KEMENTERIAN PENDIDIKAN TINGGI

POLITEKNIK MALAYSIA
BALIK PULAU

BRIGHTEN UP: PROBLEM SOLVING & PROGRAM DESIGN

Written by:
Siti Norsyahirah Binti Mohd Nor
Siti Hajar Binti Mohamadon
Shorayha A/P Eh Chong

Jabatan Teknologi Maklumat & Komunikasi
Politeknik Balik Pulau

Published by:
Politeknik Balik Pulau
Pinang Nirai, Mukim E
11000 Balik Pulau, Pulau Pinang
Malaysia
Tel: 604-8689000, Faks: 604-8692061
Emel: polibalikpulau@pbu.edu.my

BRIGHTEN UP: PROBLEM SOLVING & PROGRAM DESIGN

eISBN 978-967-2765-15-8

Content Editor
Syafiza Binti Ab Wahab

Language Editor
Shobhanambigha A/P Sivaguru

Authors
Siti Norsyahirah Binti Mohd Nor
Siti Hajar Binti Mohamadon
Shorayha A/P Eh Chong

Cataloguing-in-Publication Data

Perpustakaan Negara Malaysia

A catalogue record for this book is available from the National Library of Malaysia

eISBN 978-967-2765-15-8

Acknowledgement

We would like to extend our heartfelt thanks to everyone who contributed to the creation of this Brighten Up: Problem Solving and Program Design e-book.

First and foremost, we are grateful to our family and friends for their unwavering support and encouragement throughout this journey. Your belief in us been a constant source of motivation.

A special thank you to team members, whose insights and feedbacks were invaluable in shaping this work. Your expertise and perspective helped refine the ideas and bring them to life.

We also appreciate the contributions of Politeknik Balik Pulau, whose resources and support made this project possible.

Finally, thank you to our readers. Your interest and enthusiasm inspire us to continue sharing our thoughts and ideas.

Introduction

Welcome to Brighten Up: Problem Solving and Program Design, where we embark on a journey to enhance our problem-solving skills.

This book is aimed at aspiring students who is taking Problem Solving and Program Design course in honing their analytical skills. This guide provides a comprehensive framework for understanding and applying essential concepts in problem-solving and program design towards writing a basic programming code.

Throughout this e-book, we will explore the key principles of breaking down problems into manageable components including algorithm, pseudocode and flowchart. Each chapter is designed to build upon the last, offering practical examples, exercises, and insights drawn from real-world experiences.

Our goal is to empower readers with the tools and mindset needed to approach problems with confidence and creativity. As you progress through the pages, we encourage you to engage actively with the material, experiment with the concepts, and reflect on your own experiences.

Thank you for joining us on this exciting journey. Let's dive into the art and science of problem solving and program design!

Summary

Brighten Up: Problem Solving and Program Design will expose the user with a knowledge of programming language, key principles of breaking down problems into manageable components including algorithm, pseudocode and flowchart and idea toward writing programming code in C++.

Chapter 1: The readers will be introduced to the history of programming language with terms that frequently used in programming development. From this foundation, this will guide users to get some idea on the programming application used in real life

Chapter 2: This chapter will expose the readers to activities involved in all phases of the programming life cycle focusing on logical thinking to solve the problem using the 3 main methods (algorithm, pseudocode, flowchart .

Chapter 3: This chapter on Fundamentals of programming language is one of the important topic that will help readers with the readiness to write the programming codes.

Chapter 4: This chapter will bring the readers to start with their first writing of programming codes using C++ programming language.

Table of Contents

CHAPTER 1

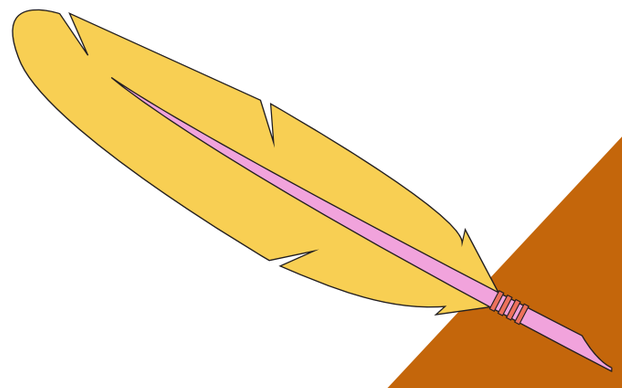
INTRODUCTION TO PROGRAMMING LANGUAGE

LEARNING OUTCOMES

1.1 Describe the programming language

1.2 Describe fundamentals of programming languages

1.1 DESCRIBE THE PROGRAMMING LANGUAGE



History of Programming Language

Before1940
The languages were Codes

The 1940s
Plankalkül , ENIAC coding system

The 1950s - 1960s
FORTRAN, LISP, COBOL

1967 – 1978
Logo, B, Pascal, Forth, C, Smalltalk, Prolog, ML, Scheme, SQL

The 1980s
C++, Objecyive-C, Ada,Common Lisp, Eiffel, Erlang, Perl, Tcl, FL

The 1990s : Internet age
Haskell, Python, Visual Basic, Ruby, Lua, CLOS, Java, Delphi, JavaScript, PHP, Rebol, D

Current trend
C#, Visual Basic . NET, F#, Scala, Factor, Windows Poer shell, Clojure, Groovy, Go

1.1 DESCRIBE THE PROGRAMMING LANGUAGE

Different Types of Programming Language

Machine language

Consist of binary code

```
01010100 01101000
00100000 01101001
01110010 01110101
01100011 10010010
11001111 11001100
```

```
section  .text
global   _start

_start:

    mov  edx, len
    mov  ecx, msg
    mov  ebx, 1
    mov  eax, 4
    int  0x80

    mov  eax, 1
    int  0x80

section  .data

msg  db  "Hello World!"
len  equ $ - msg
```

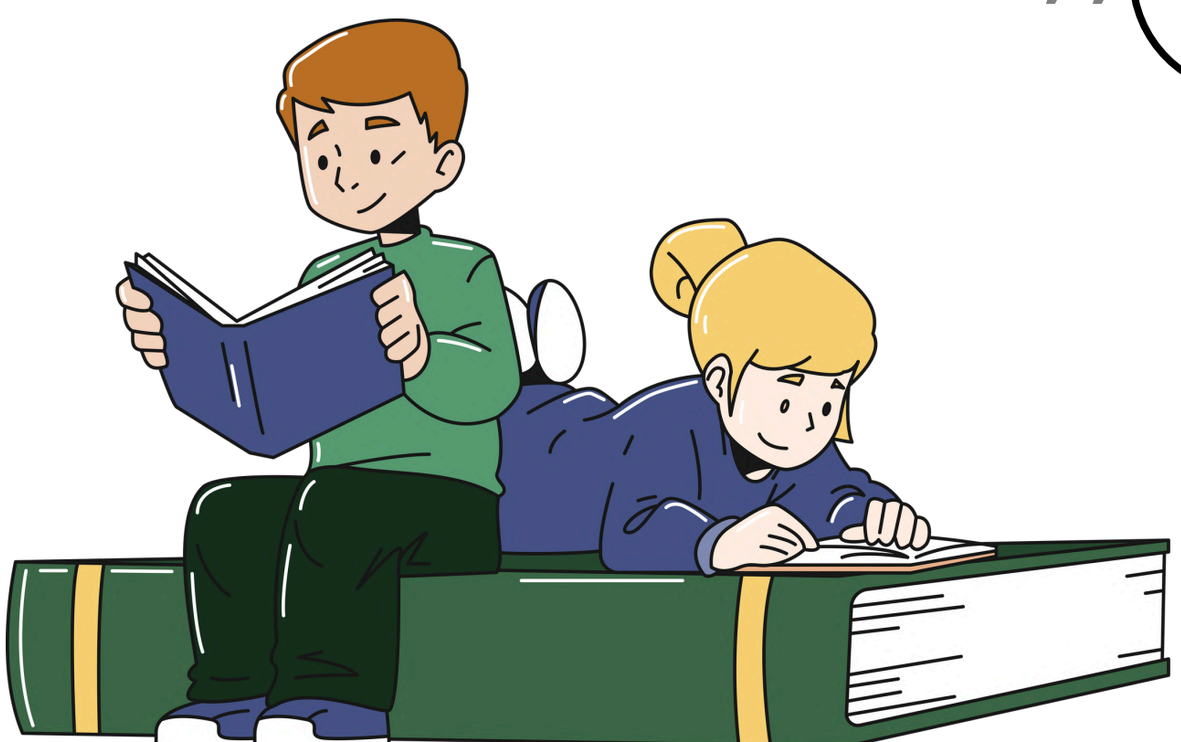
Assembly language

Using mnemonic code

High level language

Closer to human language and uses commands

```
#include <iostream>
using namespace std;

int main()
{
    int number1;
    int number2;
    int answer;

    cout<<"Insert two numbers below: "<<"\n";
    cin>> number1;
    cin>> number2;

    answer = number1 + number2;

    cout<<"Adition of the 2 numbers is "<<answer;

    return 0;
}
```



1.1 DESCRIBE THE PROGRAMMING LANGUAGE

Generations of Programming Language

1st Generation (Machine Language)

- Only languages understood by computers
- Called machine language
- Consist entirely of binary numbers
- The lowest-level programming language
- Machine-dependent

2nd Generation (Assembly Language)

- Low-level programming language for computers
- Implements a symbolic representation of the machine codes
- Use mnemonics code
- Machine-dependent

3rd Generation (High Level Language)

- High-level programming language
- A refinement of a second-generation programming language
- The languages more programmer-friendly
- Most 3GLs support structured programming
- Not machine-dependent

4th Generation (High Level Language)

- A programming environment designed with a specific purpose
- To reduce programming effort, time and cost of software development
- Closer to human languages
- Most 4GLs are used to access databases

5th Generation (High Level Language)

- Based on artificial intelligence
- Designed to make the computer solve a given problem without the programmer
- Programming language based on solving problems using constraints given to the program

Click video icon to open the video:

1.1 DESCRIBE THE PROGRAMMING LANGUAGE

Advantages and Disadvantages for Generations of Programming Language

1st Generation (Machine Language)

Advantages
- Most efficient in term of speed and storage

Disadvantages
- Instructions and syntax not standardize
- Impossible for human to understand
- Machine-dependent

2nd Generation (Assembly Language)

Advantages
- Program run faster
- Efficient use of the memory

Disadvantages
- Difficult to understand
- Errors can be made easily
- Machine-dependent

3rd Generation (High Level Language)

Advantages
- Machine independent - can be written and executed in any machine
- The languages are more programmer-friendly
- Easy to develop and maintained

4th Generation (High Level Language)

Advantages
- Machine independent - can be written and executed in any machine
- Save programming effort, time and cost of software development
- Simpler language and easy to code

5th Generation (High Level Language)

Advantages
- Machine independent - can be written and executed in any machine
- Easy to be developed and maintained

EXERCISE 1.1

1. The images uniquely represent each programming language generation that you have learned. Identify and match the programming language generation correctly.

Expression	Features

`mov edx, len`
`mov ecx, msg`
`mov ebx, 1`
`mov eax, 4`
`int 0x80`
 	
`mov eax, 1`
`int 0x80`
	A collection of binary numbers understood by a computer.

01010100 01101000 00100000 01101001 01110010 01110101 01100011 10010010 11001111 11001100	Translation to machine language is performed using compiler or interpreter.

`cout<<"Insert two number`
`cin>> number1;`
`cin>> number2;`
 	
`answer = number1 + numbe`
 	
`cout<<"Adition of the 2`
	Also known as symbolic language because it uses mnemonic code.

2. List the characteristics for each generations of programming languages below:

Generation of Programming Language	Characteristic
First generation	
Second generation	
Third generation	
Fourth generation	
Fifth generation	

1.2 FUNDAMENTALS OF PROGRAMMING LANGUAGE

Definition of Programmer, Program and Programming

Programmer

- Someone who writes computer software
- A specialist in one area of computer programming or to a generalist who writes code for many kinds of software.

Program

An organized list of instructions that causes the computer to behave in a predetermined manner when it is executed.

Programming

The process of designing, writing, testing, debugging, and maintaining the source code of computer programs.

1.2 FUNDAMENTALS OF PROGRAMMING LANGUAGE

Programming Language Translators

Assembler

A program that translates programs from assembly language to machine language.

```
section  .text
global   _start

start:

mov  edx, len
mov  ecx, msg
mov  ebx, 1
mov  eax, 4
int  0x80

mov  eax, 1
int  0x80
```

Assembly language

```
01010100 01101000
00100000 01101001
01110010 01110101
01100011 10010010
11001111 11001100
```

Machine language

Compiler

- A computer program that transforms source code written in a programming language into another computer language .
- Translate the whole program at one time .

```
include <iostream>
sing namespace std;

main()

nt number1;
int number2;
int answer;

cout<<"Insert two numbers
cin>> number1;
cin>> number2;

answer = number1 + number2

cout<<"Adition of the 2 nu
```

C++ language

```
01010100 01101000
00100000 01101001
01110010 01110101
01100011 10010010
11001111 11001100
```

Machine language

Interpreter

- Translate high level language directly to machine language.
- Translate one line at a time and executes each line after translated.
- Used to translate BASIC and SQL language.

```
ith value_per_order as

select
    order_id
    ,sum(quantity * pr
    from {{raw.e_comme
    group by order_id

lect
(total_revenue_per_orde
avg(total_revenue_per_ord
min(total_revenue_per_ord
max(total_revenue_per_ord
rom value_per_order
```

SQL language

```
01010100 01101000
00100000 01101001
01110010 01110101
01100011 10010010
11001111 11001100
```

Machine language

1.2 FUNDAMENTALS OF PROGRAMMING LANGUAGE

Relate Programming Languages Application in Real Life

Education

Example:
- SPMP
- Cidos e-Learning
- W3school

Business

Example:
- Lazada
- Amazon
- Shopee

Finance

Example:
- CIMB clicks
- Maybank2U
- TouchnGo e-wallet

Entertainment

Example:
- Youtube
- Spotify
- JOOX Music

EXERCISE 1.2

1. Define the following terms:

Terms	Definition
Program	
Programmer	
Programming	

2. Differentiate between compiler and interpreter.

Compiler	Interpreter

2. List examples of suitable application which implements programming in real life application in each field.

Field	Applications
Agricultural	• • •
Medical	• • •
Education	• • •

ARE YOU READY FOR QUIZ??

SCAN THE QR CODE BELOW TO START THE QUIZ:

CHAPTER 2

PROBLEM SOLVING METHOD

LEARNING OUTCOMES

2.1 Demonstrate Programming Life Cycle

2.2 Identify Problem Solving Concept

2.3 Describe the different types and patterns in algorithms to solve problem.

2.1 DEMONSTRATE PROGRAMMING LIFE CYCLE

Programming Life Cycle?

A framework or discipline, which uses certain techniques needed in computer programming development.

2.1 DEMONSTRATE PROGRAMMING LIFE CYCLE

7 STEPS INVOLVED IN PROGRAMMING LIFE CYCLE

Specify the Problem
Defining the issue, outlining the requirements to solve it, and stating the desired outcome, while also considering any constraints or assumptions.

Analyze the Problem
To describe in detail a solution to a problem and information needed by identifying the required inputs, processes, outputs, any constraints, and relevant formulas.

Program Design
It is a framework or flow that shows the steps in problem solving using methods like algorithms, flowcharts, and pseudocode.

Program Coding
The process of implementing an algorithm by writing a computer program and creating interface using a programming language or application development tool.

Testing and Verifying
Testing is the process of executing a program to demonstrate its correctness, while program verification is the process of ensuring that a program meets user-requirement.

Maintain and Update
An activity to modify the system to meet the current requirement and the process of changing a system after it has been applied to maintain its ability.

Documentation
A written or graphical report of the steps taken during the development of a program.

2.1 DEMONSTRATE PROGRAMMING LIFE CYCLE

THREE TYPES OF ERROR

Syntax Error

- Occurs when the rules of programming language are not applied.
- Correction is done during the program coding.
- The bug can be traced during the compilation.
- Also known as compile-time error
- Must be corrected before executing and testing the program.

Logic Error

- Cannot be traced by compiler.
- Corrected during the problem solving process.
- Example : output for average is 4, but when it runs, the output is 2.

Runtime Error

- An error that occurs while a program is being executed, after it has successfully passed the compilation stage.
- Example :
  - Division by zero
  - Array out of bounds
  - Invalid user input
  - Using more memory than what the system can provide

2.1 DEMONSTRATE PROGRAMMING LIFE CYCLE

FOUR TYPES OF MAINTENANCE

1. **Adaptive Maintenance**

> Modifications to properly interface with changing environments
> -> new hardware, OS, devices

2. **Perfective Maintenance**

> Implementing new system requirements after the system is successful

3. **Preventive Maintenance**

> Avoid possible failures, enhance the overall functioning of your computer, and increase the lives of various components.

4. **Corrective Maintenance**

> If predictive and preventive maintenance is unable to solve a problem that has occurred in your computer, we need to go for corrective maintenance.

EXERCISE 2.1

Cik Suria asks you to prepare the planning for system development. You have to study the phases and activities involves in program life cycle. Define the phase definition below to match the description with its appropriate Phase by using line in the space provided.

No	Description
1	Run the program several times using different sets of data
2	Process of modifying a software system or component after delivery to correct faults, improve performance or adapt to a changed environment
3	Identify the input, process and output of the problem
4	Determine the required form and units in which the results should be displayed
5	To build a code base on the algorithm being design
6	Develop a list of steps called an algorithm to solve the problem
7	A written or graphical report of the steps taken during the development of a program

No	Phase
	Implementation
	Documentation
	Specify the problem
	Design the algorithm
	Verification and Testing
	Analyzing
	Maintenance and update

2.2 IDENTIFY PROBLEM SOLVING CONCEPT

Problem Solving ?

The process of creating a set of instructions or algorithms to address a specific computational issue or challenge.

Problem Solving Steps

- Identify the input and output
- Identify other data and constants required to solve the problem
- Identify what needs to be computed
- Write an algorithm

2.2 IDENTIFY PROBLEM SOLVING CONCEPT

DEFINING PROBLEM USING IPO CHART

Input

Data or information received by the system.

Process

Operations performed on the input data.

Output

The final result or information produced.

"The cashier will scan the price of all items and enter the amount of money tendered. The price of each fruit and the amount of money tendered will be used as **input.**"

After scanning, the system will total up the price and calculate the balance. This is call as **process**.

Next, the price of all items and balance will be displayed on the receipt as **output**.

"Click video image here to get some idea."

EXERCISE 2.2

Problem 1

Below is the set of work flows of Auto Pay Machine for Parking Ticket. Tick the appropriate box by categorizing them into input, process and output based on the given instructions.

Set of Instructions:	**Input**	**Process**	**Output**
Insert parking ticket			
The payment will be calculated based on the duration			
Parking fee will be displayed			
Pay by notes or coins			
Calculate the change for payment balance			
Processed parking ticket			
Verify amount of money inserted			

EXERCISE 2.2

Problem 2

Below is the set of work flows of Course Registration for student. Tick the appropriate box by categorizing them into input, process and output based on the given instructions.

Set of Instructions:	**Input**	**Process**	**Output**
Save selected class for course registration in the database			
Course registration slip display at the screen			
Select course for registration			
Printed course registration slip			
Enter your ID and password			
System will validate your ID and password			
Select the class that offer the course selected before			
Save selected course registration in the database			

2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

DESCRIBE LOGIC

Method of reasoning that involves a series of statements, each of which must be true if the statement before it is true

RELATE LOGIC WITH ALGORITHM

Logic facilitates the decision-making process in algorithms, allowing them to react differently under varying circumstances

2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

ALGORITHM

An algorithm is a list of steps to be executed with the right order in which these steps should be executed.

