EST 102
PROGRAMMING IN C

MODULE 1

MODULE OVERVIEW:

Module 1

Basics of Computer Hardware and Software

Basics of Computer Architecture: processor, Memory, Input& Output devices

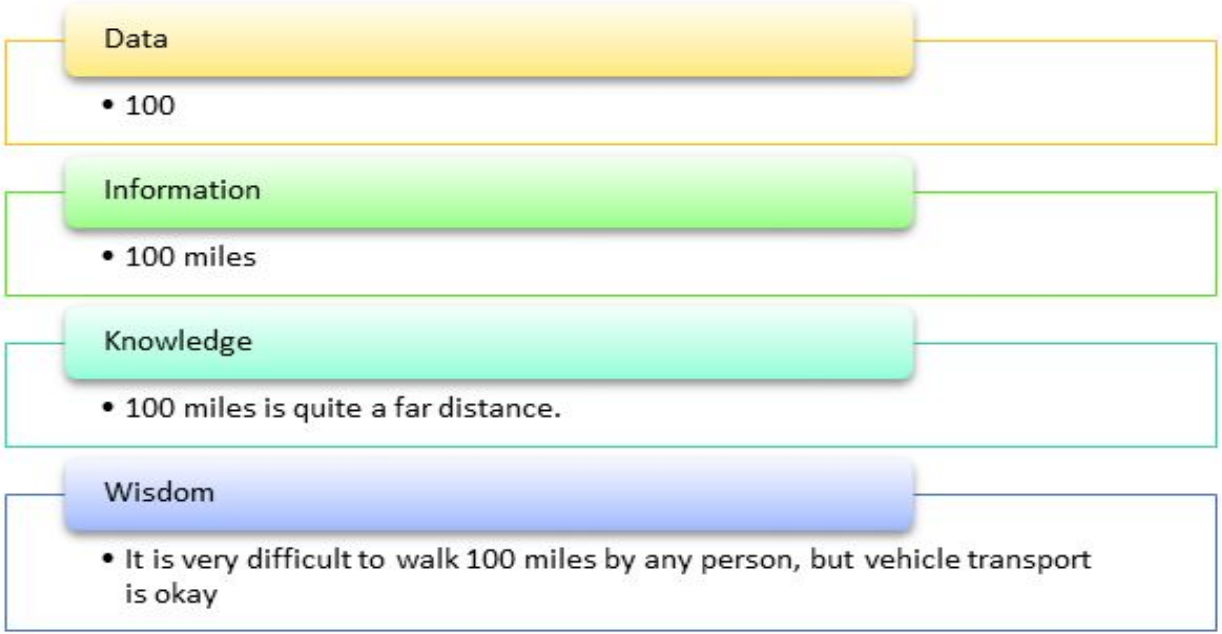
Application Software & System software: Compilers, interpreters, High level and low level languages

Introduction to structured approach to programming, Flow chart Algorithms, Pseudo code (*bubble sort, linear search - algorithms and pseudocode*)

INTRODUCTION TO COMPUTER:

- A computer is an **electronic programmable device** that can accept data, process data and produce results based on the instructions given to it.
- It is used to process **data** into **information** that is useful to people.
- The word ‘programmable’ makes it unique as compared to other electronic devices which perform only the function for which it was developed for. They are not programmable. Eg: television, radio, etc.
- Computers can be used to carry out massive and sophisticated scientific as well as commercial purposes.

DIKW Model:


Data
• 100
Information
• 100 miles
Knowledge
• 100 miles is quite a far distance.
Wisdom
• It is very difficult to walk 100 miles by any person, but vehicle transport is okay

Basic Components of a Computer System:

- The complete computer system has 4 components:

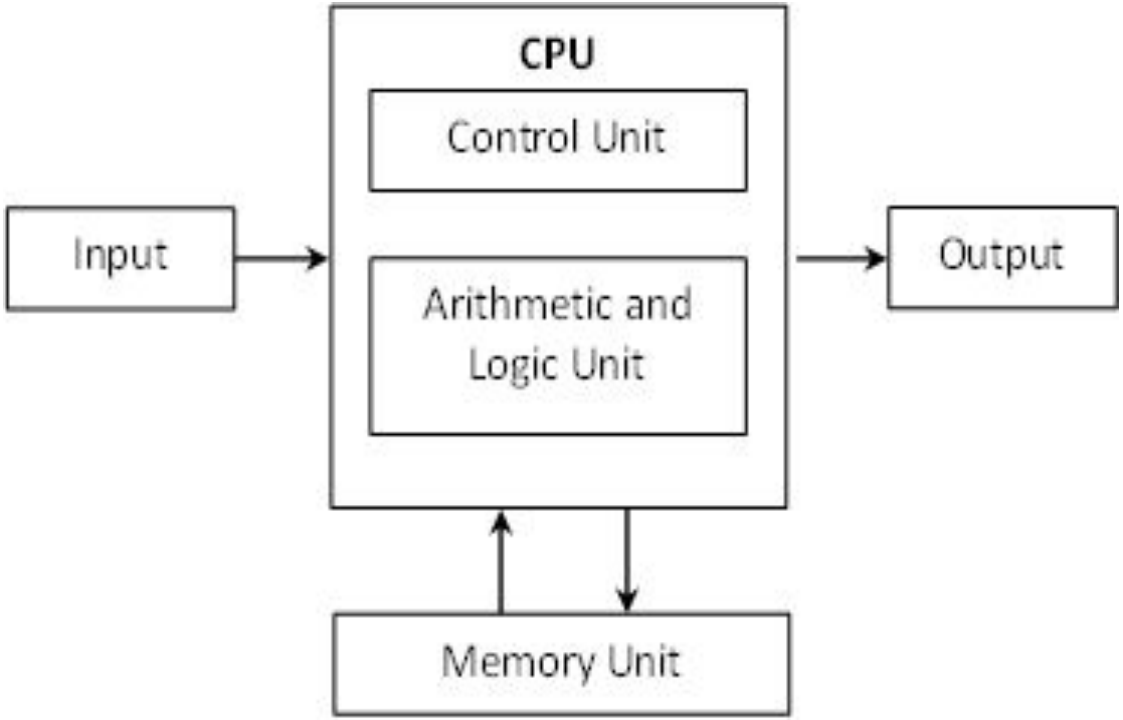
1. **Hardware**- any part of the computer system that one can see and touch.
2. **Software**- tells the computer what to do and makes the computer do meaningful work.
3. **Users-**Computers work as per directions of the user.
4. **Data-** raw facts that are manipulated by the computer according to the instructions provided by the software. Computers transform raw data into useful information.

BASICS OF COMPUTER ARCHITECTURE:

- The **hardware** of a computer system can be categorised into following four functional units :

1. Central processing unit (Processor)
2. Memory unit
3. Input devices
4. Output devices

Components of Computer Hardware(Figure):



CPU:

- Central Processing unit or Processor is the brain of the computer .
- The Central Processing Unit, also known as the *processor* is an electronic circuitry designed on the motherboard to carry out operations of the computer system.
- It contains arithmetic and logic unit (ALU), control unit (CU) and an array of registers. The processor runs a program by executing the instructions one by one.

The Memory Unit:

- The Memory unit refers to the area where instructions and data are stored in a computer.
- The processor reads instructions and data from the memory, executes them and writes the result back to it.
- Different forms of memory are used in a computer system. They vary in their size, speed and location.
- A memory chip is a collection of storage cells together with associated circuits needed to transfer information in and out of the storage.
- Anything and everything in a computer is stored in the form of bits known as binary language or machine language.
- 8 bits make up a byte.

Input Devices & Output Devices:

- **The Input Devices:** - The input unit consists of input devices that are attached to the computer. These devices take input from the user and convert it into binary language that the computer understands. Some of the common input devices are keyboard, mouse, joystick, scanner etc.
- **The Output Devices:** - The output unit consists of output devices that are attached to the computer. It converts the binary data coming from CPU to human understandable form. The common output devices are monitor, printer, plotter etc.

