CSC 128

TOPIC 1 : INTRODUCTION TO COMPUTER, PROGRAM AND PROGRAMMING LANGUAGE

By : MOHD SAIFULNIZAM ABU BAKAR

LEARNING OUTCOME

At the end of this chapter, you should be able to:

- Understand the concepts and importance of programs and programming.
- Differentiate between program, compiler, interpreter and assembler.
- Apply the steps in the program development life cycle.

CHAPTER OUTLINE

- A Brief History of Programming Language
- Introduction to Programming
  - What is a computer program and importance of computer programming
  - Importance of good programs.
  - Relationship between compilers, interpreters, assemblers and programs
  - C++ program structure
- Program Development Life Cycle
  - Problem solving phases; problem definition, algorithm design and implementation
  - Analysis, design, coding, maintenance

INTRODUCTION

- Computers can be found anywhere from the size of a desktop to smaller than the palm of one's hand such as desktop computers, notebooks, netbooks, tablet PCs and mobile devices.
- Many kinds of applications or apps can be downloaded into the tablet or smartphone.

INTRODUCTION

- There are many ways to develop these applications.
- Some websites provide templates to create apps quickly
- Users with programming knowledge can create their apps from scratch.
- Examples of systems/apps developed using programming language:
  - Automated Teller Machine (ATM) systems,
  - Student Information Systems
  - Online Ticketing Systems

OVERVIEW OF COMPUTERS AND HISTORY OF PROGRAMMING LANGUAGE

HISTORY OF COMPUTERS

19th Century	first general purpose computing device.
20th Century	early **analogue** computers have been invented.
1936	• first freely programmable computers were developed.
1940s	• electronic programmable computers using vacuum tubes and transistors have been created.
1950-1960s	• commercial computers, computer games and **programming languages** were developed.
1975	• mainframes and supercomputers were available.
1981	• IBM introduced its **personal computers**
1983	• The first personal computer with a graphical user interface was introduced

BASIC OPERATION OF A COMPUTER

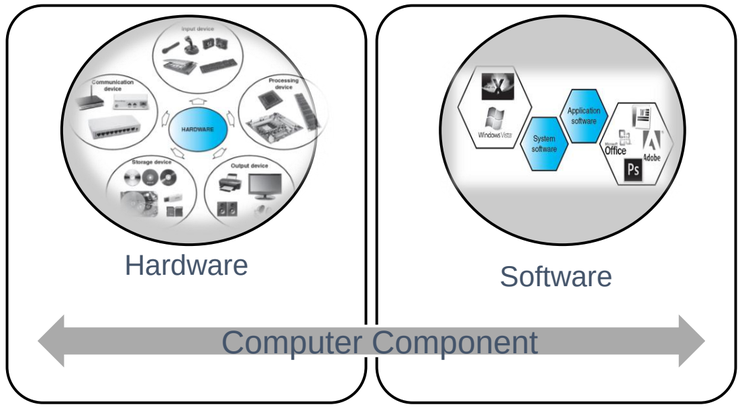
- A computer is a device that can process data.
- Data consists of raw facts or unprocessed information

Basic operation of a computer

- Input – accepts data from user
- Process – manipulate data
- Output – produce result
- Storage – store result

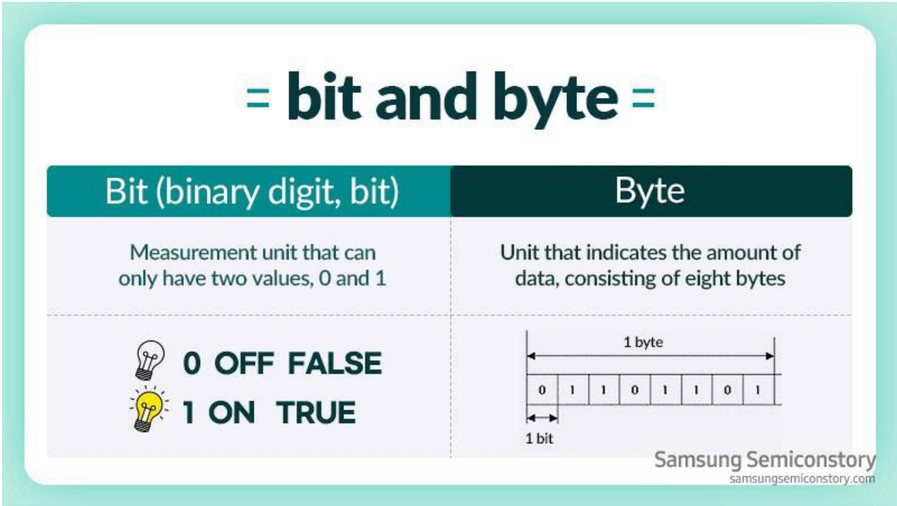
BASIC OPERATION OF A COMPUTER

- Computers are electronic devices capable of performing computations and making logical decisions at speeds faster than human beings.

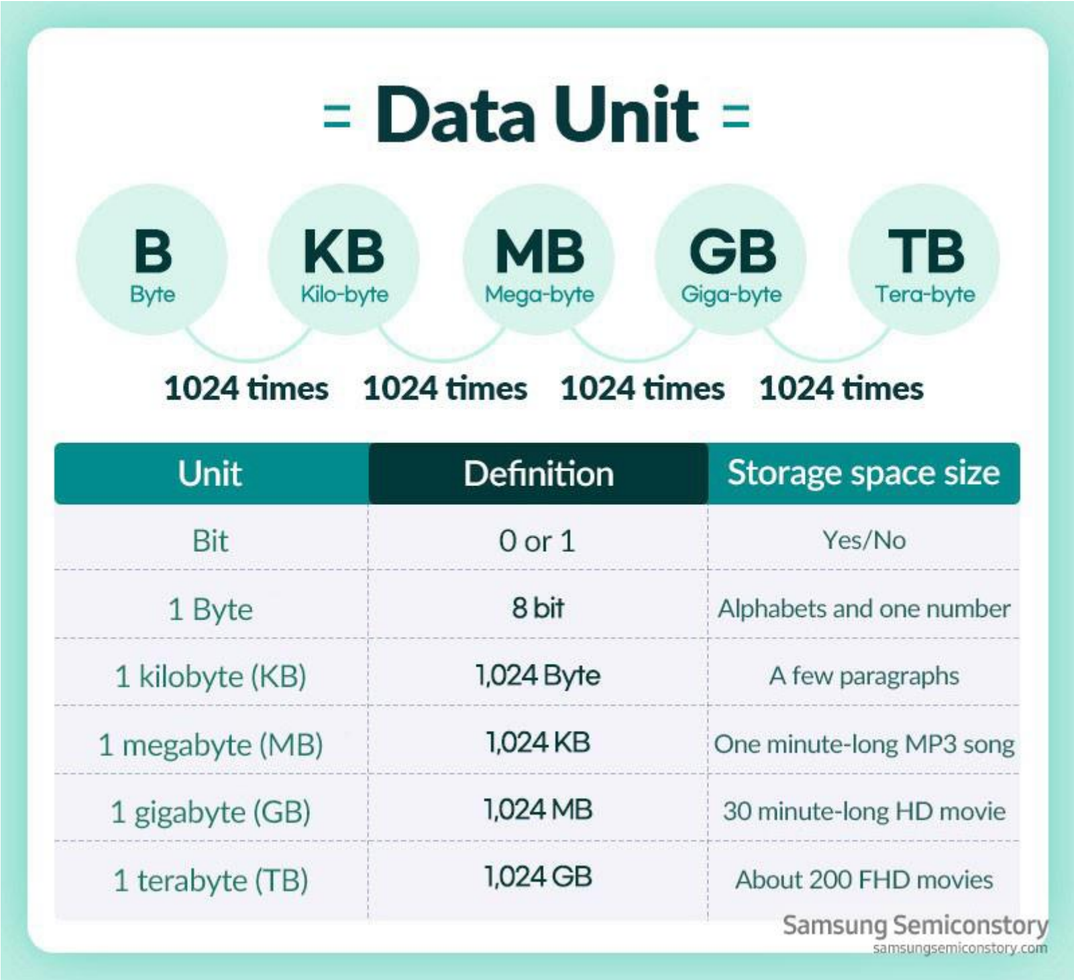


LANGUAGE OF A COMPUTER

- Computers can only understand machine language.
- Machine language is also called binary numbers or binary code, which is a sequence of 0s and 1s.
- The digits 0 and 1 are called binary digits or bits.
- A sequence of 8 bits is called a byte.