An algorithm can be represented using pseudocode or flowchart.

3 DIFFERENT PATTERNS (CONSTRUCT) IN ALGORITHM

Sequential

Line-by-line execution

Conditional

Execute depending on the condition either true or false

Selection

Iterational

Allow the execution of block of statement multiple time until a special condition is met

Repetition

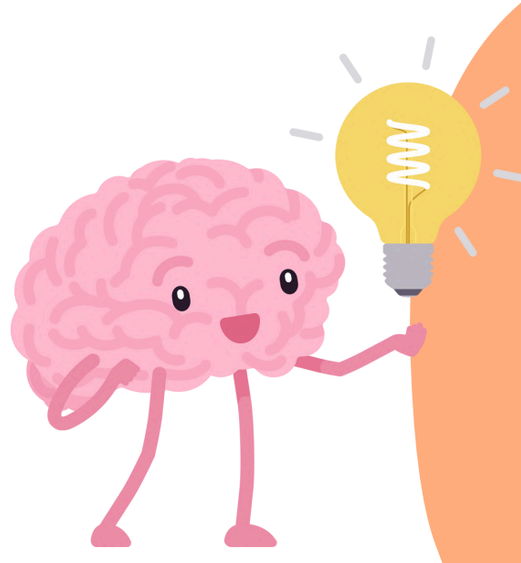
2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

CONCEPT OF ALGORITHM

- Must may have input(s) and must have output(s)
- Should not be ambiguous (there should not be different interpretations to it)
- Must be general (can be used for different inputs)
- Must be correct and it must solve the problem for which it is designed
- Must execute and terminate in a finite amount of time
- Must be efficient enough so that it can solve the intended problem using the resource currently available on the computer

2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

WRITE AN ALGORITHM



Aiman went to the book store and bought a few of pen and pencils. Price for each pen is RM1.20 and price for pencil is RM0.75 each. Calculate the total amount Aiman need to pay.

IPO CHART

Input	no_of_pen, no_of_pencils
Process	total_charged = (no_of_pen*1.20) + (no_of_pencil*0.75)
Output	total_charges

ALGORITHM

1.Input no_of_pen, no_of_pencils

2.Calculate total charge by using this formula:

total_charged = (no_of_pen*1.20) + (no_of_pencil*0.75)

3.Print total_charged

EXERCISE 2.3A

You have to write an **algorithm** for the problem given.

PROBLEM	PROBLEM DESIGN (ALGORITHM)
Problem 1: You are tasked with calculating the average of three test scores for a student.	
Problem 2: You need to convert a temperature from Celsius to Fahrenheit. The formula for conversion is: Fahrenheit=(Celsius×9/5)+32	
Problem 3: You need to calculate the total time a runner takes to complete a race, given the time for each lap. The runner completes 4 laps, and the time for each lap is recorded in minutes.	
Problem 4: You are tasked with calculating the total cost of items purchased in a grocery store, including a fixed 10% tax. • 3 apples (RM1.20 each) • 5 oranges (RM1.80 each) • 2kg Cucumber (RM12.00 per kg)	

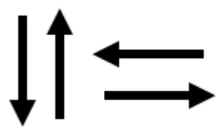
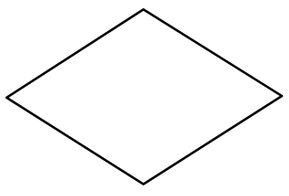
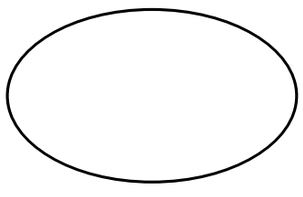
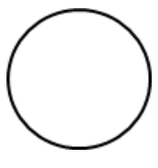
2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

FLOWCHART

A graphical representation of data, information and workflow using certain symbols that are connected to flow lines to describe the instructions done in problem solving.

It shows the flow of the process from the start to the end of the problem solving.

STANDARD SYMBOLS USED IN FLOWCHART

Symbol	Explaination
 *Flow Lines*	• Indicate the direction of data flow. • Used to connect a block to another block.
 *Process*	• Indicates operations/process involved.
 *Input / Output*	• Receive/read value • Display value
 *Decision*	• Execute decision based on condition. • Test is performed and the program flow continues, based on the result
 *Start / End Flow Lines*	• Indicates the beginning and end of a flowchart.
 *On-page Connector Flow Lines*	• Show the continuing flowchart in the same page.

2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

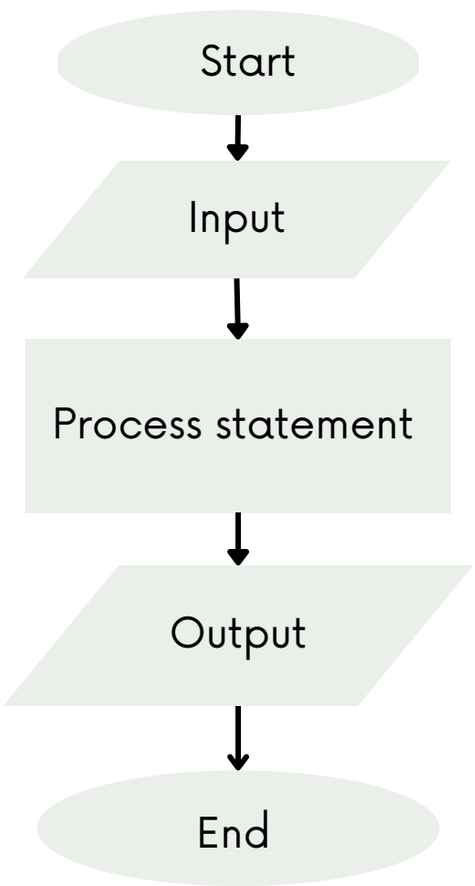
DISTINGUISH BETWEEN FLOWCHART AND PSEUDOCODE

	FLOWCHART	PSEUDOCODE
Layout	Graphical structure	Structure for the code of the program
Benefits	For smaller concepts and problems	More efficient for larger programming languages
Structure	Symbols and shapes	Linear text-based structure
Depth	Detail can cause confusion	More flexibility with detail

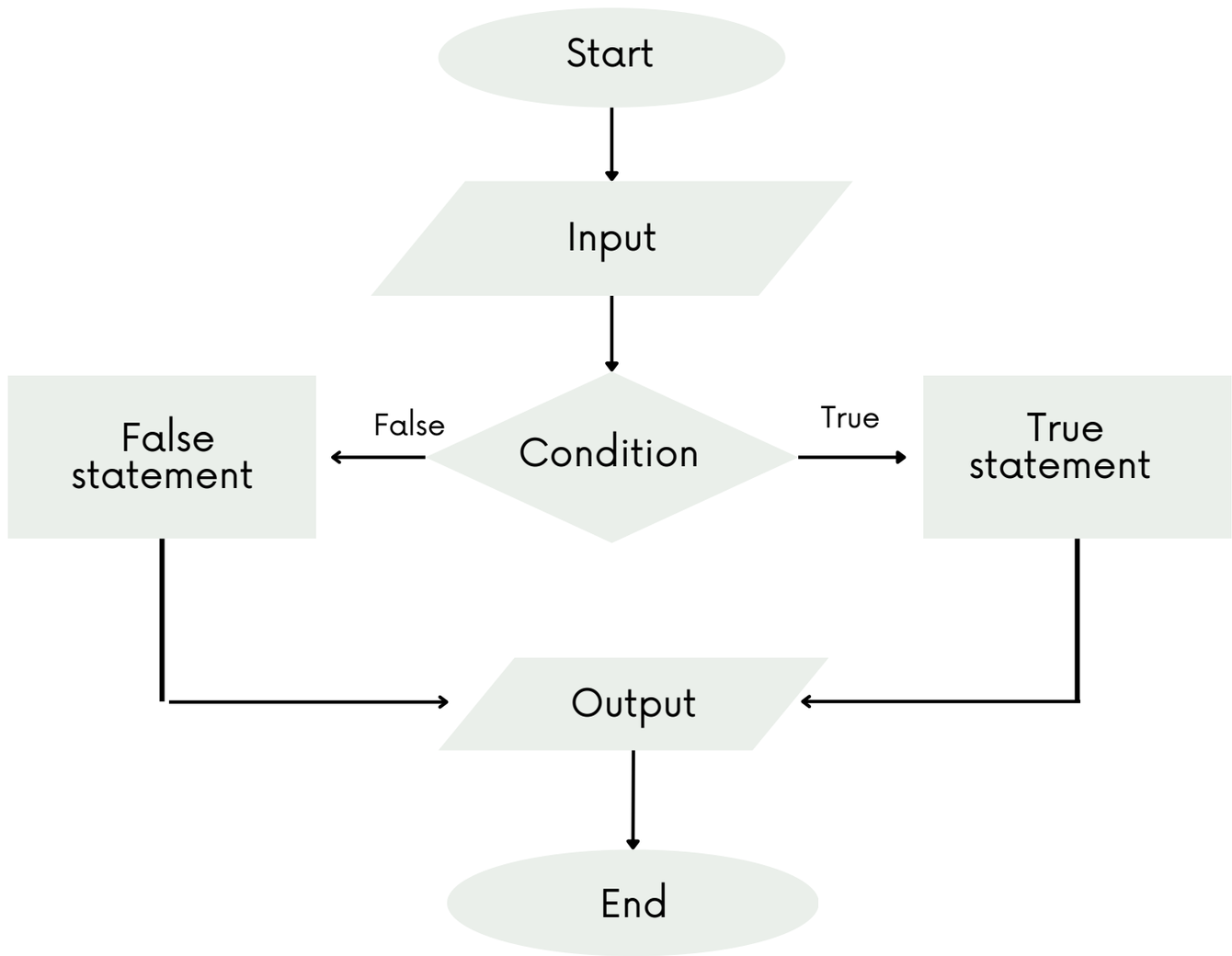
2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

REPRESENT THE ALGORITHM PATTERN IN A FLOWCHART

SEQUENCE



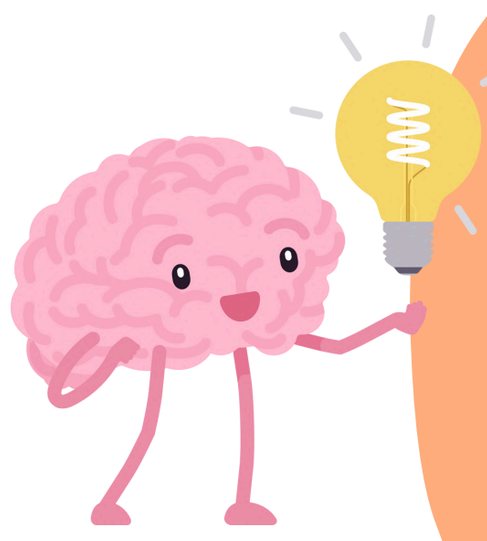
CONDITIONAL



ITERATIONAL

2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

ILLUSTRATE FLOWCHART



Aiman went to the book store and bought a few of pen and pencils. Price for each pen is RM1.20 and price for pencil is RM0.75 each. Calculate the total amount Aiman need to pay.

ALGORITHM

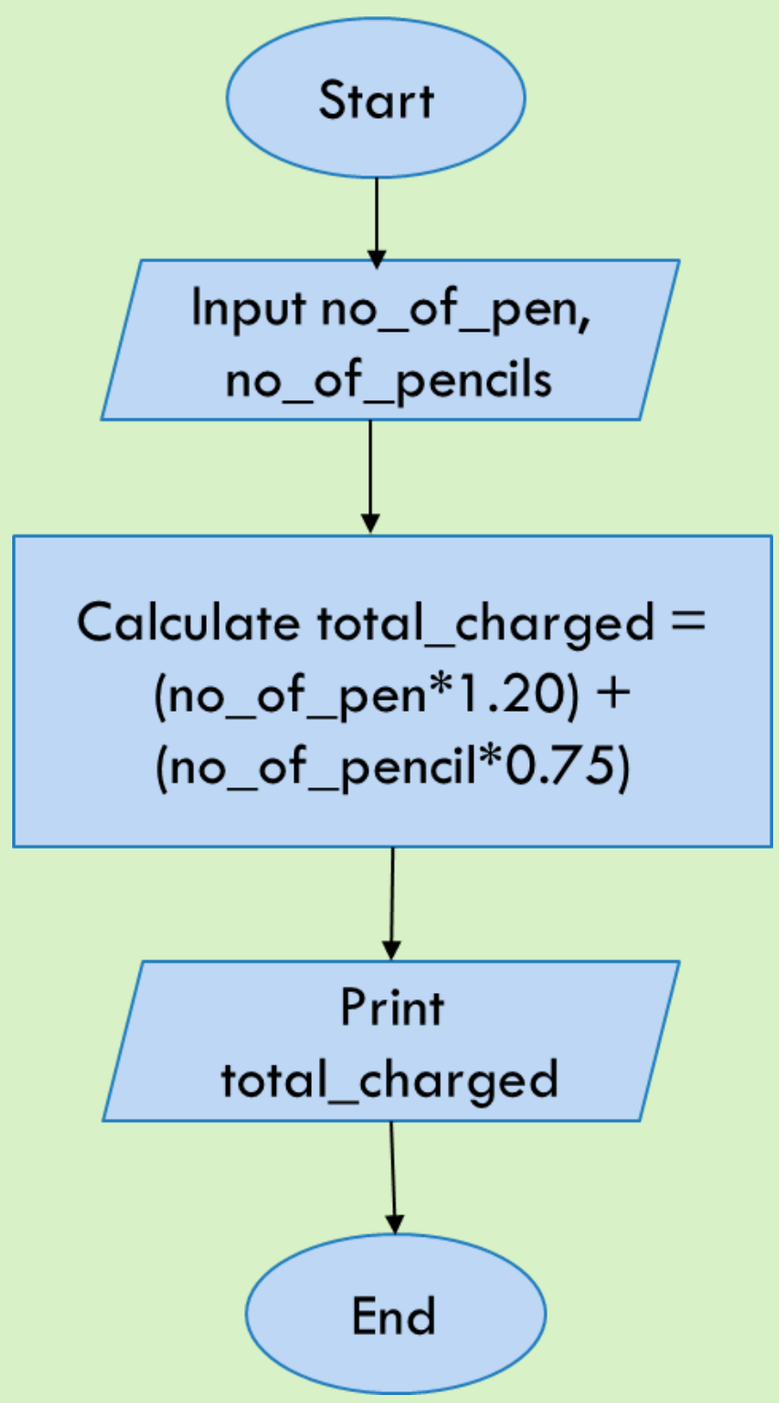
1.Input no_of_pen, no_of_pencils

2.Calculate total charge by using this formula:

total_charged = (no_of_pen*1.20) + (no_of_pencil*0.75)

3.Print total_charged

FLOWCHART



EXERCISE 2.3B

You have to draw a **flowchart** for the problem given.

PROBLEM	PROBLEM DESIGN (FLOWCHART)
Problem 1: You are tasked with calculating the average of three test scores for a student.	
Problem 2: You need to convert a temperature from Celsius to Fahrenheit. The formula for conversion is: Fahrenheit=(Celsius×9/5)+32	
Problem 3: You need to calculate the total time a runner takes to complete a race, given the time for each lap. The runner completes 4 laps, and the time for each lap is recorded in minutes.	
Problem 4: You are tasked with calculating the total cost of items purchased in a grocery store, including a fixed 10% tax. • 3 apples (RM1.20 each) • 5 oranges (RM1.80 each) • 2kg Cucumber (RM12.00 per kg)	

2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

PSEUDOCODE

Steps in problem solving that is written half in programming code and half in human language.

PURPOSE OF USING PSEUDOCODE

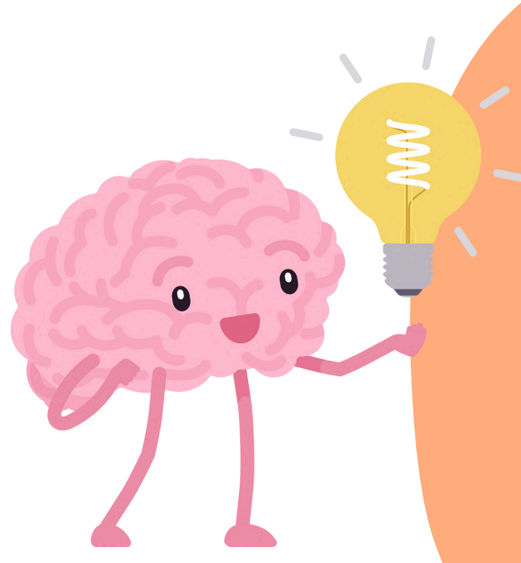
- Easier for people to understand than conventional programming language code.
- Easier to identify mistakes in a program or function's logic.
- Can be quickly and easily converted into an actual programming language as it is similar to a programming language.

GUIDELINES AND BEST PRACTICES WRITING PSEUDOCODE

- Easier for people to understand than conventional programming language code.
- Easier to identify mistakes in a program or function's logic.
- Can be quickly and easily converted into an actual programming language as it is similar to a programming language.

2.3 DESCRIBE THE DIFFERENT TYPES AND PATTERN IN ALGORITHM TO SOLVE PROBLEM

ILLUSTRATE PSEUDOCODE



Aiman went to the book store and bought a few of pen and pencils. Price for each pen is RM1.20 and price for pencil is RM0.75 each. Calculate the total amount Aiman need to pay.

ALGORITHM

1.Input no_of_pen, no_of_pencils
2.Calculate total charge by using this formula:
total_charged = (no_of_pen*1.20) + (no_of_pencil*0.75)
3.Print total_charged

PSEUDOCODE

```
Start
    Input no_of_pen, no_of_pencils
    total_charged = (no_of_pen*1.20)
                    + (no_of_pencil*0.75)
    Print total_charged
End
```

EXERCISE 2.3C

You have to write a **pseudocode** for the problem given.

PROBLEM	PROBLEM DESIGN (PSEUDOCODE)
Problem 1: You are tasked with calculating the average of three test scores for a student.	
Problem 2: You need to convert a temperature from Celsius to Fahrenheit. The formula for conversion is: Fahrenheit=(Celsius×9/5)+32	
Problem 3: You need to calculate the total time a runner takes to complete a race, given the time for each lap. The runner completes 4 laps, and the time for each lap is recorded in minutes.	
Problem 4: You are tasked with calculating the total cost of items purchased in a grocery store, including a fixed 10% tax. • 3 apples (RM1.20 each) • 5 oranges (RM1.80 each) • 2kg Cucumber (RM12.00 per kg)	

SCAN THE QR CODE BELOW TO START THE QUIZ:

CHAPTER 3

FUNDAMENTAL OF PROGRAMMING LANGUAGE

LEARNING OUTCOMES

3.1 Explain Data and Identifier

3.2 Show a Problem Solving Skills using Operators in a Program

3.3 Describe control structures in problem solving

3.1 UNDERSTAND DATA & IDENTIFIER

DATA

- Input for the program
- Pre-processing fact

DATA TYPE

- classification of the type of data
- 2 main data types:
  1) Numeric 2) Non-numeric

IDENTIFIER

The given name to multiple elements in programming such as constant, variables and function.

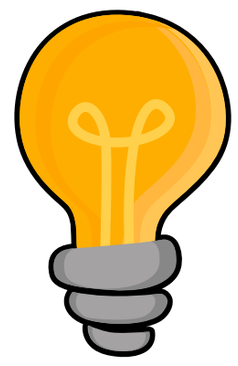
CONSTANT

- expression with a fixed value
- value are not changeable during entire programming execution.