CENTRAL PROCESSING UNIT:

- Central Processing unit or **Processor** is the **brain of the computer** .
- It is an electronic circuitry designed on the motherboard to carry out operations of the computer system.
- It contains **arithmetic and logic unit** (ALU), **control unit** (CU) and an array of registers.
- The processor runs a program by executing the instructions one by one.
- Each instruction execution cycle (IC) is made up of three phases;

  -instruction fetch

  -instruction decode

  -instruction execution.

- **System Clock:** Within the CPU, events happen at a pace controlled by the vibrations of a tiny quartz crystal called system clock.
- If CPU is the *brain* of the computer, then system clock is the *heart* of it.
- The system clock pulses at a rapid rate measured in millions of cycles per second(MHz).
- A single beat of clock is called the clock cycle or clock tick.
- Depending on the type of processor, it may take one or more clock cycles to carry out an instruction.
- System clock thus synchronises internal activities of the computer.
- As the clock speed is doubled, the number of instructions executed per second is also doubled and so the operating speed of the CPU.

- Every CPU has essentially two sub components: **Arithmetic Logic unit & Control Unit.**
- **Arithmetic Logic unit:**

  -Computer handles all types of data as numbers.

  -So, the processing involves comparing these numbers and/or carrying out arithmetic operations on them.

  -All arithmetic and logical operations are done by the ALU.

ARITHMETIC	LOGICAL
Add	Equal to, Not equal to
Subtract	Greater than, Not greater than
Multiply	Less than, Not less than
Divide	

- **Control Unit:**

  -Acts as a supervisor by managing all resources of the computer system.

  -It initiates, monitors and stops the flow of data from one part of computer to the other.

  -It neither process data nor store data, but simply directs and coordinates the actions of other units of the system.

  -**The control unit controls and coordinates all operations of the CPU by providing appropriate control and timing signals**.

  -

- **Registers:**

  -High speed memory locations in CPU.

  -Used to hold data being processed at a time.

  -There are different registers each designated to perform a specific function.

  -Registers can receive information, hold it temporarily and pass it on as directed by the CU.

  -The size of registers determines the amount of data with which the computer can work at a given time.

CPU REGISTERS:

TYPE	PURPOSE
Storage	To hold information on its way to and from memory.
Instruction	To hold an instruction while it is being executed.
Address	To hold the address of a memory needed
Accumulator	To accumulate results

The Memory Unit:

- CPU registers are small areas that can hold only a few bytes of data at a time.
- CPU needs to have a larger space where it can store programs and data while they are being used. This area is called memory.
- The Memory unit refers to the **area where instructions and data are stored in a computer.**
- The processor reads instructions and data from the memory, executes them and writes the result back to it.
- The memory is also referred to as internal memory, main memory, system memory or primary memory.
- Memory consists of a large number of cells called **memory locations**.
- Each memory location can store one binary word of information.
- The word length is specific for each processor.

- Most of the memory in a computer is of a temporary nature.
- The data and programs stored in these memory devices will be available for use as long as the computer is not switched off.
- So it is very important to have some other means to keep the information permanently. Such devices used in computers are called storage devices or secondary memory.
- Magnetic devices or semiconductors are used to fabricate memory devices.
- Primary memory appears in the form of a chip which is attached to the motherboard.
- Some memory chips can retain the data they hold even without electric power. Such a memory is called **non-volatile(permanent) memory.**
- On the other hand, memory chips whose contents are not retained when electric power is switched off are called **volatile memory.**

The Registers:

- **Fastest** accessible units of memory by the processor.
- They are highly expensive and hence are limited in number.
- Some registers are used to store data while some are reserved to store address.
- There are some general purpose registers as well.
- Size of the registers depends on the processor.
- Accumulator is a register found on all processors, which is mainly used to store the operand of an arithmetic operation.
- Program counter (PC) always stores the address of the next instruction to be executed by the processor.

Primary Memory:

- The memory chips that the processor can directly access are referred to as primary memory.
- The three types of primary memory are RAM, ROM and CMOS.
- RAM is volatile whereas ROM as CMOS are non-volatile memories.
- They are metal oxide semiconductor memory cells built on silicon based ICs.