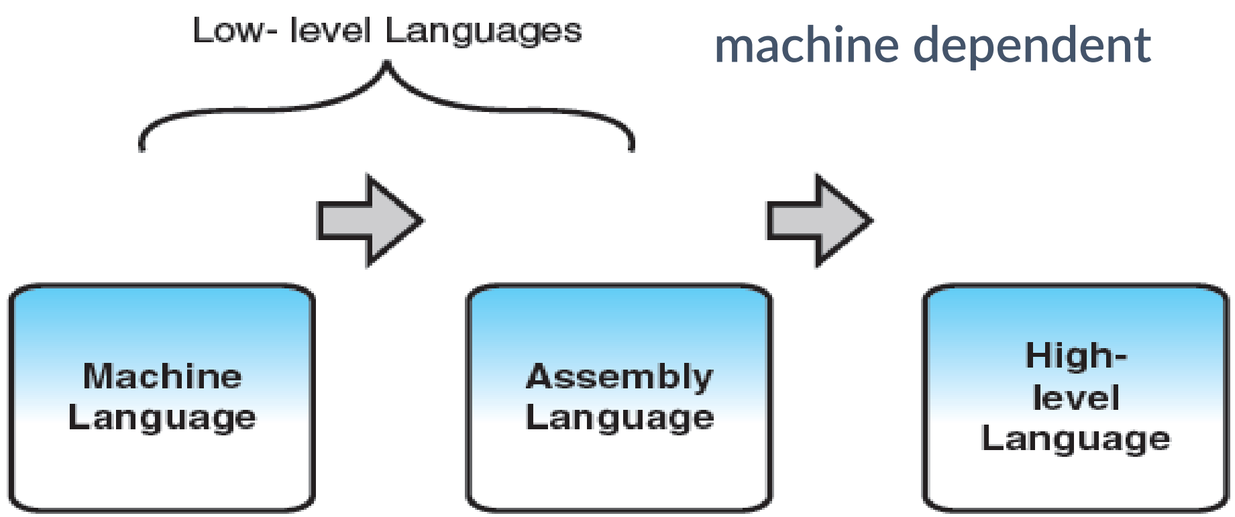


LANGUAGE OF A COMPUTER



Unit	Definition	Storage space size
Bit	0 or 1	Yes/No
1 Byte	8 bit	Alphabets and one number
1 kilobyte (KB)	1,024 Byte	A few paragraphs
1 megabyte (MB)	1,024 KB	One minute-long MP3 song
1 gigabyte (GB)	1,024 MB	30 minute-long HD movie
1 terabyte (TB)	1,024 GB	About 200 FHD movies

TYPES OF PROGRAMMING LANGUAGE



- Early computers were programmed in machine language

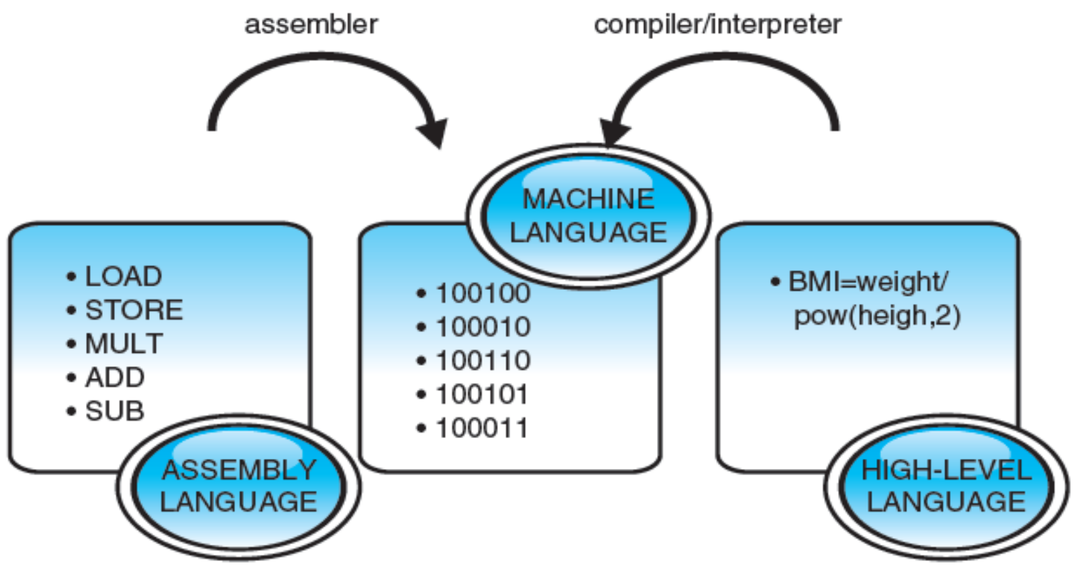
- language is mnemonic
- use assembler as translator to translate to machine language

- machine independent
- the instructions are quite English-like
- use compiler/interpreter as translator to translate to machine language
- Example: JAVA, C++, COBOL

TYPES OF PROGRAMMING LANGUAGE

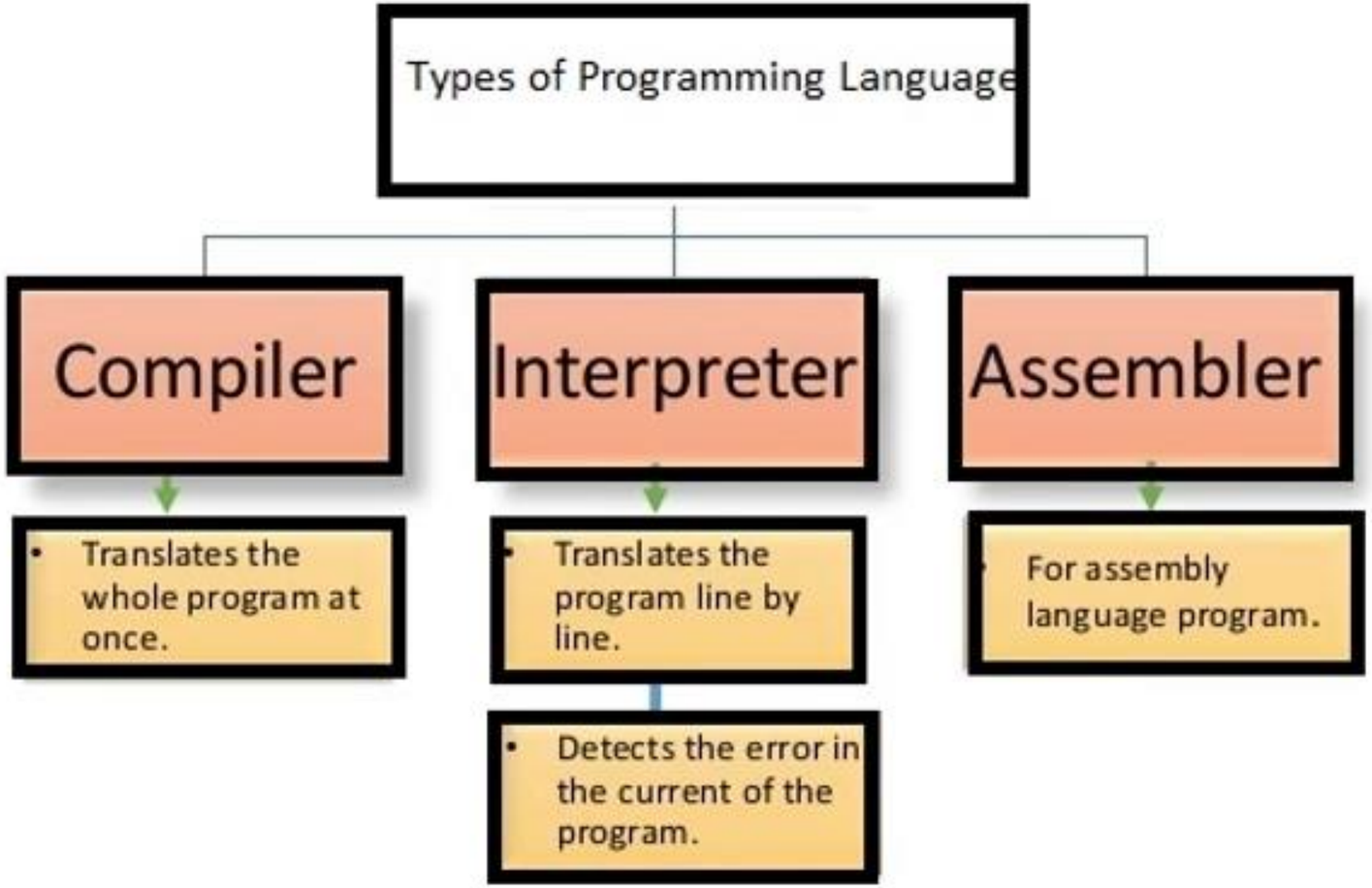
- Example:
- To calculate the BMI of a user given the formula:

$$\text{BMI} = \frac{\text{weight (kg)}}{\text{height (m) x height(m)}}$$

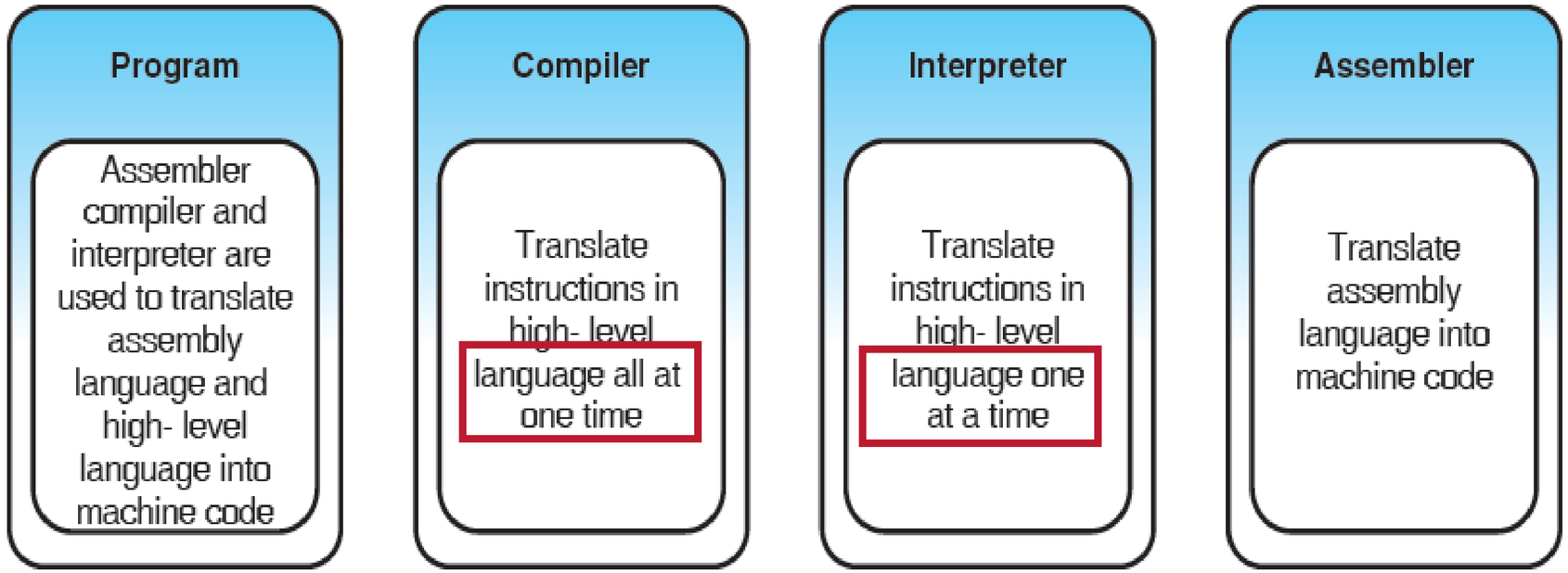



```
100100 010001    // Load
100110 010010    // Multiply
100010 010011    // Store
```

TYPES OF PROGRAMMING LANGUAGE



C++ RELATIONSHIP BETWEEN PROGRAMS, COMPILER, INTERPRETER AND ASSEMBLER



DIFFERENCES BETWEEN PROGRAMS AND PROGRAMMING

- A program is a set of instructions that tell the computer how to solve a problem or perform a task.
- Programming is the process of designing and writing computer programs.
- A program is like a recipe. It contains a list of ingredients (variables) and a list of directions (statements) that tell the computer what to do with the variables.
- A program can be as short as one line of code, or as long as several million lines of code.
- Computer programs guide the computer through orderly sets of actions specified by the computer programmers.
- The programmer must decide what the programs need to do, develop the logic of how to do it and write instructions for the computer in a programming language that the computer can translate into its own language and execute.

THE IMPORTANCE OF COMPUTER PROGRAMMING

- Able to perform difficult tasks without making human-type errors such as lack of focus, energy, attention or memory.
- Capable of performing extended tasks at greater serial speeds than conscious human thoughts.
- Human brain cannot be duplicated or ‘re-booted’ like computers, and has already achieved ‘optimization’ through design by evolution, making it difficult to upgrade.
- Human brain does not physically integrate well, externally or internally with current hardware and software.
- Non-existence of boredom in computers when performing repetitive tasks allows jobs to be done faster and more efficiently.

THE IMPORTANCE OF WRITING A GOOD PROGRAM

Names for variables, types and functions

- Variables and constants are storage locations in the computer's memory that match with associated names known as identifiers.
- The following are some standards that can be used when naming variables, constants, types and functions:
  1. Function names will start with a lowercase letter.
     Example: `double calculateBMI (double, double);`
  2. Variable names start with a lowercase letter and the length must not be more than 40 characters.
     Example: `double weight, height;`
  3. Constant names can be all in capital letters.
     Example: `const int MAX_SIZE =10;`

THE IMPORTANCE OF WRITING A GOOD PROGRAM

UNIVERSITI TEKNOLOGI MARA

Indentation styles and spacing

- In order to improve readability in programming, indentation can be used to format the program source code.
  - A text editor is used to create a program by following the rules or syntax of different programming languages.
- Spaces can also be added in between sentences to make programs much more readable.
- A new level of indentation should be used at every level of statement nesting in the program.
- The minimum number of spaces at each indentation should be at least three.
  - Many programmers use a tab mark (typically 8 spaces) which will be easier when indenting source code.

THE IMPORTANCE OF WRITING A GOOD PROGRAM

Style 1

```
if   (marks>50)
{
     return PASS;
}
else
{
     return FAIL;
}
```

Style 2

```
if   (marks>50)
{return PASS;}
else{return
FAIL;}
```

C++ PROGRAM STRUCTURE

EXAMPLE 1.1 shows the general form of a C++ program.

```
//introductory comments
//file name, programmer, date written or modified
//purpose of the program

//header files
#include<iostream>
using namespace std;

//main function
int main()
{
    variable declaration;
    constant declaration;

    executable statements;
    return 0;
}
```

EXAMPLE 1.2 shows a simple C++ program.

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

int main()
{
    //display greeting message

    cout <<"Hi. Good Morning.";

    return 0;
}
```

compiler directive/header file

main function

comments

statement

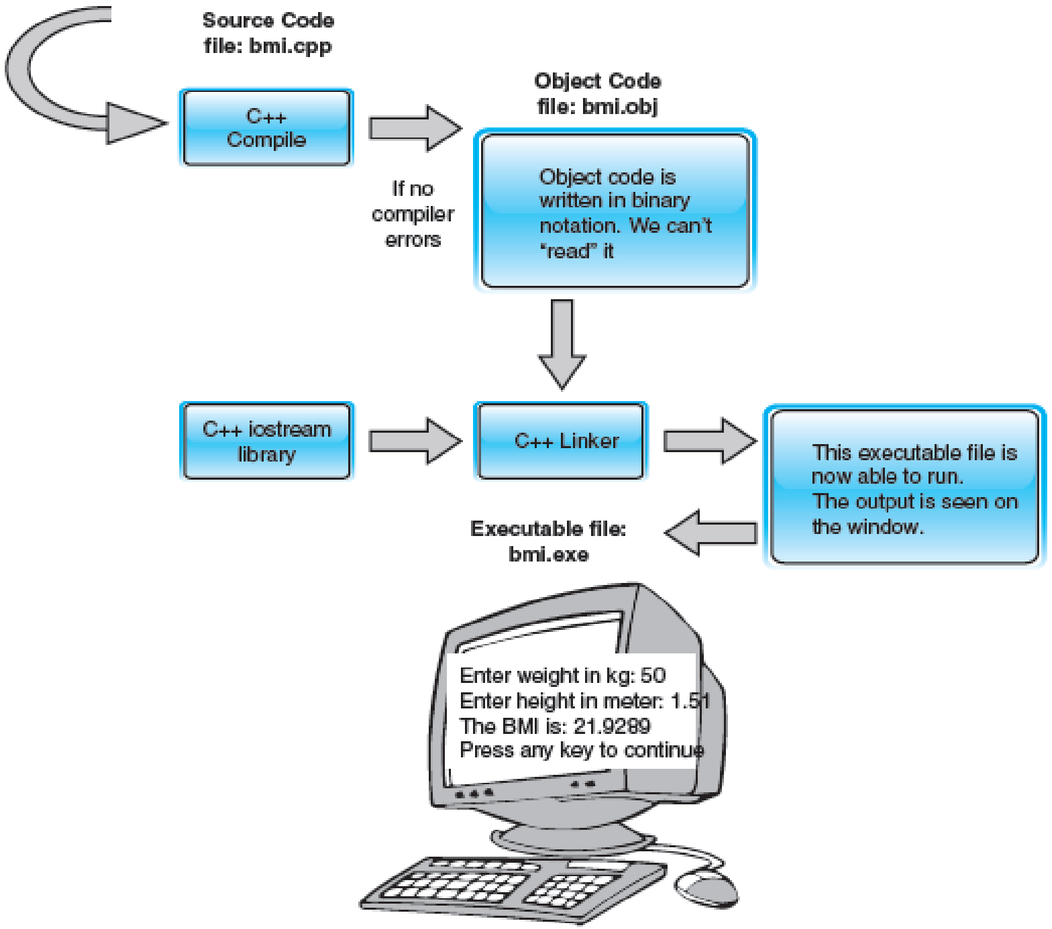
OUTPUT
Hi. Good Morning.

C++ PROGRAM STRUCTURE

EXAMPLE 1.3 shows a C++ program with the source code file name: bmi.cpp