VARIABLE

- memory location
- keep data value
- value are changeable during entire programming execution

3.1 UNDERSTAND DATA & IDENTIFIER



DATA TYPE

Numeric

Only numbers data

Non-numeric

Can be characters, string, numbers, symbols

Example:

"Home", 'P', "123t_"

Integer

Integers only

Example 1:

-3, 20, 19

Floating

Can be integers or floating (decimal) numbers

Example 2:

-7, 9, 29.89

3.1 UNDERSTAND DATA & IDENTIFIER

IDENTIFIER

Declaring Constant

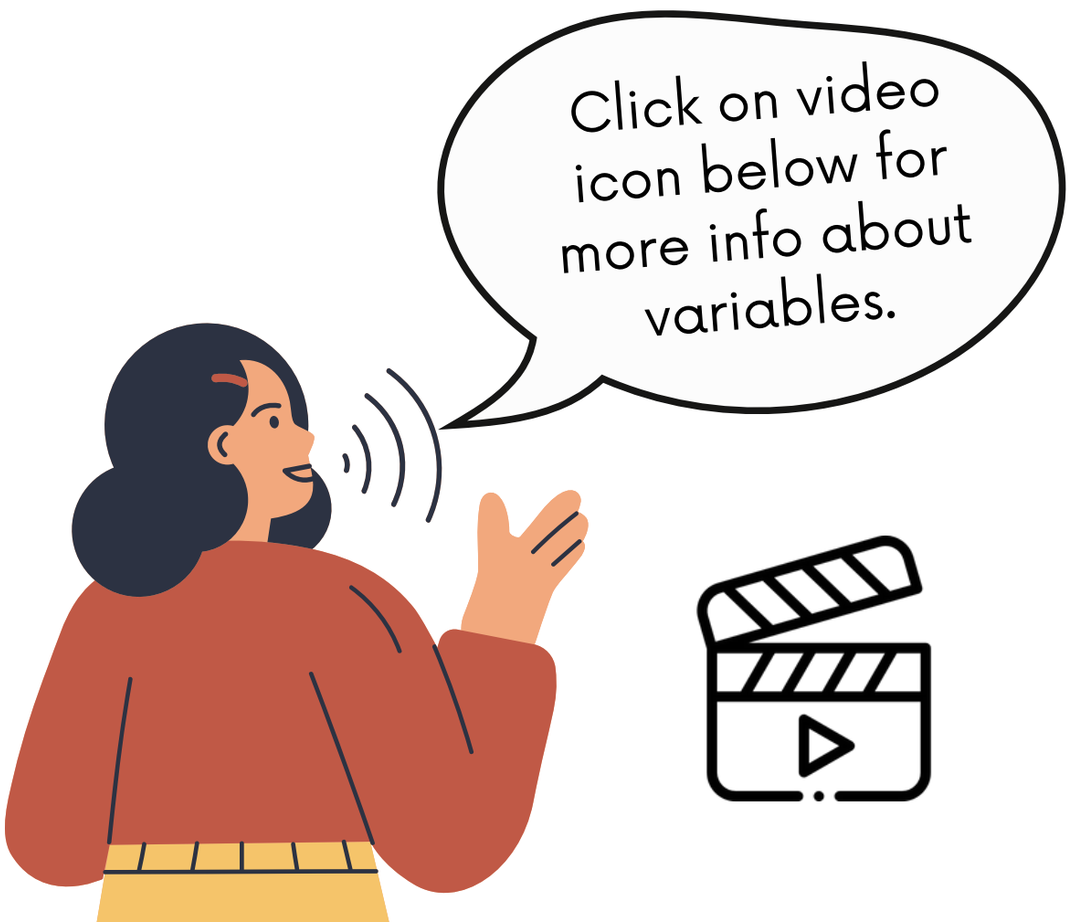
- const int days_in_year = 365;
- const float FlourPerKg= 17.5;
- const float pi = 3.142;

Writing Variable

- Combination of:
  1. Character (A - Z), (a - z)
  2. Digit (0 - 9)
  3. Underscore (_)
- Cannot start with digit
- Do not have reserve words
- No blank space
- No limit of character usage but the system will identify first 32 character
- Case sensitive

Declaring variable:

- int age;
- float area;
- char grade;



EXERCISE 3.1

1. Identify the data type for the expression below.

Expression	Data type
Number123	
2024	
-0.0000005	
0.741	
X	
&	
grade	

2. Identify the variables and data type based on the problems.

Problems	Variable	Data type
Calculate area of a circle using a given radius.		
Age can be calculated based on the birth year and the current year.		
You are creating a simple program to identify if a student passed or fail for their marks in the mid semester test.		

EXERCISE 3.1

3. Based on given reserved words in table below, answer TRUE or FALSE.

Reserved Word	TRUE/FALSE
break	
number	
return	
braces	
void	
while	

4. Which of the following variables, identify valid or invalid? If invalid, give the reason.

Identifier name	Valid/ Invalid	Reason
ic@number		
TotalIncome		
#validhashtag		
100student		
my_polytechnic		
student name		

EXERCISE 3.1

5. State "TRUE" or "FALSE" for each of the variables declaration below

Variable declaration	TRUE/FALSE
int class;	
float priceS;	
int &studentName;	
int Student IC;	
int _ICnumber;	
char 4saleItem;	
int register;	

6. Based on the following variables declaration, identify valid or invalid. If invalid, give the reason.

Variable declaration	Valid/ Invalid	Reason
int enum;		
char switch;		
char S;		
float we,ght;		
string student_name;		
float ^age^;		

3.2 SOLVE PROBLEM USING OPERATORS IN A PROGRAM

OPERATOR

A symbol to represent particular computer operation.

OPERAND

Objects that are operated.

TYPE OF OPERATORS:

1. ASSIGNMENT OPERATOR
2. ARITHMETIC OPERATOR
3. RELATIONAL OPERATOR
4. LOGICAL OPERATOR
5. INCREMENT & DECREMENT OPERATOR

3.2 SOLVE PROBLEM USING OPERATORS IN A PROGRAM

1. ASSIGNMENT OPERATOR

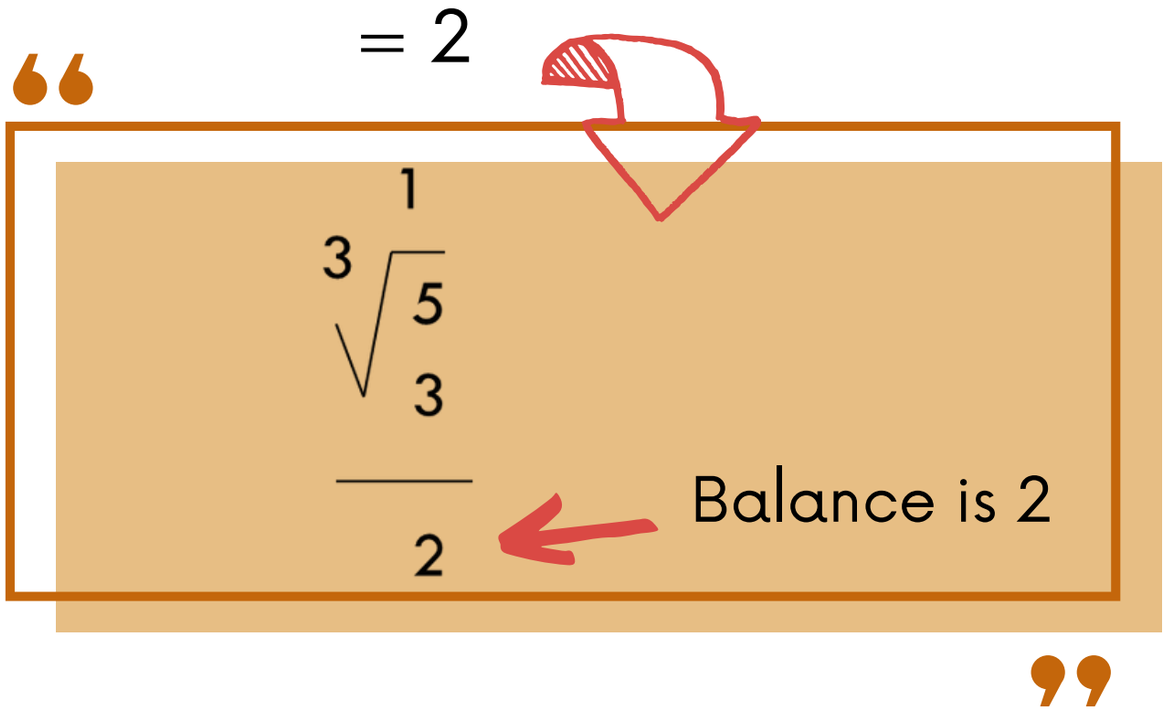
Assignment operator can be combined into a single operator with some other operators to perform a combination of two operations in one single statement.

Syntax	Operator	Expression
x + = y	Addition	**x = x + y**
x - = y	Substraction	**x = x - y**
x * = y	Multiplication	**x = x * y**
x / = y	Division	**x = x / y**
x % = y	Modulus	**x = x % y**

Example 1:

x = 5 % 3

= 2



3.2 SOLVE PROBLEM USING OPERATORS IN A PROGRAM

2. ARITHMETIC OPERATOR

- It has 5 basic operator in programming language
- The execution of process should follow the operator priority.

Arithmetic	Operator	Expression
*	Multiplication	**x = x + y**
/	Division	**x = x - y**
%	Modulus	**x = x * y**
+	Addition	**x = x / y**
-	Substraction	**x = x % y**

highest

lowest

bracket ( ) has the highest priority among the operators.

Example 1:

$$x = 5 + 5 * (6 - 2)$$
$$= 5 + 5 * 4$$
$$= 5 + (5*4)$$
$$= 5 + 20$$
$$x = 25$$

EXERCISE 3.2A

1. Find the value for the following expression. Show your steps for solution.

Expression	Solution steps
a. (1 + 2) + 6 * 4 / 2 – 1	
b. (6 + 4) / 2 – 4	
c. 6 * 3 / 6 + 9	
d. 5 + 3 * (7 – 2)	

2. Given integer variables x = 10, y = 7 and z = 2.
   Determine the value of each of the arithmetic expression.

Variable declaration	Solution steps
a. x + 2y – z	
b. x / z – (x * x + y)	
c. 5 (x + y + z) – x / z	
d. (x * y) % z	

3. Given the following declaration and initial assignments p = 6, q = 2 and r = 3. Determine the value of each of the following assignments.

Variable declaration	Solution steps
a. p + 3q - r	
b. p / r + (p * p + r)	
c. r (p +q) (p –q)	
d. 12/ p + (p + q – r) – 6 / r	

3.2 SOLVE PROBLEM USING OPERATORS IN A PROGRAM

3. RELATIONAL OPERATOR

- Used to compare 2 operators
- Only can compare between same data type.
- The result is either true or false.

Symbol	Description	Expression
>	Greater than	**10 > 5**
<	Less than	**3 < 10**
>=	Greater than or equal	**10 >= 9**
<=	Less than or equal	**5 <= 8**
==	Equal with	**9 == 9**
!=	Not equal with	**10 != 9**

Example 1:

a. 8>=5, result is True
b. 3 < 2, result is False
c. 8 != 7, result is True
d. 10 > 3, result is True

EXERCISE 3.2B

1. Determine whether the expression below is TRUE or FALSE if X = 2, Y = 6 and Fish = FALSE.

Expression	TRUE/FALSE
(X == Y) || (Y <= 3)	
(X >= 0) && (X <= Y)	
Fish && (X > Y)	
Fish && (! Fish)	
! Fish || ! (! Fish)	

2. Given the value a = 0, b = 6 and c = 3. Write TRUE for the true expression and FALSE for the false expression. Show all the steps clearly.

Expression	TRUE/FALSE
a. ! (a == 0) && (b != 2)	
b. ((a == c) && (b == c))|| (a < 1)	
c. ! (!(c !=4) || !(4 ==2)|| !(a==0))	

3.2 SOLVE PROBLEM USING OPERATORS IN A PROGRAM

4. LOGICAL OPERATOR

- Used to test some operations.
- Consist of 3 logical operators.
- Result is based on truth table

Symbol	Description	Expression
&&	AND	10 > 5
||	OR	3 < 10
!	NOT	10 != 9

TRUTH TABLE

X	Y	X && Y	X || Y
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

TRUE = 1
FALSE = 0

EXERCISE 3.2C

1. Determine whether the expression below is TRUE or FALSE if X = 2, Y = 6 and Fish = FALSE.

Expression	TRUE/FALSE
a. (X == Y) || (Y <= 3)	
b. (X >= 0) && (X <= Y)	
c. Fish && (X > Y)	
d. Fish && (! Fish)	
e. ! Fish || ! (! Fish)	

2. Given the value a = 0, b = 6 and c = 3. Write TRUE for the true expression and FALSE for the false expression. Show all the steps clearly

Expression	TRUE/FALSE
a. ! (a == 0) && (b != 2)	
b. ((a == c) && (b == c))|| (a < 1)	
c. ! (!(c !=4) || !(4 ==2)|| !(a==0))	

3. Given the value a = 4, b = 8. Find the value of x by showing all the steps clearly.

Expression	Solution Steps
a. x = !(a==b) && (a>b)	
b. x = !(a==2) || (a<b)	
c. x = ((a + b > b) && (a == a)) || (b == a)	

3.2 SOLVE PROBLEM USING OPERATORS IN A PROGRAM

5. INCREMENT & DECREMENT OPERATOR

- Sometimes, we need to increment or decrement a single value in programming
- Valid for variables only (invalid for constant)

Operator	Description	Expression x = 5
++x	pre increment	**y = ++x, y = 6**
x++	post increment	**y = x++, y = 5**
--x	pre decrement	**y = --x, y = 4**
x--	post decrement	**y = x--, y = 5**

Example 1:

Pre-increment

Given **a = 3**
b = ++a

Output

a = 4
b = 4

Pre-decrement

Given **a = 3**
b = --a

Output

a = 2
b = 2

increase/decrease the **value** of a **first**, **then assign** the current value to b

Example 2:

Post-increment

Given **a = 3**
b = a++

Output

a = 4
b = 3

Post-decrement

Given **a = 3**
b = a--

Output

a = 2
b = 3

assign the current **value** to b **first**, **then increase /decrease** the value a

EXERCISE 3.2D

1. Determine whether the expression below is TRUE or FALSE.

Expression	TRUE/FALSE

a. x = x+1;
is the same as
x++;	

b. x = x-1; is the same as x--;	
c. int a = 5 a++ answer = a is now 6	
d. int a = 5 int c c = a++ answer = c is now 6	
e. int a = 7 ++a answer = a is now 7	
f. int a = 10 a-- answer = a is now 10	
g. int a = 5 --a answer = a is now 4	

3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

3 Types of Program Control Structures:

- Sequence
- Selection
- Repetition

Sequence

- Command set which is executed line by line
- Follows logic flow

Selection / Conditional

- The expression which result on **TRUE** or **FALSE**
- Consists 2 keywords:
  - if
  - else
- Categorized into 4 groups:
  - **If** statement
  - **If-else** statement
  - **If-else (nested)** statement
  - **Switch** statement

Repetition / Iterative

- Using loop structure.
- Allows a series of instructions executed repeatedly until the condition set is met
- 3 types of loop:
  - **While**
  - **Do .. While**
  - **For**

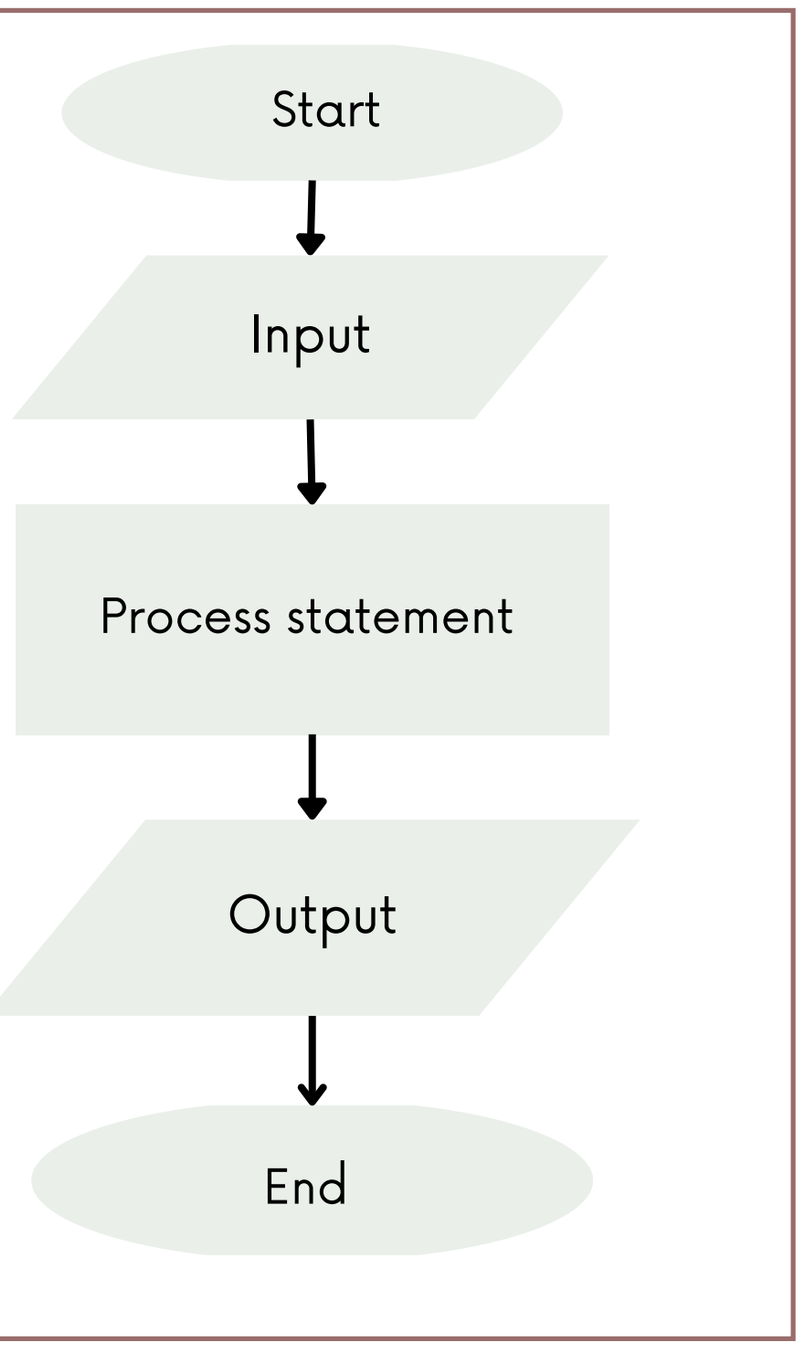
3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

1. SEQUENCE

Algorithm

1. Input data
2. Process statement
3. Display output

Flowchart



Pseudocode

```
Start
        Input data
        Process statement
        Display output
End
```

Example 1:

Write a program that help a lecturer to calculate the average of marks for 3 quizzes for a student.

