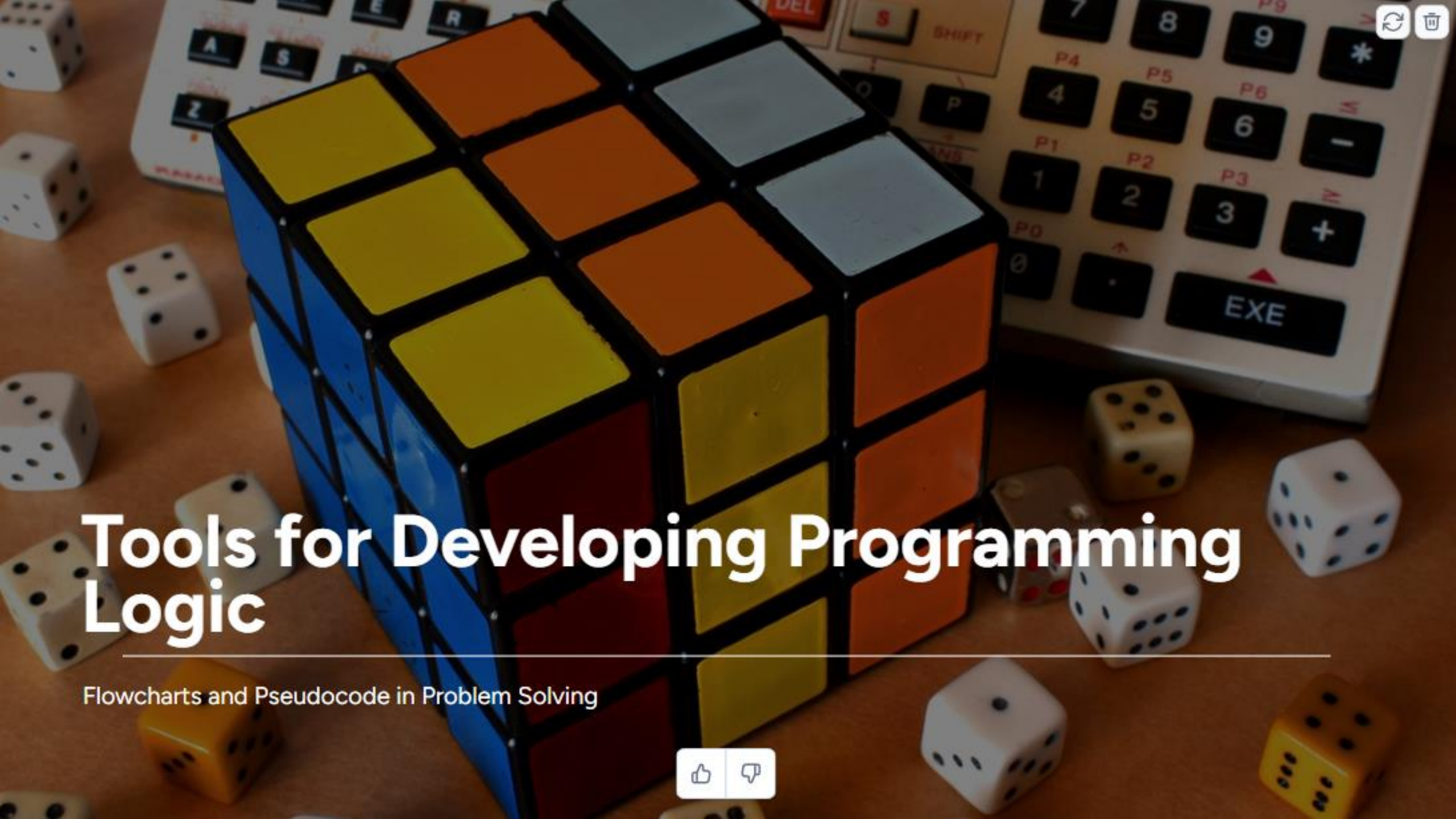


Tools for Developing Programming Logic

Flowcharts and Pseudocode in Problem Solving

Table of contents

Introduction to Programming Logic

What is Programming Logic?

The basic way programmers understand and organize their code to produce desired results is known as programming logic. Fundamentally, it involves decomposing issues into smaller, more manageable components and formulating a plan of action to address each one.

Key Elements of Programming Logic:

1. Sequence:

Programming logic involves arranging commands in a sequential order, allowing the computer to execute them one after another.

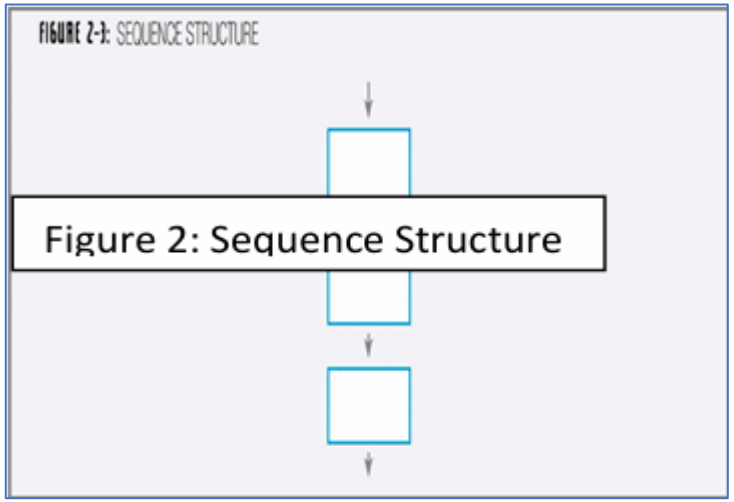
2. Selection (Conditional Statements):

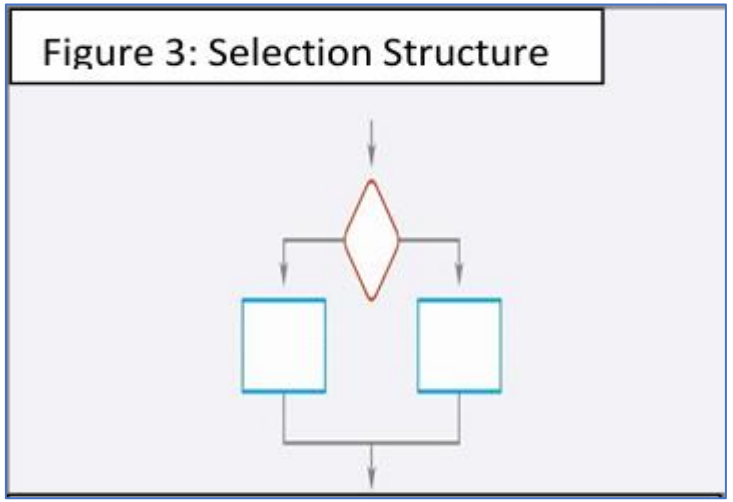
Conditions or logical tests are used to direct the flow of a program. For instance, "if-else" statements allow the code to make decisions based on specific conditions.

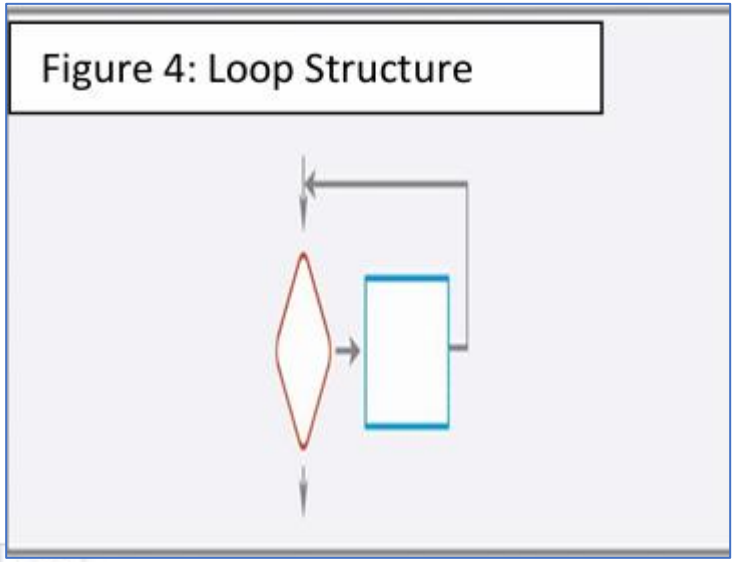
3. Iteration (Loops):

Loops enable the repetition of certain tasks until a condition is met. This helps in automating repetitive tasks and managing data efficiently.