Random Access Memory (RAM):

- RAM is a volatile memory and loses its contents when the power is turned off.
- The circuit is so designed that any random location can be directly accessed without the need to scroll up or down the memory.

- The program to be executed is loaded into RAM from the non-volatile backup memory.
- The processor reads instructions and data from the RAM, executes them and writes data back to the RAM. Hence RAM is also known as the main memory of the computer.
- Everything from the RAM is copied to the non-volatile backup memory before the computer is turned off.
- The two main types of RAM are **Dynamic Ram** and **Static RAM.**
- **DRAM**:

  -Each memory cell in a DRAM is made of one transistor and one MOS capacitor, which store one bit of data.

  - However, this cell starts losing its charge and hence data can be retained for less than thousandth of a second.

  -So it needs to be refreshed thousand times a second, which takes up processor time.

  -Primary memory of most of the personal computers is made of DRAM.

- **SRAM:**

  -Each cell in SRAM is made of several transistors in a cross coupled flip flop configuration that stores one bit.

  - It retains its bit till the power supply is on and doesn’t need to be refreshed like DRAM.

  - It also has shorter read-write cycles as compared to DRAM.

  - SRAM is used in specialized applications.

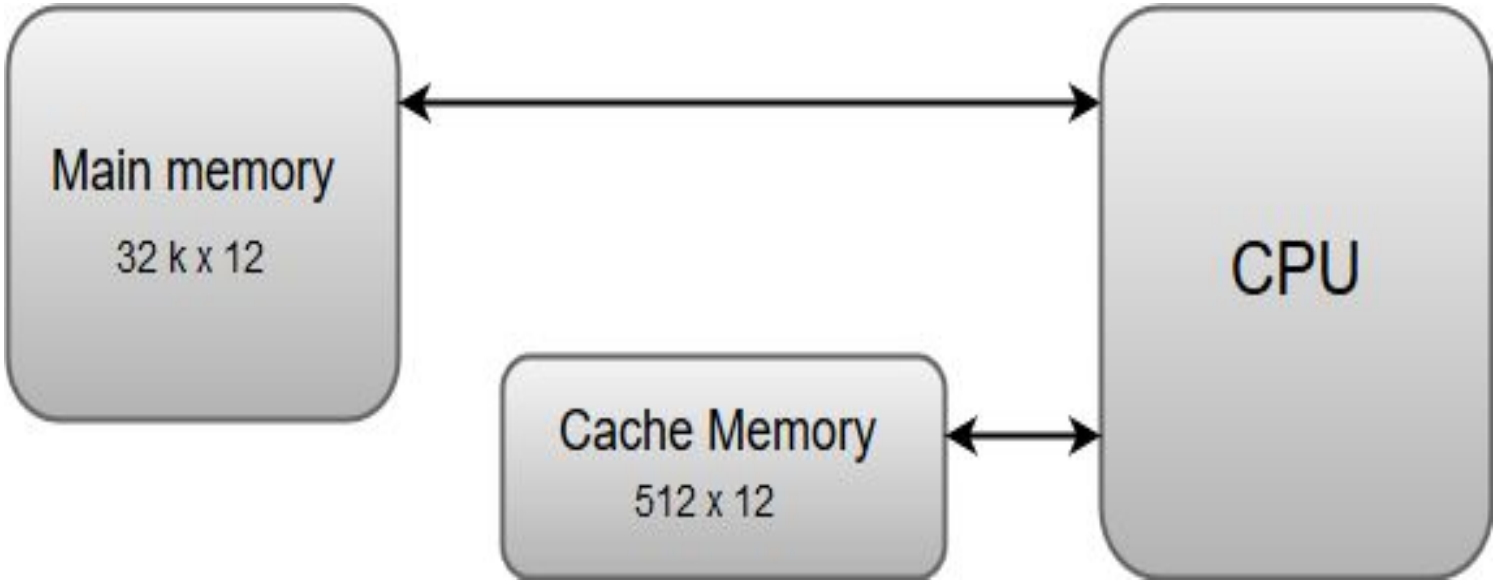
Read Only Memory (ROM):

- ROM is a **non-volatile** memory and hence retains its contents even when the computer is switched off.
- It is used for storing software that is rarely changed during the life of the system, sometimes known as firmware.
- Almost every computer comes with a small amount of ROM containing the boot firmware which is essential for the boot-up.
- BIOS is a firmware used to perform hardware initialization during the booting process and to provide runtime services for operating systems and programs.
- The BIOS firmware comes pre-installed on the ROM and it is the first software to run when the computer is powered on.