Answer:

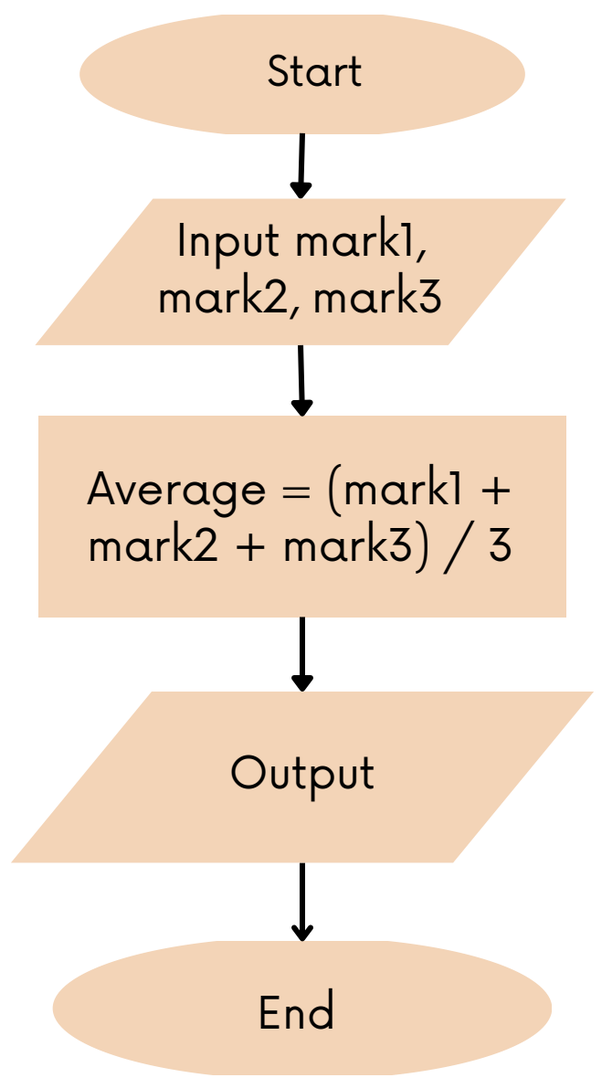
Algorithm

1. Input mark1, mark2, mark3
2. Calculate Average:
   Average = (mark1 + mark2 +mark3) / 3
3. Display Average

Pseudocode

```
Start
        Input mark1, mark2, mark3
        Average =  (mark1 + mark2 +mark3) / 3
        Display Average
End
```

Flowchart



EXERCISE 3.3A

1 Mr Wong is a mathematics teacher. He wants to teach his students about how to convert time in hour into the times in minute.
Help Mr Wong to solve his problem by construct an algorithm, a pseudocode and flowchart.

Answer
a. Algorithm
b. Pseudocode
b. Flowchart

EXERCISE 3.3A

2 You have been hired by Prima Hotel as a programmer. Your task is to develop a program that can count the total rate by multiply room rate with number of days, and add 10% room service charges. Construct a pseudocode and flowchart.

Answer
a. Pseudocode
b. Flowchart

3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

2. SELECTION/ CONDITIONAL

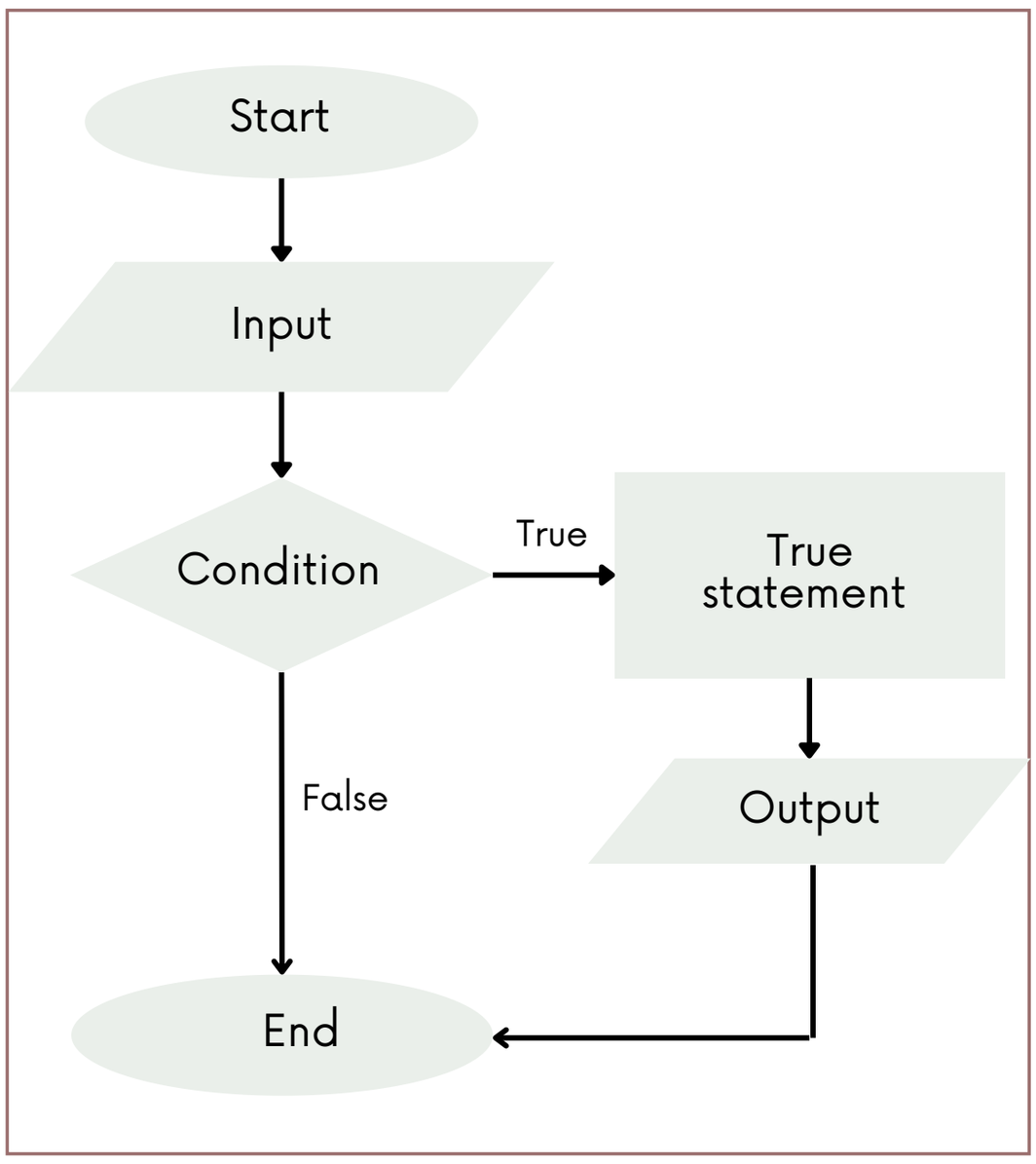
if...end If

The IF statement will perform an action if the condition is true and ignore the action if the condition is false

Algorithm

1. Input data
2. If condition
3. Compute True statement
4. End if
5. Display output

Flowchart



Pseudocode

```
Start
        Input data
          If condition
             True statement
          end if
        Display output
End
```

Example 2:

If speed > 110,
print penalty

Answer:

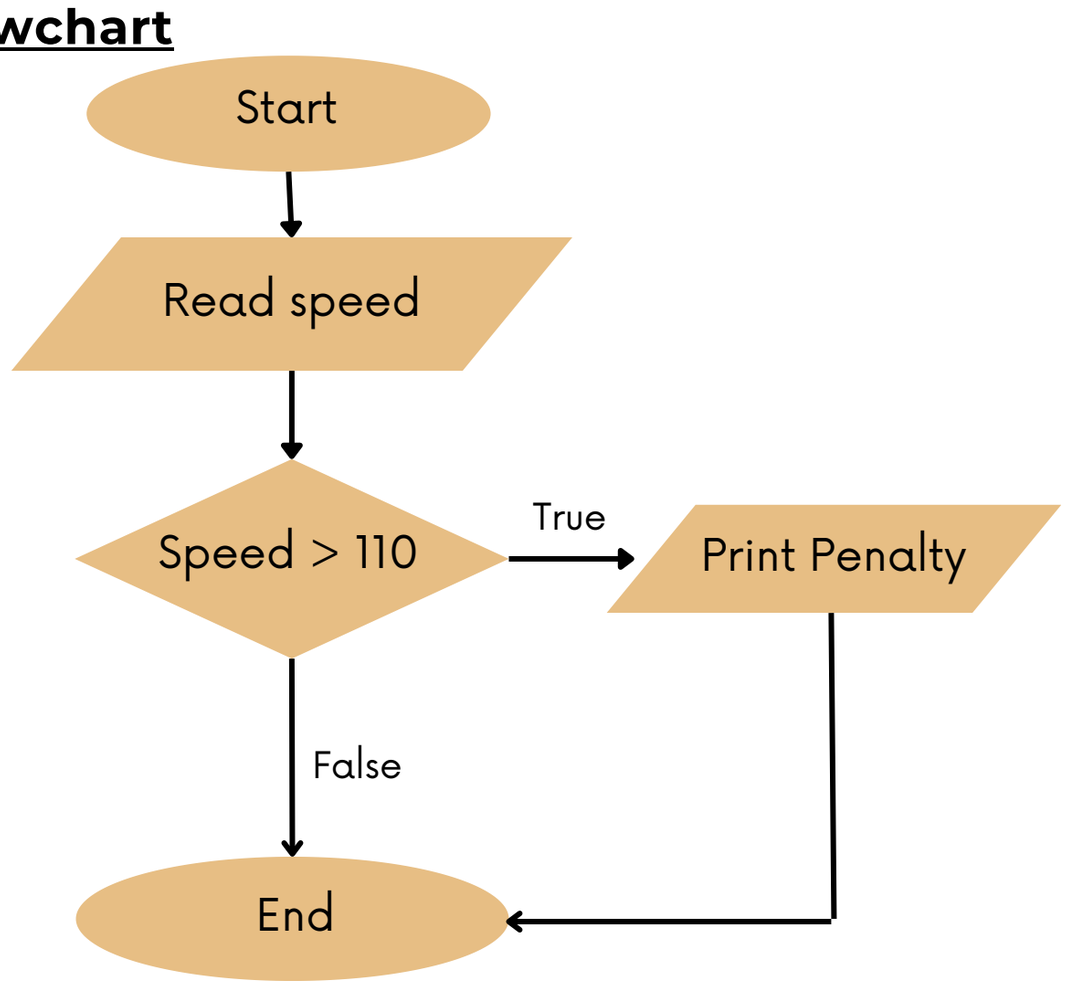
Algorithm

1. Read speed
2. If speed is greater than 110, Print Penalty
3. End If

Pseudocode

```
Start
        Read speed
        If speed > 110
        Print Penalty
        End If
End
```

Flowchart



EXERCISE 3.3B

1 Write a program that show the message "You have exceeded the speed limit" if the speed is exceed 110km/hour.

Answer
a. Pseudocode
b. Flowchart

3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

2. SELECTION/ CONDITIONAL

If... Else

The IF statement will perform an action if the condition is true and perform another action if the condition is false.

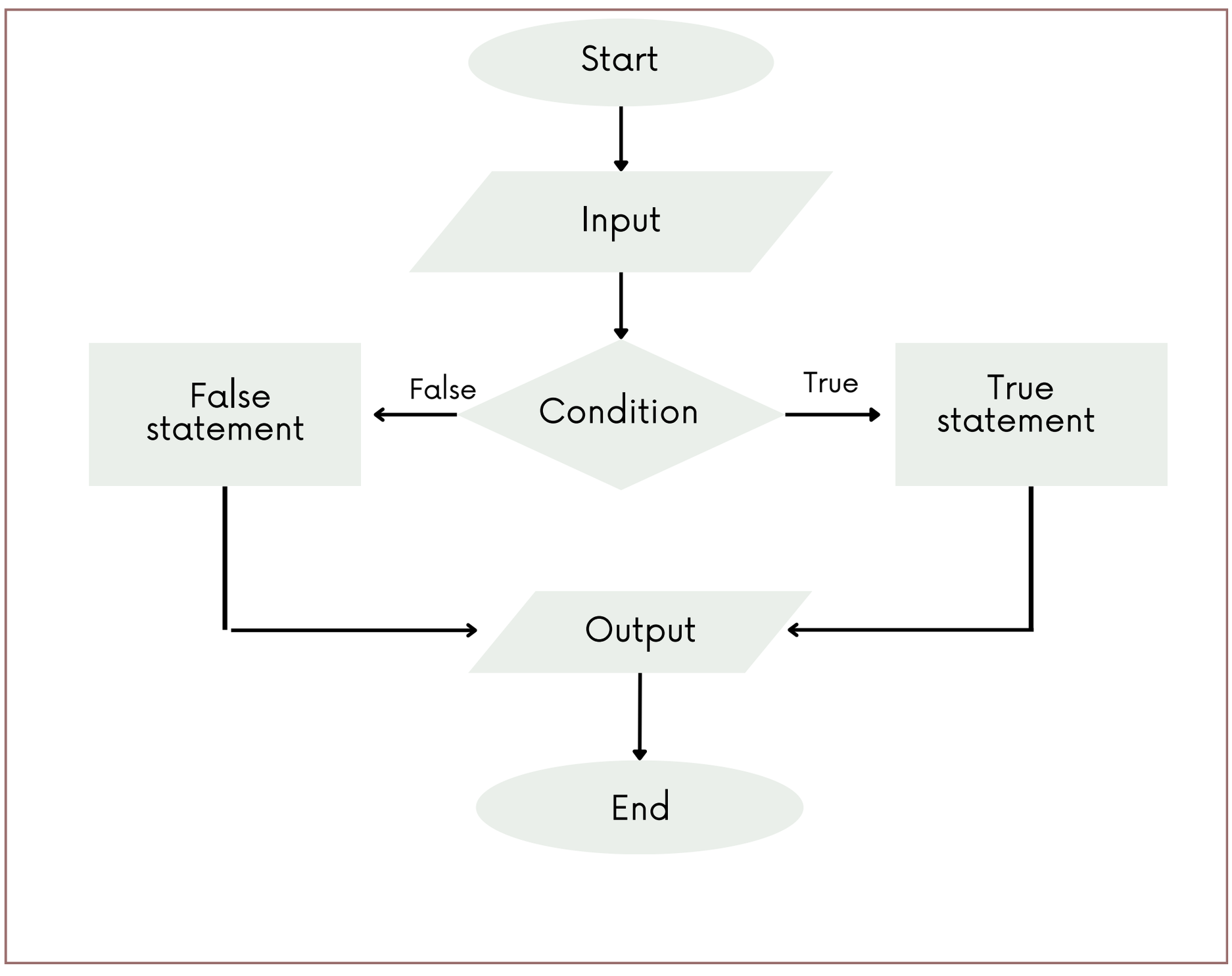
Algorithm

1. Input data
2. If condition
3. Compute True statement
4. Else compute False statement
5. End If

Pseudocode

```
Start
        Input data
          If condition
             True statement
                else False statement
          end if
End
```

Flowchart



3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

Example 3:

If speed > 110,
print "Penalty"
else print "No penalty"

Answer:

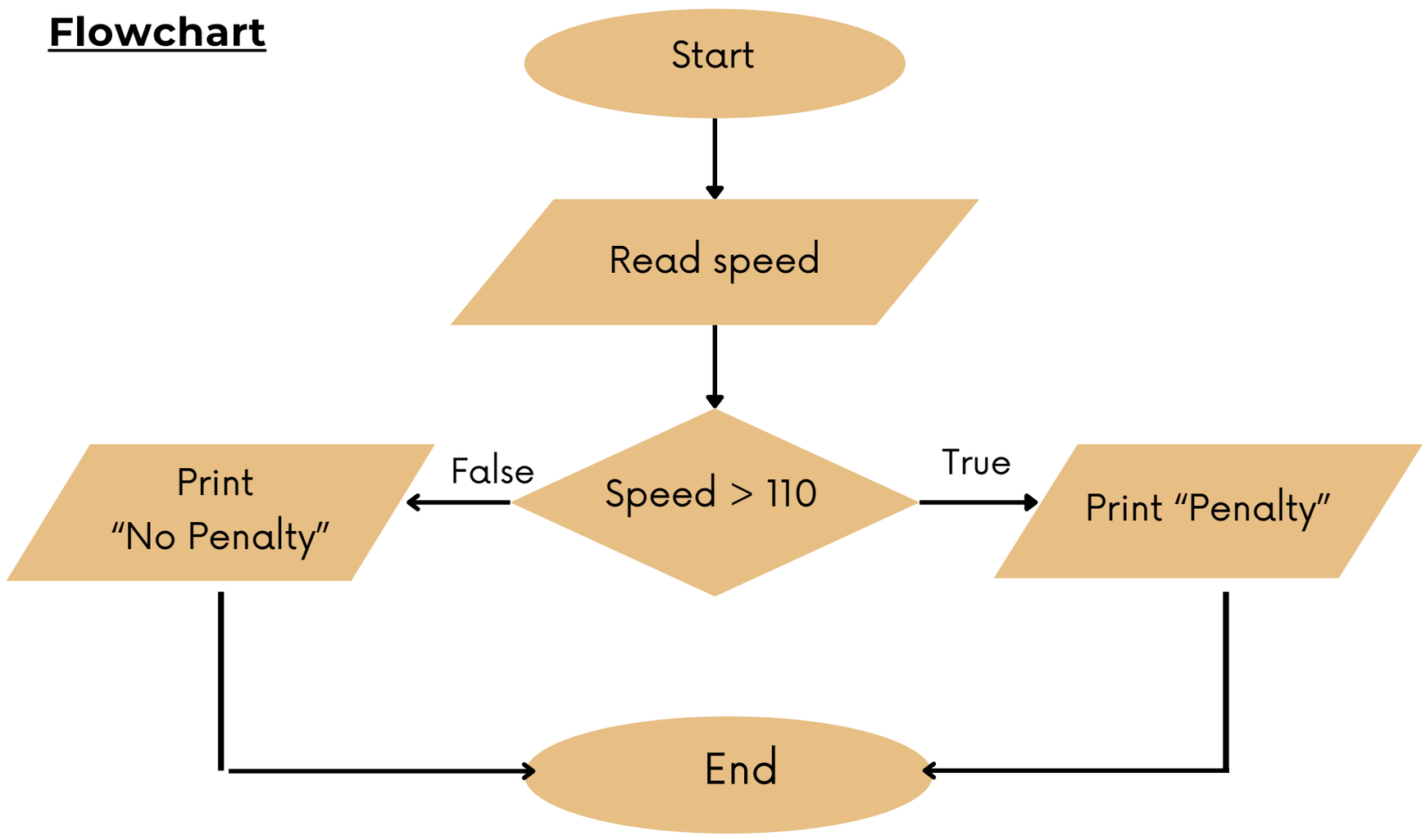
Algorithm

1. Read speed
2. If speed is greater than 110,
   Print "Penalty"
3. Else print "No penalty"
4. End If

Pseudocode

```
Start
    Read speed
      If speed > 110
          Print "Penalty"
           Else print "No Penalty"
      End If
End
```

Flowchart



EXERCISE 3.3C

1. Create a program that can help cashier to calculate the net price of curry mee bought by customers. If the total price purchased is more than RM50 in one receipt, customer will get 3% discount of the their bill. The bill should display total price of purchased curry mee and net price to be paid.

Answer
a. Pseudocode
b. Flowchart

3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

2. SELECTION/ CONDITIONAL

If... Else (nested)

IF-ELSE nested statement is a condition when we have an 'if' in another 'if' body. Use this control structure if we have many selection to be handle.

