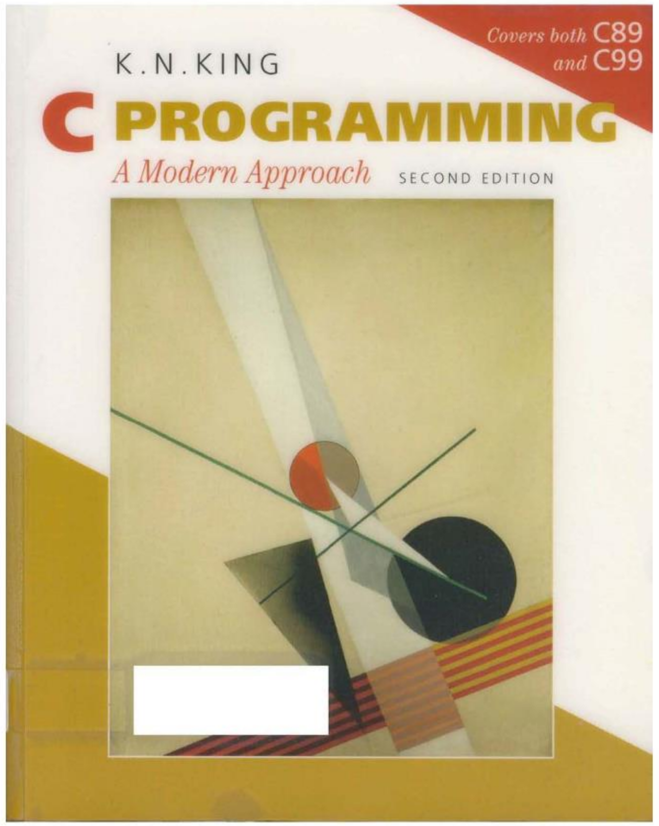
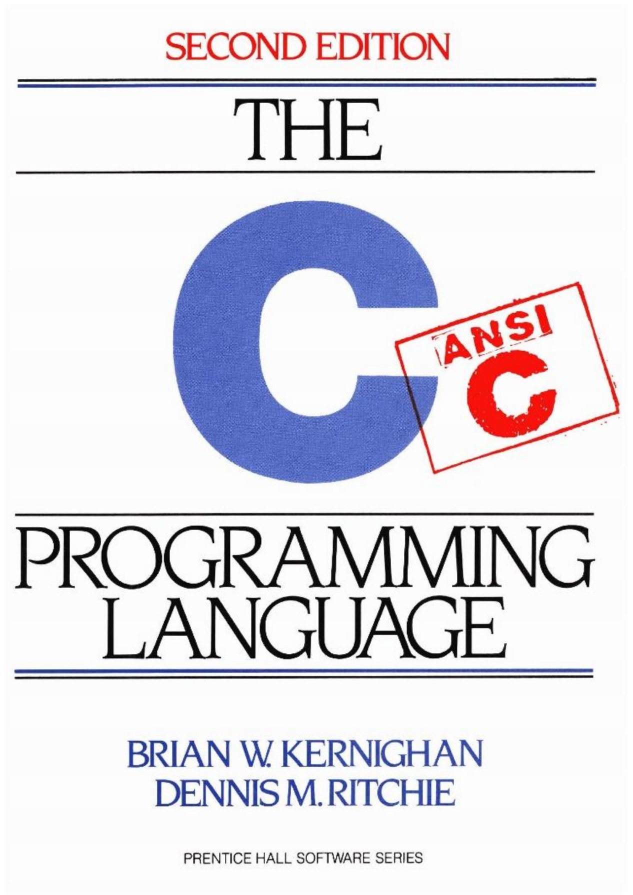
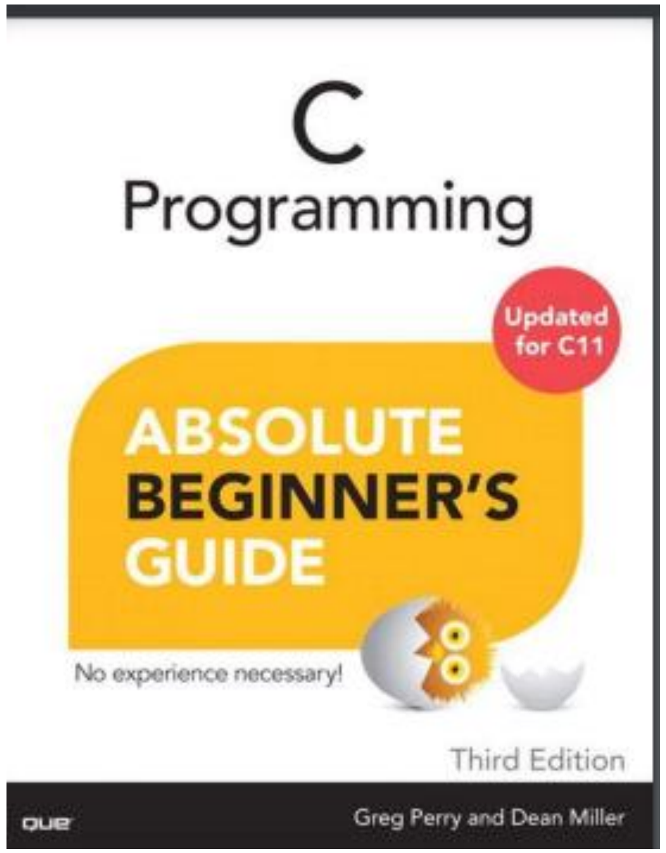
Fundamentals of Programming CCS1063/CSE1062

Professor Noel Fernando

Recommended books

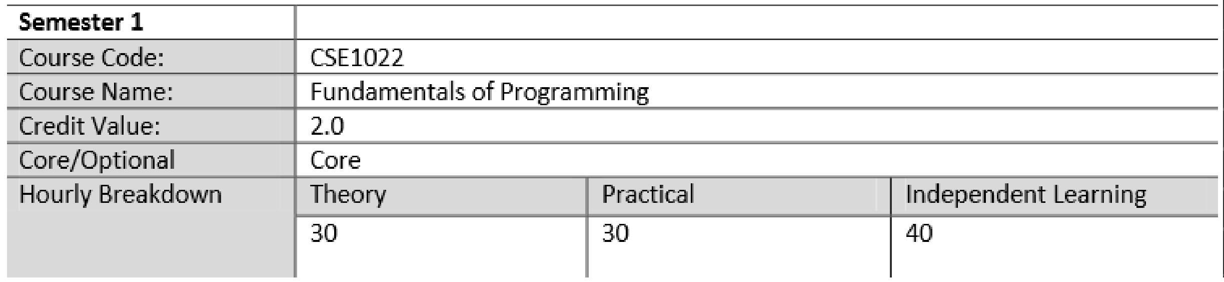




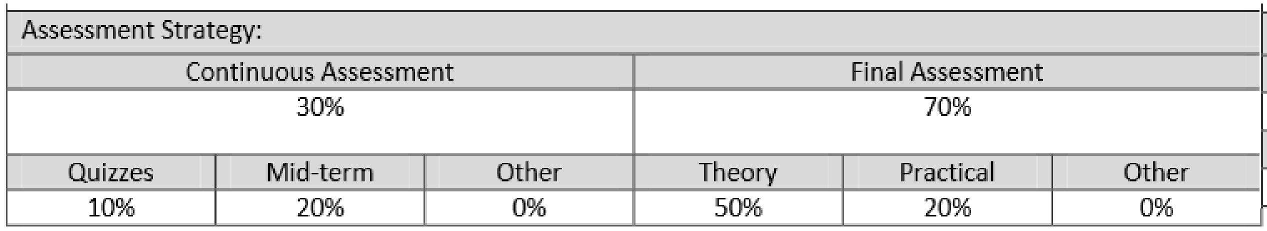


Assessment Strategy:

BCompHons (Soft Eng)

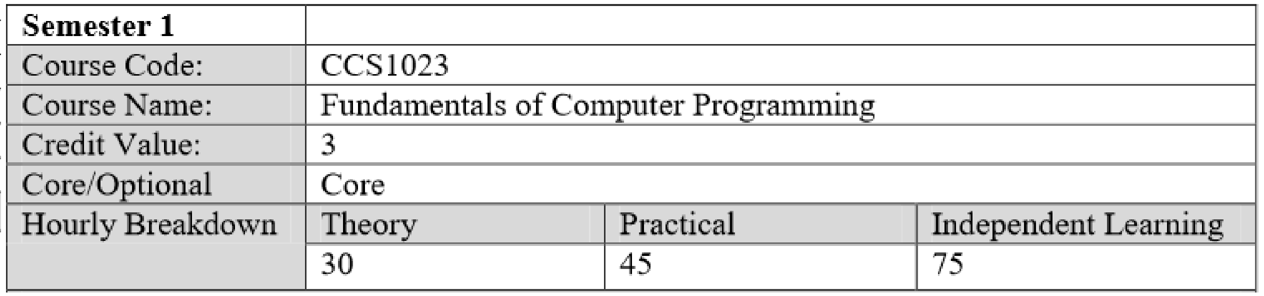


Semester 1			
Course Code:	CSE1022		
Course Name:	Fundamentals of Programming		
Credit Value:	2.0		
Core/Optional	Core		
Hourly Breakdown	Theory	Practical	Independent Learning
	30	30	40

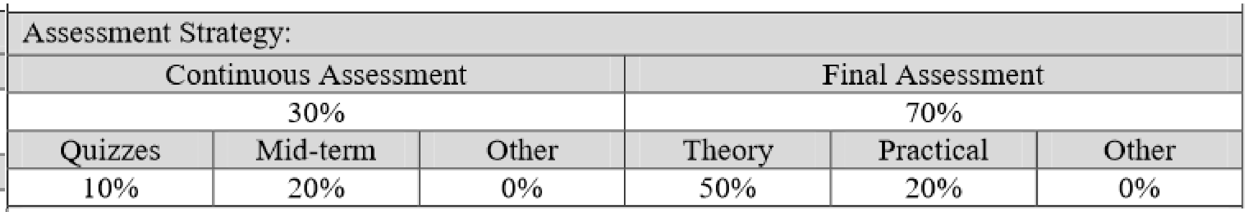


Assessment Strategy:					
Continuous Assessment			Final Assessment		
30%			70%		
Quizzes	Mid-term	Other	Theory	Practical	Other
10%	20%	0%	50%	20%	0%

- BCompHons (Comp Sc)



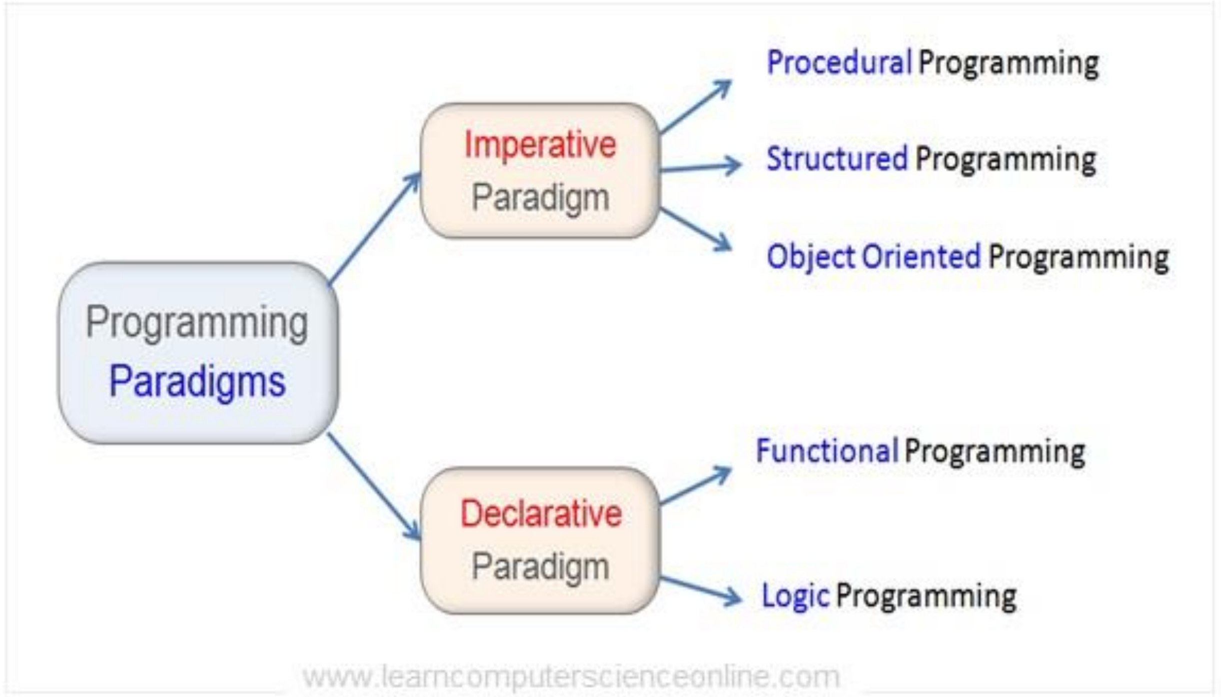
Semester 1			
Course Code:	CCS1023		
Course Name:	Fundamentals of Computer Programming		
Credit Value:	3		
Core/Optional	Core		
Hourly Breakdown	Theory	Practical	Independent Learning
	30	45	75



Assessment Strategy:					
Continuous Assessment			Final Assessment		
30%			70%		
Quizzes	Mid-term	Other	Theory	Practical	Other
10%	20%	0%	50%	20%	0%

Introduction to programming Languages

- What is a programming Language/ Computer Program/algorithm?
- Development of flow charts and pseudo codes
- Difference between the Syntax and Semantics
- What are the types of programming language ?
- Does the world need new languages?
- Introduction to Procedural and Object Oriented Programming (OOP)
- Programming fundamentals and problem solving



What is a programming Language?