```
/*                                  another way to represent comments
File Name              : bmi.cpp
Programmer             : Aiddamirah
Matrix No              : 2014253684
Topic                  : Example 1
Program purpose        : To calculate bmi for a user.
Date                   : 25 November 2014
*/

#include <iostream>                 preprocessor directives or header file
#include <math.h>                   predefined function
using namespace std;

int main()
{                                   begin block
    double weight, height, bmi;     variable declaration
    cout << "Enter weight in kg: ";
    cin >> weight;
    cout << "Enter height in metres: ";
    cin >> height;                  input, process and
    bmi = weight/pow(height,2);     output statements
    cout << "The BMI is : "<<bmi;
    return 0;
}                                   end block
```

THE FLOW OF PROGRAM EXECUTION



COMMENTS

- Important in any program as they are very useful for documentation but it is not compulsory to write comments in programs.
  - ✓ Comments are not source code, thus they will not be read by the compiler.
- Can be written in any way, according to the programmers' preferences
- Explain the purpose, parts of the program and keep notes regarding changes to the source code
- Store programmers' name for future reference.

Symbols used	Purpose	Example
// OR double slash	use for one line of comment only	//This is an example of a comment //This is the second line
/* */ OR Open comment: slash asterisk Close comment: asterisk slash	use for one or more than one line of comments in a group	/*This is an example of a comment. This is the second line*/

PRE-PROCESSOR DIRECTIVE

- Also known as a header file in C++.
- It is a line of code that begins with the # symbol.
- Header files are not executable code lines but instructions to the compiler.
- Header files are usually included at the top of the program before the main function.
- The most common and important header file is `#include<iostream>.`
- This header file tells the pre-processor to include the contents of the file `<iostream>` which are the input and output operations (such as printing to the screen).

FUNCTION

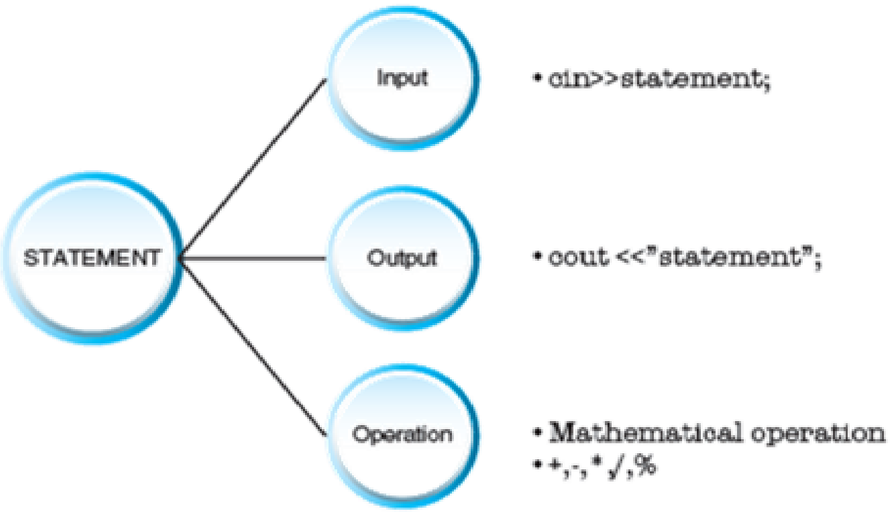
- Main Function
- Every C++ program contains one or more functions, but one of it must be named as **`main`** .
- A function is a block of code that carries out specific tasks.
- Generally, a main function starts with the function type **`void`** or **`int`** before it is followed by the word **`main`** and a pair of parentheses ().
- It is more advisable to write the function type int in the main function but the word **`return 0`** must be written at the end of the function before it is closed.

FUNCTION

- Braces
- Body of the main function which is enclosed in braces {}.
- Used to mark the beginning and end of blocks of code in any program.
- The open brace { is placed at the beginning of code after the main function and the close brace } is used to show the closing of code.
- The code after the } will not be read/evaluated by the compiler.

FUNCTION

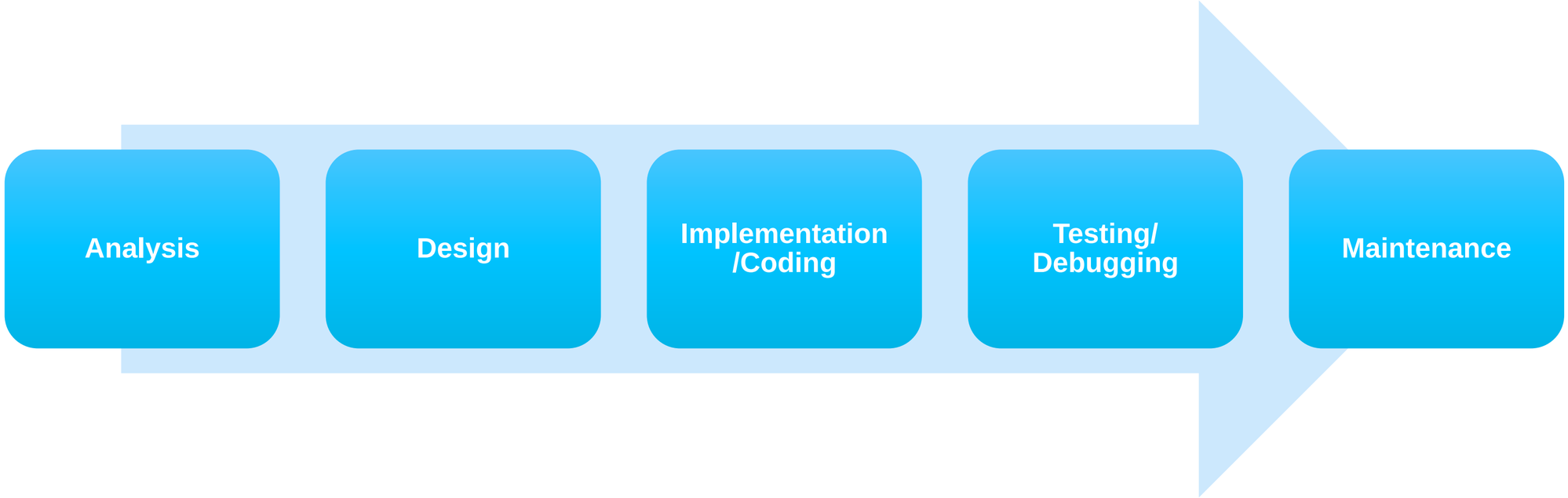
- Statement
- A function consists of a sequence of statements that perform the work of the function.
- Every statement in C++ ends with a semicolon (;).



PROGRAM DEVELOPMENT LIFE CYCLE

- The process of developing a program is called program development.
- The process associated with creating successful application programs is called the Program Development Life Cycle (PDLC).

PROGRAM DEVELOPMENT LIFE CYCLE



STEP 1 : ANALYSIS

- First step in the Program Development Life Cycle (PDLC).