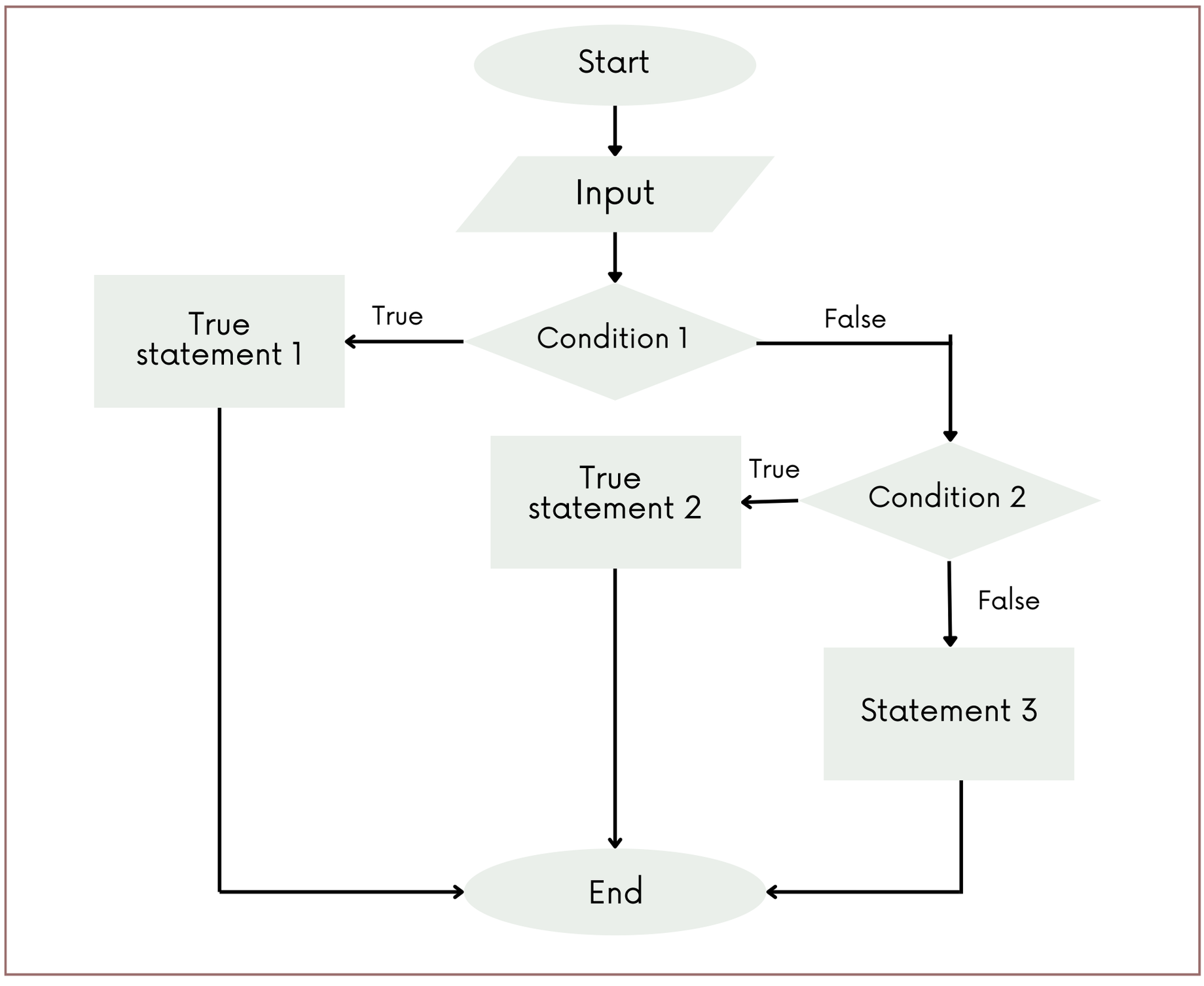
Algorithm

1. Input data
2. If condition 1
3. Compute True statement 1
4. Else If condition 2
5. Compute True statement 2
6. Else compute statement 3
7. End If

Pseudocode

```
Start
        Input data
          If condition 1
             True statement 1
                else If condition 2
                 True statement 2
                   else statement 3
          end if
End
```

Flowchart



3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

Example 4:

Create a program that can recognize and print the price based on user prompt code. If user enter a non-available value, the program will print "code not recognize".

Code	Price
1	RM2.00
2	RM4.00
3	RM6.00
4	RM8.00

Answer:

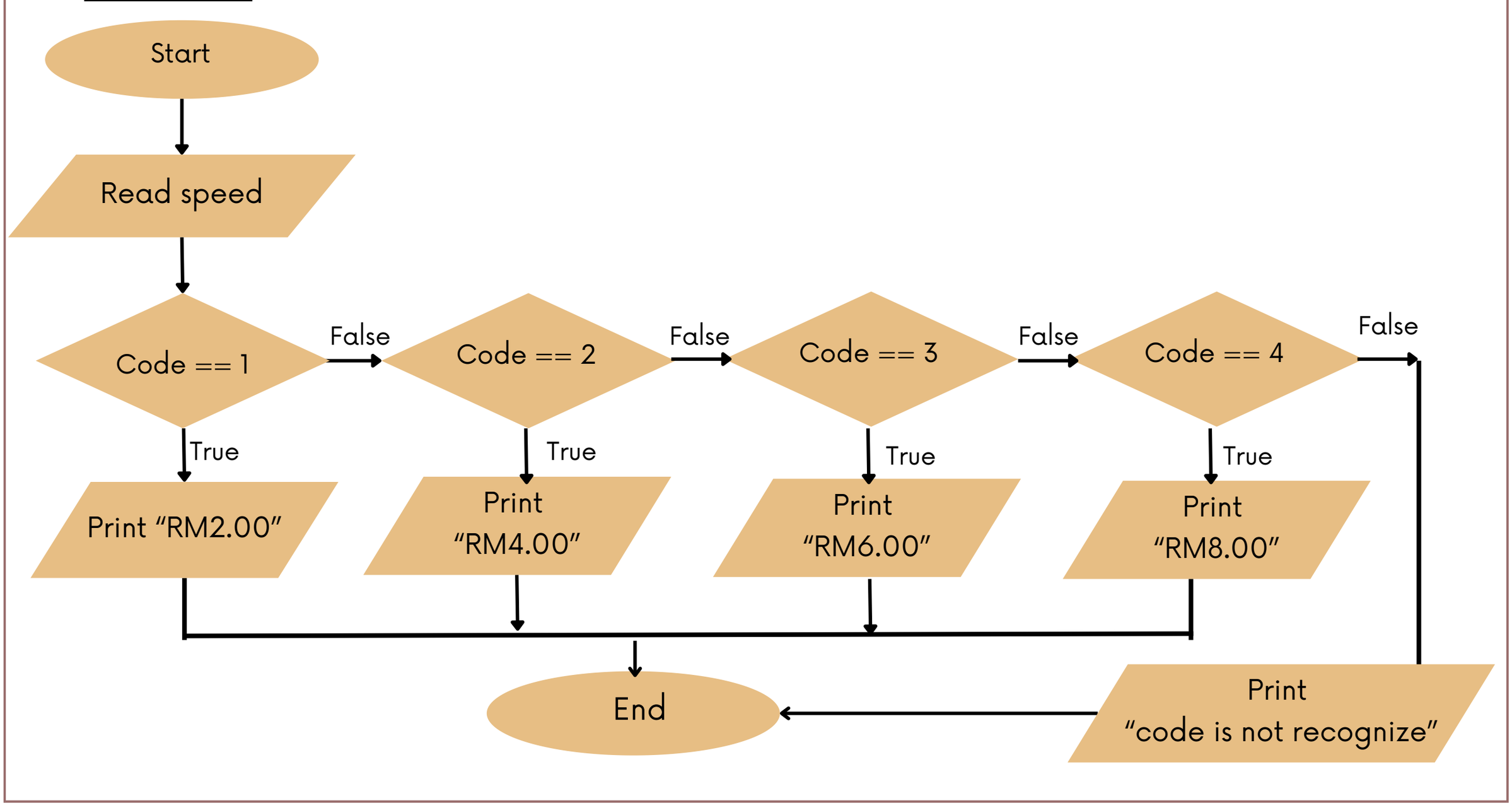
Algorithm

1. Read code
2. If code is equal 1, print RM2.00
3. Else if code is equal 2, print RM4.00
4. Else if code is equal 3, Print RM6.00
3. Else if code is equal 4, print RM8.00
4. Else print "Code is not recognize"
5. End If

Pseudocode

```
Start
    Read code
      If code == 1, print RM2.00
        Else if code == 2, print RM4.00
        Else if code == 3, print RM6.00
        Else if code == 4, print RM8.00
        Else print "Code is not recognize"
      End If
End
```

Flowchart



EXERCISE 3.3D

1 Program that receives the current temperature as input. If the temperature is 45 degrees or more, output a message telling the user to go swimming, otherwise, if the temperature is 30 degrees or more, output a message to go running, otherwise stay inside.

Answer
a. Pseudocode
b. Flowchart

3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

2. SELECTION/ CONDITIONAL

switch

Can **ONLY** use for 2 data type, **integer** and **character**.

Keyword: switch, break, default

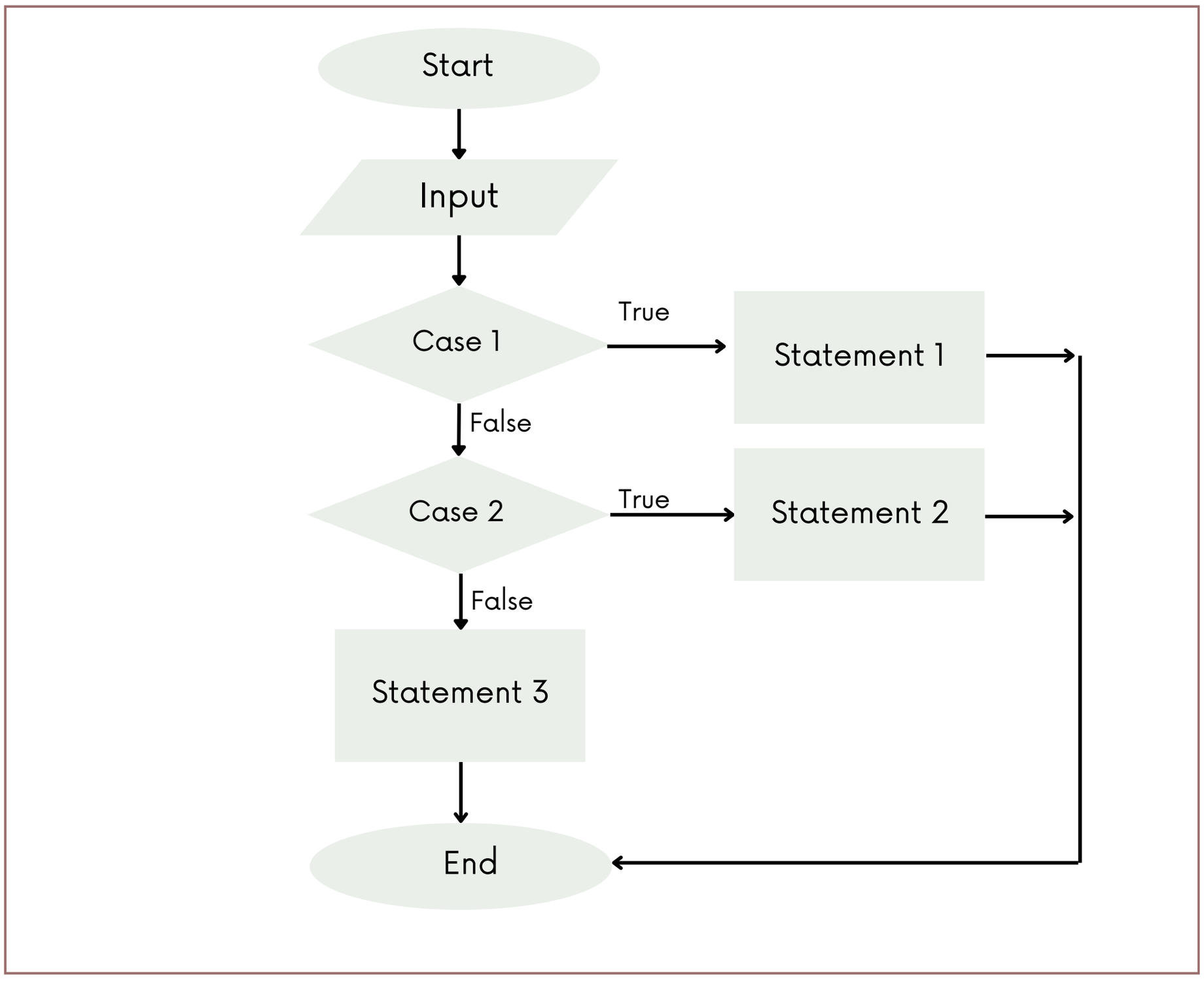
Algorithm

1. Input data
2. Switch (data)
3. Case 1
4. Statement 1, break
5. Case 2
6. Statement 2, break
7. Default statement 3
8. End switch

Pseudocode

```
Start
    Input data
      Switch (data)
        Case 1
            Statement 1, break;
        Case 2
            Statement 2, break;
        Default statement 3
      end switch
End
```

Flowchart



3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

Example 5: switch

Create a program that can recognize and print the coffee name based on user prompt code. If user enter a non-available code the program will print "code is not recognize".

Code	Coffee
1	Latte
2	Mocha
3	Machiato
4	Cappuccino

Answer:

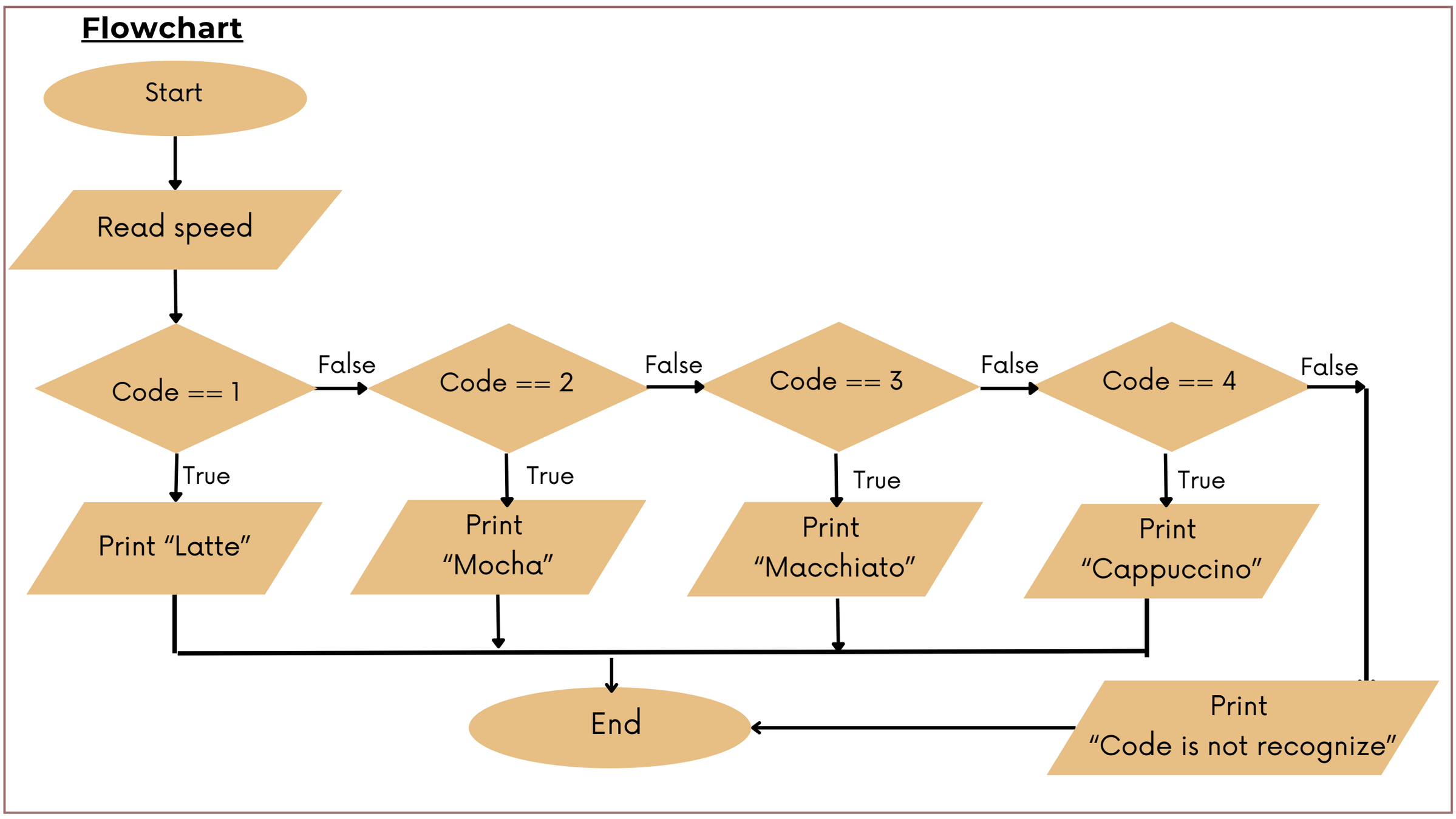
Algorithm

1. Read code
2. Switch (code), print Latte
3. Case 1, print Latte, break
4. Case 2, print Mocha, break
5. Case 3, print Macchiato, break
6. Case 4, print Cappuccino, break
4. Else print "Code is not recognize"
5. End switch

Pseudocode

```
Start
    Read code
      switch (code)
      Case 1: print Latte; break;
      Case 2: print Mocha; break;
      Case 3: print Macchiato; break;
      Case 1: print Cappuccino; break;
      Else print "Code is not recognize";
      End switch
End
```

Flowchart



EXERCISE 3.3E

1 A coffee vending machine has a few choices of coffee with different prices. Based on the table given below, create a program that can display the user selected coffee and it's price otherwise display "Type a correct code".

Coffee Code	Description	Price
001	Mocha	RM10.99
002	Latte	RM8.59
003	Cappuccino	RM11.25
004	Matcha	RM7.75

Answer
a. Algorithm
b. Pseudocode
c. Flowchart

3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

3. ITERATIONAL(LOOPING)

1

Initialize

- Set a value to start a loop.
- Executed only once before looping.

2

Condition

- A Boolean expression to check the condition.
- If not given, it is assumed to be true.

3

Counter

- Either to decrease or increase the initialize value.

while

```
Initialize;      1
while (Condition)  2
{
True statement

Counter          3
}
```

do..while

```
Initialize;      1
do
{
True statement

Counter          3
}
while (Condition)  2
```

For..loop

1 2 3

```
for (initialize; condition; counter)
{
statement 1;
statement 2 ;
}
```

3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

3. ITERATIONAL

- while

Consists of a block of code and a condition.
If the condition evaluated is true, the code within the block is executed.
This repeats until the condition becomes false.
While loops check the condition before the block is executed (**Pre-test Loop**)

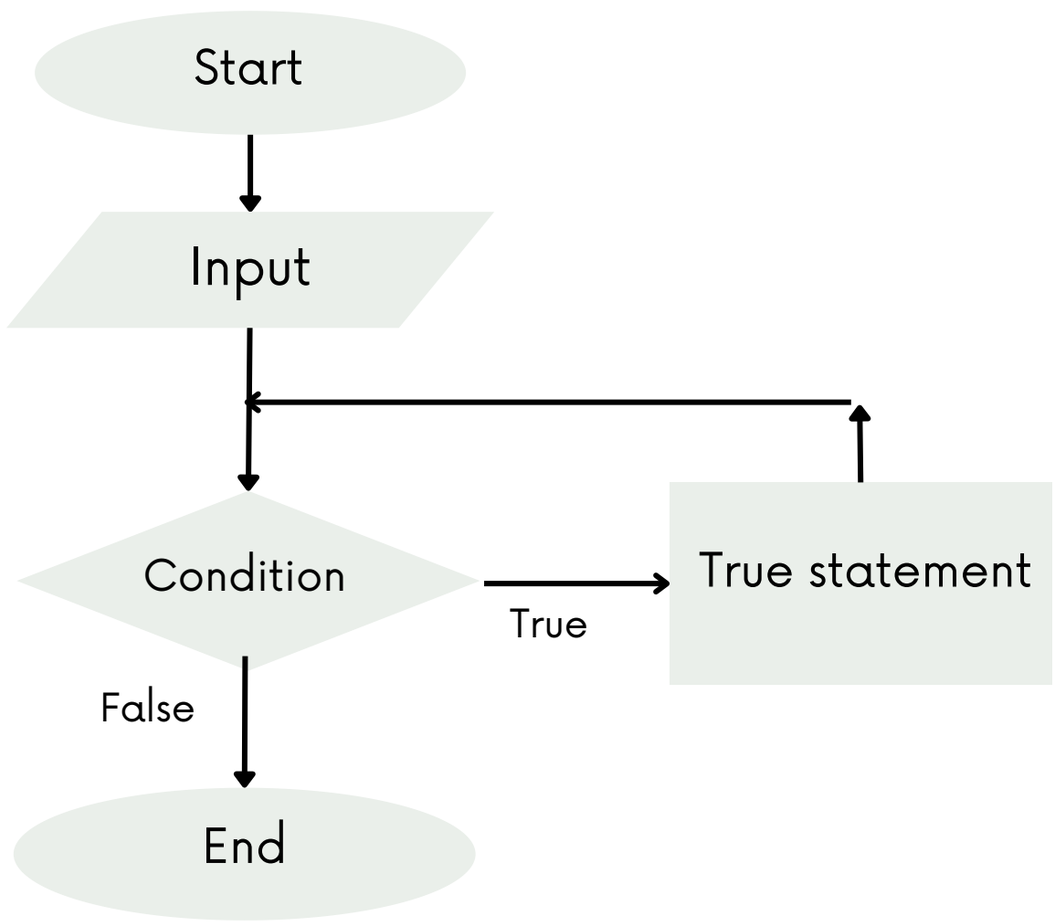
Algorithm

1. Input data (1)
2. While condition (2)
3. Compute True statement repeatedly
4. Counter (3)
5. End while

Pseudocode

```
Start
    Input data (1)
    While (condition) (2)
        Compute True statement;
        counter (3)
    End while
End
```

Flowchart



3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

Example 1: while

Create a program that can print the value which is less than 5.