Introduction to Programming Logic

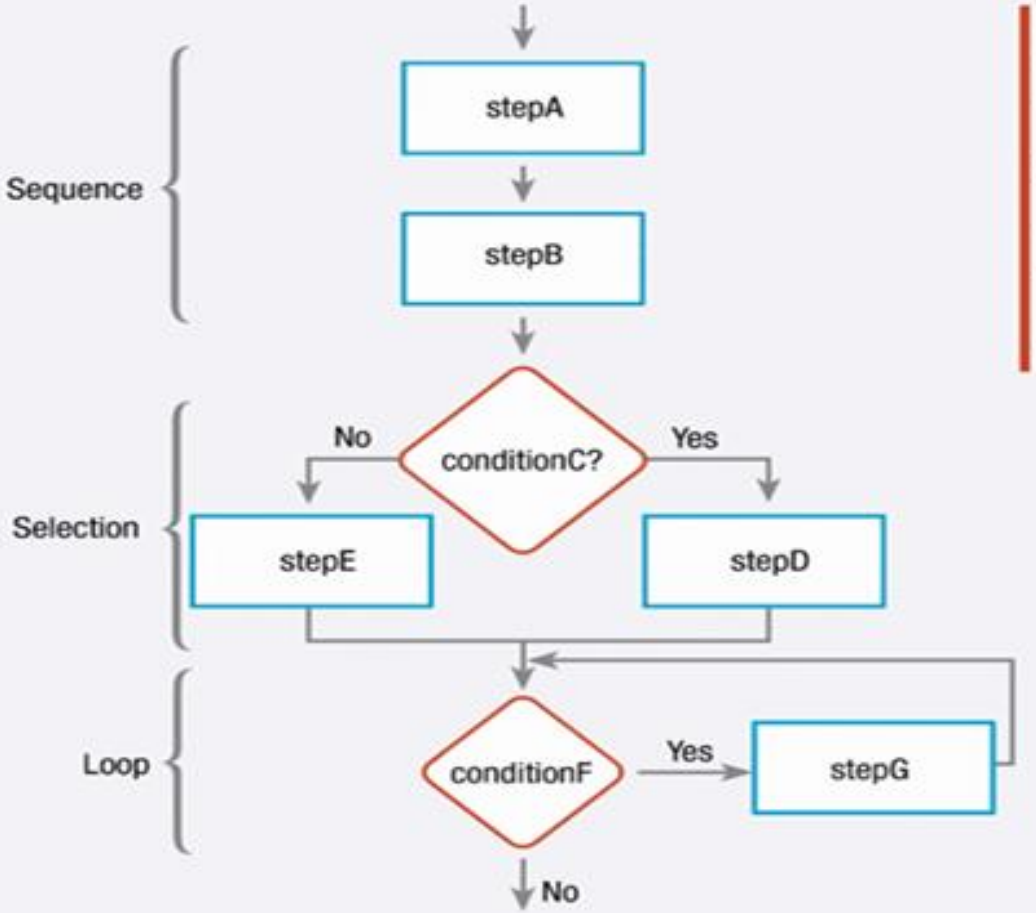






Introduction to Programming Logic

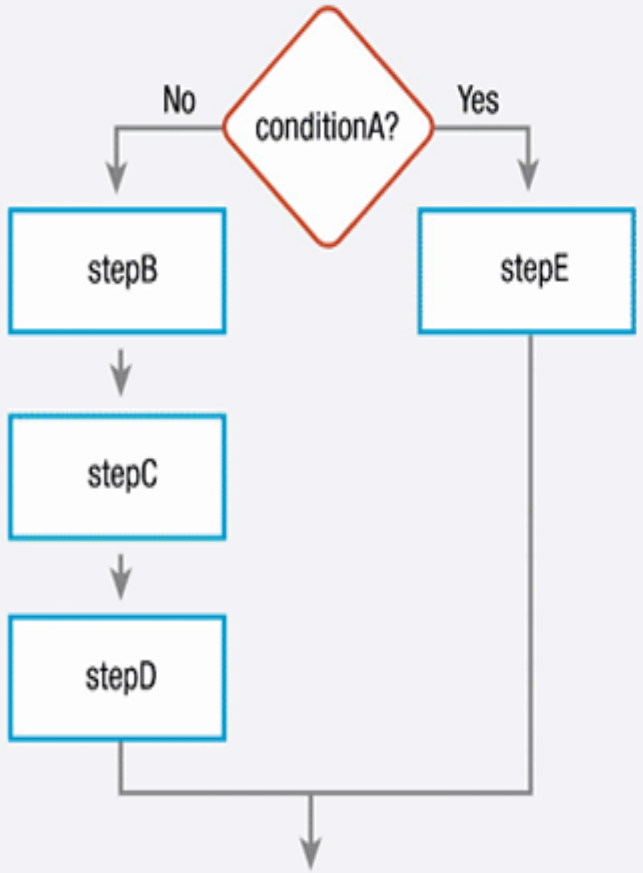
FIGURE 2-1: STRUCTURED FLOWCHART AND PSEUDOCODE




```
do stepA
do stepB
if conditionC is true then
      do stepD
else
      do stepE
endif
while conditionF is true
      do stepG
endwhile
```

Introduction to Programming Logic

FIGURE 2-8: FLOWCHART AND PSEUDOCODE SHOWING A SEQUENCE NESTED WITHIN A SELECTION




```
if conditionA is true then
     do stepE
else
     do stepB
     do stepC
     do stepD
endif
```

Introduction to Programming Logic

Why is Programming Logic Important?

Understanding programming logic is crucial for several reasons:

1. Problem Solving:

Programming logic helps in breaking down complex problems into smaller, manageable parts. This simplification enables developers to solve problems systematically.

2. Efficient Code Writing:

By using logical structures, programmers can write code that is not only understandable but also efficient. It helps in avoiding redundancy and streamlining the execution process.

3. Enhancing Debugging Skills:

Logical thinking assists in identifying errors within the code and debugging more effectively. It becomes easier to trace the flow of a program and identify where issues might occur.

Introduction to Programming Logic

Building Blocks of Programming Logic:

Variables:

Variables are used to store data within a program. They can be manipulated and used in logical operations.

Operators:

Operators perform specific operations on variables and values. For instance, arithmetic operators (+, -, *, /) are used for mathematical operations.

Control Structures:

Control structures like loops (for, while, do-while) and conditional statements (if, else if, else) control the flow of the program.

Introduction to Programming Logic

Examples of Programming Logic:

1. Conditional Statements:

```
$age = 25;

if ($age < 18) {
    echo "You are a minor.";
} else {
    echo "You are an adult.";
}
```

2. Looping:

```
for ($i = 0; $i < 9; $i++) {
    echo "Count: $i <br>";
}
```

Introduction to Programming Logic

How to Develop Strong Programming Logic?

Solving problems and practicing regularly are necessary to develop strong programming reasoning. Here are some actions you can take to sharpen your reasoning:

1. Start with Simple Problems:

Begin by solving simple problems using a structured approach. Break down tasks and analyze the steps to solve them.

2. Practice Regularly:

Regular practice and writing code help reinforce programming logic. Experiment with different problems to understand how logic applies to diverse scenarios.

3. Learn from Others' Code:

Reviewing and understanding code written by experienced developers can provide insight into how logic is applied in different contexts.

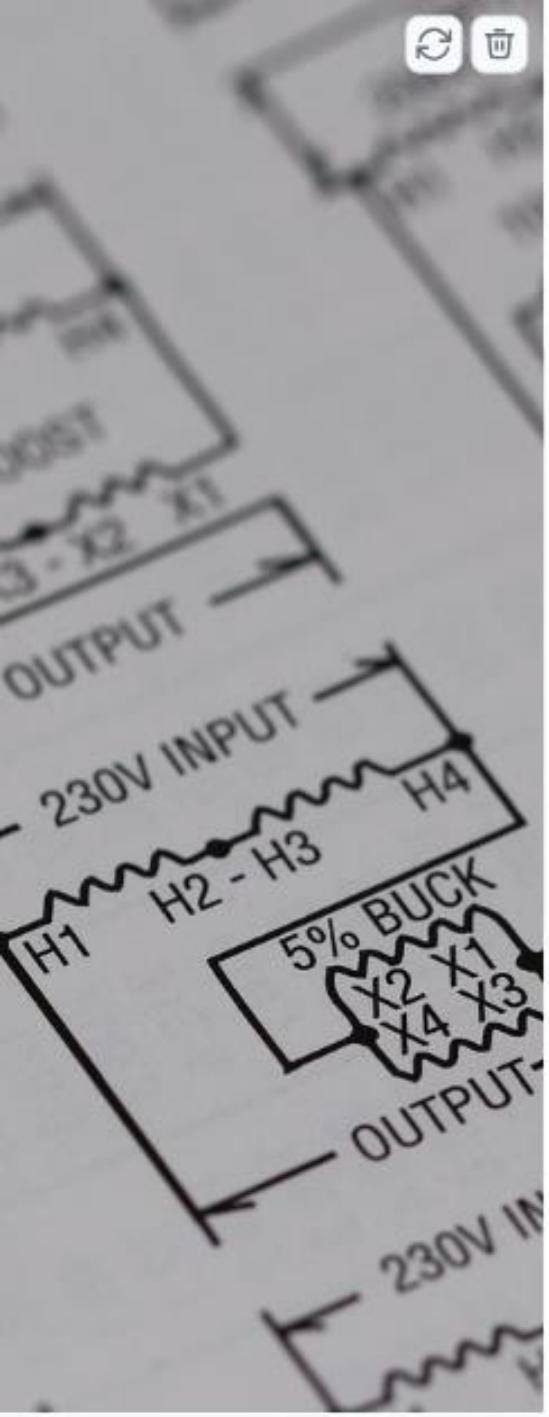
Introduction to Programming Logic

Conclusion:

Programming logic forms the foundation of coding. It's the systematic approach of thinking and organizing instructions to create functional software. Embracing logical thinking, breaking down problems, and applying structured solutions are key components for any budding programmer.