- A programming Language is a computer language programmers use to develop software programs, scripts, or other set of instructions for computers to execute.

- a script is a program or sequence of instructions that is interpreted or carried out by another program
- e.g Perl, Rexx (on IBM mainframes), JavaScript, and Tcl/Tk (Tcl/Tk :It is a scripting language that aims at providing the ability for applications to communicate with each other)

What is a programming Language?

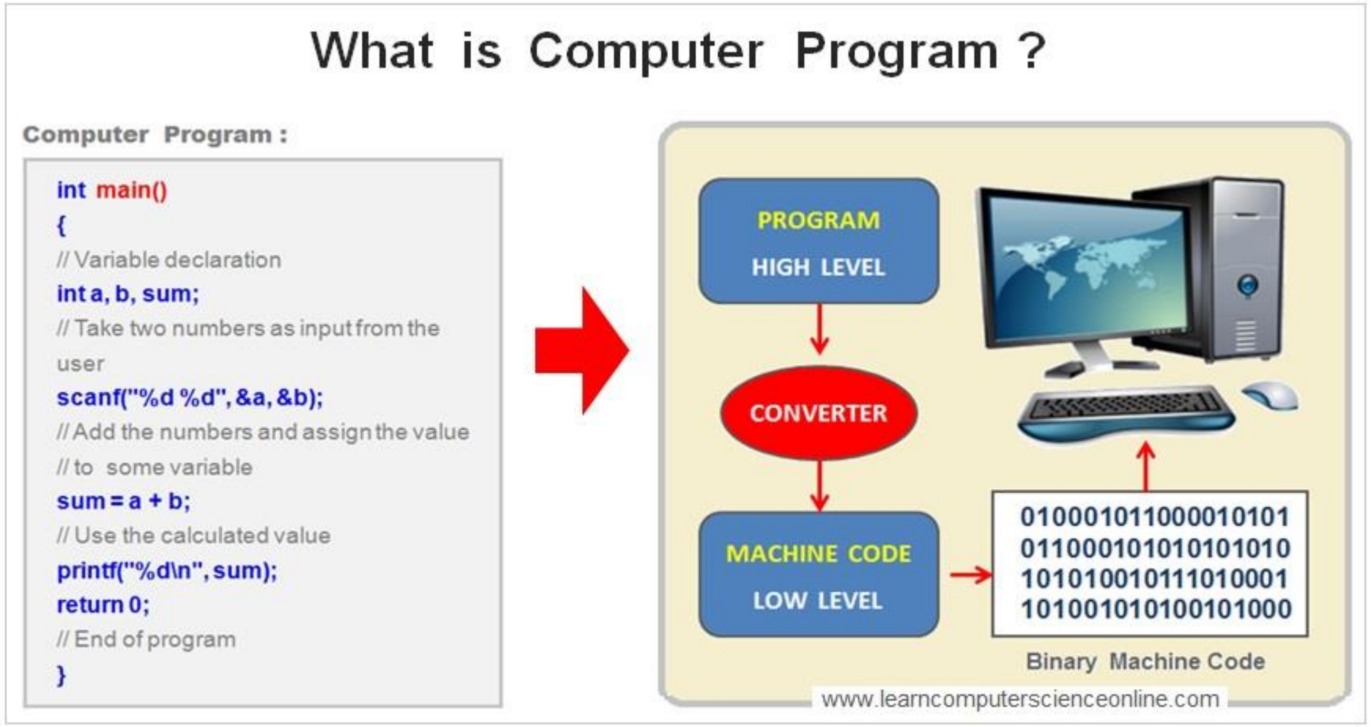
- A programming language is a set of rules that provides a way of telling a computer what operation to perform.
- A programming language is a set of rules for communication an algorithm
- It provides a linguistic framework for describing computations.

What Is a Computer Program?

A set of instructions that directs a computer to perform the tasks necessary to process data into information.

Or

A program is simply a sequence of instructions telling a computer what to do.



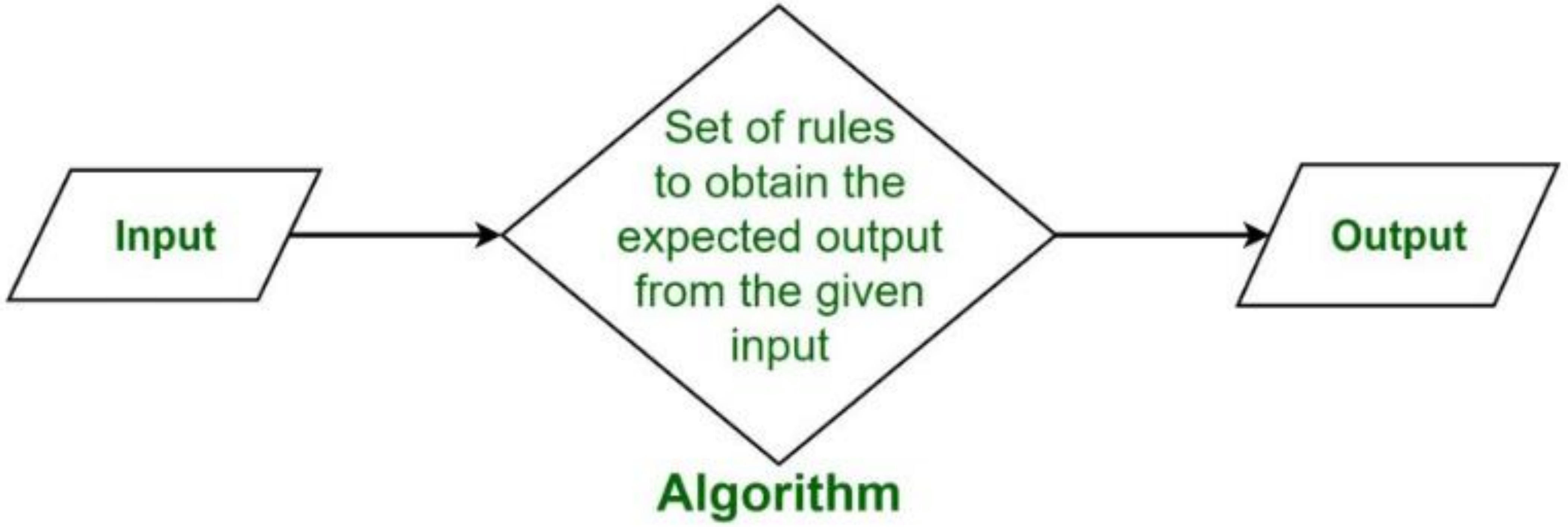
```
int main()
{
// Variable declaration
int a, b, sum;
// Take two numbers as input from the
user
scanf("%d %d", &a, &b);
// Add the numbers and assign the value
// to  some variable
sum = a + b;
// Use the calculated value
printf("%d\n", sum);
return 0;
// End of program
}
```

A Few Definitions

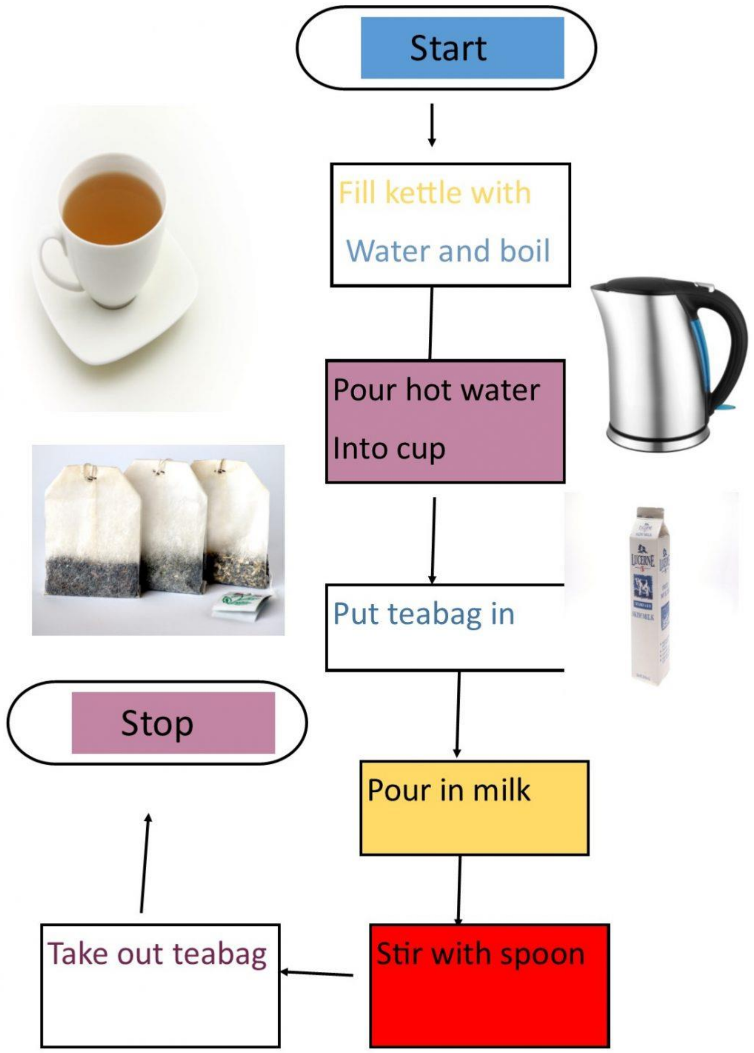
- **Programming** is also known as **software development**.
- **Programming** is problem-solving procedures which follows a six-step process in developing instructions or programs.
- **Programming** is the creation of lists of instructions for a computer.

What is an Algorithm?

What is Algorithm?

