4. Representing Information As Bits

Signed integers → Usually represented using two's complement

Real numbers → Represented using floating-point notation (IEEE 754)

b) Characters / Text

Represented by assigning binary codes:

ASCII → 7-bit / 8-bit code (e.g., A = 65 = 01000001)

Unicode (UTF-8/UTF-16) → Supports characters from all languages

c) Images

Stored as pixels, each pixel represented by bits:

4. Representing Information As Bits

Black & White → 1 bit per pixel

Grayscale → 8 bits (0–255 shades)

Color (RGB) → 24 bits (8 bits each for Red, Green, Blue)

d) Audio

Represented by sampling sound waves at regular intervals.

Each sample stored as binary (e.g., CD quality = 44.1 kHz, 16-bit per sample).

e) Video

Sequence of images (frames) + audio, all represented in bits.

Parul® University

5. Binary System

- What is the Binary System?

The Binary Number System is the fundamental language of computers.

It uses only two digits: 0 and 1 (called bits – binary digits).

Every piece of information a computer processes (numbers, text, images, sounds) is ultimately represented in binary.

- Why Binary in Computers?

- Digital Nature of Hardware

Parul® University

5. Binary System

Computer circuits have two stable states:

ON (1) – current flows

OFF (0) – no current

- Perfect for representing binary digits.
- Simplicity & Reliability
- Easy to design circuits using two states.
- Less chance of error compared to multiple states.
- Efficient Processing

Logical operations (AND, OR, NOT, XOR) can be directly implemented using binary.

5. Binary System

- Logical operations (AND, OR, NOT, XOR) can be directly implemented using binary.

◆ Place Value in Binary

Binary is a base-2 number system.

Each digit's place represents a power of 2:

Example:
Binary number 1011 ⟶

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11(decimal)$$

5. Binary System

- Representing Data in Binary

- Numbers → directly converted into binary.
- Text → stored using character encoding (e.g., ASCII, Unicode).
- Images → pixels represented as binary values.
- Sound → sampled as binary sequences.

The binary system is the foundation of all computer operations, since hardware works in ON/OFF states, and all data is reduced to combinations of 0s and 1s.

6. Storing Integers

1. Representation of Integers

Computers store integers as binary numbers (0s and 1s). Since computers work with bits, integers are represented using a fixed number of bits.

- Unsigned integers → only positive numbers (including 0).
- Signed integers → both positive and negative numbers.

(i) Unsigned Integers

If we have n bits:

Range = 0 to $(2^n - 1)$

6. Storing Integers

Example: With 3 bits → numbers from 0 to 7

000 → 0

001 → 1

111 → 7

(ii) Signed Integers

There are multiple ways to represent signed integers:

Sign-and-Magnitude

6. Storing Integers

Leftmost bit → sign (0 = positive, 1 = negative).

Example (with 4 bits):

0101 = +5

1101 = -5

One’s Complement

Negative number = invert all bits of the positive number.

Problem: Has two zeros (+0 and -0).

6. Storing Integers

Two’s Complement (most used)

Negative number = invert all bits + add 1.

Advantages:

- Only one zero.
- Easy arithmetic (addition/subtraction works directly).

6. Storing Integers

Example (with 4 bits):

0101 = +5

1011 = -5

Range (for n bits):

-2^{n-1} to $(2^{n-1} - 1)$

For 8 bits → −128 to +127

6. Storing Integers

2. Integer Storage in Memory

Integers are stored in bytes (1 byte = 8 bits).

Common sizes: 8-bit, 16-bit, 32-bit, 64-bit.

Example (32-bit signed int):

Range: −2,147,483,648 to +2,147,483,647

3. Endianness (Storage Order in Memory)

When integers occupy multiple bytes :-

6. Storing Integers

Big-endian → Most significant byte stored first.

Little-endian → Least significant byte stored first.

(Example: Intel processors use little-endian.)

7. Storing Fractions

Fractions (like ½, 0.25, 3/8) are usually stored using floating-point representation because integers can't directly represent non-whole numbers.

1. Fixed-Point Representation

The decimal (or binary) point is placed at a fixed position.

Example: If we allow 2 digits after the decimal point,
3.14 → stored as 314 with an implicit scale of 0.01.

Simple, but limited range and precision.

2. Floating-Point Representation (IEEE 754 Standard)

This is the standard way computers store fractions.

7. Storing Fractions

A floating-point number has three parts:

- Sign bit (S) → 0 = positive, 1 = negative
- Exponent (E) → stores the range (where the decimal point "floats")
- Mantissa / Significand (M) → stores the actual digits of the fraction

3. Rational Representation (Fraction Form)

Sometimes, especially in math software, fractions are stored as a pair of integers:

Numerator / Denominator

Example: 3/8 stored as (3, 8)

7. Storing Fractions

More exact than floating-point, but less common in hardware-level systems.
Example in IEEE 754 (32-bit).