- This process is done by reviewing the program specifications.
- Other criteria must also be identified, especially the data that will be used as input, the process involved and the output that will be produced.

The equation BMI = weight (kg) / (height (m) x height (m))

can be written in C++ as: BMI = weight/pow(height,2);

EXAMPLE 1.4 Problem: Calculate BMI for a user

Objectives: Calculate BMI for a user

Output: bmi

Input: weight, height (in metres)

Process: bmi=weight/pow(height,2)

STEP 1 : ANALYSIS

- Done by reviewing the program specifications.
  - Eliminate ambiguities in the problem statement.
  - Other criteria must be identified especially the data that will be used as input, process involve and output that will be produced.

- Indicating what the new system should do.
- In this step, the objectives, outputs, inputs, and processing requirements are determined.
- The program objectives are the problems that you are trying to solve.

STEP 1 : ANALYSIS

Example of Problem Solving Exercise

- Write a program to calculate the bmi of a user

Objective :

- To calculate the bmi of a user

Output :

- bmi

Input :

- weight , height

Process :

- Declare -> double bmi, weight, height
- Calculate -> bmi= weight/pow(height,2)

The equation
BMI = weight (kg) / (height (m) x height (m))

can be written in C++ as:
BMI = weight/pow(height,2);

STEP 2 : DESIGN

- A programmer needs to develop a series of steps with logical order, which when applied would produce the output of the problem.
- A solution is created using a structured programming technique known as algorithm, which consists of pseudocode and flowchart.
    - A procedure or formula for solving a problem.
    - A step-by-step problem solving process where the result is attained in a limited amount of time.

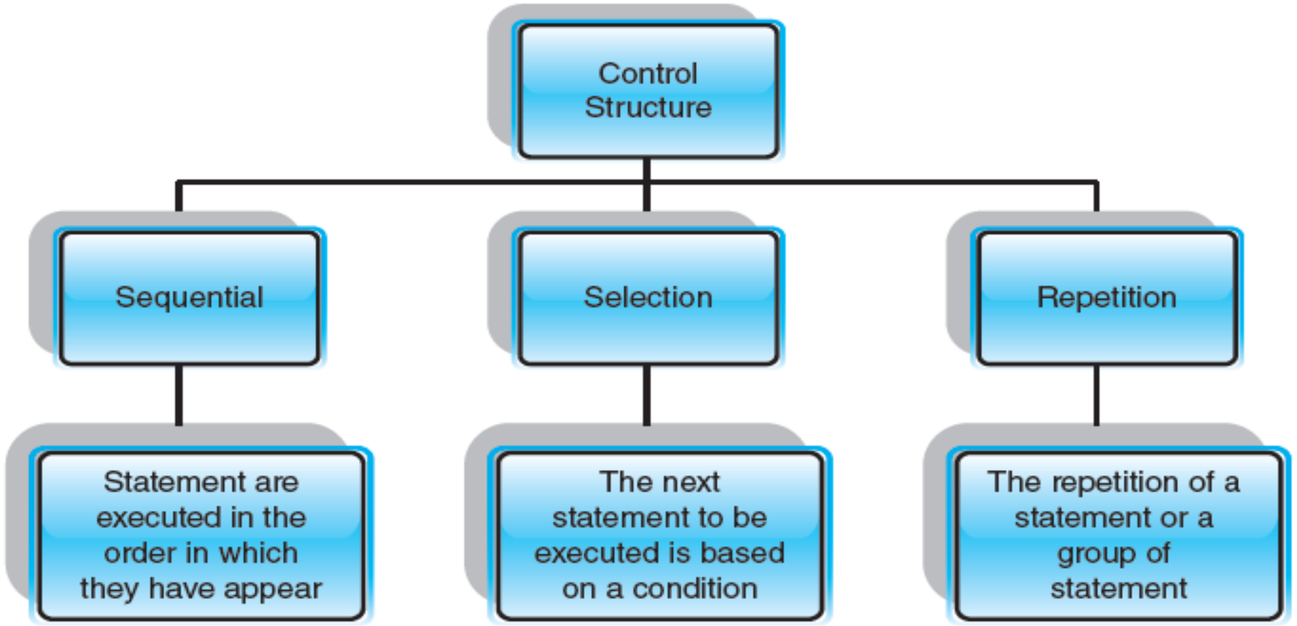
STEP 2 : DESIGN

Algorithm

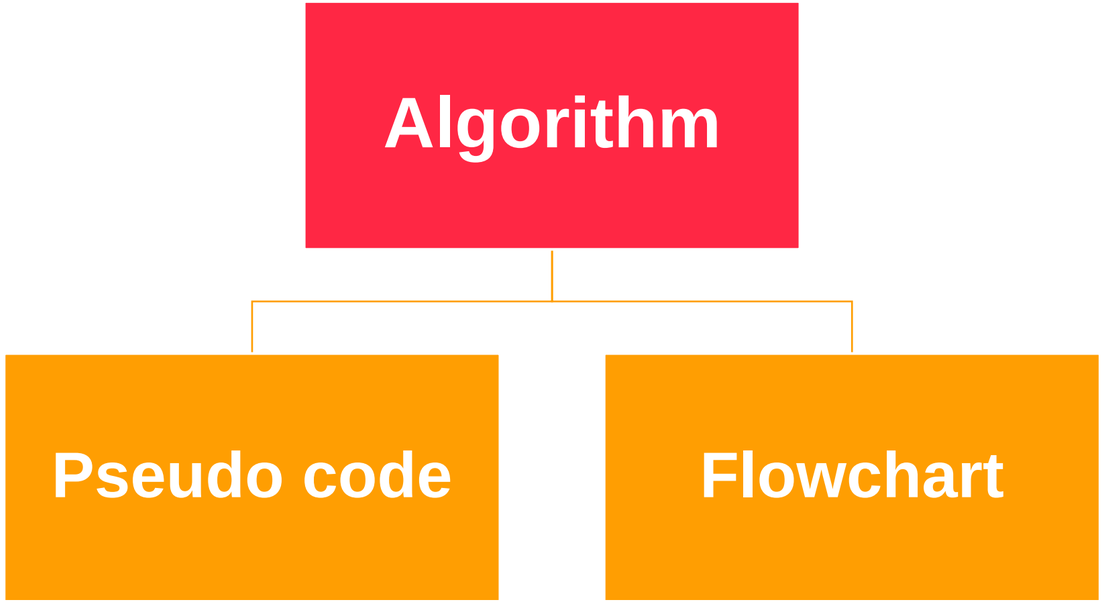
- A step-by-step problem solving process in which a solution is arrived at a finite amount of time.
- A sequence of a finite number of steps arranged in a specific logical order which when executed will produce the solution for that problem.
- An algorithm must satisfy some requirements which are:
  - Unambiguousness (clear)
  - Generality (unspecific)
  - Correctness
  - Finiteness (limitation)

STEP 2 : DESIGN

- Before an algorithm is created, the three types of control structure should be understood first.
- A control structure is a pattern to control the flow of a program module.



STEP 2 : DESIGN



STEP 2 : DESIGN – PSEUDO CODE

- A semiformal, English-like language with a limited vocabulary used to design and describe algorithms.
- Every statement in pseudocode involves keywords which define the process and operands.
- Each pseudocode statement should be written in a separate line.

STEP 2 : DESIGN – PSEUDO CODE

EXAMPLE 1.5 show pseudocode for sequential control structure.

Problem: Calculate BMI for a user

```
BEGIN
DECLARE double bmi, weight, height
READ weight, height
CALCULATE bmi=weight/pow(height,2)
DISPLAY bmi
END
```

EXAMPLE 1.6 show pseudocode for sequential control structure.

Problem: Find the area of a circle

```
BEGIN
DECLARE double radius, area; const double PI=3.142
READ radius
CALCULATE area=PI*pow(radius,2)
DISPLAY area
END
```

EXAMPLE 1.7 shows the pseudocode for selection control structure.

Problem: Identify whether a number is a positive or negative number

```
BEGIN
DECLARE int num
GET num
if (num>0)
    DISPLAY " positive number"
else
    DISPLAY "negative number"
END
```

EXAMPLE 1.8 shows the pseudocode for repetition control structure.

Problem: Get three numbers and find the total sum and average of the three numbers