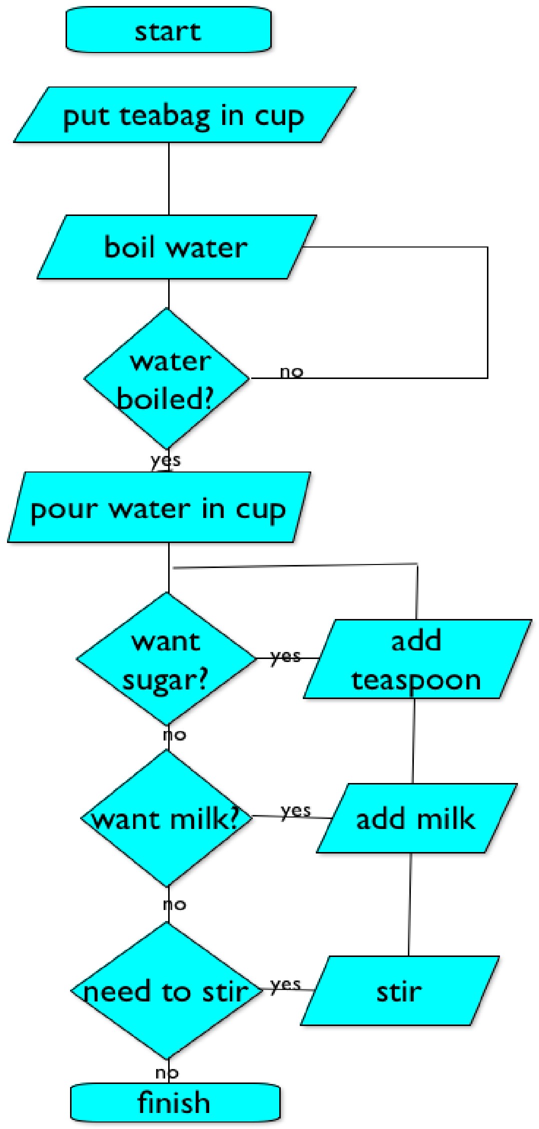

Input
Set of rules to obtain the expected output from the given input
Algorithm
Output

An Algorithm - For Making A Cup Of Tea



An Algorithm For Making A Cup Of Tea

Flowchart



Pseudocode

```
PROGRAM make tea

put teabag in cup

WHILE (water not boiled)
    boil water
    ENDWHILE
pour water in cup

WHILE (sugar needed)
    add sugar
ENDWHILE

WHILE (milk needed)
    add milk
ENDWHILE

WHILE (need to stir)
    stir tea
ENDWHILE

Server and finish
```

What is an Algorithm?

- An algorithm is a mathematical process to solve a problem using a finite number of steps.
- Or
- It’s an organized logical sequence of the actions or the approach towards a particular problem. A programmer implements an algorithm to solve a problem. Algorithms are expressed using natural verbal but somewhat technical annotations.

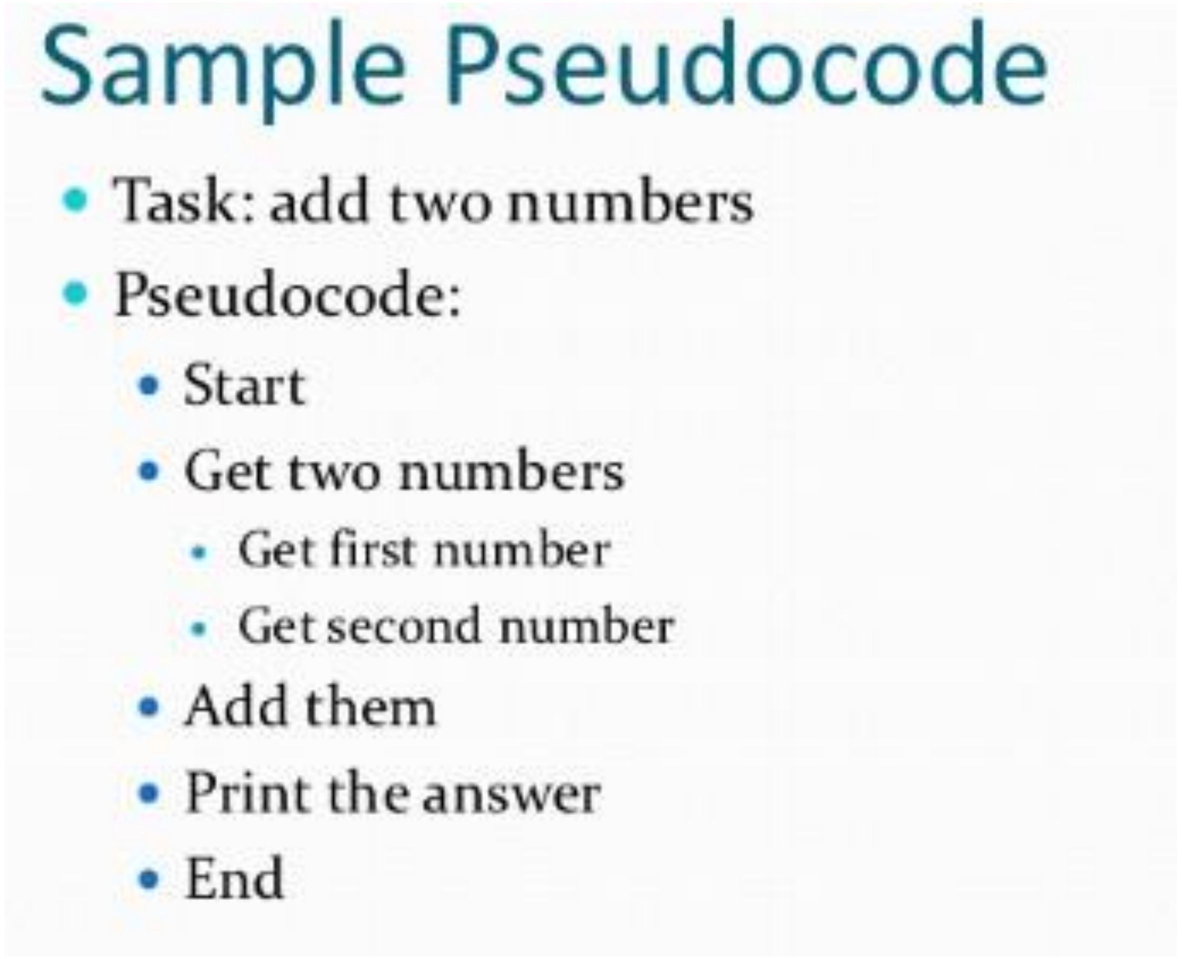
Algorithm

- Algorithm is required to be:
- Well defined – it is clearly and unambiguously specified
- Effective : its steps are executable
- Finite : it terminates after a limited number of steps

Must produce a correct solution 100% of the time

Pseudo code

- **Pseudo code:** It’s simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming language and thus can’t be compiled or interpreted by the computer.



Sample Pseudocode

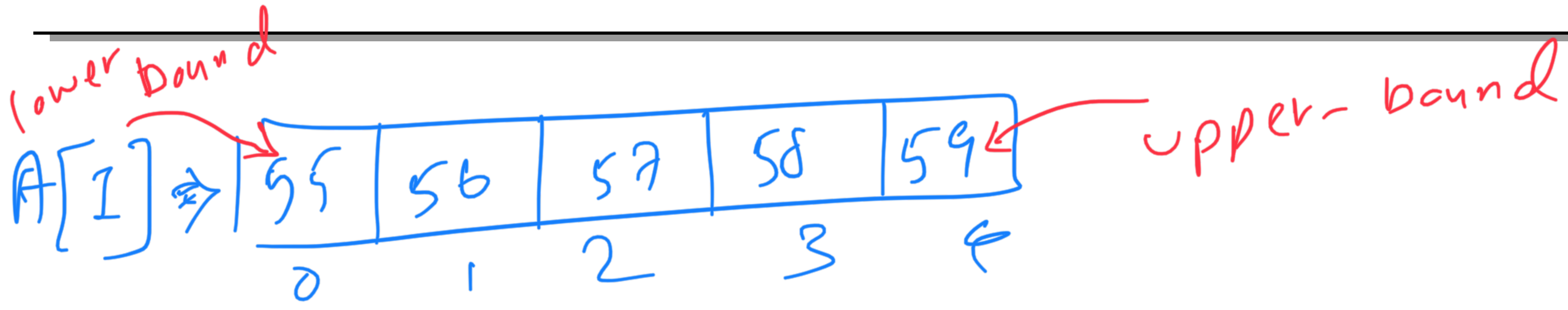
- Task: add two numbers
- Pseudocode:
  - Start
  - Get two numbers
    - Get first number
    - Get second number
  - Add them
  - Print the answer
  - End

Pseudocodes

- We establish the standard for the pseudo-codes, which we will follow in this class:
- Statements are written in simple English
- Each instruction is written on a separate line
- Each set of instructions is written from top to bottom, with only one entry and one exit
- Group of statements may be formed into modules and that given a name

Algorithm for array traversal

```
Step 1: [INITIALIZATION] SET I = lower_bound
Step 2: Repeat Steps 3 to 4 while I <= upper_bound
Step 3:         Apply Process to A[I]
Step 4:         SET  I = I + 1
        [END OF LOOP]
Step 5: EXIT
```



Semantics and Syntax

Just like communication with English:

- The meaning (**semantics**) of the sentence, as well as
- The grammar (**syntax**) of the sentence, so that others can understand, e.g.,

✓ ✓ √ he has	X he have ✓ ✗
✓ ✓ √ we are	X we is ✓ ✗

Semantics and Syntax

- Different languages have different syntax; this applies to both spoken languages and computer programming languages
- To make sure the Python shell can understand your Python program, your program has to follow the **syntax** of the Python language

Difference Between Syntax and Semantics

Syntax:	Semantics:
It refers to the rules and regulations for writing any statement in a programming language like C/C++.	It refers to the meaning associated with the statement in a programming language.
It does not have to do anything with the meaning of the statement.	It is all about the meaning of the statement which interprets the program easily.
A statement is syntactically valid if it follows all the rules.	Errors are handled at runtime.
It is related to the grammar and structure of the language.	

What are the types of programming language ?

Programming Languages

- C is one example of a programming language
- Others includes:
- C ++
- Python
- Fortran,
- Java,
- Perl,
- MATLAB, etc.

Programming Languages

Computing will influence deeply many areas of thinking, making and doing. Thus data, syntax and semantics are three ideas which are fundamental in Science and other intellectual fields.

The concepts of data, syntax and semantics is through the task of modelling and analyzing programming languages

Programming languages abound in Computer Science and range from large general-purpose languages to the small input languages of application packages

Who must use language definitions?

Other language designers

Implementors ← software Developers

Programmers (the users of the language)

Programming languages

Data : the information to be transformed by the program

Syntax :the text of the program (**Syntax** is the grammar, structure, rules or order of the elements in a language)

Semantics : the behavior of the program (Semantics are the meaning of various elements in the program)

Programming Concepts and paradigms

Millions of programing languages have been invented, and several thousands of them use.

Compared to the natural languages that developed and evolved independently, programming languages are far more similar to each other.

Does the world need new languages?

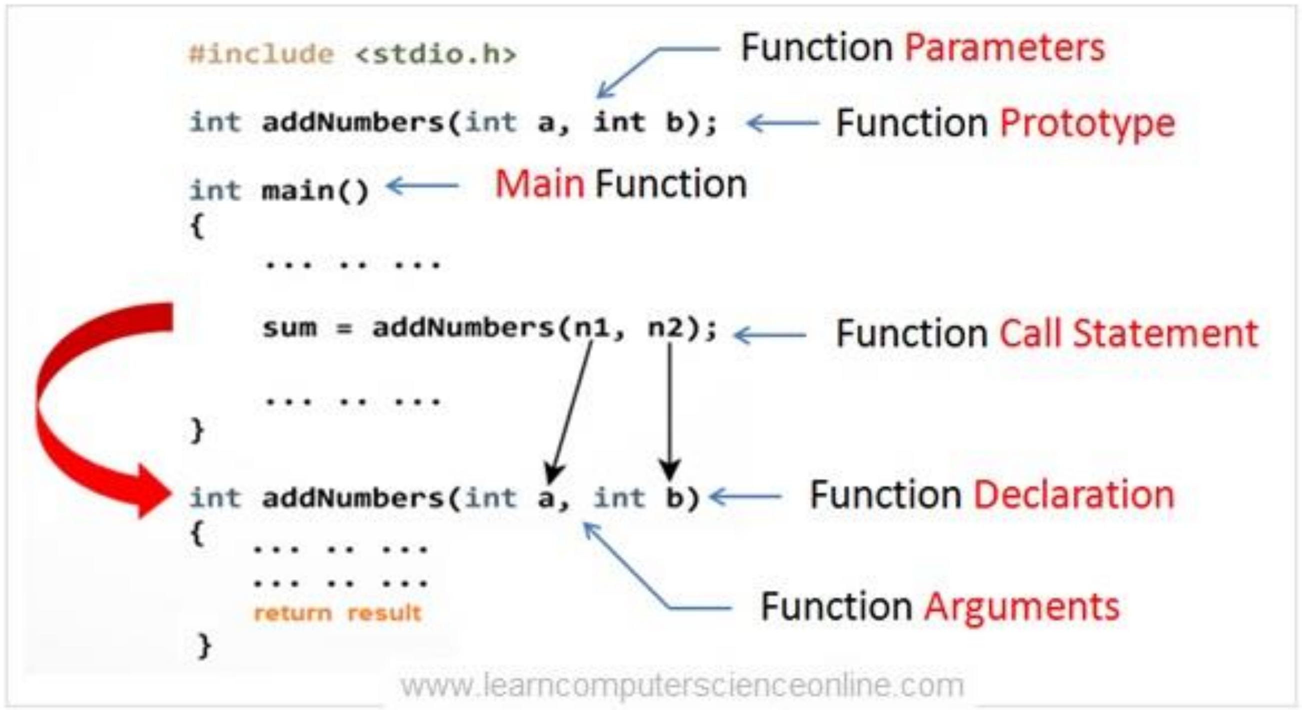
- New programming languages are needed in the modern world for several reasons:
- Evolving Technology:
  - As technology advances, new programming languages are often developed to cater to specific needs and take advantage of emerging paradigms, frameworks, or hardware architectures
  - For example, Swift and Kotlin were created to address the requirements of mobile app development, while languages like Rust focus on system-level programming with a focus on memory safety and concurrency.
- Increased Productivity:
  - New programming languages can offer improved productivity and developer efficiency.
  - They may provide higher-level abstractions, better tooling, or a more expressive syntax, making it easier and faster for developers to write code and build software applications.