By grasping programming logic, beginners pave the way for a deeper understanding of programming languages and the ability to develop efficient and effective software solutions.

Understanding programming logic is just the first step in an exciting journey towards becoming a proficient developer!

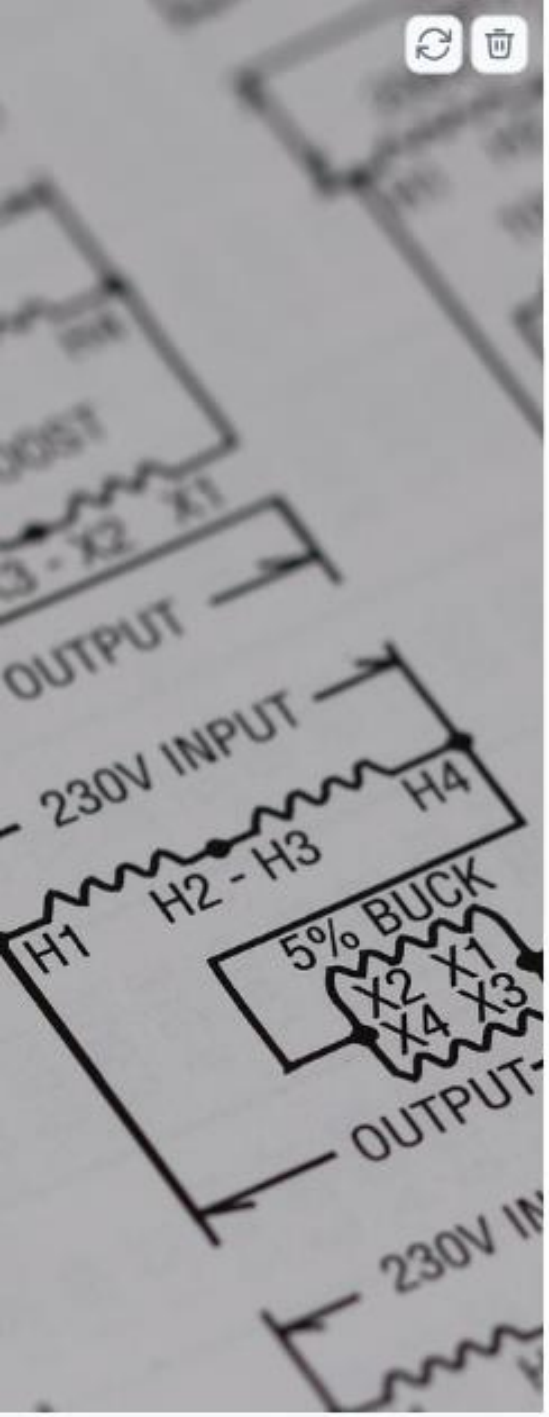


Understanding Flowcharts

What is process mapping?

> "Process mapping provides a visual representation of all the steps necessary to complete a process. You'll often hear it called a process flowchart, workflow map, or business process mapping."

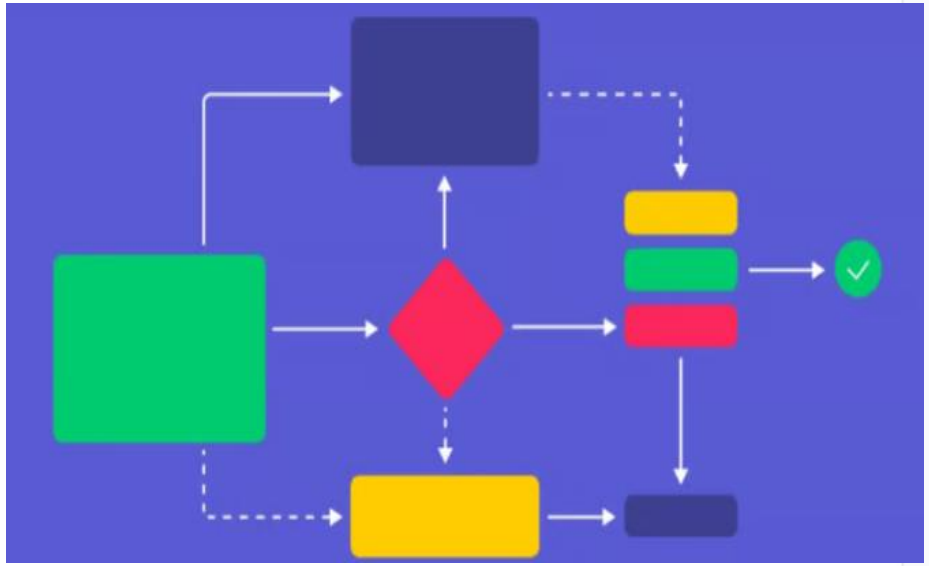
Did You Know: Process mapping is sometimes referred to as flowcharting. This method was initially created in the 1900s by Frank Gilbreth.

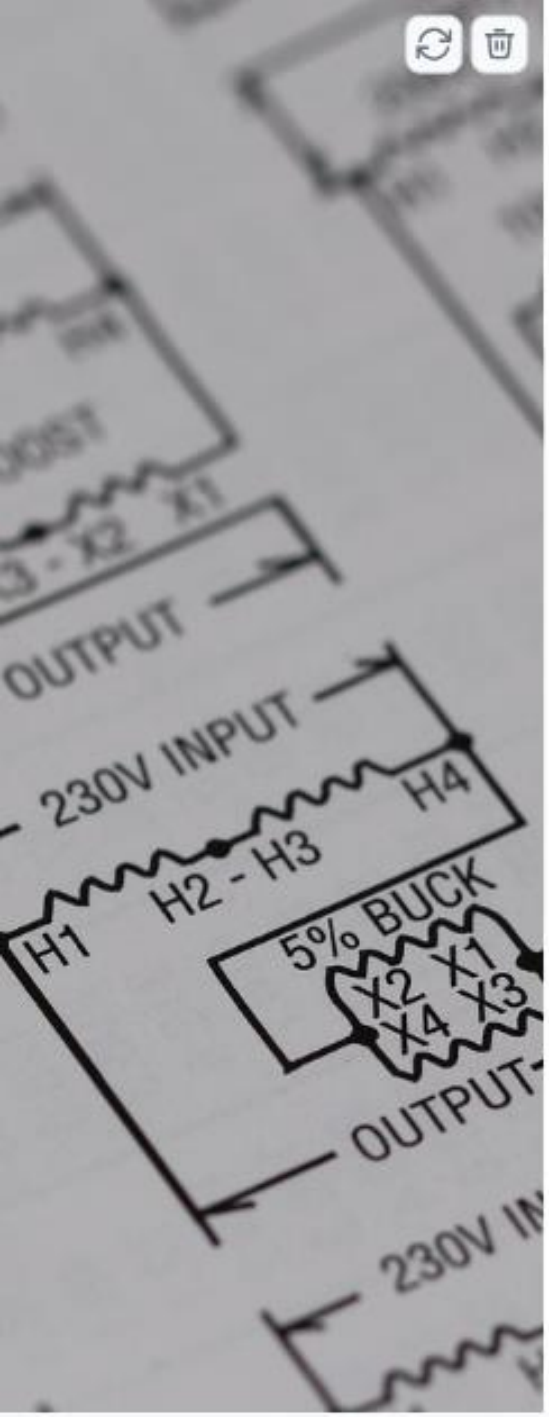


Understanding Flowcharts

The purpose of using process map symbols in a process map is to visually communicate a process's steps, flow, inputs, outputs, and other aspects. Think of them as a universal language for workflow charts.

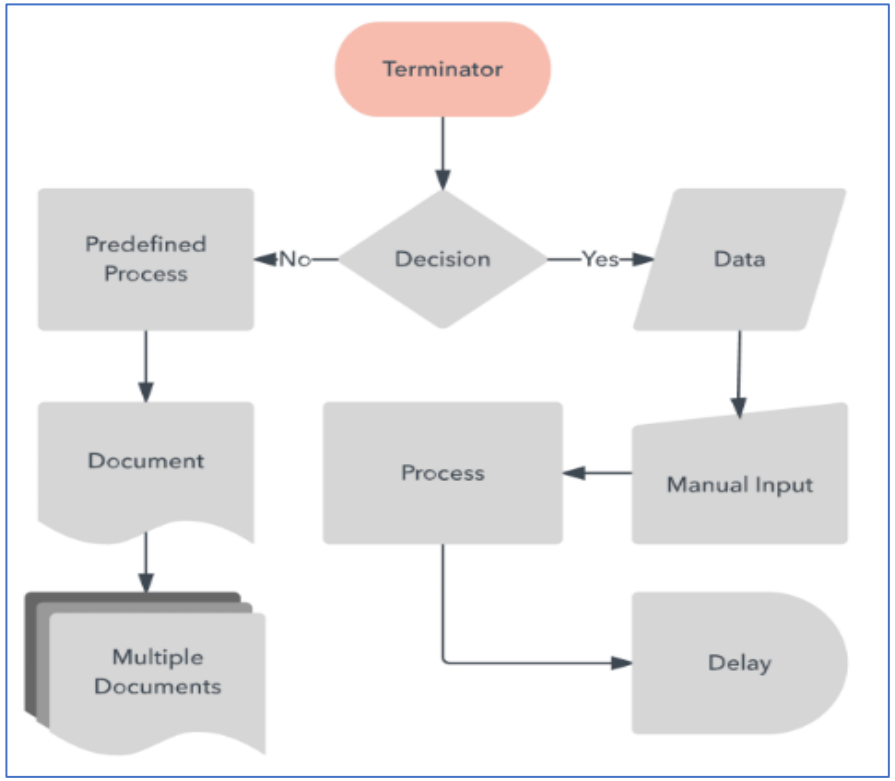
Each shape, line, and squiggle holds a specific meaning, guiding you and your team through your process.