```
BEGIN
DECLARE int no, sum, count; double avg,
INITIALIZE sum=0, count=0
while (count<3)
    GET no;
    CALCULATE sum=sum+no;
    CALCULATE count++;
COMPUTE avg=sum/count
PRINT sum, avg
END
```

STEP 2 : DESIGN – FLOW CHART

- A graphic presentation of the detailed logical sequence of steps needed to solve programming problems.
- Uses geometric symbols where different symbols are used to represent different actions such as start/stop, decision, input/output, processing and looping.
- Similar to pseudocode, keywords are written in uppercase, while variable and constant names as well as operations for the statements are written in lower case.

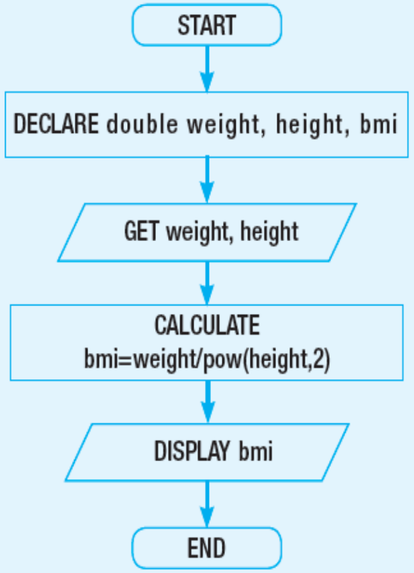
STEP 2 : DESIGN – FLOW CHART

Symbol	Name	Description
	Flowline	To connect symbols and indicate logic flow
OR	Terminal	Used to represent the beginning (Start) or the end (End) of a program
	Input/Output	Used for input and output operations, such as reading and printing.
	Processing	Used for arithmetic and data manipulation operations.
	Decision	Used for any logic or comparison operations. This symbol has one entry and two exit paths. The path chosen depends on whether the answer to the question is 'yes' or 'no'
	Connector	Used to join different flowlines
	Off-page Connector	Used to indicate that the flowchart continues to the next page

STEP 2 : DESIGN – FLOW CHART

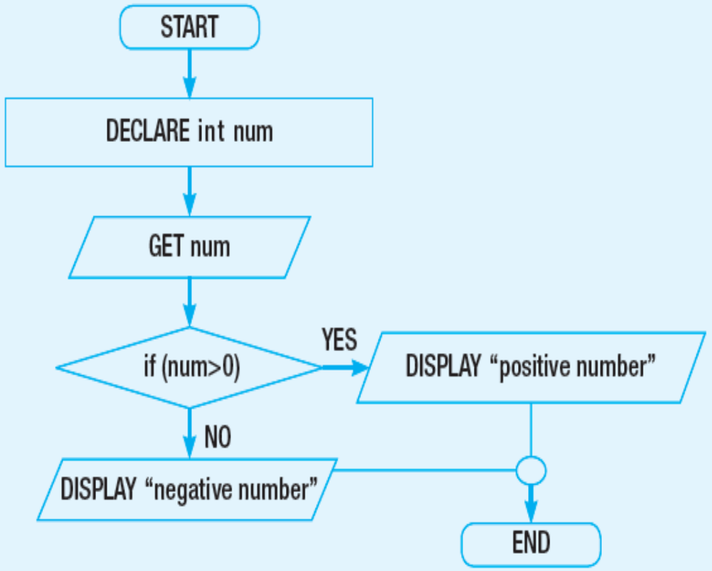
EXAMPLE 1.9 shows the flow chart for sequential control structure.

Problem: Calculate BMI for a user



EXAMPLE 1.10 shows the flow chart for selection control structure.

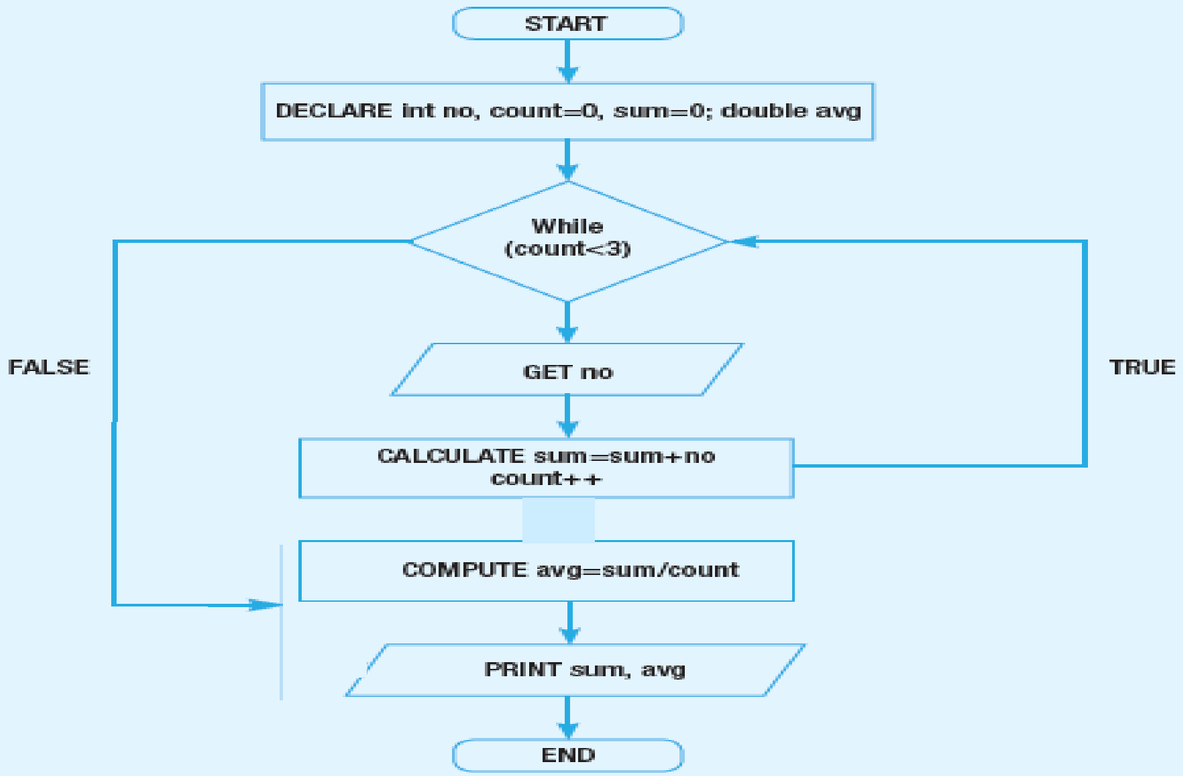
Problem: Identify whether a number is a positive or negative number



STEP 2 : DESIGN – FLOW CHART

EXAMPLE 1.11 shows the flow chart for repetition control structure.

Problem: Get three numbers and find the total sum and average of the three numbers

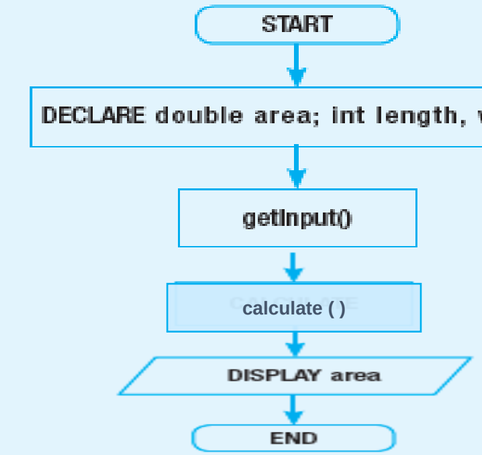


STEP 2 : DESIGN – FLOW CHART

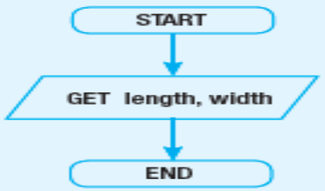
EXAMPLE 1.12 shows the flow chart for a function.

Problem: Calculate the area of a room by using 2 functions: getInput() and calculate()