Does the world need new languages?

- Domain-Specific Languages (DSLs):
  - Sometimes, specific domains or industries require specialized languages to model and solve their unique problems effectively.
  - DSLs are tailored to a specific domain, enabling concise and expressive code that closely matches the problem domain's concepts and requirements.
- Language-Specific Innovations:
  - New languages often introduce innovative features or programming paradigms that improve software development practices.
  - For example, functional programming languages like Haskell and Scala emphasize immutability and higher-order functions

Differences between Procedural and Object Oriented Programming?

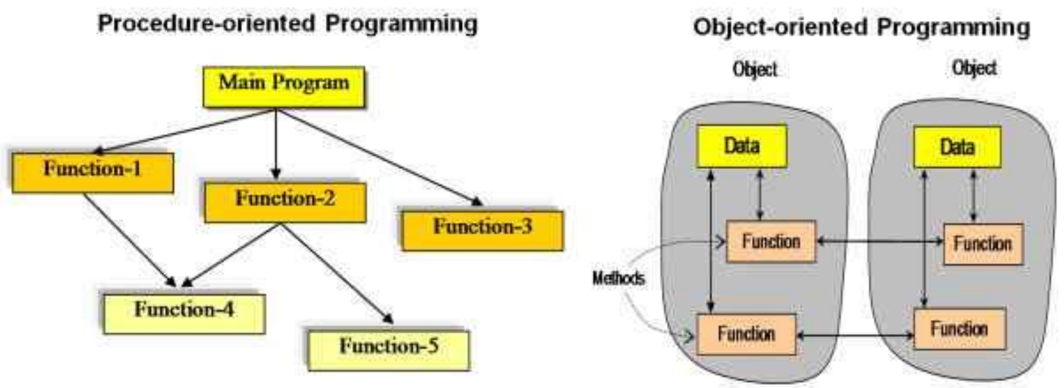
- **Procedural Programming**
- Procedural Programming is derived from structured programming, based upon the concept of calling procedure.
- Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out.
- The aim of the procedural language is dividing the large program into smaller program called procedures(Subprograms, Subroutines methods or functions)
- E.g. BASIC, FORTRAN, ALGOL, C, COBOL, and Pascal

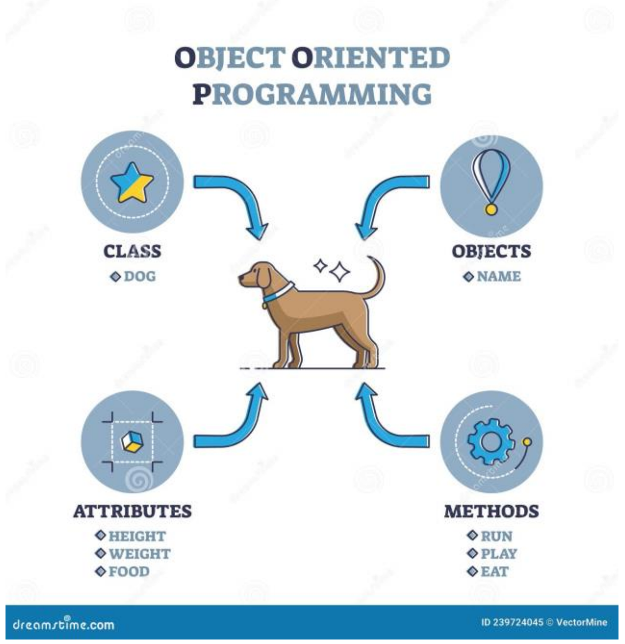


```
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
   ... .. ...
   ... .. ...
   return result
}
```

Object Oriented Programming?

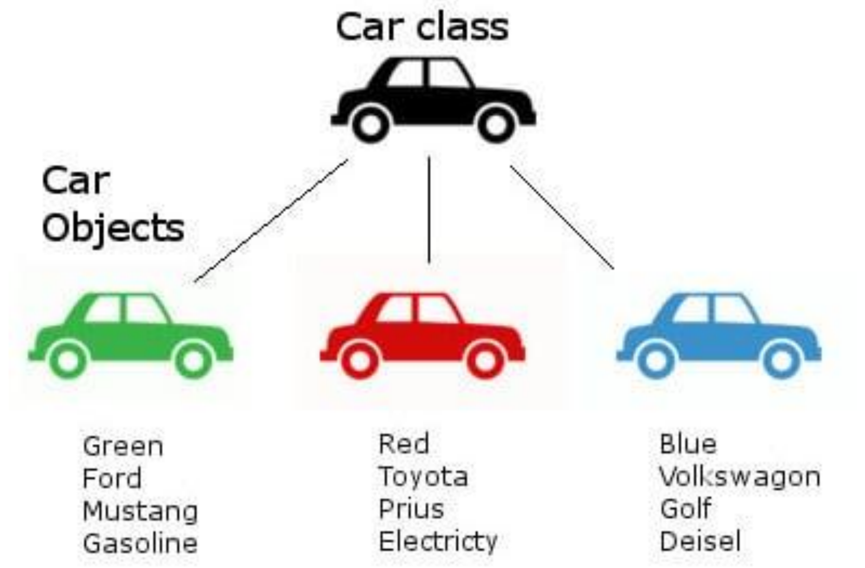
- Uses classes and objects to create models based on the real world environment.
- These objects contain data in the form of attributes and program codes in the form of methods
- The computer programs are designed by using the concept of objects that can interact with the real world entities.
- C#, Python, C++, Java, PHP, Scala, Perl, etc.





What are the features of object oriented programming

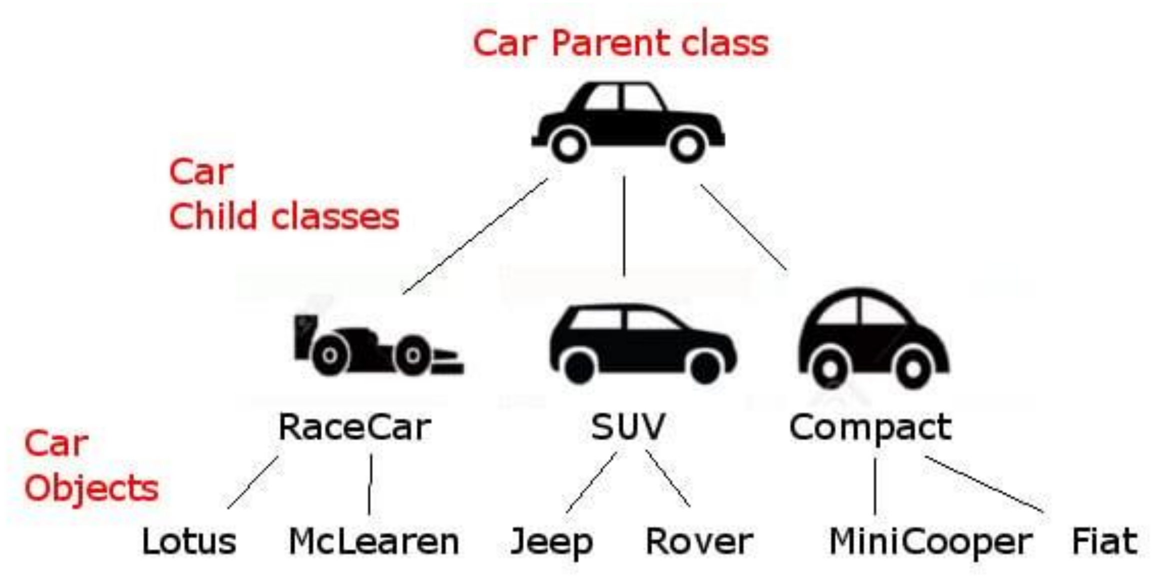
- C++ is an object-oriented programming language.
- Everything in C++ is associated with classes and objects, along with its attributes and methods
- For example: in real life, a car is an **object**.
- The car has **attributes**, such as weight and color, and **methods(behavior)**, such as drive and brake.



The car has **attributes/state**, such as weight and color, and **methods/behavior**, such as drive and brake.

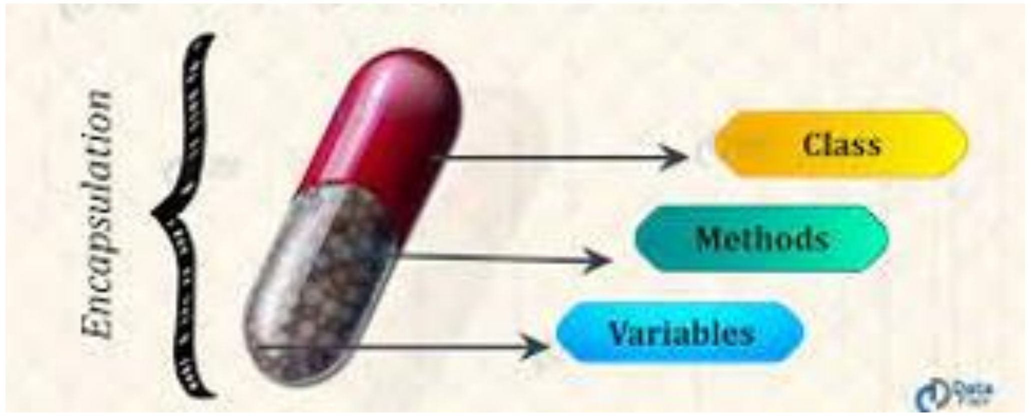
What are the features of object oriented programming?

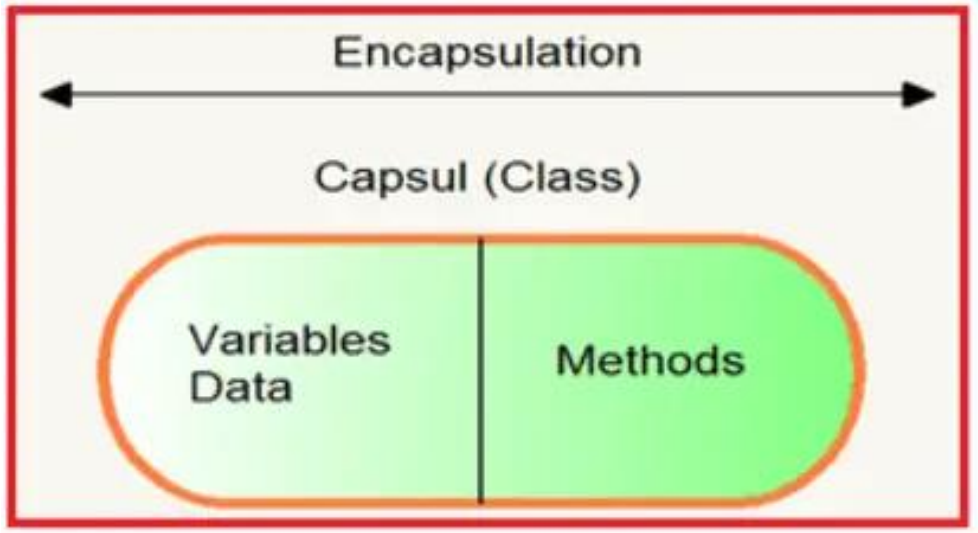
- Classes and Objects: OOP revolves around the concept of classes and objects.
- A class is a blueprint or template that defines the structure and behavior of objects.
- Objects are instances of classes that hold their own state (data/attributes) and behavior (methods).