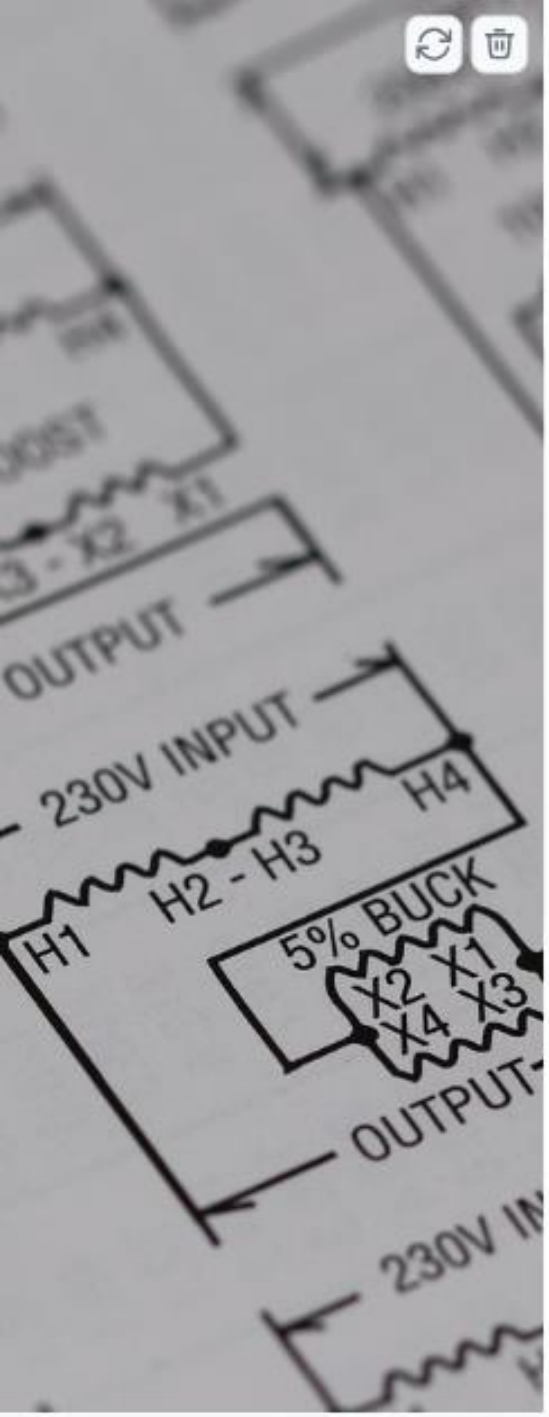


Understanding Flowcharts

There are quite a few shapes in a detailed flowchart, but here are a few common examples:



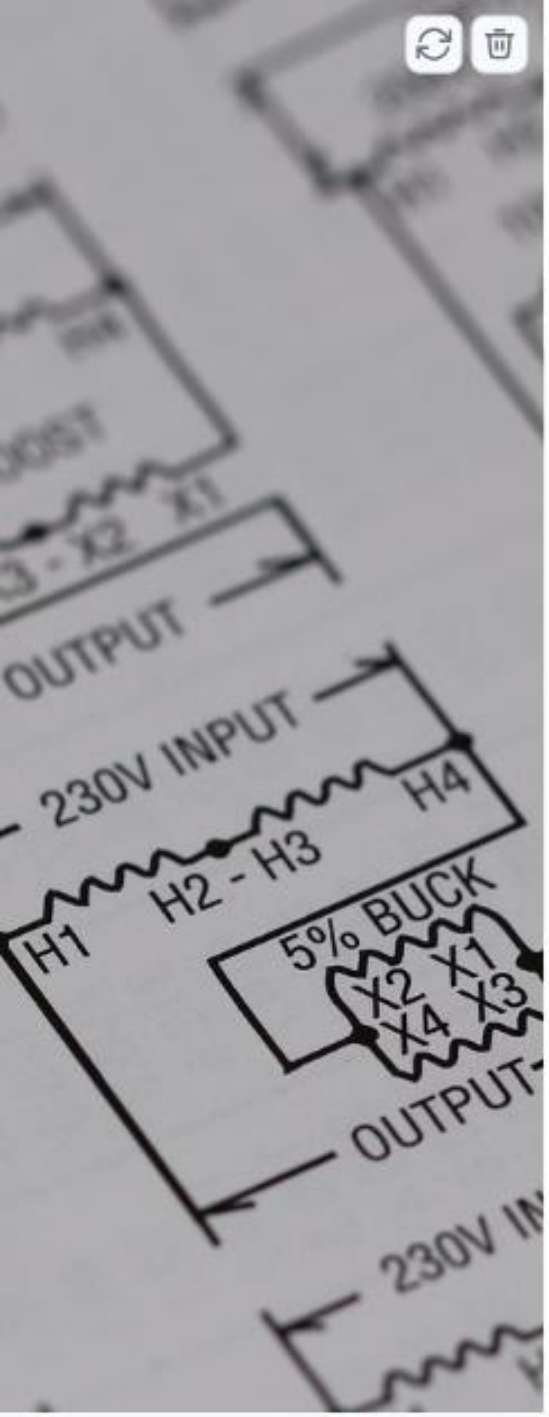
- **Ovals** represent terminal activity, or Starts and stops
- **Boxes** represents steps in the process
- **Parallelograms** require input or output
- **Rombi or diamons** signify steps that are a decision point
- A **bullet shape** (square w/rounded edge) represents temporary delay
- **Squares** with a wavy bottom indicate the use of a written document
- **Arrow** demonstrate the flow between tasks or steps



Understanding Flowcharts

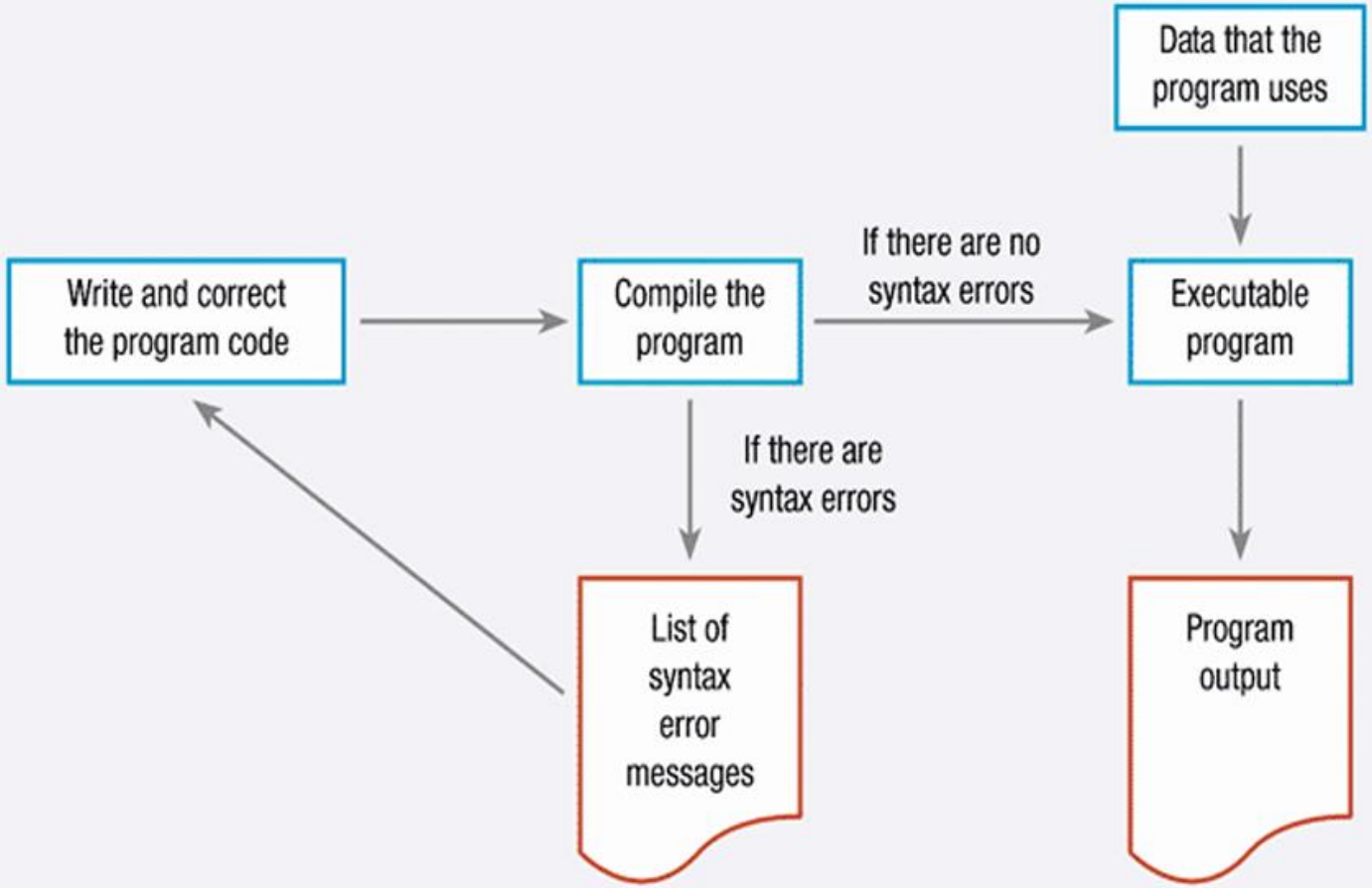
What are the process map template steps?