- ROM memories have gradually evolved from fixed read-only memories to memories than can be programmed and then re-programmed.

1. ROM (Read Only Memory)
2. PROM (Programmable Read Only Memory)
3. EPROM (Erasable Programmable Read Only Memory).
4. EEPROM (Electrically Erasable Programmable Read Only Memory)

- **CACHE MEMORY:**
  - Moving data between CPU registers and RAM is the most time consuming task.
  - Cache memory provides a solution for this problem.
  - Cache memory is similar to RAM, but extremely faster.
  - Once CPU reads data from RAM, it keeps one copy in the registers and other in cache memory.
  - For subsequent requirements of same data, CPU gets it from the cache.
  - Cache is made up of high speed semiconductor devices that are costlier than RAM.


Main memory
32 k x 12
CPU
Cache Memory
512 x 12

Memory hierarchy:

MEMORY HIERARCHY DESIGN

INPUT DEVICES:

- Devices that enables user to input data and commands into the computer.
- Such devices connects the external environment with the internal computer system.
- Input devices can send data to other devices, but can’t receive data from other devices.
- Input unit performs the following functions:

  -accept the data and instructions from the external environment.

  -convert it into machine language.

  -supply the converted data to computer system.

- Commonly used input devices are keyboard, mouse, microphone, webcam, MICR, e-pens etc.

Keyboard:

- Most common input device for entering text and numbers.
- It **accept text/numeric input from the computer user** and sends that data to a computer.
- Keyboards are available in many styles, but with identical layouts.
- The most common layout is the IBM enhanced keyboard with more than 100 keys which are grouped generally into 4 categories based on their usage.

Mouse:

- Palm sized **pointing device** with a ball built into the bottom.
- When mouse is moved over a smooth surface, the ball rolls and the pointer on the display moves in the same direction.
- It allows to **choose commands from easy to use menus and dialog boxes**.
- This frees the use from using a keyboard up to a large extent.

Microphone:

- Receives the sound generated by an input source and sends that sound to a computer.

Bar Code Reader:

- Bar code is a pattern of printed bars on products, that can be converted into a code, that computer can understand.
- Bar code reader emits a beam of laser light that is reflected by the bar code image and is identified by a light sensitive detector.
- Now the individual bar patterns are converted into numeric digits and fed to the computer as if the number has been typed on a keyboard.
- Used widely in supermarkets and stores to determine the items being sold and to retrieve the details of item from data storage.

Magnetic Ink Character Reader:

- MICR can read characters printed in a certain style with special ink containing magnetic particles.
- Some of the details on a cheque are printed using the magnetic ink, which can be recognised by its magnetic field.

Electronic Pen:

- E-pens allow user to write directly on or point at a special pad or screen of a pen based computer.
- Handwriting recognition being a complex task, pens are usually used for making selection from a list or for creating short messages.
- This type of input is allowed by most of the handheld computers.

OUTPUT DEVICES:

- Devices that are capable of representing information from a computer.
- Such devices connects the internal system of a computer to the external environment.
- It provides the result of any computations, or instructions to the outside world.
- Some common output devices are monitors, printers, plotters etc.,
- Output can take the form of a **hard copy** or **soft copy.**
- **Hard copy** does not require an electronic interface like computers or mobiles etc to read and display.
- **Soft copy** requires an electronic interface like computers or mobiles etc to read and display.