What are the features of object oriented programming?

- Encapsulation: Encapsulation refers to the bundling of data and methods within an object.
- It allows for the hiding of internal details and provides controlled access to the object's properties and methods.
- Encapsulation enhances data protection and helps manage complexity by keeping related code and data together.





Polymorphism

- Polymorphism in C++ means, the same entity (function or object) behaves differently in different scenarios.
- **example**:
- The " +" operator in c++ can perform two specific functions at two different scenarios

- when the "+" operator is used in numbers, it performs addition.

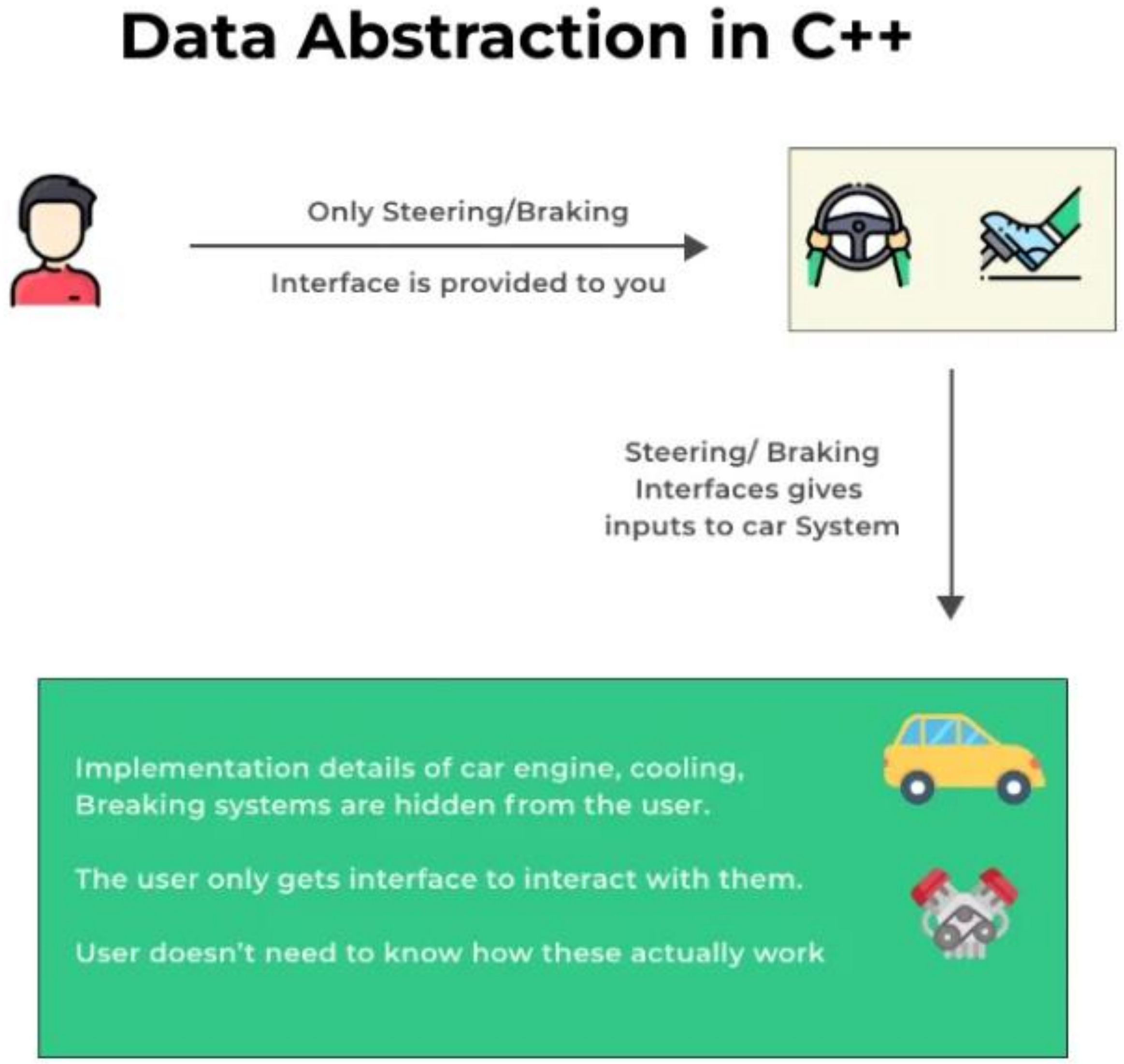
```
int a = 6;
int b = 6;
int sum = a + b; // sum =12
```

- And the same "+" operator is used in the string, it performs concatenation

```
string firstName = "Great ";
 string lastName = "Learning";
// name = "Great Learning "
string name = firstName + lastName;
```

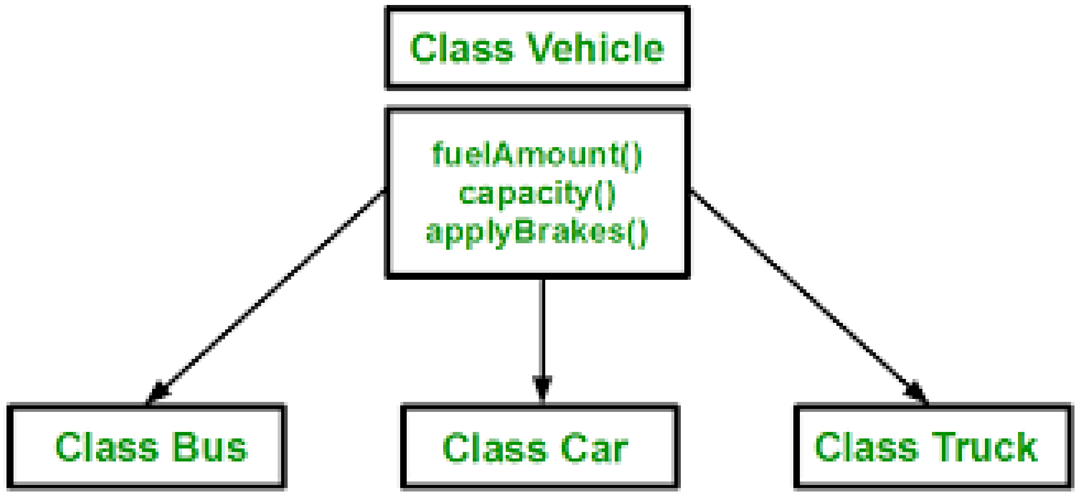
Abstraction

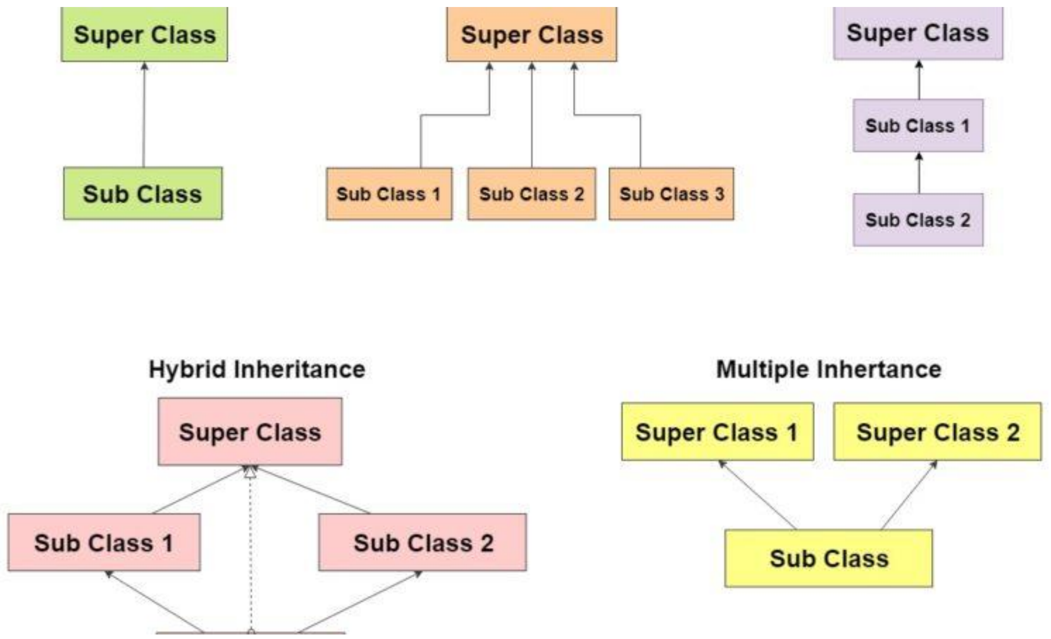
- Abstraction is the process of only showing the necessary details to the user and hiding the other details in the background.
- Control and data are the two types of abstraction in C++.
- Abstraction in C++ is achieved through classes, header files, and access specifiers (public, private, protected).


Data Abstraction in C++
Only Steering/Braking
Interface is provided to you
Steering/ Braking Interfaces gives inputs to car System
Implementation details of car engine, cooling, Breaking systems are hidden from the user.
The user only gets interface to interact with them.
User doesn't need to know how these actually work

Inheritance

- Inheritance allows the creation of new classes (child classes or derived classes) based on existing classes (parent classes or base classes).
- Child classes inherit the properties and methods of their parent classes, allowing code reuse and promoting the concept of "is-a" relationships.
- Inheritance facilitates code organization, extensibility, and modularity.





The difference between object oriented programming and procedural programming languages

	Object-oriented programming (OOP)	procedural programming
Focus	OOP focuses on objects that encapsulate data and behavior. It emphasizes the organization of code around objects, which interact with each other through methods and messages.	Focuses on procedures or routines that perform specific tasks. It follows a top-down approach, where the program's logic flows from one procedure to another, manipulating data as needed
Data and Behavior:	OOP combines data and behavior within objects. Objects are instances of classes, which define the structure (data) and behavior (methods) of objects.	Data and behavior are typically separate in procedural languages. Data is stored in variables or data structures, and functions or procedures operate on this data.