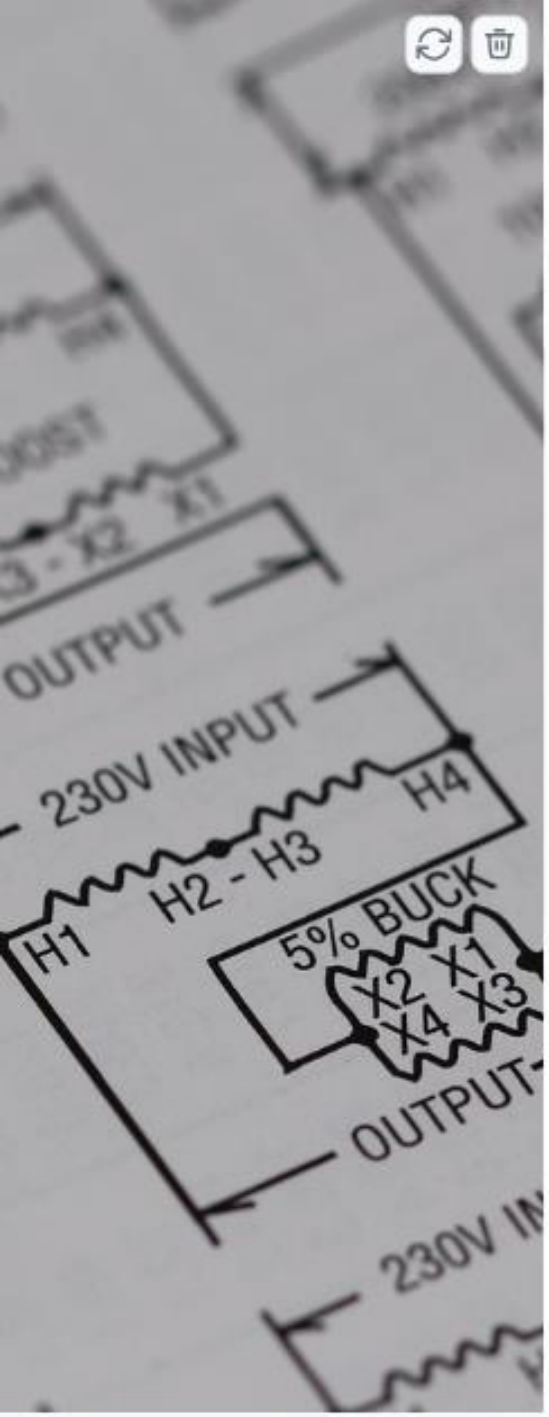
1. Identify the problem
2. Give your flowchart some structure
3. Write down each step of the process in a verb/noun format
4. Create an order or hierarchy for the process
5. Analyze the process map and make necessary adjustments



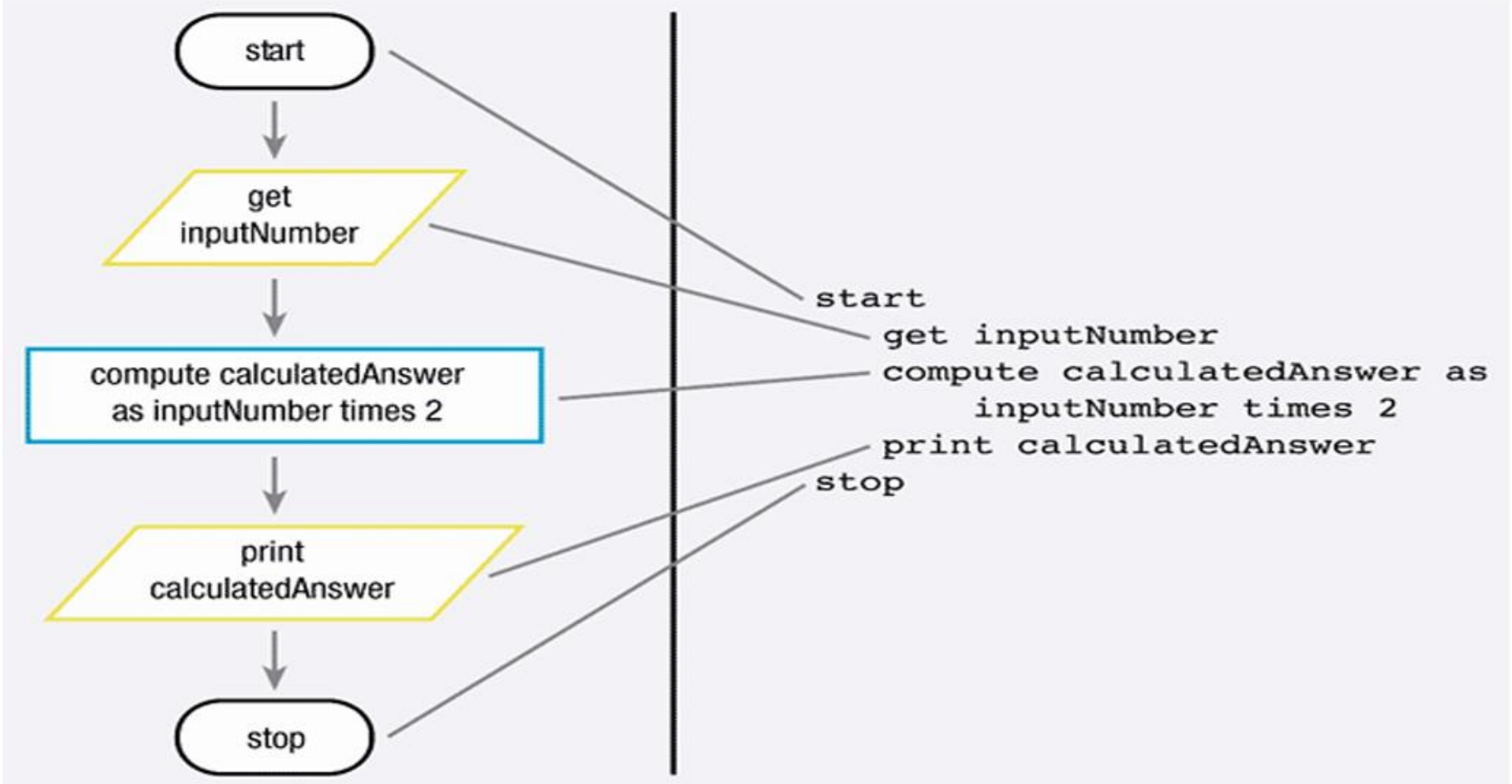
Understanding Flowcharts

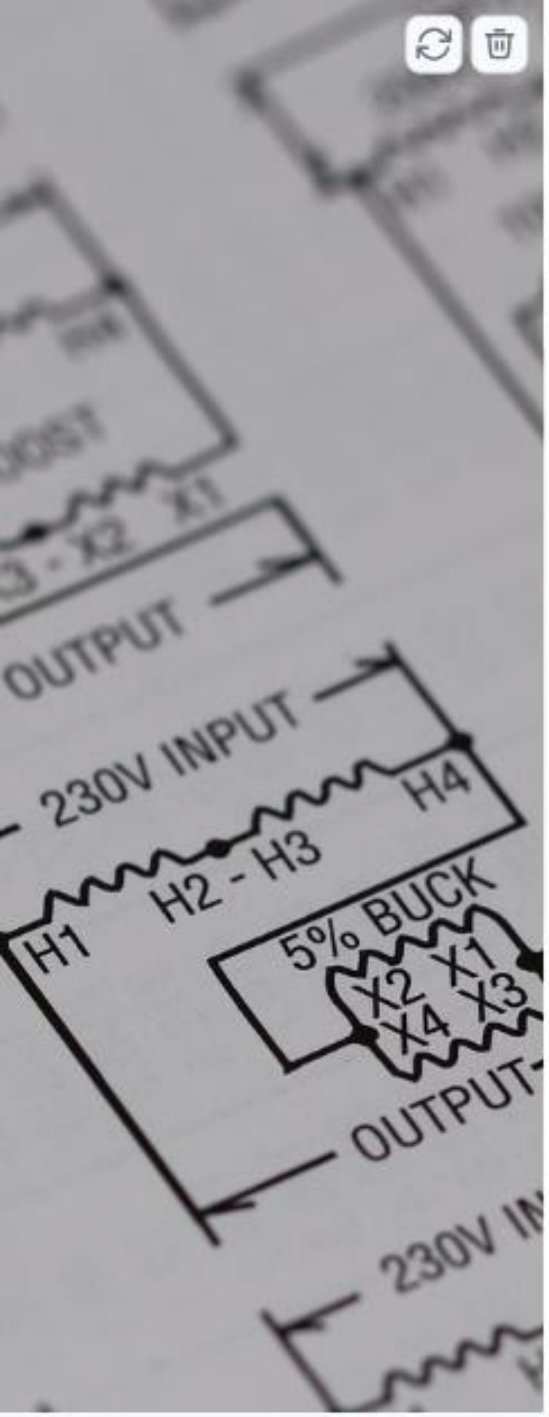
FIGURE 1-1: CREATING AN EXECUTABLE PROGRAM