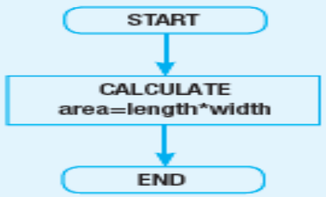
main()



getInput()

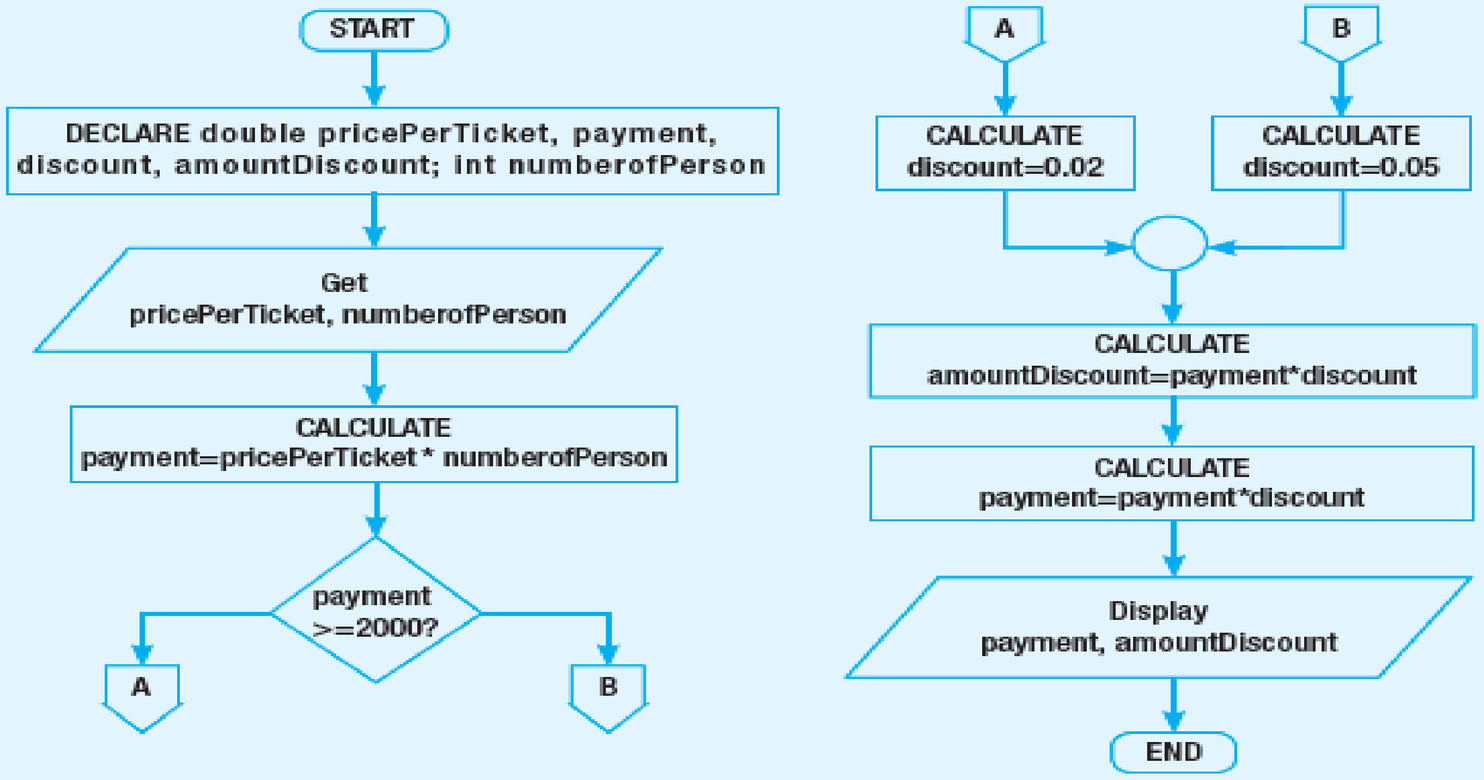


calculate()



STEP 2 : DESIGN – FLOW CHART

EXAMPLE 1.13 shows a flow chart which use off-page connectors.



STEP 3 : IMPLEMENTATION/CODING

- The *pseudocode* and *flow chart* which have been done in the design step will be converted into a program by using certain programming languages such as BASIC, JAVA, C or C++.
- This step solves the problem by enabling the user to start writing the programs.

STEP 3 : IMPLEMENTATION/CODING

- Coding is the actual process of creating a program in a programming language.
- The coded program is referred to as source code.
  - Must follow certain rules which are called syntax.
  - Must then be saved as a program which has the extension '.cpp'.
- To be executed, the program is converted by the computer into object code using a special program or translator such as a compiler or interpreter.

STEP 4 : TESTING/DEBUGGING

- The step for checking and verifying the correctness of the program.
- The process of making sure a program is free of errors or ‘bugs’ is called debugging.
- Preliminary debugging begins after the program has been entered into the computer system.

STEP 5 : MAINTENANCE

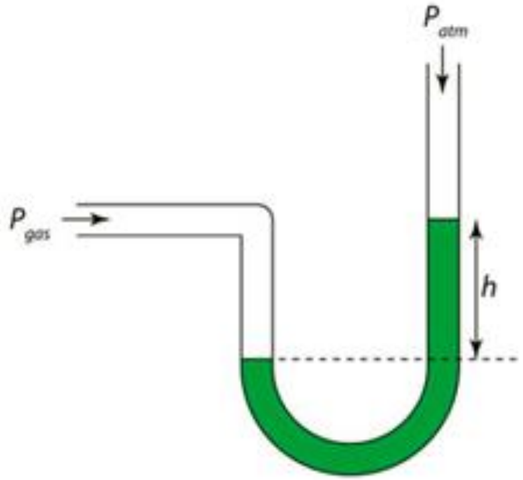
- Last step in the Program Development Life Cycle (PDLC).
- Essentially, every program, if it is to last a long time, requires ongoing maintenance.
- A process of updating software for any changes, corrections, additions, moving to a different computing platform and others so that it continues to be useful.
- A costly process.
- Can be very useful especially on extending the life of a program.

EXERCISE

QUESTION 1

- Write a program to calculate pressure using the formula given.

Manometer



$$P = P_{atm} + h\rho g$$

P_{gas} = *Pressure* (*Pa or N m*$^{-2}$)

P_{atm} = *Atmospheric Pressure* (*Pa or N m*$^{-2}$)

g = *gravitational field strength* (*N kg*$^{-1}$)

ANALYSIS

ANALYSIS
OUTPUT
INPUT
PROCESS

QUESTION 1

- Write a program to calculate pressure using the formula given.

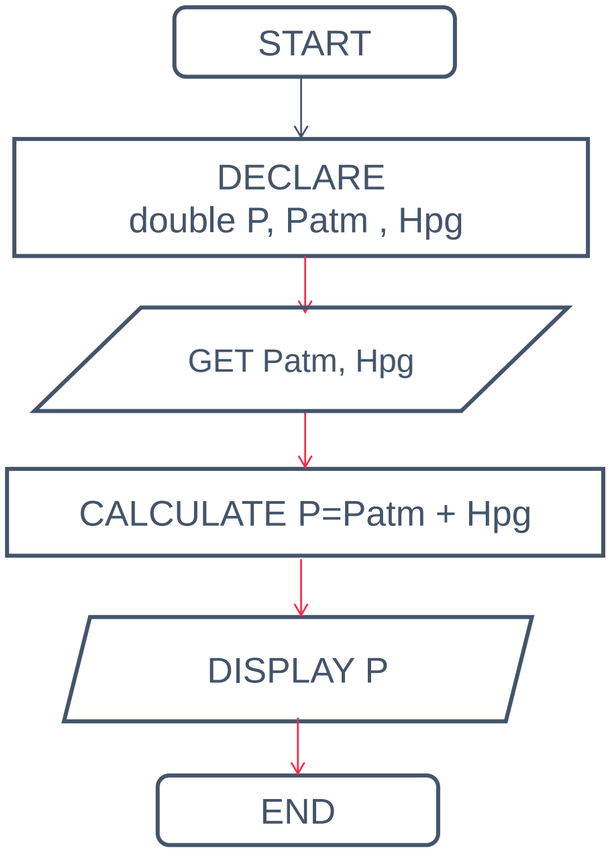
PSEUDOCODE