The difference between object oriented programming and procedural programming languages

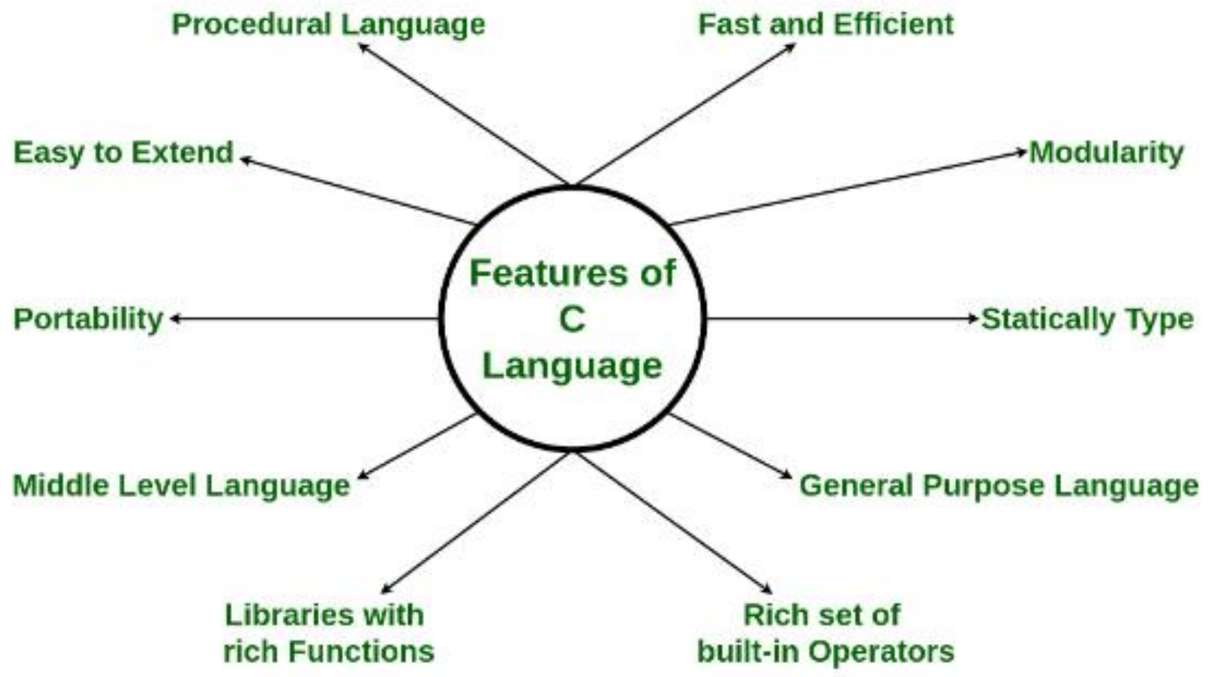
	Object-oriented programming (OOP)	procedural programming
Code Reusability:	OOP promotes code reuse through the concept of inheritance and object composition. Inheritance allows objects to inherit properties and methods from parent classes, while composition allows objects to be built by combining other objects.	Code reuse is typically achieved through functions or procedures. Functions can be called multiple times from different parts of the program to perform specific tasks.
Encapsulation and Modularity:	OOP provides encapsulation, where data and methods are bundled together within objects. Objects encapsulate their internal state, and access to it is controlled through methods, providing better data protection and modularity.	Procedural languages usually lack strong encapsulation mechanisms. Data and functions are often accessible from anywhere in the program, making it harder to control and manage dependencies.
Polymorphism	OP inherently supports polymorphism, which allows objects of different classes to be treated interchangeably through method overriding and dynamic binding	Polymorphism is not a built-in concept in procedural languages,

History of C Language

- **C programming language** was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), U.S.A.
- It was developed to overcome the problems of previous languages such as B, BCPL, etc.
- Initially, C language was developed to be used in **UNIX operating system**. It inherits many features of previous languages such as B and BCPL.

Features of C Language

- Simple
- Machine Independent or Portable
- Mid-level programming language (it supports the feature of both low-level and high-level languages)
- structured programming language
- Rich Library
- Memory Management
- Fast Speed
- Pointers
- Recursion
- Extensible



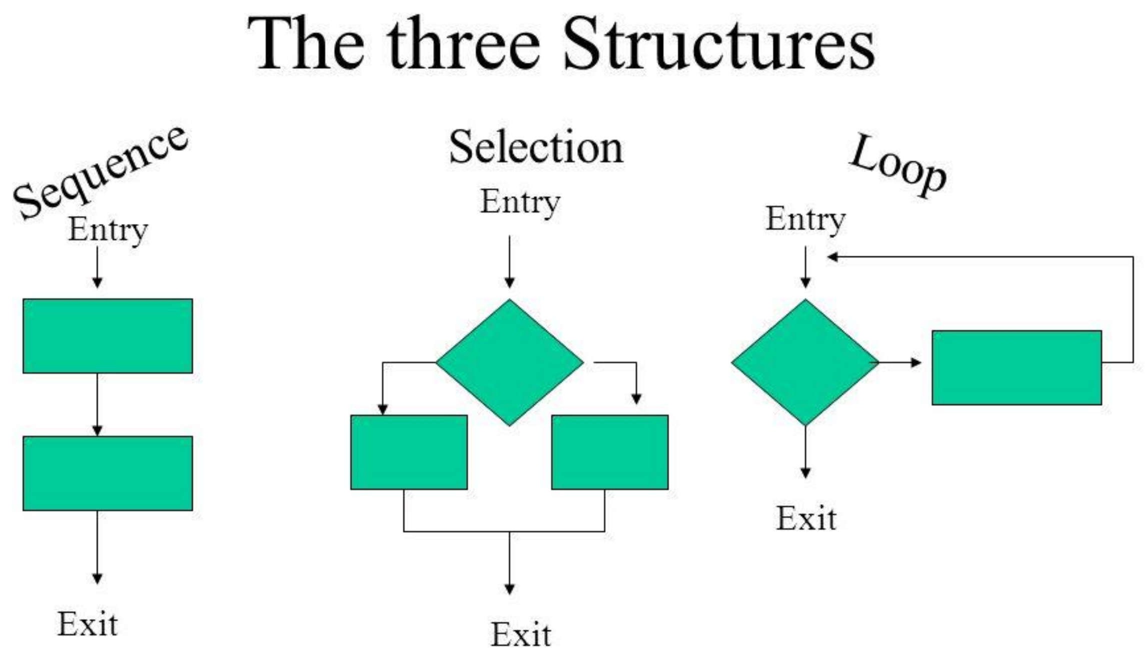
Programming fundamentals

- Algorithm
- It is a formula, a recipe or a step by-step procedure to be followed in order to obtain the solution to a problem.
- To be useful as a basis for writing program.
- The algorithm must;
  - Arrive at a correct solution within a finite time.
  - Be clear, precise and unambiguous.
  - Be in a format which lends itself to an elegant implementation in a programming language.

Control Structures

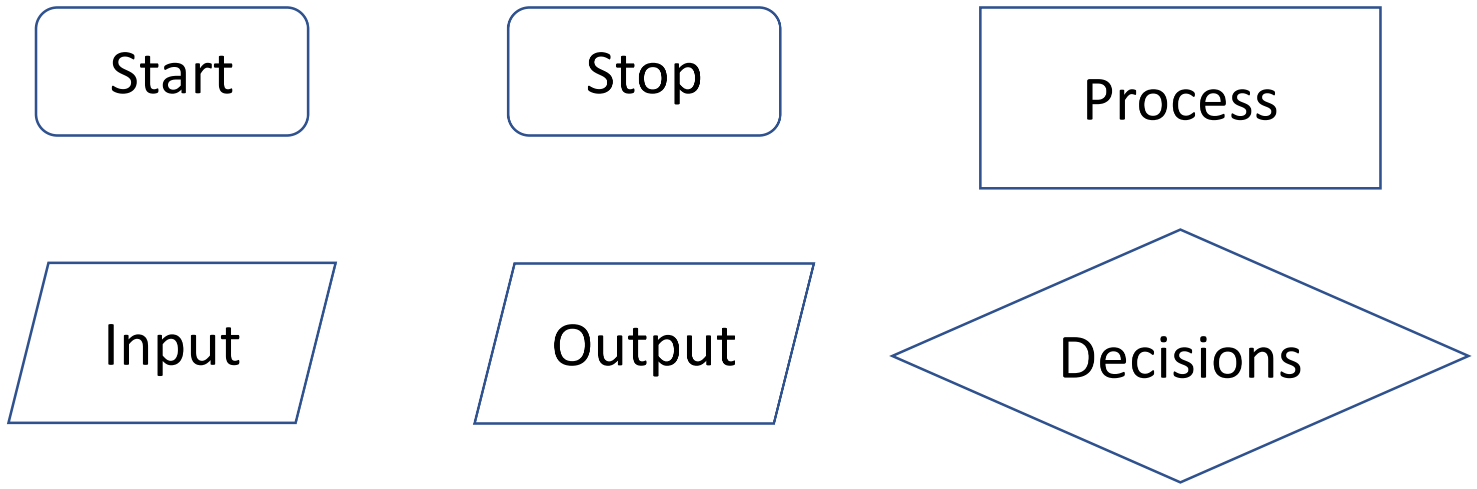
- The key to elegant algorithm design lies in limiting the control structures to only three constructs.

    1. Sequence
    2. Iteration
    3. Selection



Flow Charts

- Flow charts can be used as a way of expressing algorithms.
- Note that there are standard symbols to indicate:

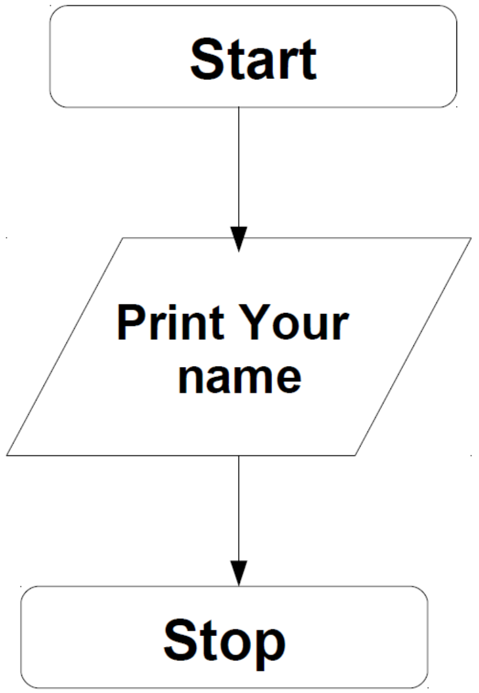


Problem 1:

- Draw a flow chat to print your name.

Problem 1:

Draw a flow chat to print your name.

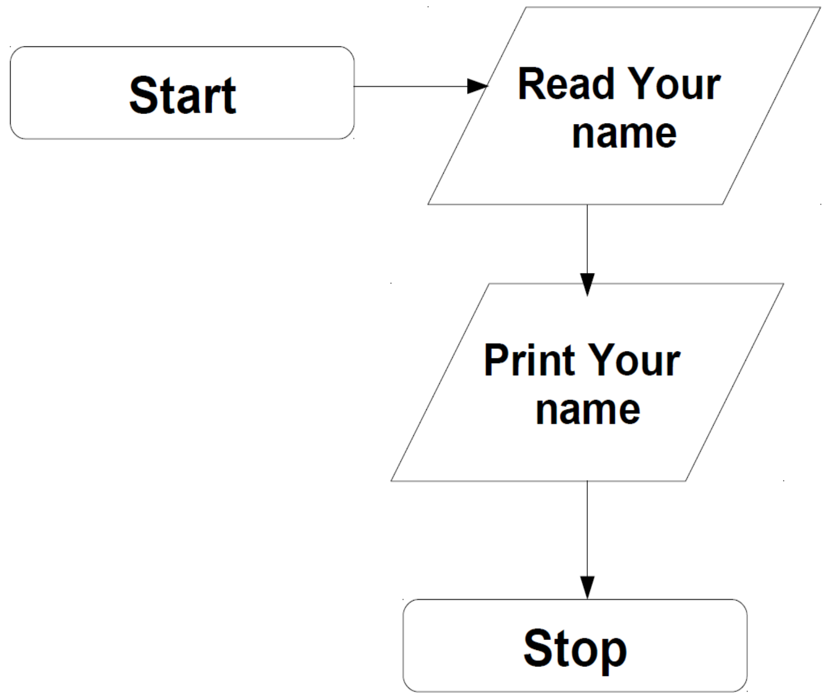


Problem 2:

- Draw a flaw chart to read and print your name.

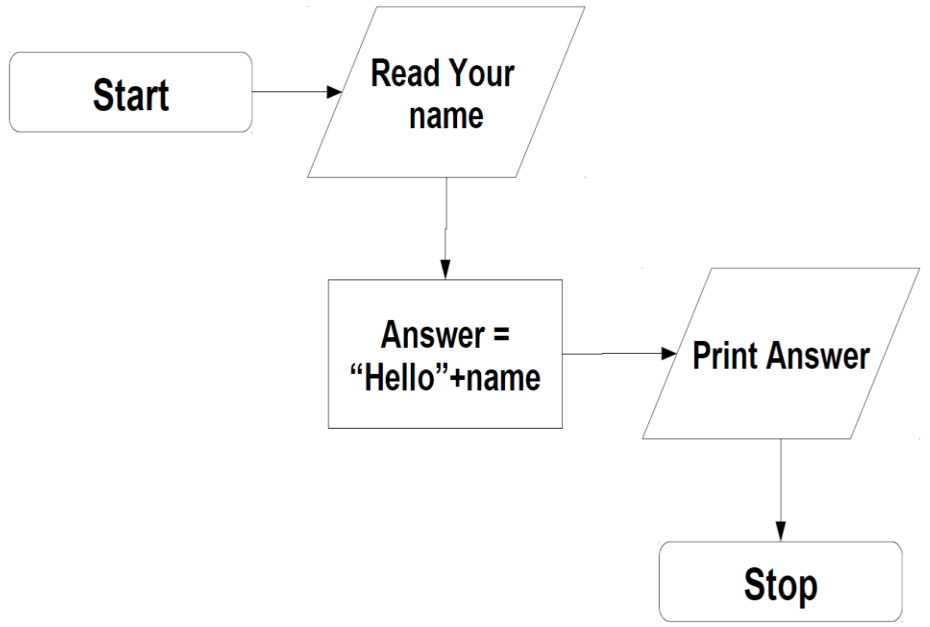
Problem 2:

- Draw a flaw chart to read and print your name.