Answer:

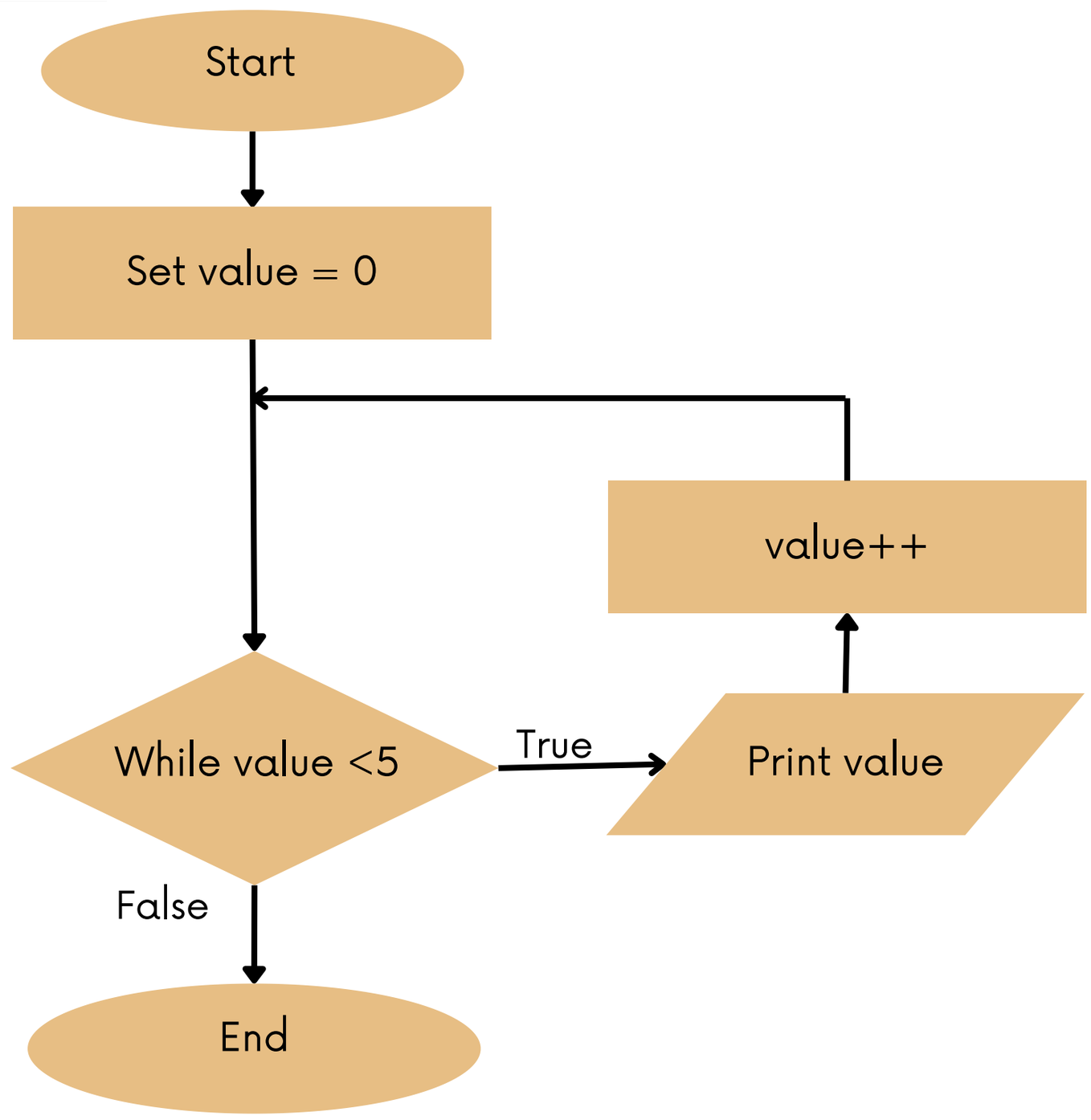
Algorithm

1. Set value = 0
2. While value less than 5
3. Print value
4. Value++
5. Repeat step 2
6. End while

Pseudocode

```
Start
        Set value = 0
          While (value < 5)
              Print value;
              Value ++;
          End while
 End
```

Flowchart



3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

3. ITERATIONAL

do...while

Consists of a block of code and a condition.
The code within the block is executed, then condition is evaluated.
This repeats until the condition becomes false.
Do...while loops check the condition after the block is executed (**Post-test Loop**)

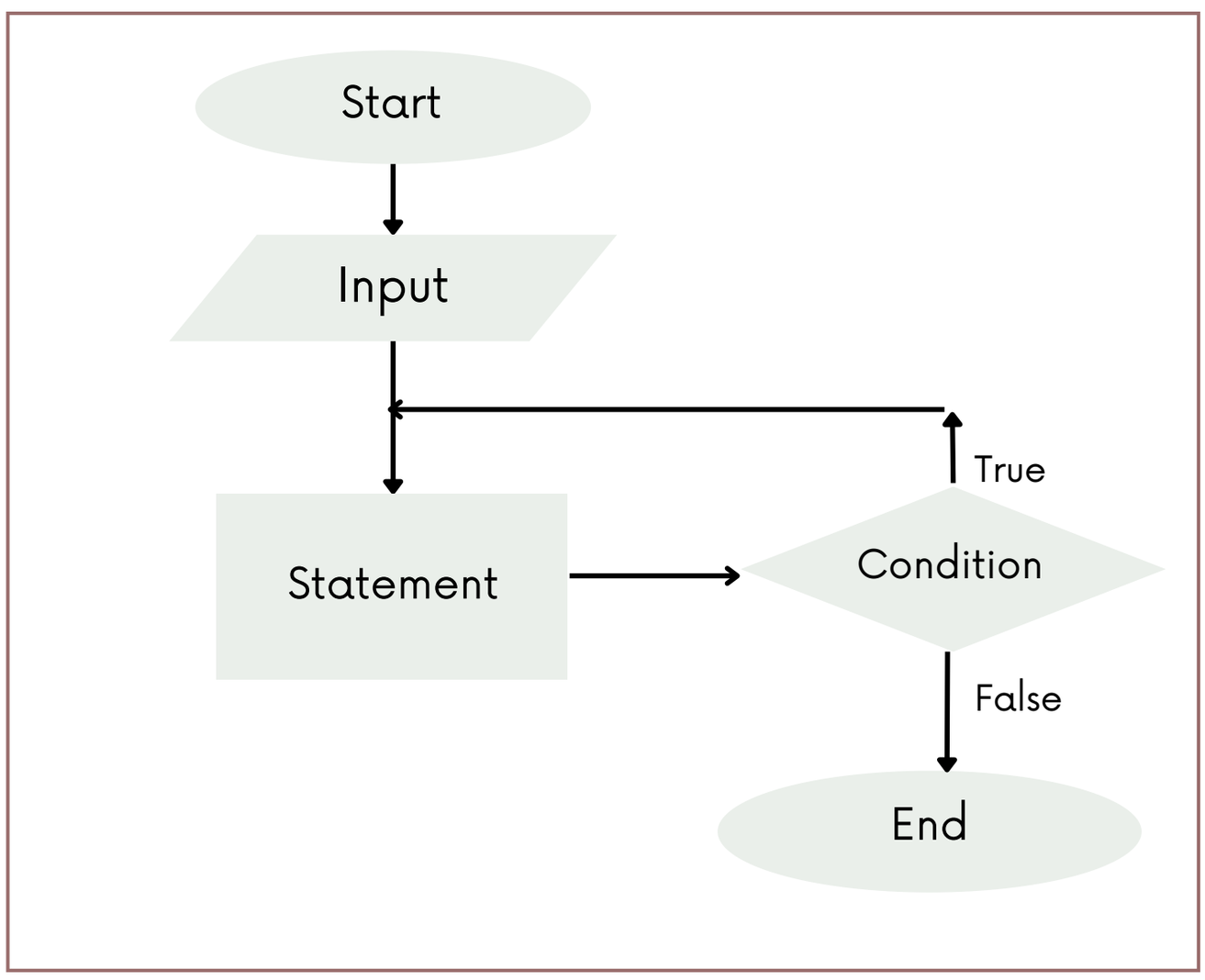
Algorithm

1. Input data 1
2. Repeat compute statement
3. Counter 3
4. While condition is false 2
5. End while

Pseudocode

```
Start
    Input data 1
      Repeat compute statement;
       Counter 3
       While (condition) is false 2
       End while
End
```

Flowchart



3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

Example 2: do..while

Create a program that can print the value which is less than 5

Answer:

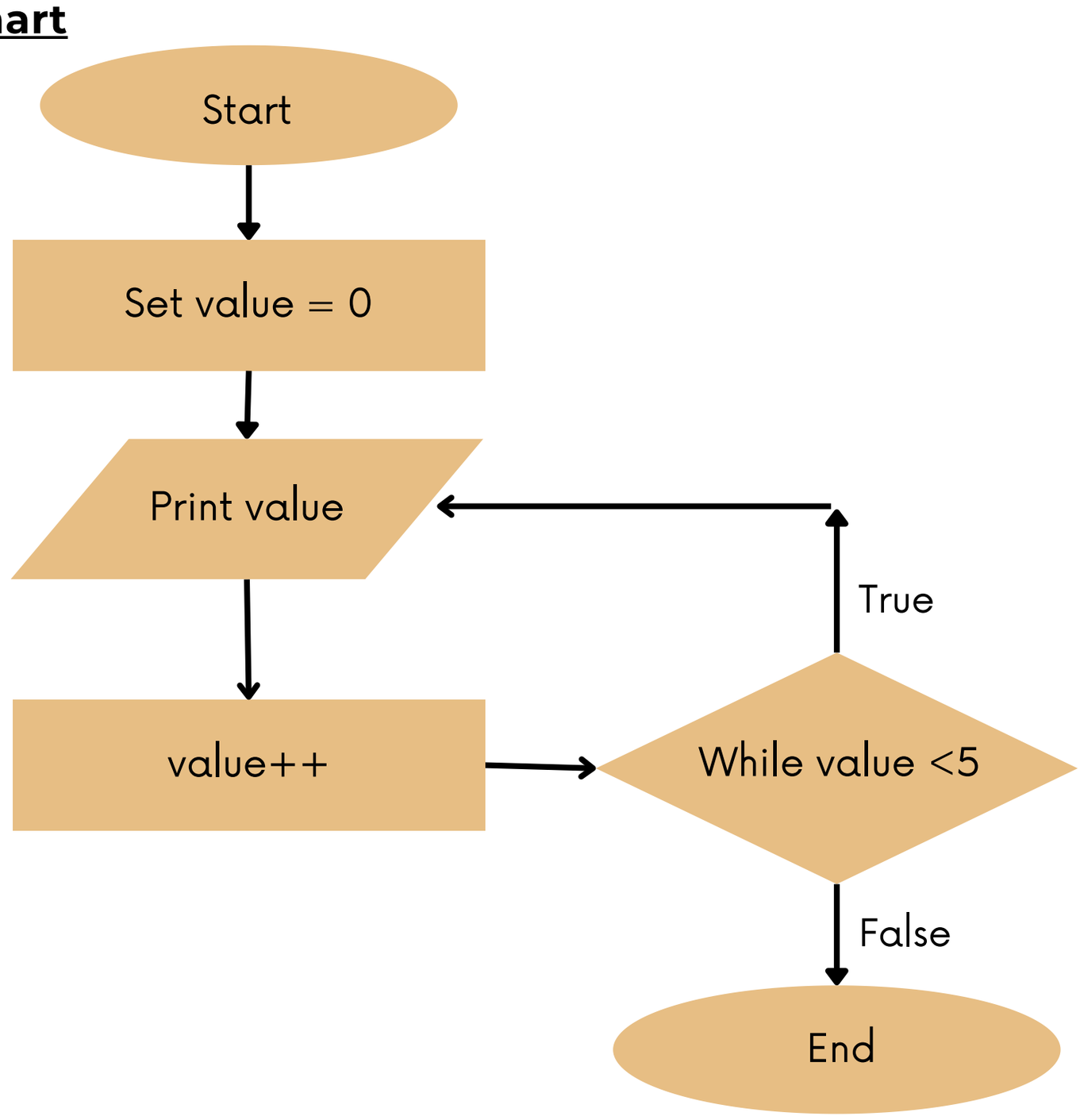
Algorithm

1. Set value = 0
2. Print value
3. Value++
4. While value less than 5
5. Go to step 2 repeatedly
6. End while

Pseudocode

```
Start
        Set value = 0;
          Print value;
          Value ++;
          While (value < 5)
          End while
 End
```

Flowchart



EXERCISE 3.3F

1 Based on the scenario given below, prepare an algorithm and pseudocode using while and do..while loop repetition control structures.

> Your goal is to walk 10,000 steps in a day. Each time you check your pedometer, you record how many steps you've taken since the last check. You want to keep checking your steps until you reach or exceed 10,000. After achieve the goal, display "Congratulations! You've walked 10000 steps today!"

Answer: Algorithm
i. While loop
ii. Do...While loop

EXERCISE 3.3F

Answer: Pseudocode
i. While loop
ii. Do...While loop

3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

3. ITERATIONAL

- for

Typically used if number of iterations known before looping.
Often distinguished by an explicit loop counter or loop variable.
Allows the body of the for loop to know about the sequencing of each iteration.

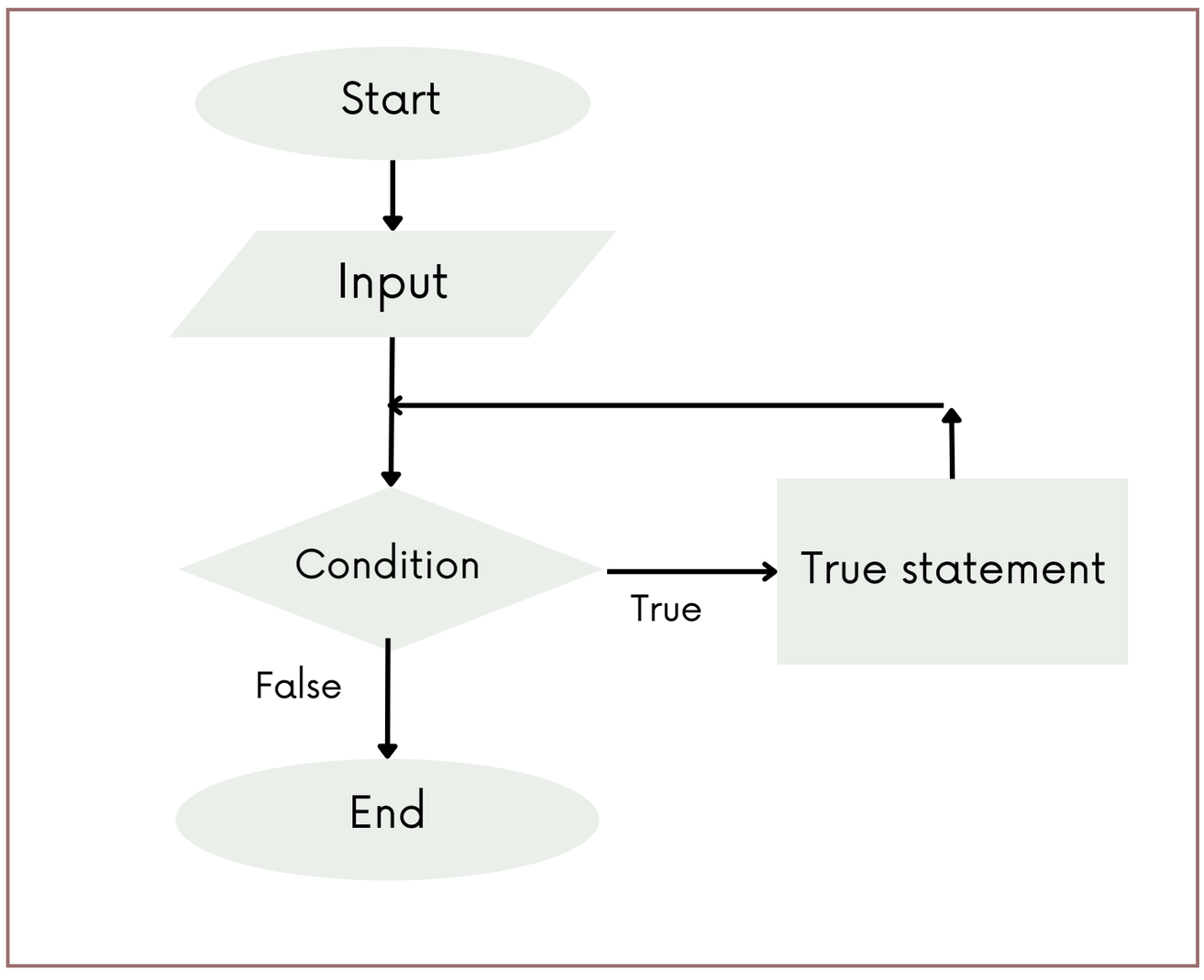
Algorithm

1. Input data (1)
2. For condition is True (2)
3. Compute True statement repeatedly
4. Counter (3)
5. End for

Pseudocode

```
Start
      Input data (1)      (2)      (3)
        For (initialize, condition, counter)
            Repeated compute True
            statement;
        End for
End
```

Flowchart



3.3 APPLY PROGRAM CONTROL STRUCTURES IN PROBLEM SOVLING

Example 3: for loop

Create a program that can print the value of i which is less than 5.