Understanding Flowcharts





Understanding Flowcharts

FIGURE 1-12: FLOWCHART USING THE CONNECTOR

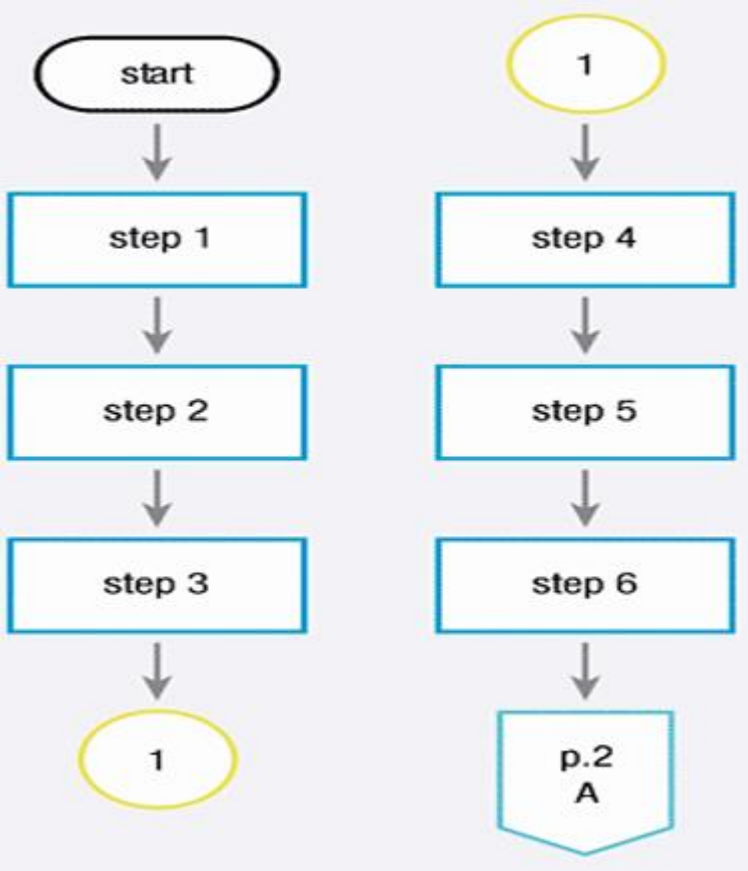
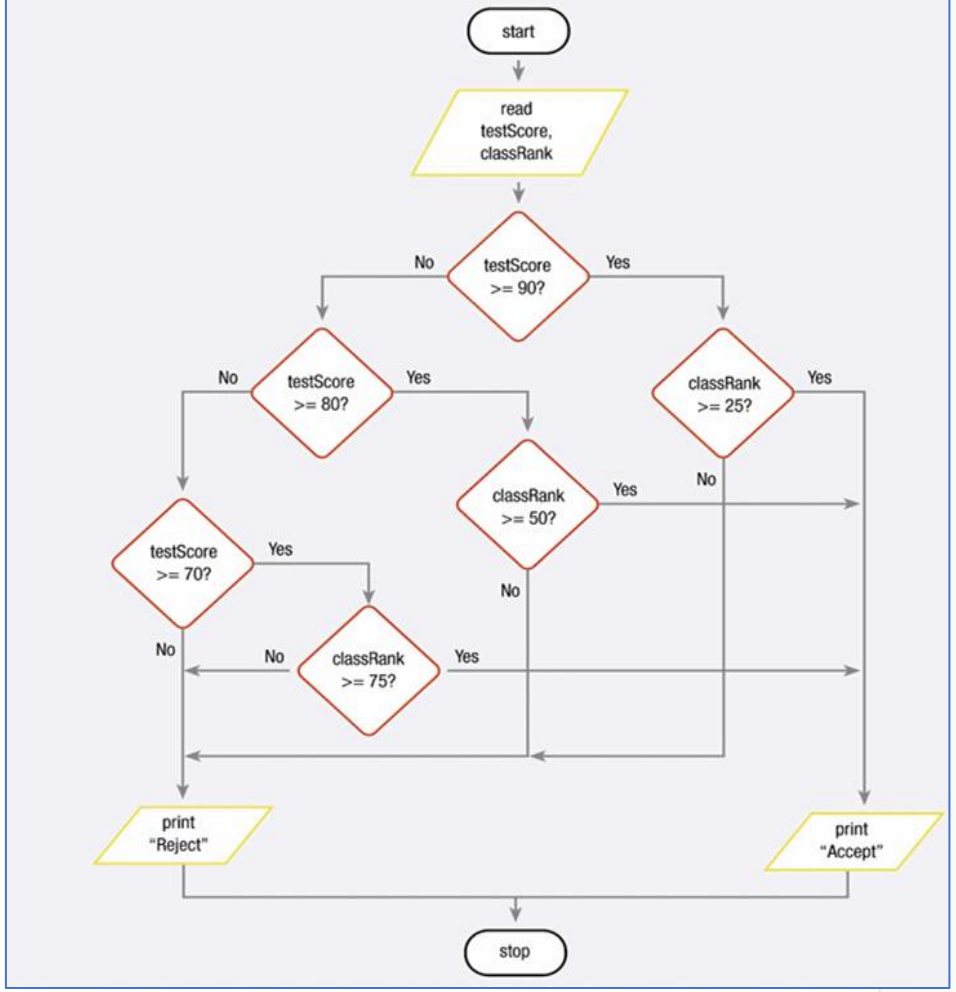
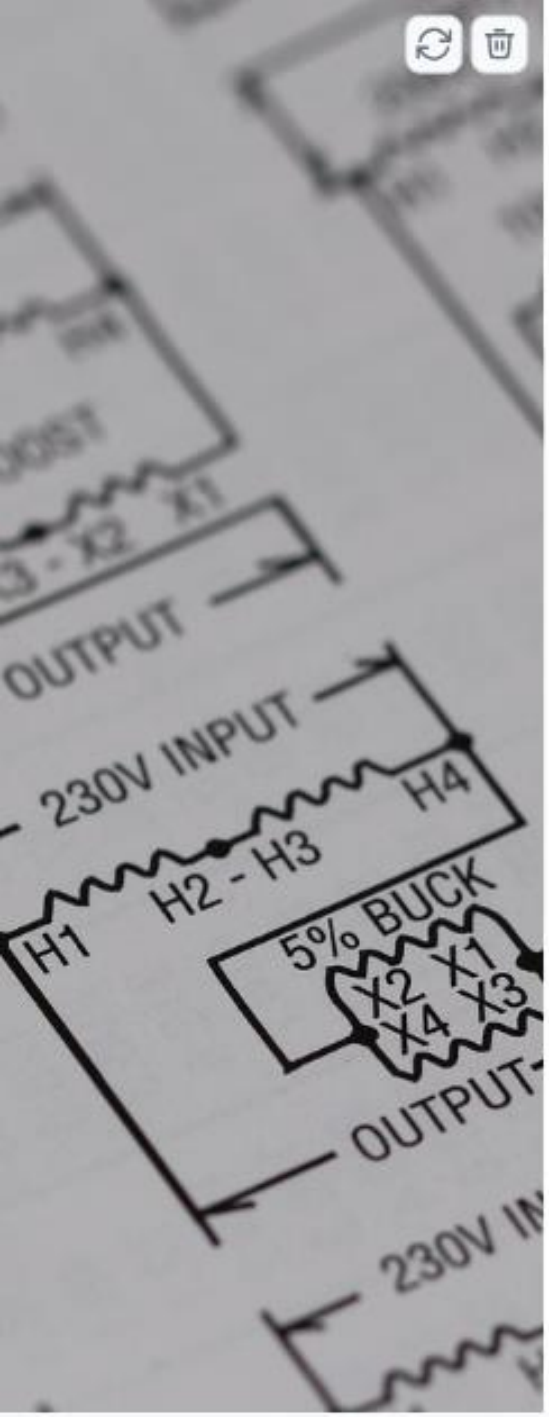
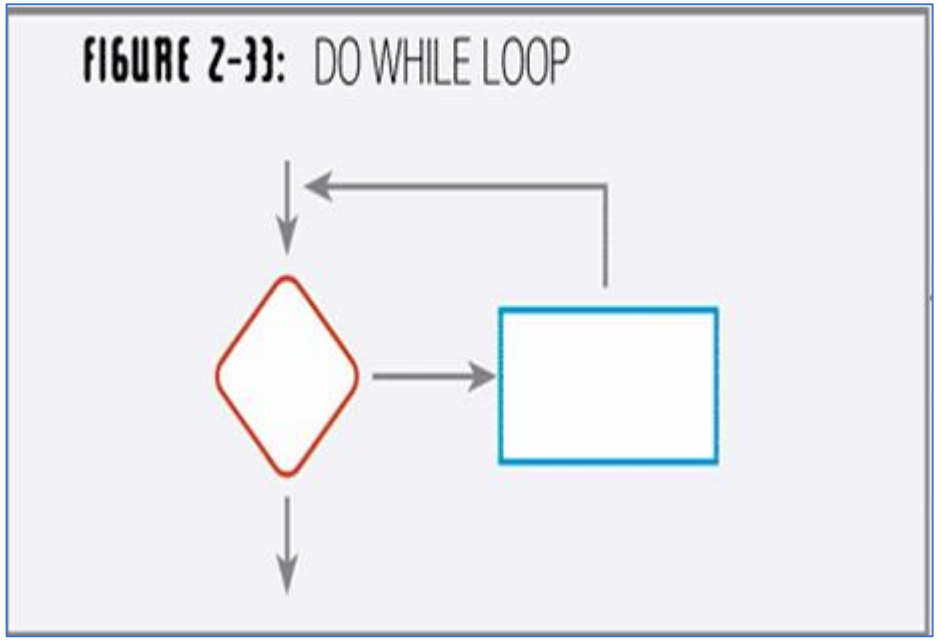