```
BEGIN

DECLARE double P, Patm , Hpg

READ/GET Patm, Hpg

COMPUTE/CALCULATE  P = Patm + Hpg

PRINT/DISPLAY P

END
```

QUESTION 1

- Write a program to calculate pressure using the formula given.

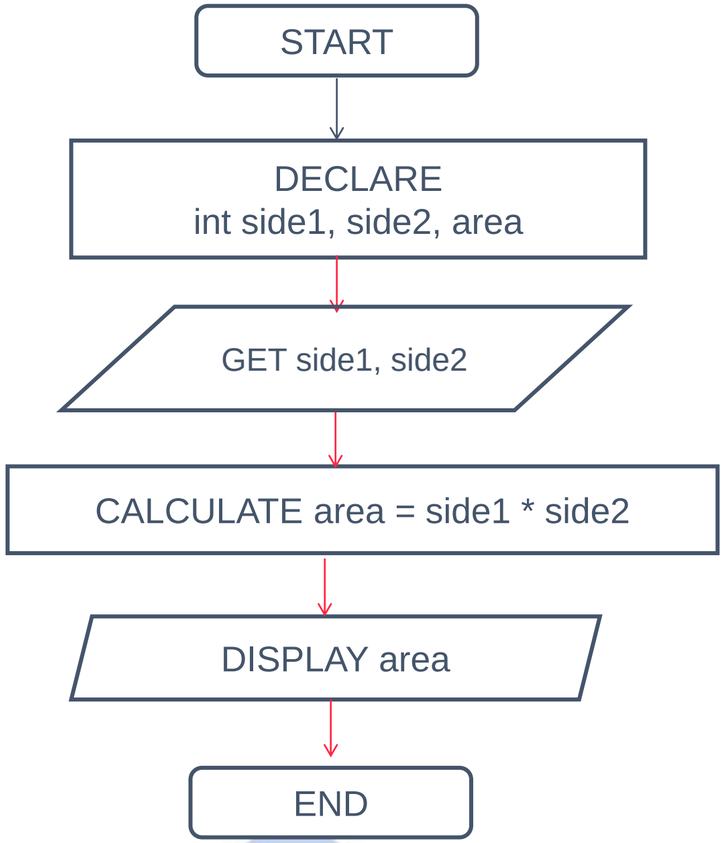
FLOW CHART



QUESTION 2

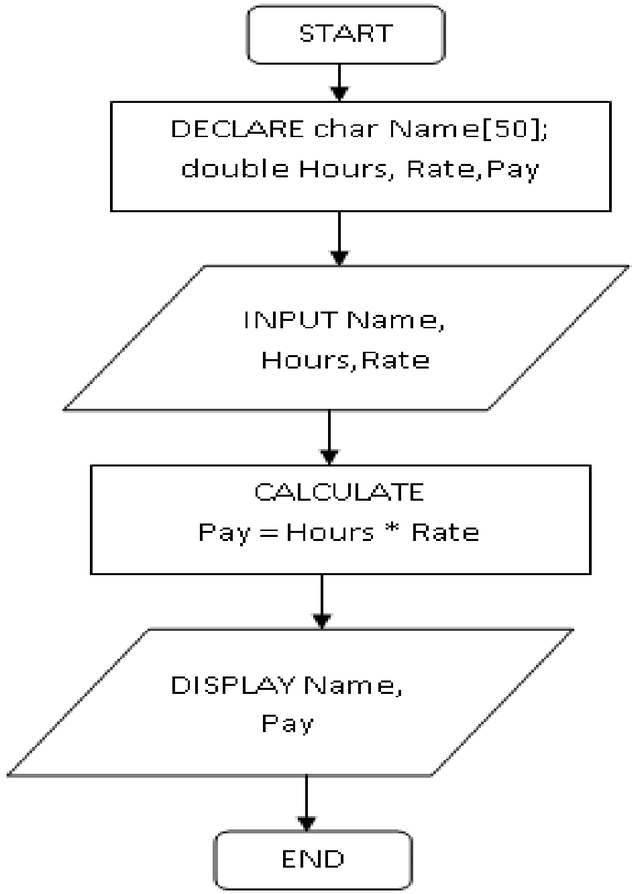
- Write the flowchart for the pseudocode given below:

```
BEGIN
DECLARE int side1,int side2,area
GET side1,side2
CALCULATE area = side1 * side2
DISPLAY area
END
```



QUESTION 3

- Write pseudocode for the flowchart given below:

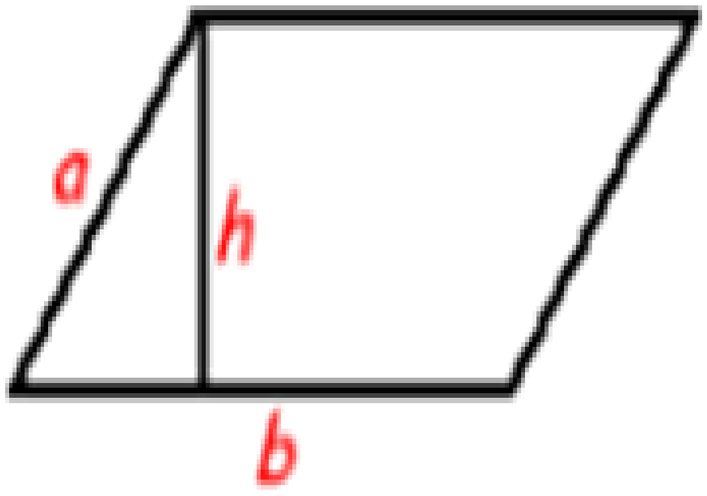



```
BEGIN
DECLARE char Name[50];
                double Hours, Rate, Pay
INPUT Name, Hours, Rate
CALCULATE   Pay = Hours * Rate
DISPLAY Name, Pay
END
```

QUESTION 4

- Draw a flowchart based on the pseudocode given below:

```
BEGIN
DECLARE double b,h ,A
GET  b,h
COMPUTE A=b*h
PRINT A,b,h
END
```

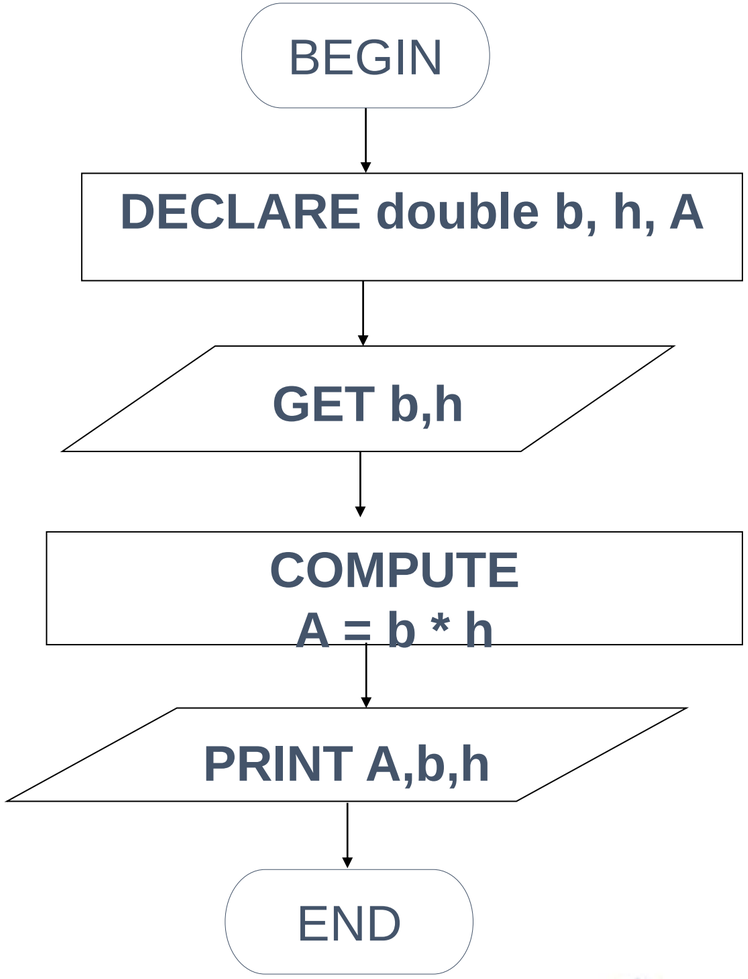


(Hint: A-Area of parallelogram, b-base,h-height)

Formula: A=b*h

QUESTION 4

```
BEGIN
DECLARE double b,h ,A
GET  b,h
COMPUTE A=b*h
PRINT A,b,h
END
```



QUESTION 5

A retail store grants its customers a maximum amount of credit. Each customer's available credit is his or her maximum amount of credit minus the amount of credit used. Write a pseudo code and flowchart algorithm for a program that asks for a customer's maximum amount of credit and amount of credit used. The program should then display the customer's available credit.

QUESTION 5

A retail store grants its customers a maximum amount of credit. Each customer's available credit is his or her maximum amount of credit minus the amount of credit used. Write a pseudo code and flowchart algorithm for a program that asks for a customer's maximum amount of credit and amount of credit used. The program should then display the customer's available credit.

Answer:
Program Analysis / Specification

OUTPUT	INPUT	PROCESS
balance	maxAmount, amountUsed	Balance=maxAmount-amountUsed

QUESTION 5

PSEUDO CODE

```
BEGIN

DECLARE double balance, maxAmount, amountUsed

READ maxAmount,amountUsed

CALCULATE balance = maxAmount – amountUsed

DISPLAY balance

END
```

FLOW CHART