Let’s take 0.75:

Decimal → Binary = 0.11

Normalize: 1.1 × 2^-1

Sign = 0 (positive)

Exponent = 127 - 1 = 126 (bias = 127)

Mantissa = .1

Parul® University

7. Storing Fractions

Stored as:
0 | 01111110 | 10000000000000000000000

Fractions are stored as fixed-point, floating-point (most common), or sometimes numerator/denominator pairs.

The IEEE 754 floating-point standard is the backbone of modern computing for storing fractions.

8. Examples of Computational Problems

(1)Arithmetic Problems

(2)Sorting Problem

(3)Searching Problems

(4)Sorting names alphabetically.

(5)String Processing Problems

(6)Checking whether a word is a palindrome.

(7)Counting the number of vowels in a string

8. Examples of Computational Problems

(1)Arithmetic Problems:

Adding, subtracting, multiplying, or dividing large numbers.

Computing factorial of a number (e.g., n!).

Finding the greatest common divisor (GCD) of two numbers.

(2)Searching Problems:

Finding a specific element in a list (e.g., linear search, binary search).

Locating a record in a database by key.

(3)Sorting Problems:

Arranging numbers in ascending or descending order (e.g., Bubble Sort, Merge Sort, Quick Sort).

(4) String Processing Problems:

Checking whether a word is a palindrome.

Counting the number of vowels in a string.

pattern matching (finding a substring inside a text).

Parul® University

8.Examples of Computational Problems

(5)Graph/Network Problems:

Finding the shortest path between two cities (Dijkstra's algorithm).

Determining if a graph is connected.

(6)Data Encoding/Decoding Problems:

Converting decimal to binary and vice versa.

Compressing or encrypting data

Parul®
University

9.Iterative and recursive Approches to solve problem

1. Iterative Approach:

The iterative approach solves problems step by step in a loop until the communication is complete.
Example Problem:
A sender wants to send a message to a receiver, character by character.

ALGORITHM:
Algorithm (Iterative):

Start

Initialize message = "HELLO"

Parul®
University

Iterative and recursive Approches to solve problem

For each character in the message:

Send character

Wait for acknowledgment

Repeat until all characters are sent

Stop

8.Iterative and recursive Approches to solve problem

Pseudocode (Iterative):

```
function sendMessage(message):
 for i = 0 to length(message)-1:
 send(message[i])
 wait for acknowledgment
 print("Message Sent Successfully")
```

Explanation:
Here, the loop ensures that each character is transmitted step by step.

8.Iterative and recursive Approches to solve problem

2. Recursive Approach

The recursive approach solves the communication problem by letting the function call itself until the entire message is sent.

Example Problem:
Same message sending problem but solved recursively.
Algorithm (Recursive):

Start

If the message is empty → stop

8.Iterative and recursive Approches to solve problem

Otherwise:

Send the first character

Wait for acknowledgment

Recursively call the function for the remaining part of the message

End

8.Iterative and recursive Approches to solve problem

Pseudocode (Recursive):

```
function sendMessage(message):
if message is empty:
 print("Message Sent Successfully")
 return
send(message[0])
wait for acknowledgment
sendMessage(message[1:])   # Recursive call
```

Explanation:

The recursive approach breaks down the problem:

Send one character

Parul® University

Difference Between Iterative and Recursive

Key Difference in Communication Problem Solving:

Iterative: Uses loops (for, while) → More memory efficient.

Recursive: Uses function calls → Simpler to express but consumes more stack space.

You want the iterative and recursive approaches to solve a communication problem in Fundamentals of Computer System (FCS).

In FCS, a communication problem usually means sending and receiving data between processes, systems, or components. To understand this, let's break it into two solving approaches:

In Fundamentals of Computer Systems, computational problems are usually divided into easy problems and hard problems, based on how much time and resources they need to solve.

Parul® University

9.Easy and Hard Computational Problem

Easy Computational Problems:
These are problems that can be solved efficiently (in polynomial time, like O(n), $O(n^2)$, etc.).
They usually have simple algorithms and don't need too much memory or processing power.

Example:
Addition, subtraction, multiplication, division of integers.
Converting numbers between binary, decimal, octal, and hexadecimal.

Hard Computational problems:
These are problems that are difficult to solve efficiently because they may require exponential time (like $O(2^n)$) or nondeterministic approaches.
They often belong to NP-hard or NP-complete categories.

Parul®
University

9.Easy and Hard Computational Problem

Example:

Traveling Salesman Problem (TSP): Find the shortest route visiting all cities once.

Graph Coloring Problem: Coloring graph nodes with minimum colors such that no two adjacent nodes have the same color.

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in