Answer:

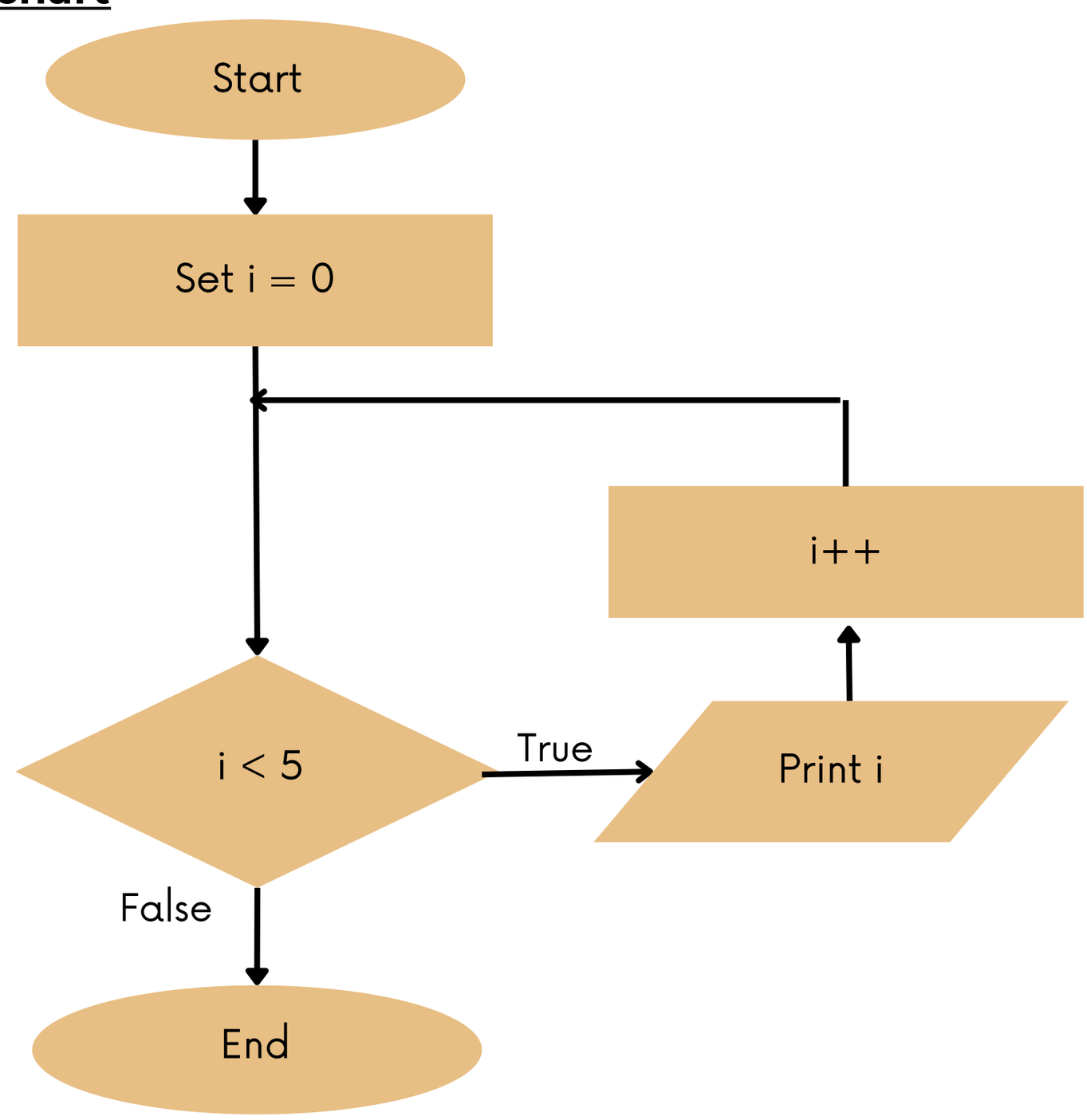
Algorithm

1. Set i = 0
2. For i less than 5
3. Print i
4. i = i + 1
5. If i less than 5, go to step 2
6. If false, go to step 7
7. End for

Pseudocode

```
Start
        for (i = 0; i < 5; i++)
          Print i;
          i ++;
        End for
 End
```

Flowchart



EXERCISE 3.3G

1 Based on the algorithm below, prepare a pseudo code and flowchart for each type of repetition control structures.

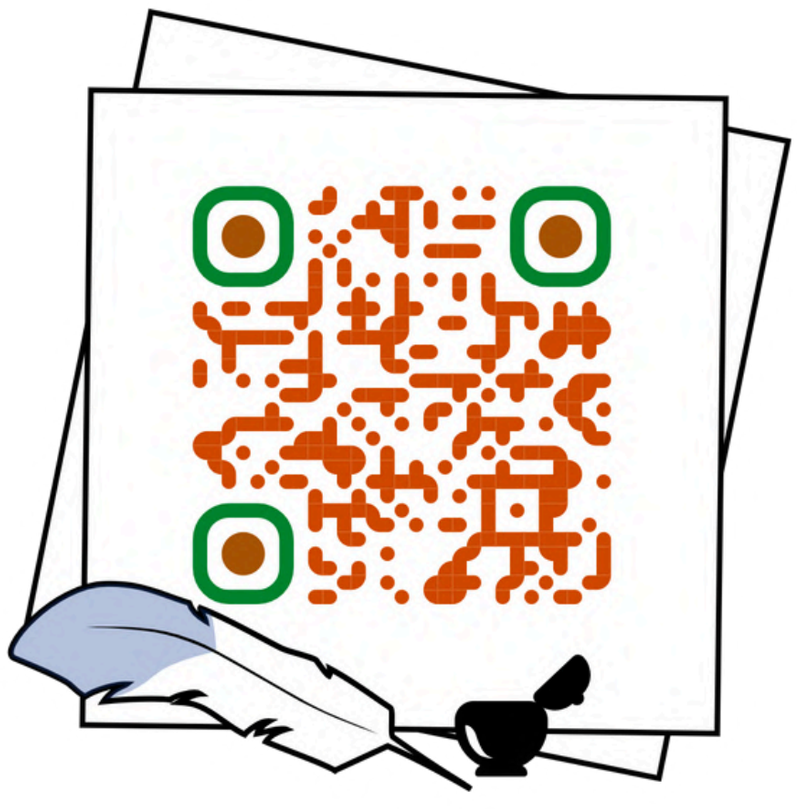
1. Initialize Counter = 1; average = 0; Total = 0;
2. Input number.
3. Add Total using formula:
   Total = Total + number
4. Add Counter using formula:
   Counter = Counter + 1
5. Compare whether Counter is greater than 5
6. If yes, go to step 8.
7. If no, go to step 2.
8. Calculate Average of numbers using formula:
   Average = Total / 5
9. Display Average.

Answer: Pseudocode
i. While loop
ii. Do...While loop

EXERCISE 3.3G

iii. For loop
Answer: Flowchart
i. While loop and For loop
ii. Do...While loop

SCAN THE QR CODE BELOW TO START THE QUIZ:



CHAPTER 4

BASIC OF PROGRAMMING CODES

LEARNING OUTCOMES

4.1 Describe Basic Programming Concept

4.2 Identify Standards and Best Practices

4.3 Apply the Basics of Programming Language

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

ELEMENTS OF PROGRAMMING LANGUAGE

- Comment
- Preprocessor directive
- Standard header file
- Main function
- Reserved word
- Identifiers
- Special symbol
- Statements
- Punctuation
- Blocks

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

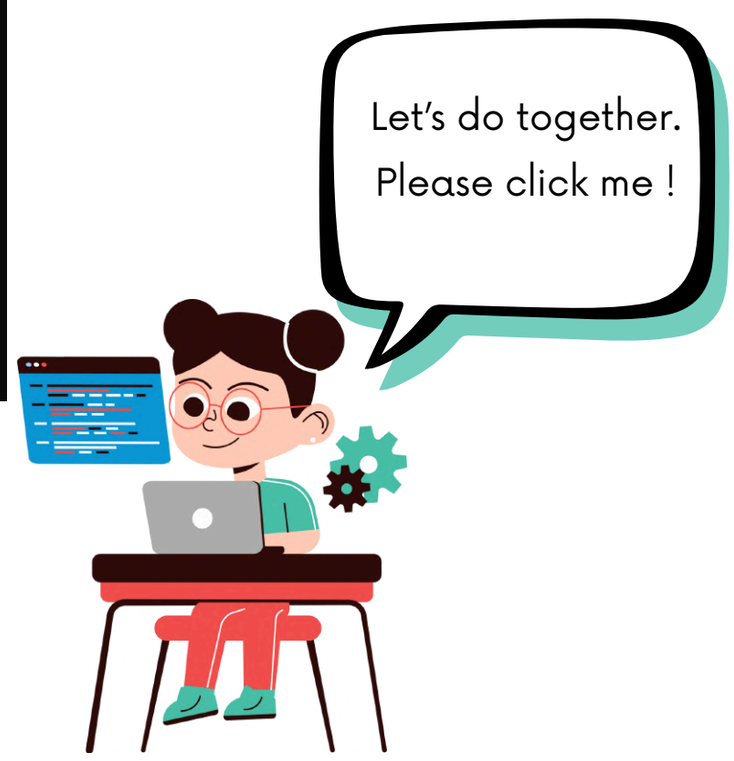
1. COMMENT

Comments are used to explain your code.
They are ignored by the compiler and are meant for human readers.

- Single-line Comment: Starts with //
- Multi-line Comment: Starts with /* and ends with */

C++

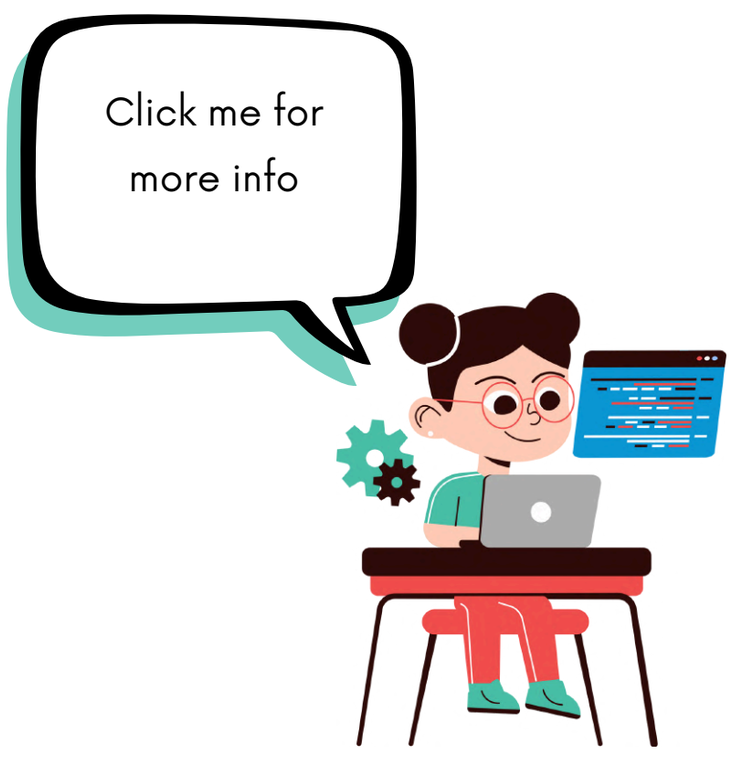
```cpp
#include <iostream>

using namespace std; // Allows us to use standard library features without std::

int main() {
    // This is a single-line comment
    cout << "Hello, World!" << endl; /* This is a multi-line comment */
    return 0;
}
```



2. PREPROCESSOR DIRECTIVES

These are commands that tell the compiler to include libraries or define constants before compiling the program.

- **#include**: Used to include standard libraries.
- **#define**: Used to define constants.



C++

```cpp
#include <iostream>

#define PI 3.14 // Defines a constant for PI

int main() {
    cout << "Value of PI: " << PI << endl; // Using the constant PI
    return 0;
}
```

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

3. MAIN FUNCTION

- The **main()** function is where every C++ program starts its execution.

C++

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "This is the main function." << endl;
    return 0; // Return 0 indicates the program ended successfully
}
```

4. STANDARD HEADER FILE

Contains declarations of functions, classes, and variables which can be used in different program files.
Access to the functionalities provided in that file, like input/output, data manipulation, algorithms, and more.
Example of common standard header file:

- **<stdio.h>**
- **<fstream>**
- **<iostream>**
- **<list>**

C++

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

5. RESERVED WORD

Known as **keywords**, are predefined words in C++ that have special meanings to the compiler.
These words **cannot be used as identifiers** (e.g., variable names, function names) in your programs because they are reserved for specific functionalities.

CONTROL STRUCTURES

- if
- else
- switch
- case
- default
- break
- continue
- for
- while
- do

DATA TYPE

- int
- float
- double
- char
- bool
- void
- string

COMPLETE LIST OF RESERVED WORDS IN C++:

alignas	alignof	and	and_eq	asm
auto	bitand	bitor	bool	break
case	catch	char	char16_t	char32_t
class	compl	const	constexpr	const_cast
continue	decltype	default	delete	do
double	dynamic_cast	else	enum	explicit
export	extern	false	float	for
friend	goto	if	inline	int
long	mutable	namespace	new	noexcept
not	not_eq	nullptr	operator	or
or_eq	private	protected	public	register
reinterpret_cast		return	short	signed
sizeof	static	static_assert	static_cast	struct
switch	template	this	thread_local	throw
true	try	typedef	typeid	typename
union	unsigned	using	virtual	void
volatile	wchar_t	while	xor	xor_eq

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

6. IDENTIFIERS

Identifiers are names assigned to variables, functions, classes, and other entities in C++.

Rules:

- Must start with a letter (A-Z, a-z) or underscore (_).
- Can contain letters, digits (0-9), or underscores, but cannot start with a digit.
- Case-sensitive: age, Age, and AGE are different identifiers.
- No reserved words or special characters (@, #, $).
- No spaces allowed in identifiers.

C++

```cpp
#include <iostream>
using namespace std;

// Variable identifier
int age = 25;

// Function identifier
void printMessage() {
    cout << "Welcome to C++ Programming!" << endl;
}

int main() {
    // Local variable identifier
    int number = 10;

    // Output the value of 'age' and 'number' using their identifiers
    cout << "Age: " << age << endl;
    cout << "Number: " << number << endl;

    // Call the function using its identifier
    printMessage();

    return 0;
}
```

EXPLANATION

1. **age**: A variable identifier used to store the age value.
2. **printMessage**: A function identifier that prints a welcome message.
3. **number**: A local variable identifier used inside the main() function to store a number.

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

7. SPECIAL SYMBOL

Special symbols have specific meanings and purposes within the code. These symbols are used for operations, data structuring, controlling program flow, and more.

COMMON SPECIAL SYMBOLS IN C++

1. **Braces { }** define code blocks.
2. **Parentheses ( )** are used in functions and control structures.
3. **Square Brackets [ ]** are used for arrays.
4. **Semicolon ;** ends statements.
5. **Comma ,** separates multiple variables or parameters.
6. **Asterisk** * is used for pointers.
7. **Ampersand &** obtains the address of a variable.
8. **Arrow -> and Dot .** access class/struct members.
9. **Pound #** is for preprocessor directives.

```
int main() {
    // Code inside the braces
    return 0;
}
```

BRACES { }

```
int add(int a, int b) {
    return a + b;
}
```

PARENTHESES ( )

```
int numbers[5] = {1, 2, 3, 4, 5};
cout << numbers[0];
```

SQUARE BRACKETS [ ]

```
int x = 10;
```

SEMICOLON ;

```
int a = 10, b = 20;
```

COMMA ,

```
int *ptr;
```

ASTERISK *

```
int x = 5;
int *ptr = &x;
```

AMPERSAND &

```
ptr->function();
```

ARROW ->

```
object.memberVariable;
```

DOT .

```
#include <iostream>
```

POUND #

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

8. STATEMENT

A statement in C++ is a command or instruction that tells the computer to perform an action. Statements are the building blocks of a C++ program and control how the program behaves.

EXPLANATION

- **Expression Statements:**
Perform calculations or assignments.

- **Compound Statements:**
Group of statements enclosed in { }.

- **Selection Statements:**
Make decisions (if, else, switch).

- **Iteration Statements:**
Repeatedly execute code (for, while, do-while).

- **Jump Statements:**
Alter the program flow (break, continue, return).

- **Declaration Statements:**
Declare variables, constants, or functions.

EXAMPLE

- **Declaration Statement:**
int number = 10;
- **Expression Statement:**
number > 5;
- **Selection Statement:**
if (number > 5)
- **Iteration Statement:**
for (int i = 0; i < 3; i++)
- **Return Statement:**
return 0;

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

9. PUNCTUATION

Punctuation symbols in C++ are used to organize the structure of the code and define the relationships between different elements in a program. They play an essential role in defining the syntax of the language.

EXPLANATION

- **Semicolon ;:**

Ends each statement.

- **Comma ,:**

Separates multiple variables in a declaration.

- **Parentheses ( ):**

Used in the if condition and function calls.

- **Braces { }:**

Define the block of code inside the if statement.

- **Square Brackets [ ]:**

Accesses elements of the array numbers[].

- **Hash #:**

Used for including the input/output library.

EXAMPLE

- **Semicolon ;:**

Ends statements.

- **Comma ,:**

Separates variables or expressions.

- **Parentheses ( ):**

Encloses parameters and expressions.

- **Braces { }:**

Encloses code blocks.

- **Square Brackets [ ]:**

Used with arrays.

- **Colon ::**

Used in switch cases or initialization lists.

- **Double Colon :::**

Used for scope resolution.

- **Dot .:**

Accesses class or structure members.

- **Arrow ->:**

Accesses members through pointers.

- **Pound/Hash #:**

Preprocessor directives.

4.1 DESCRIBE BASIC PROGRAMMING LANGUAGE

10. BLOCKS

Block in C++ is a group of statements enclosed within curly braces { }. Blocks are used to group multiple statements together so that they can be treated as a single unit. Blocks are commonly used in functions, loops, conditional statements, and other control structures.

KEY FEATURES OF BLOCKS

- **Grouping of Statements:**
Multiple statements inside a block { } are executed together.

- **Scope:**
Variables declared inside a block have block scope and are only accessible within that block.

- **Used in Control Structures:**
Blocks are used in loops (for, while), conditionals (if, else), and function definitions.

TYPE OF BLOCKS IN C++

- **Function Block:**
Every function in C++ has a block that defines its body.

- **Conditional Block (if, else, switch):**
Used to group statements that will execute based on a condition.

- **Loop Block (for, while, do-while):**
The body of a loop is enclosed in a block.

- **Nested Block:**
A block inside another block, often used to manage the scope of variables and for control structures.

- **Compound Statement Block:**
A compound statement groups several statements into one block.

EXERCISE 4.1

1

CASE STUDY

Infinity Design Solution Sdn. Bhd, has received your application as an industrial student to run your practical in this company for three months. Miss Suria whom is the IT programmer for Infinity Design Solution is appointed to be your supervisor. As you are new in working environment; Miss Suria plans to brief you on the programming language elements before major tasks are appointed to you.

INSTRUCTION:
Your task is to convince Miss Suria that you are familiar with the programming language elements by filling up all the boxes:

DIAGRAM 1

```
/* This is a simple program in C++ to calculate the circumference of a circle*/   A

#include <iostream>   B   C

const double PI = 3.142;   D

using namespace std;

int main()   E
{
    int r;
    double circumference;   F

    cout << "Please enter the radius : ";
    cin >> r;

    circumference = 2*PI*r;

    cout << "The radius entered is : " << r << endl;
    cout << "The circumference calculated is : " << circumference << endl;

    system("pause");

    return 0;   G
}
```

Your task is to convince Miss Suria that you are familiar with the programming language elements by filling up all the boxes:

Problem	**Solution**
Problem 1: Based on the case study programming language, identify the elements noted in A-G	A: B: C: D: E: F: G:

Problem 2:
You are required to identify the basic elements of the programming code below:
 	
// Program to display the message
// Welcome to a new language
#include <iostream>
using namespace std;
 	
int main()
{
 cout << " Welcome to C++ Programming Language! ";
 return 0;
}
	a. Comment:
	
	b. Preprocessor directive :
	
	c. Main function:
	
	d. Reserved word :
	
	e. Special symbol :
	
	f. Punctuation:
	
	g. Statements:

4.2 IDENTIFY STANDARDS AND BEST PRACTICES

THE STANDARDS AND BEST PRACTICES IN WRITING PROGRAM CODES

1. Use of comment

- Comment helps the programmer understand what exactly is happening on the code.
- Avoid obvious comments

2. Consistent indention

- Consistent alignment of each block
- Make code easier to read

3. Consistent Naming Scheme

- Easy to read and is not confusing
- camelCase: The first letter of each word is capitalized, except the first word (eg. dateOfBirth)
- underscores: Underscores between words (eg. date_of_birth)

4. Avoid Deep Nesting

- Least levels of nesting can make code easier to read and follow

```
function do_stuff() {

// ...

        if (is_writable($folder)) {

                if ($fp = fopen($file_path,'w')) {

                        if ($stuff = get_some_stuff()) {

                                if (fwrite($fp,$stuff)) {

                                        // ...