Problem 3:

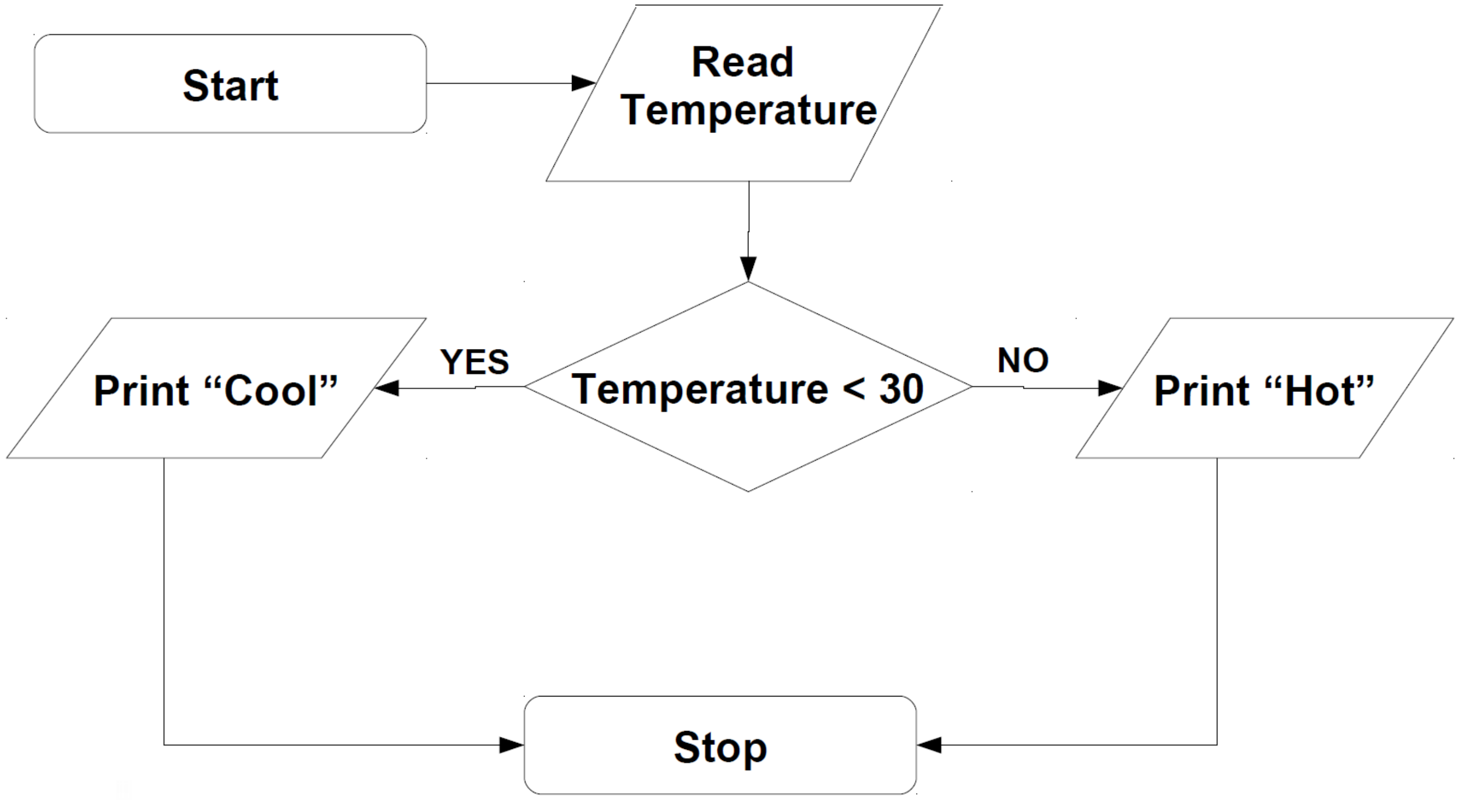
Write a program to read and say “Hello” to your name



Problem 4:

Write a program to read temperature and print "Cool" or "Hot"

- If temper



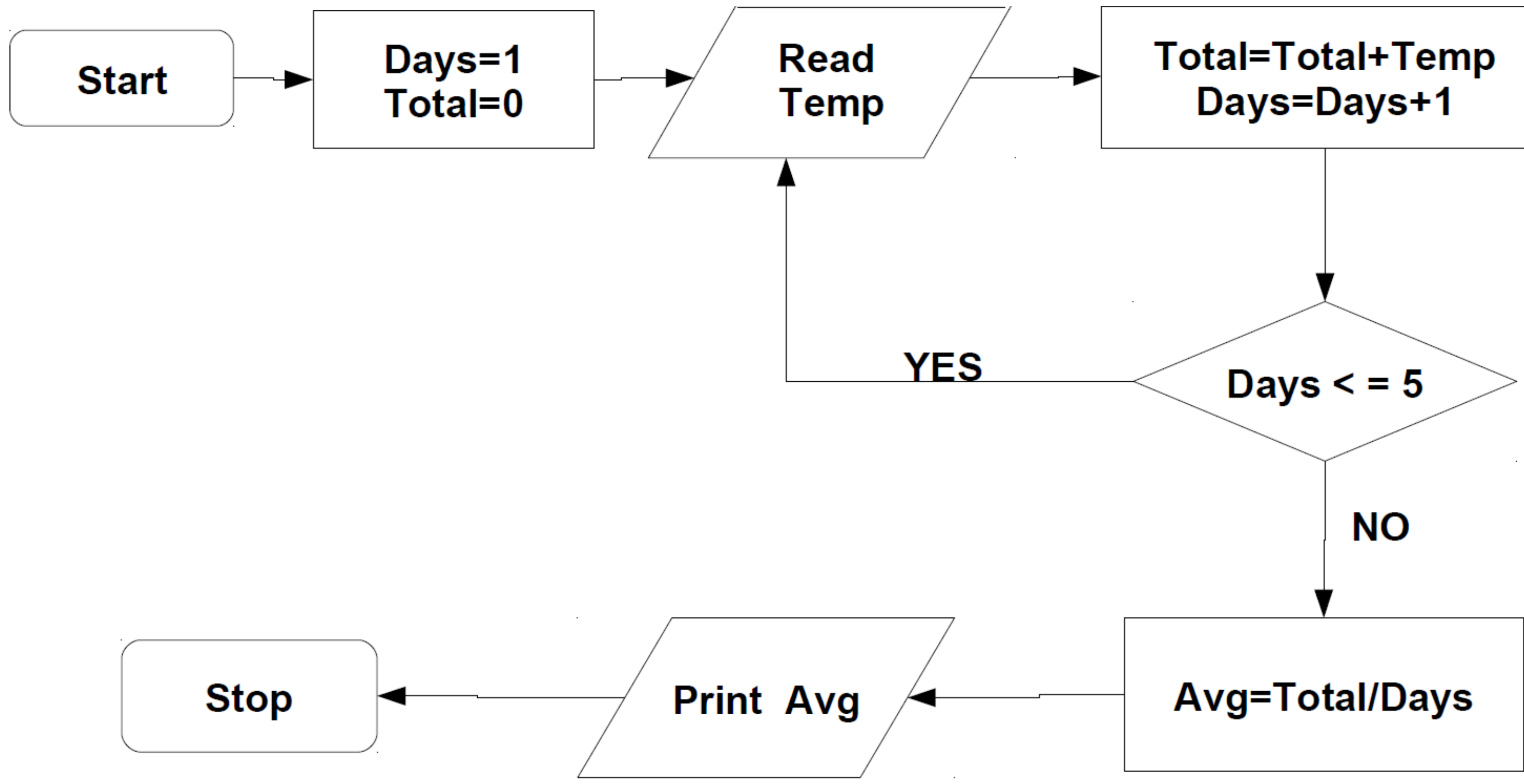
Problem 4:

Write a program to read temperature and print “Cool” or “Hot”

- If temperature is less than 30: Cool Otherwise: Hot

Problem 5:

Read temperature for 5 days and print average.



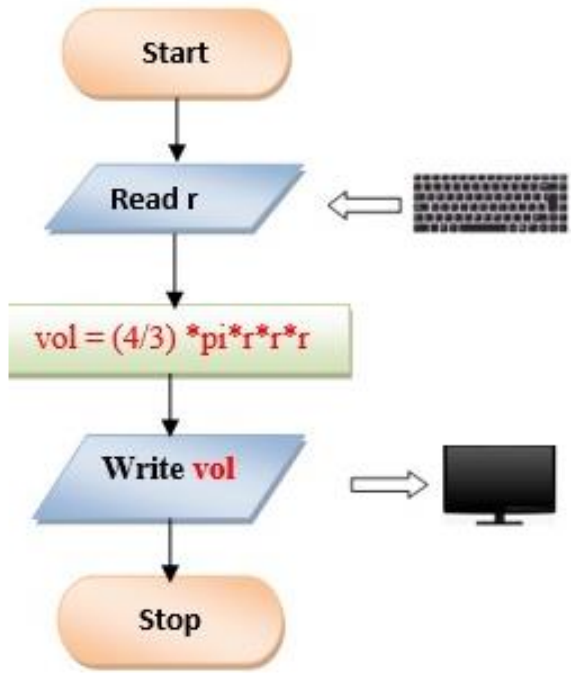
Problem 6:

- Draw a flaw chart and write a pseudocode algorithm to compute the volume of a sphere. Use the formula: V = (4/3) *pi*r^3 where pi is equal to 3.1416 approximately. The **r** is the radius of sphere. Display the result.

Problem 6:

- Draw a flaw chart and write a pseudocode algorithm to compute the volume of a sphere. Use the formula: $V = (4/3) * pi * r^3$ where pi is equal to 3.1416 approximately. The **r** is the radius of sphere. Display the result.

Flowchart



Algorithm

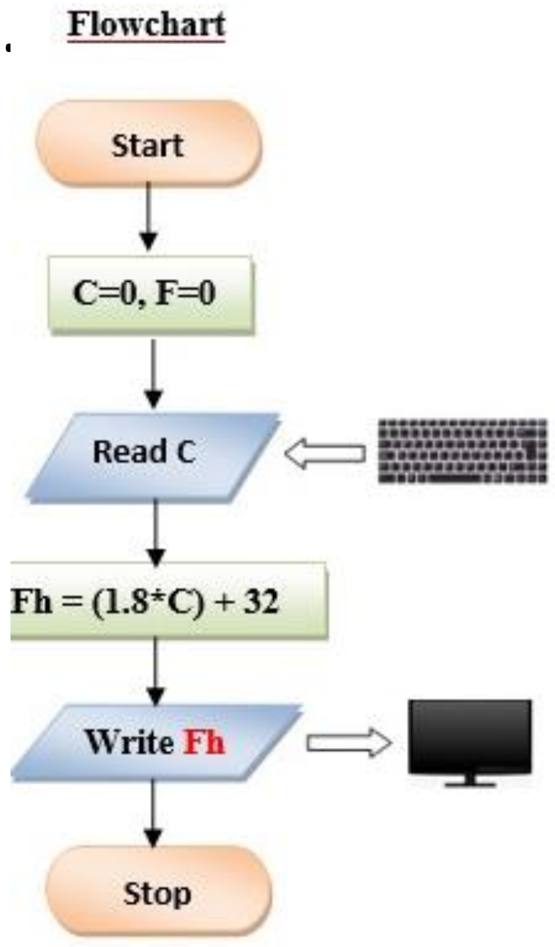
1. Start
2. Read r
3. vol = (4/3) *pi*r*r*r
4. Print or display vol
5. Stop

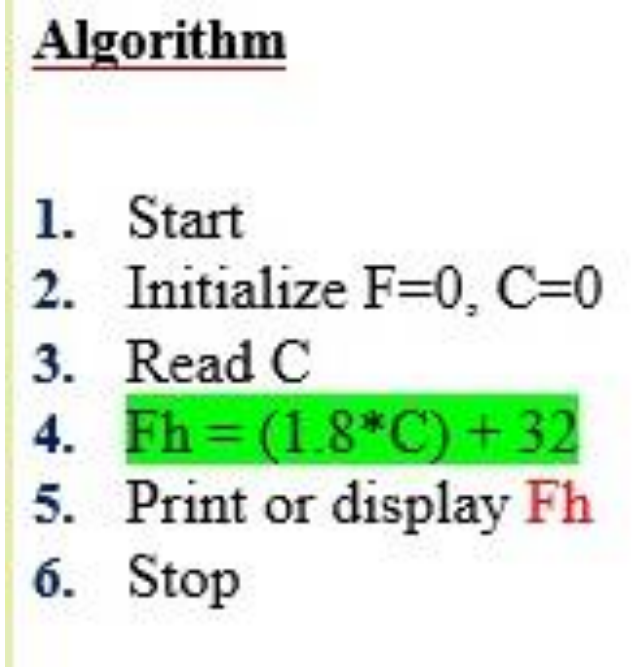
Problem 7:

- Draw a flow chart and write a pseudo code algorithm to converts the input Celsius degree into its equivalent Fahrenheit degree. Use the formula: $F = (9/5) * C+32$.