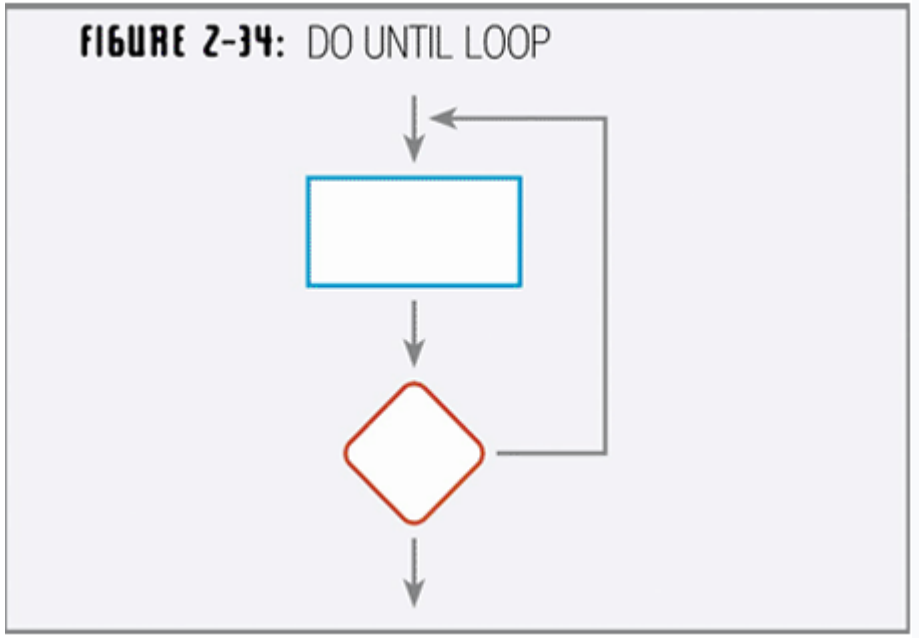
FIGURE 2-2: SPAGHETTI CODE EXAMPLE





Understanding Flowcharts



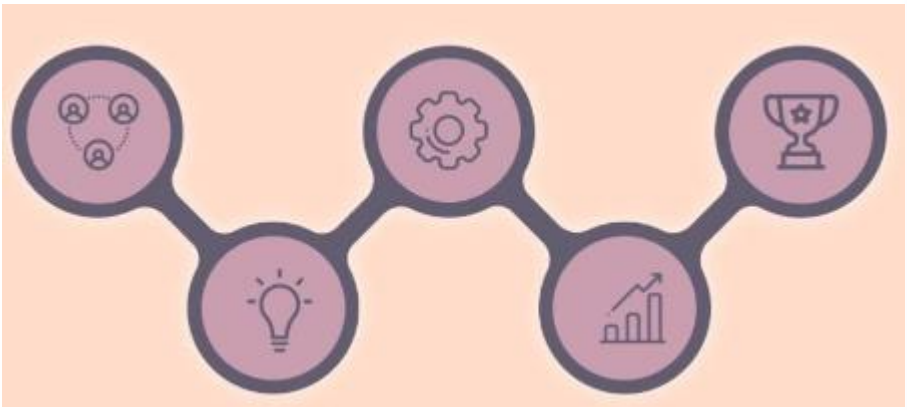


Advantages and Limitations of Flowcharts

Advantages of Flowchart

Listed are a few advantages of Flowcharts:

- Easy to make
- Communication becomes effective and easy to understand
- Mistakes can be easily identified
- Analysis becomes effective
- Synthesis becomes effectual
- Debugging becomes possible
- Logics can be easily interpreted.



Advantages and Limitations of Flowcharts

Disadvantages of Flowchart

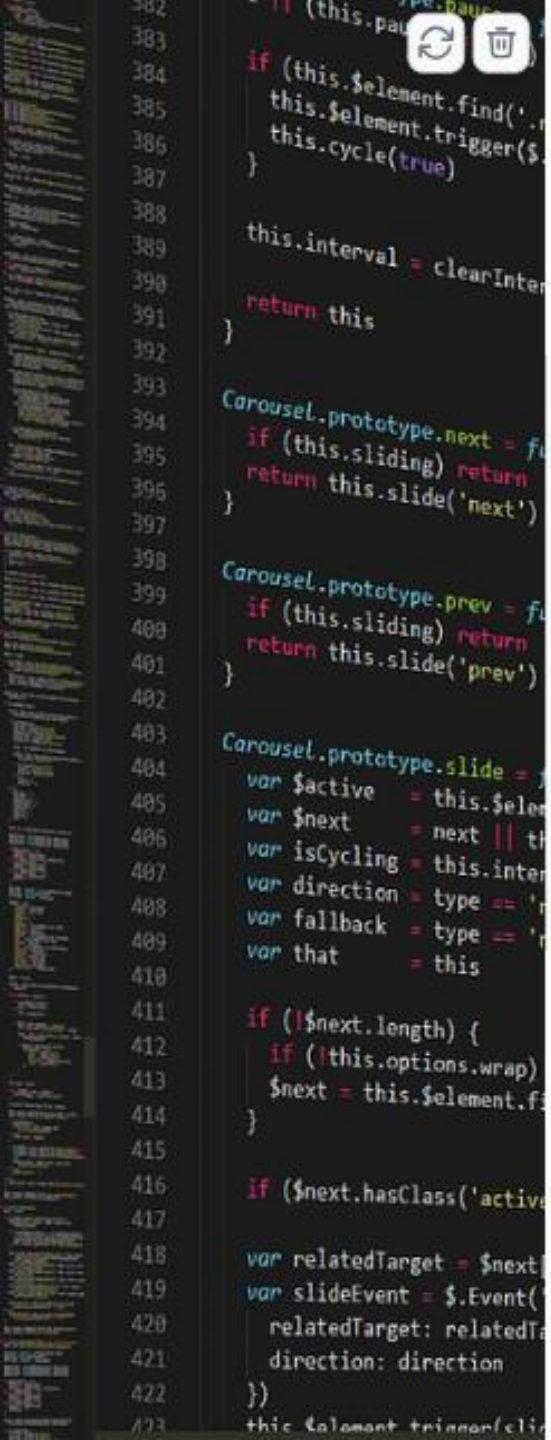
A few disadvantages of Flowcharts are as follows:

- Difficulty in presenting complex programs and tasks.
- No scope for alteration or modification
- Reproduction becomes a problem
- It's a time-consuming process
- Difficult to understand for people who don't know flowchart symbols.
- No man to computer communication.

Introduction to Pseudocode

*A **Pseudocode** is defined as a step-by-step description of an algorithm. Pseudocode does not use any programming language in its representation instead it uses the simple English language text as it is intended for human understanding rather than machine reading.*
*Pseudocode is the **intermediate state between an idea and its implementation(code)** in a high-level language.*

We use pseudocode in various fields of programming, whether it be app development, data science or web development. Pseudocode is a technique used to describe the distinct steps of an algorithm in a manner that's easy to understand for anyone with basic programming knowledge. Although pseudocode is a syntax-free description of an algorithm, it must provide a full description of the algorithm's logic so that moving from pseudocode to implementation is merely a task of translating each line into code using the syntax of any given programming language.



Benefits and Conventions of Pseudocode

What is the need for Pseudocode

Pseudocode is an important part of designing an algorithm, it helps the programmer in planning the solution to the problem as well as the reader in understanding the approach to the problem. Pseudocode is an intermediate state between algorithm and program that plays supports the transition of the algorithm into the program.

Introduction to Pseudocode

How to write Pseudocode?

Before writing the pseudocode of any algorithm the following points must be kept in mind.

- Organize the sequence of tasks and write the pseudocode accordingly.
- At first, establishes the main goal or the aim. **Example:**

> *This program will print first **N** numbers of Fibonacci series.*

- Use standard programming structures such as **if-else**, **for**, **while**, and **cases** the way we use them in programming. Indent the statements if-else, for, while loops as they are indented in a program, it helps to comprehend the decision control and execution mechanism. It also improves readability to a great extent. **Example:**

```
IF "1"
    print response
        "I AM CASE 1"
```

```
IF "2"
    print response
        "I AM CASE 2"
```

- Don't write the pseudocode in a programming language. It is necessary that the pseudocode is simple and easy to understand even for a layman or client, minimizing the use of technical terms.