                                } else {
                                        return false;
                                }
                        } else {
                                return false;
                        }
                } else {
                        return false;
                }
        } else {
                return false;
        }
}
```

4.2 IDENTIFY STANDARDS AND BEST PRACTICES

THE STANDARDS AND BEST PRACTICES IN WRITING PROGRAM CODES

5. Limit Line Length

- More comfortable when reading tall and narrow columns of text.
- Avoid writing horizontally long lines of code

6. File and Folder Organization

- Proper file naming
- Separation of file accordingly for a better experience of maintenance

7. Code Grouping

- Keep statement within separate blocks of code, with some spaces between them.

```
function foo() {
        if ($maybe) {
                do_it_now();
                again();
        } else {
                abort_mission();
        }
        finalize();
}
```

```
function foo()
{
        if ($maybe)
        {
                do_it_now();
                again();
        }
        else
        {
                abort_mission();
        }
        finalize();
}
```

```
function foo()
{       if ($maybe)
        {       do_it_now();
                again();
        }
        else
        {       abort_mission();
        }
        finalize();
}
```

4.2 IDENTIFY STANDARDS AND BEST PRACTICES

THE DISADVANTAGES OF NOT FOLLOWING STANDARDSAND BEST PRACTICES WHILE WRITING CODE.

1. Difficult to read the code

- Some misunderstanding and confusion may occur

2. Difficult to identify mistakes or logic error

- Logic error unable to detect by compiler, need human experience to identify
- Inconsistent naming convention scheme make it difficult to spot error

3. Increase development time

- Using a deep nesting or long programming code may take time for developer to manage the code

4. Limited Flexibility

- Developer may find themselves limited by the framework's conventions and may need to write additional code to achieve their desired functionality

EXERCISE 4.2

Based on the given code below, list out the best practices that been applied in writing programming code.

```
#include <iostream>
using namespace std;

int main()
{
    int package = 699;
    float discount, NewPrice; //declare variable

    discount = 0.15*package; //calculate discount
    NewPrice = package - discount; //calculate new price

    cout<< "Discount amount: " << discount<<endl;
    cout<< "New Price is "<<NewPrice<<endl;

    return 0;
}
```

Answer:

4.3 APPLY THE BASICS OF PROGRAMMING LANGUAGE

Scan to download and install Dev C++

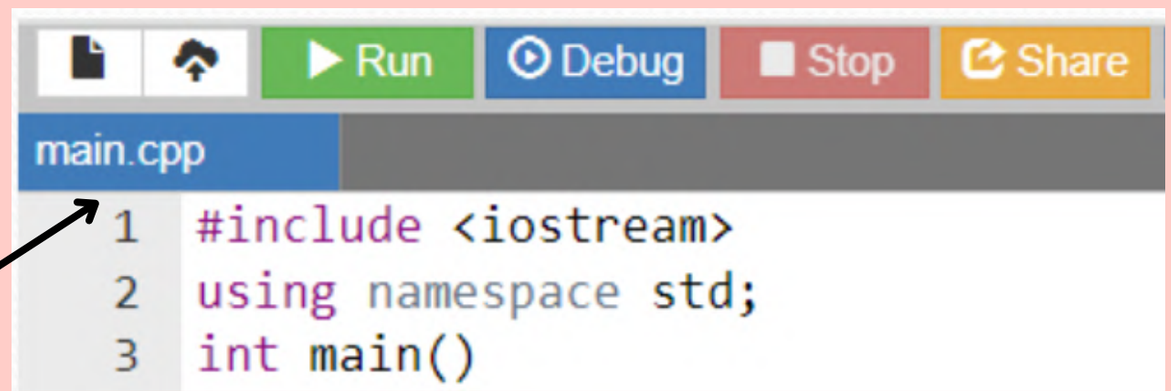
Let's Start!

DOWNLOAD NOW

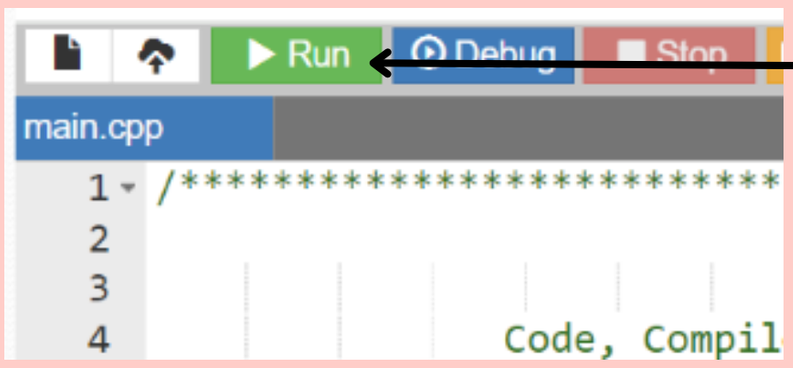
4.3 APPLY THE BASICS OF PROGRAMMING LANGUAGE

IDENTIFY THE STEPS IN CREATING A PROGRAM

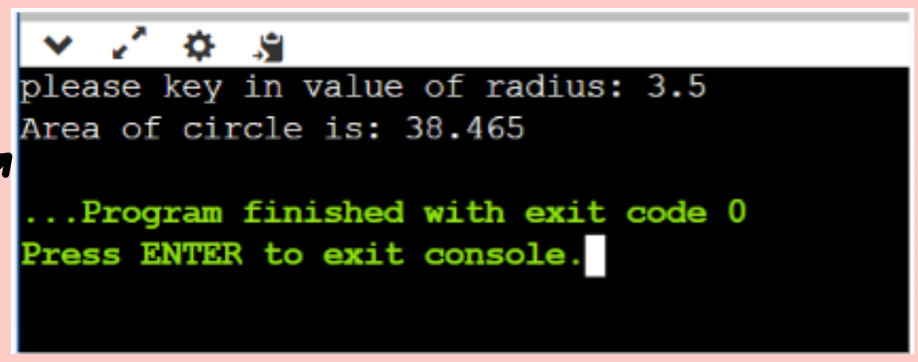
- **Source:**
Create a file name for source code – type your source code (eg. main.cpp)



- **Compile:**
Once finish, click Run to compile your code.



- **Output:**
Code is successfully compile, you will get the output as below:



4.3 APPLY THE BASICS OF PROGRAMMING LANGUAGE

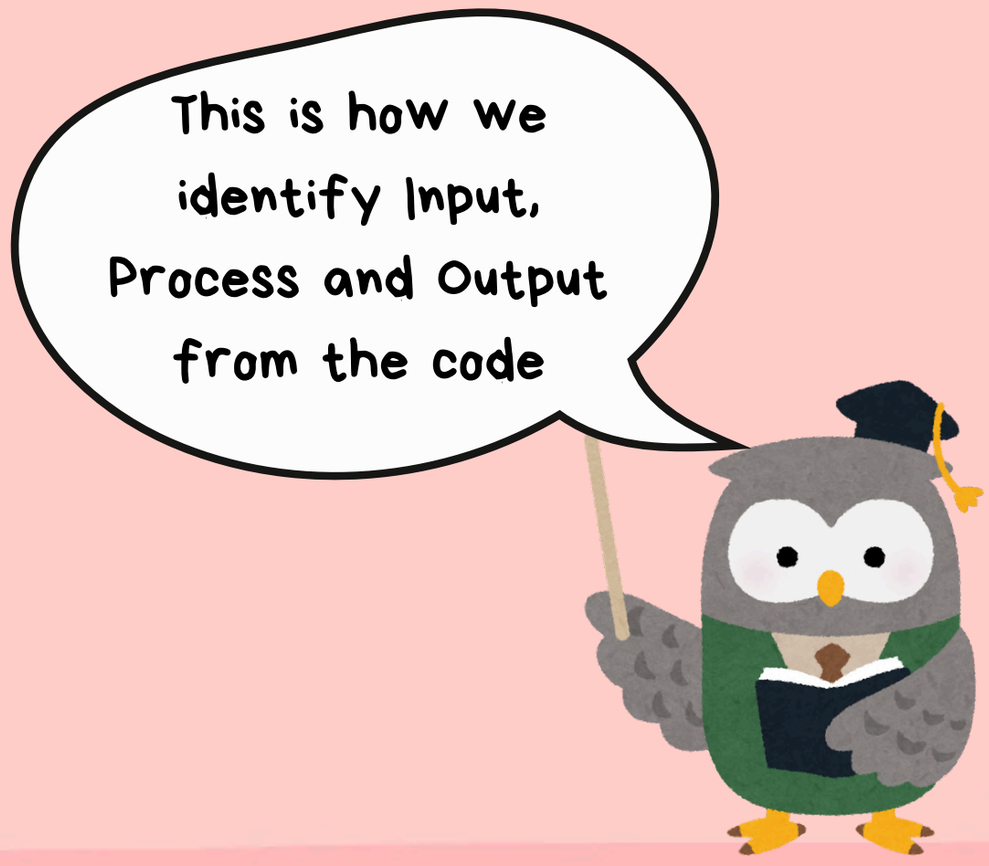
ANALYZE A PROGRAM TO IDENTIFY INPUT, PROCESS AND OUTPUT

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7     int width,length,area;
8
9     cout<< "Enter Width: ";
10    cin>> width;
11    cout<< "Enter Length: ";
12    cin>>length;
13    area = width*length;
14    cout<< "Area: " << area<<endl;
15
16    return 0;
17  }
18
```

INPUT

PROCESS

OUTPUT



4.3 APPLY THE BASICS OF PROGRAMMING LANGUAGE

IDENTIFY DATA TYPES & DECLARATION OF CONSTANT AND VARIABLES

How to identify data type from the given code?

Answer:

You must know what is the value to be stored in your variable.

main.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      float radius, area;
6      const double pi = 3.14;
7
8      cout<<"please key in value of radius: ";
9      cin>> radius;
10
11     //formula to calculate area of a circle
12     area = pi * radius * radius;
13
14     cout<<"Area of circle is: "<< area;
15
16     return 0;
17 }
```

Data type for radius and area is **float.** Declaration of **variables** name: **float radius, area;**

Data type for pi is **double.** Declaration of **constant**: **const double pi = 3.14;**

How to declare variable and constant?

Answer:

First, write down the data type, then follow with identifier name.

4.3 APPLY THE BASICS OF PROGRAMMING LANGUAGE

WRITE BASIC PROGRAMMING CODE

Given Problem:

Salam and Zalihar were planning for a honeymoon. Salam was decided to take the honeymoon package for RM699 with 15% discount. You are required to create a program to calculate how much Salam need to pay for the honeymoon.

1st Step: Identify Input, Process and Output

Input: none
Process: package = 699
discount = (15/100)*package
NewPrice = package – discount
Output: NewPrice

2nd Step: Design Algorithm

1. Set package = 699
2. Calculate the discount by multiply (15/100) with package
3. Calculate the NewPrice after discount by minus discount from package
4. Print NewPrice

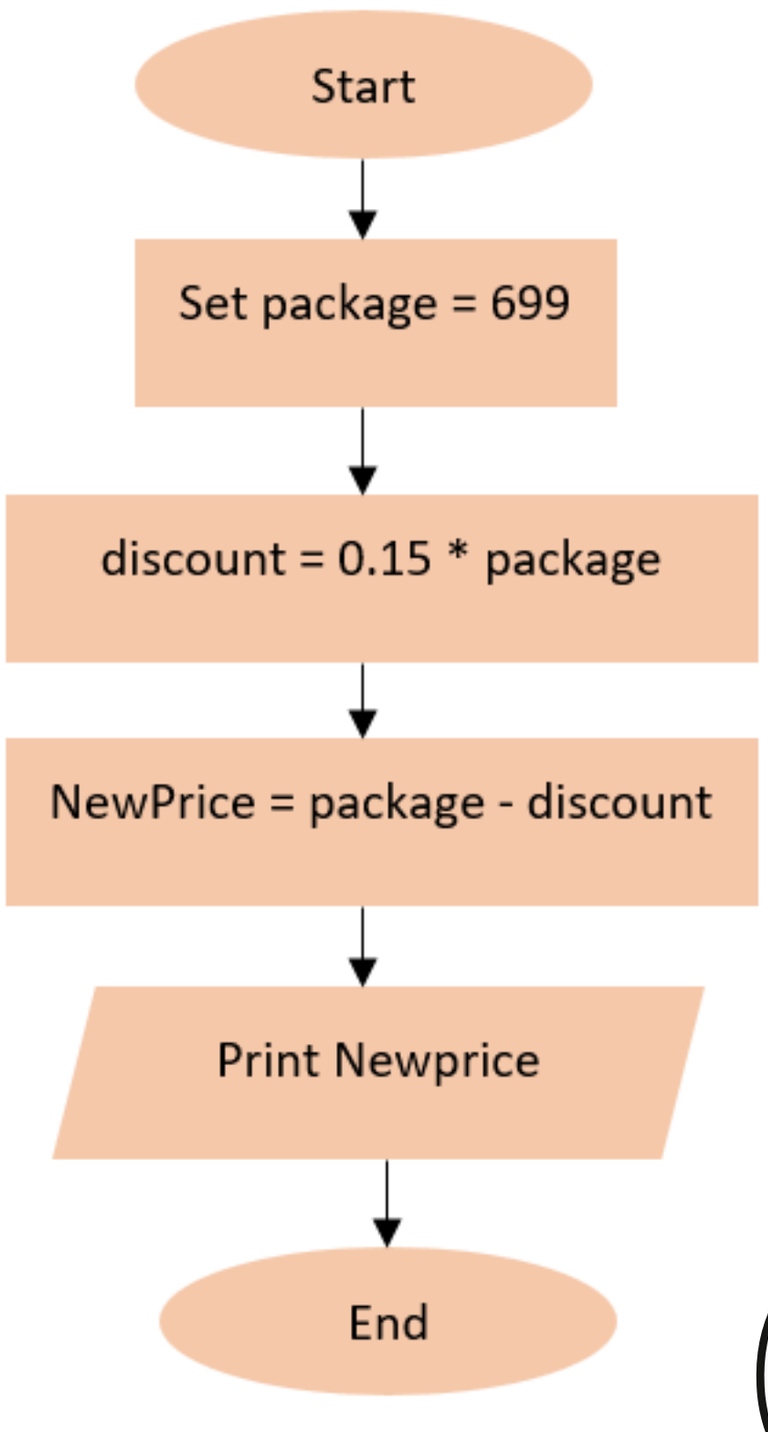
4.3 APPLY THE BASICS OF PROGRAMMING LANGUAGE

WRITE BASIC PROGRAMMING CODE

3rd Step: Write Pseudocode

```
Start
    Set package = 699
    discount = (15/100)*package
    NewPrice = package – discount
    Print NewPrice
End
```

4th Step: Draw Flowchart



4.3 APPLY THE BASICS OF PROGRAMMING LANGUAGE

WRITE BASIC PROGRAMMING CODE

5th Step: Program code

```
#include <iostream>
using namespace std;

int main()
{
    int package = 699;
    float discount, NewPrice; //declare variable

    discount = 0.15*package; //calculate discount
    NewPrice = package - discount; //calculate new price

    cout<< "Discount amount: " << discount<<endl;
    cout<< "New Price is "<<NewPrice<<endl;

    return 0;
}
```

EXERCISE 4.3A

1 **Miss Suria has provided you with a task to write the algorithm and illustrate a flowchart based on the codes given.**

main.cpp

```
#include <iostream>
using namespace std;

int main()
{
  int a, b, total ;

   cout << "Input numbers to be added: " << endl;

   cin >> a >> b ;
   total = a + b ; // AddTwoNumbers

   cout << "The sum is " << total
   << endl;

   return 0;
}
```

Answer: Algorithm

Answer: Flowchart

EXERCISE 4.3A

2 **Analyze the given programming codes below, then identify it input, process and output. Then, transform into a pseudocode.**

ain.cpp

```
#include <iostream>
using namespace std;

int main()
{
        int Q_Orange, Q_Apple;
    float P_Orange, P_Apple, total_price; // declare variables

    cout << "Please enter quantity of Orange: ";
    cin >> Q_Orange;

    cout << "Please enter price of Orange, RM: ";
    cin >> P_Orange;

    cout << "Please enter quantity of Apple: ";
    cin >> Q_Apple;

    cout << "Please enter price of Apple, RM: ";
    cin >> P_Apple;

    total_price = (Q_Orange*P_Orange) + (Q_Apple*P_Apple);

    if (total_price > 50)
        {
        cout << "You will get a lucky draw! Your total price is RM "<< total_price;
        }
        else
        {
            cout << "Your total price is RM "<< total_price;
        }

    return 0;
}
```

Answer: IPO

Answer: Pseudocode

EXERCISE 4.3B

1

Create a program that prompt 10 students' marks, then calculate the total and average of 10 student marks. Finally print total mark and the average. Please use either for loop or while loop for writing the codes.

Answer: Pseudocode

Answer: Codes

SHALL WE START THE QUIZ NOW??

SCAN THE QR CODE BELOW TO START THE QUIZ:

References

Halim, Y., Harun, N., & Ismail, R. (2021). Quick Learner Problem Solving & Program Design (2nd ed.). Politeknik Kuala Terengganu. https://anyflip.com/toehg/ggmh/basic

Spindt, S. (2019, December 16). Problem Solving in Computer Science: What Does It Really Mean? Lead by Learning. https://weleadbylearning.org/2019/12/16/problem-solving-in-computer-science-what-does-it-really-mean/

C++ Basics: The Easiest Guide to Understand Basic Concepts of C++. (2024, July 23). Simplilearn. https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-basics#

ASM 100: x86 Assembly Code (69 pts). (n.d.). https://samsclass.info/127/proj/ASM100.htm

C++ Comments. (n.d.). W3schools. https://www.w3schools.com/cpp/cpp_comments.asp

C++ Preprocessor. (n.d.). Tutorialspoint. https://www.tutorialspoint.com/cplusplus/cpp_preprocessor.htm

Functions in C++ - 12 (Part - 1). (2016, December 7). [Video]. YouTube. https://www.youtube.com/watch?v=2UqzHffkFbU

Thinks, A. (2022, February 7). How to install Dev C++ on Windows 11 [Video]. YouTube. https://www.youtube.com/watch?v=BRKQnirXw7o

Generations of programming language || Explained in brief. (2017, January 9). [Video]. YouTube. https://www.youtube.com/watch?v=js1I1nSIfp4

IPO Cycle. (2022, June 4). [Video]. YouTube. https://www.youtube.com/watch?v=gbTMrR1F36U

Coding Basics: Variables | Programming for Beginners |. (2020, August 20). [Video]. YouTube. https://www.youtube.com/watch?v=ghCbURMWBD8

Computer Science Basics: Sequences, Selections, and Loops. (2018, October 4). [Video]. YouTube. https://www.youtube.com/watch?v=eSYeHlwDCNA

BRIGHTEN UP : PROBLEM SOLVING & PROGRAM DESIGN

e ISBN 978-967-2765-15-8

9 789672 765158

POLITEKNIK BALIK PULAU

(online)