Problem 7:

- Draw a flow chart and write a pseudo code algorithm to converts the input Celsius degree into its equivalent Fahrenheit degree. Use the formula: F = (9/5) *C+32.





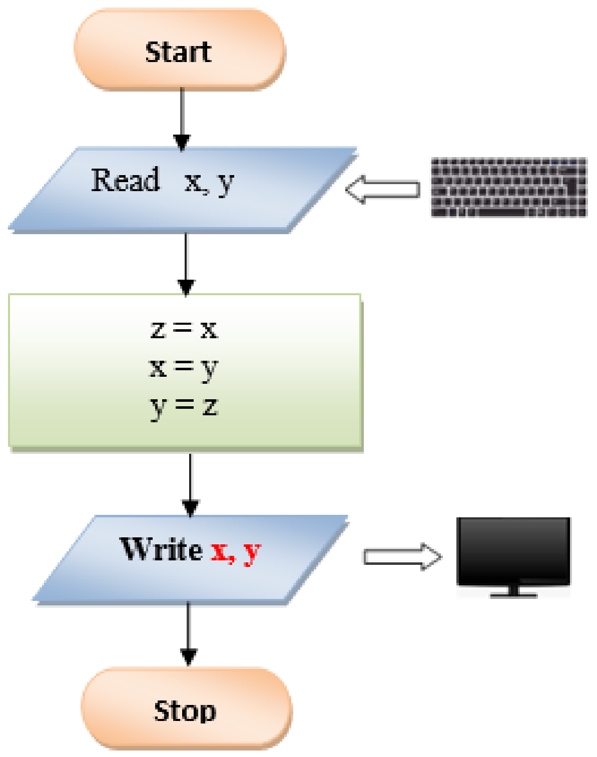
Algorithm

```
1. Start
2. Initialize F=0, C=0
3. Read C
4. Fh = (1.8*C) + 32
5. Print or display Fh
6. Stop
```

Problem 8:

- Draw a flow chart and Write a pseudocode algorithm to exchanges the value of two variables: x and y. The output must be: the value of variable y will become the value of variable x, and vice versa.

Flowchart



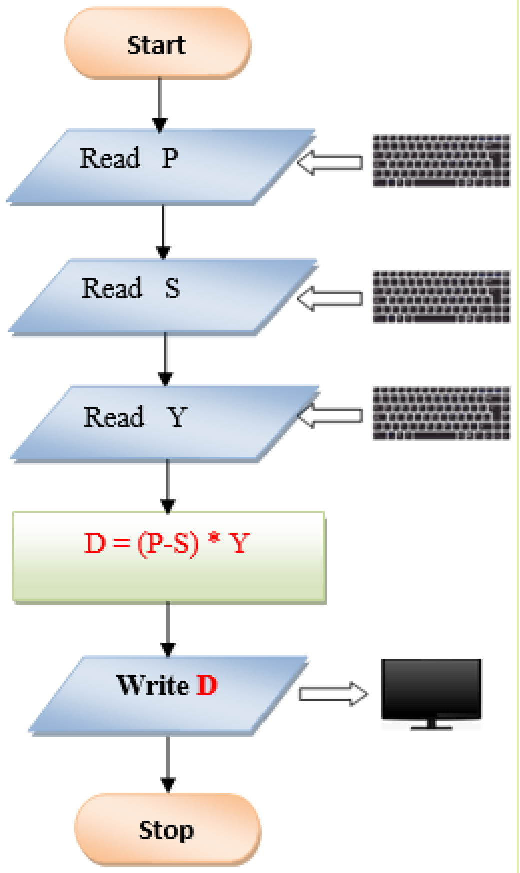
Algorithm

```
1. Start
2. Read x, y
3. Declare third variable, z
            z = x
            x = y
            y = z
4. Print or display x, y
5. Stop
```

Problem 9:

- Write a program that takes as input the purchase price of an item (P), its expected number of years of service (Y) and its expected salvage value (S). Then outputs the yearly depreciation for the item (D). Use the formula: D = (P – S) Y.

Flowchart



Algorithm

1. Start
2. Read P
3. Read S
4. Read Y
5. D = (P-S) * Y
6. Print or display D
7. Stop

Problem 10

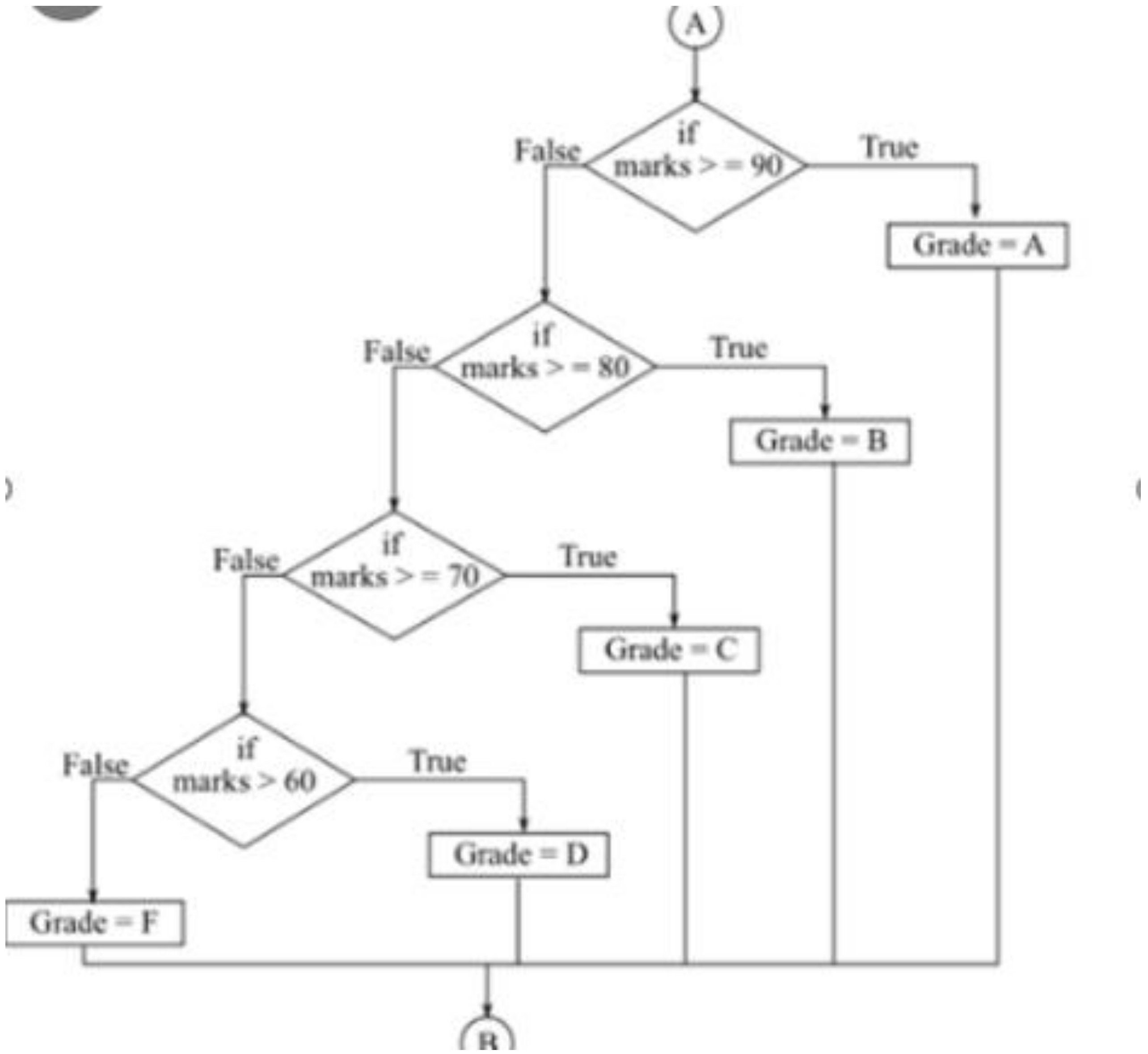
- Draw a flow chart, Write pseudo code algorithm and finally calculate the Grade
- Hint : marks >=90 “A GRADE”

  marks >=80 and =< 89 “B GRADE”

  marks >=7 0and =< 79 “C GRADE”

  marks >=60 and =< 69 “D” GRADE”

  Otherwise “ F GRADE”


A
if marks > = 90
False
True
Grade = A
if marks > = 80
False
True
Grade = B
if marks > = 70
False
True
Grade = C
if marks > 60
False
True
Grade = D
Grade = F
B

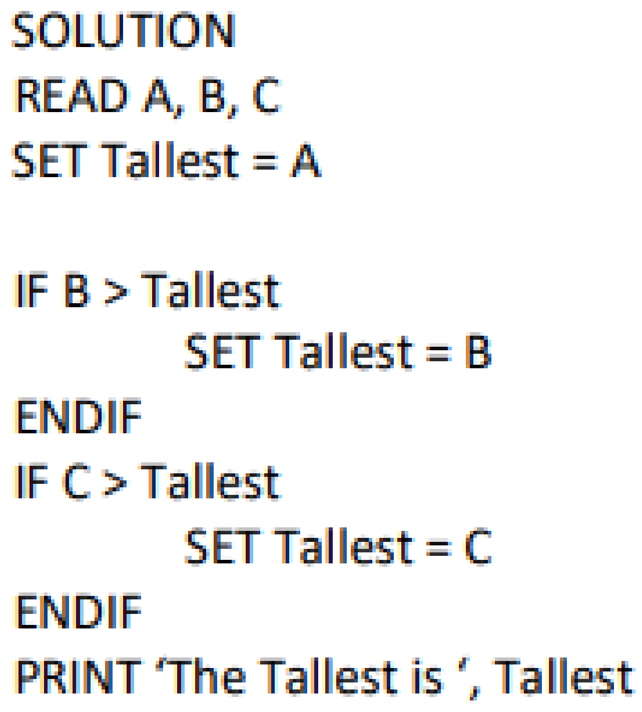
Problem 10:

Write Pseudocode to print the height of the tallest boy in a three-guy basketball team. Their height is stored as A, B, C.

Problem 10:

Write Pseudocode to print the height of the tallest boy in a three-guy basketball team. Their height is stored as A, B, C.

- SOLUTION In the following algorithm, it’s assumed that the first boy A is the tallest, then the second boy B is compared to this tallest. If the second happens to be taller than the assumed tallest, the second is stored as the new tallest, otherwise the first number is still the tallest. Algorithm does the same with the third boy.



```
SOLUTION
READ A, B, C
SET Tallest = A

IF B > Tallest
        SET Tallest = B
ENDIF
IF C > Tallest
        SET Tallest = C
ENDIF
PRINT ‘The Tallest is ’, Tallest
```