Introduction to Pseudocode

Good vs Bad ways of writing Pseudocode:

Pseudocode Example

Good

```
FOR each index of string
    IF string(index) is equal to hash THEN
        RETURN "Hash found"
    END IF
END FOR
```

Bad

```
FOR i = 0 to 20
    IF string[i] = '#'
        RETURN "Hash found"
    END IF
END FOR
```

```
FOR each index of string
    if string[i] = '#'
        return "Hash found"
    END if
END FOR
```

Introduction to Pseudocode

Difference between Algorithm and Pseudocode

Algorithm	Pseudocode
An Algorithm is used to provide a solution to a particular problem in form of a well-defined step-based form.	A Pseudocode is a step-by-step description of an algorithm in code-like structure using plain English text.
An algorithm only uses simple English words	Pseudocode also uses reserved keywords like if-else, for, while, etc.
These are a sequence of steps of a solution to a problem	These are fake codes as the word pseudo means fake, using code like structure and plain English text
There are no rules to writing algorithms	There are certain rules for writing pseudocode
Algorithms can be considered pseudocode	Pseudocode cannot be considered an algorithm
It is difficult to understand and interpret	It is easy to understand and interpret

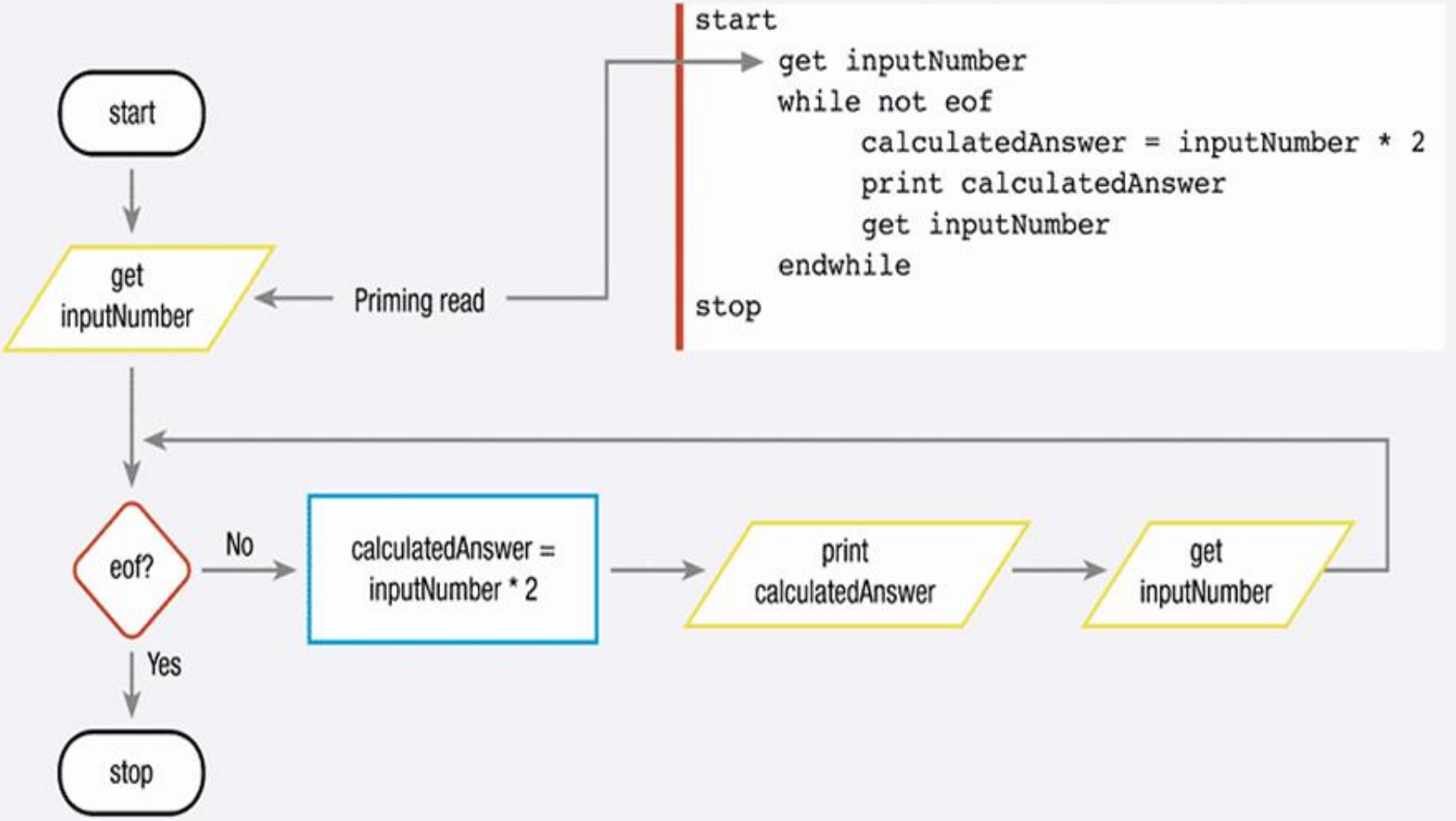
Flowcharts vs Pseudocode: When to Use Each

Difference between Flowchart and Pseudocode

Flowchart	Pseudocode
A Flowchart is pictorial representation of flow of an algorithm.	A Pseudocode is a step-by-step description of an algorithm in code like structure using plain English text.
A Flowchart uses standard symbols for input, output decisions and start stop statements. Only uses different shapes like box, circle and arrow.	Pseudocode uses reserved keywords like if-else, for, while, etc.
This is a way of visually representing data, these are nothing but the graphical representation of the algorithm for a better understanding of the code	These are fake codes as the word pseudo means fake, using code like structure but plain English text instead of programming language
Flowcharts are good for documentation	Pseudocode is better suited for the purpose of understanding

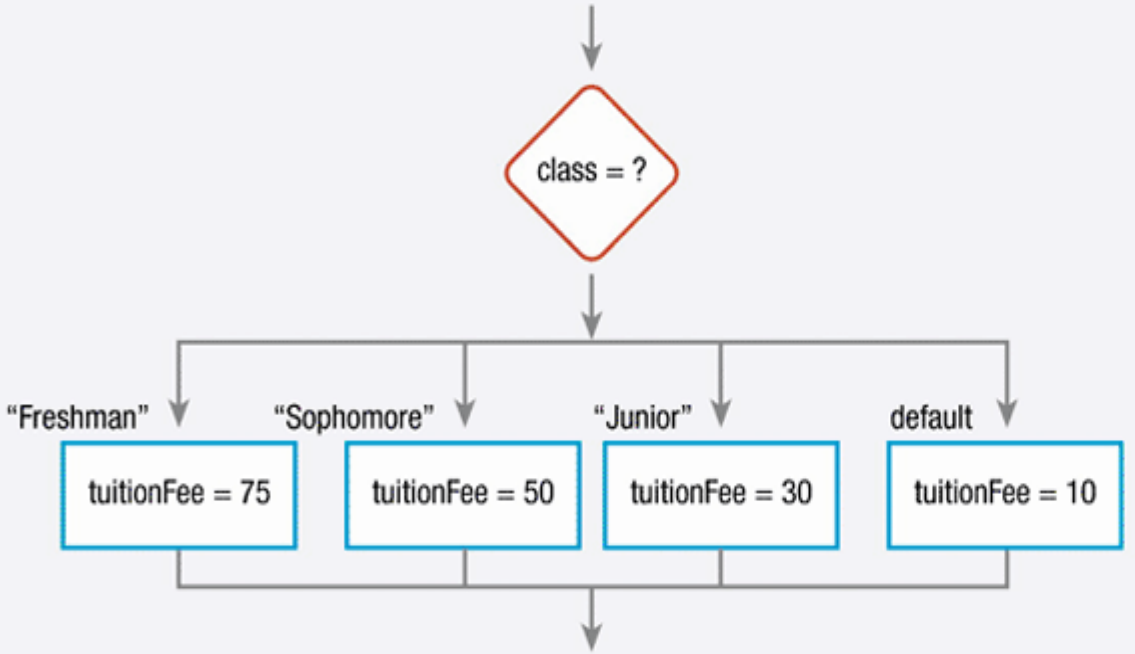
Flowcharts vs Pseudocode: When to Use Each

FIGURE 2-13: FUNCTIONAL, STRUCTURED FLOWCHART AND PSEUDOCODE FOR THE NUMBER-DOUBLING PROBLEM



Flowcharts vs Pseudocode: When to Use Each

FIGURE 2-32: FLOWCHART AND PSEUDOCODE OF CASE STRUCTURE




```
case based on class
      case "Freshman"
            tuitionFee = 75
      case "Sophomore"
            tuitionFee = 50
      case "Junior"
            tuitionFee = 30
      default
            tuitionFee = 10
endcase
```

Conclusion

Software development is complex and usually involves many parties working together. Therefore, planning out a project before beginning to program is essential for success.

QUIZ #2

Applying Program Logic Design Tools

- Flowcharting
- Pseudo Coding