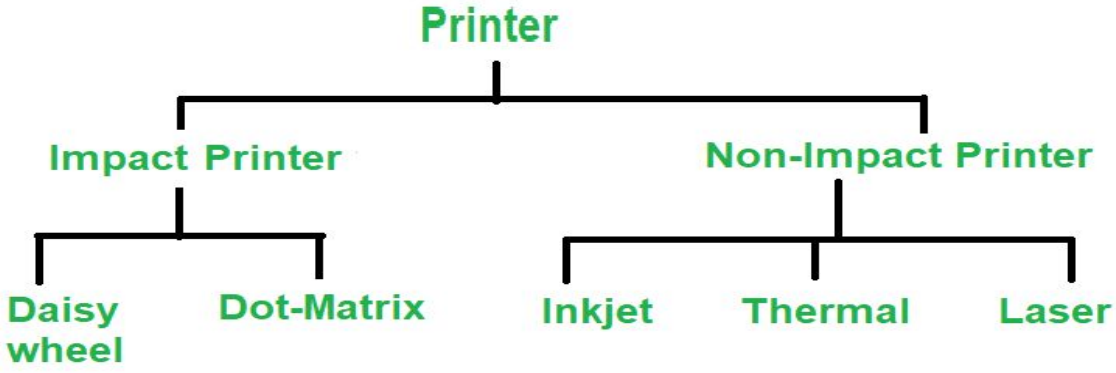
Monitors:

- Receives data from a computer and displays that information as text and images for users to view.
- Monitors are categorised based on several ways like colours they display, the technology used to generate images, their screen size and other performance characteristics.

Printers:

- There are several real-life situations where it is essential to have printed documents or hard copy.
- A **printer** is an output device that prints paper documents. This includes text documents, images, or a combination of both.
- The printed output produced by a **printer** is often called a hard copy, which is the physical version of an electronic document.

- Printers can be broadly classified into two **based on technology** used;



Impact and Non-Impact Printer

- Impact printers involve mechanical components for conducting printing while in Non-Impact printers, no mechanical moving component is used.

- **Based on the speed**, printers can be classified into three;

  -Character printer: can print only one character at a time.
  Eg:Dot matrix printer, Daisy wheel printer.
  -Line printer: can print an entire line at a time.
  Eg: Drum printer, Band printer
  -Page printer: can print an entire page at a time as a whole.
  Eg: Laser printer, ink jet printer.

Plotters:

- Plotter is used to print images on larger formats such as engineering drawings created using CAD.
- Plotters are available in two forms: **Drum type** and **Flat bed type.**
- In drum plotters, both paper and pen move whereas in Flat bed type, paper is fixed and pen moves.

INPUT/OUTPUT DEVICES:

- There are some devices that can accept input and produce output. Such devices are called Input/Output devices.
- Eg:Modem, USB Flash drive, CD-RW drive, DVD- RW drive etc.,

SOFTWARE:

- Computers need clear-cut instructions to tell them;

  -what to do

  -how to do

  -when to do.

- A set of.instructions to carry out these functions is called a **program.**
- A group of such programs that are put into a computer to operate and control its activities is called the software.
- Software is a collection of instructions that enable the user to interact with a computer, its hardware and perform tasks.
- Software is an essential part of the computer system and there are two categories for a software- **Application software and System software.**

Application Software:

- Application software (app for short) is computing software designed **to carry out a specific task** other than one relating to the operation of the computer itself, typically to be used by end-users.
- All programs that enable the computer users to apply the computer to do their work, can be called application software.
- Examples:

1. Railways reservation software
2. CAD software
3. Microsoft word
4. Microsoft powerpoint
5. Database management software etc.

System Software:

- System software includes the programs that are dedicated to managing the computer itself.
- ie,system software includes general programs written for the system which provide the background to facilitate the development and usage of application software.
- System software **provides platform to other softwares.**
- These are commonly prepared by computer manufacturers.
- **Eg:**

1. Operating system
2. Anti-virus software
3. Disk formatting software
4. Computer language translators.

- These software consist of programs written in low-level languages used to interact with the hardware at the very basic level.
- System software serves as the interface between the hardware and the end users.
- The most important features of the system software includes:

  -High speed.

  -Closeness to the system

  -Difficult to manipulate

  -Written in low-level language

  -Difficult to design

Operating System:

- A set of programs that control and coordinate the operation of a computer.
- OS is a type of system software that manages computer's hardware and software resources.
- OS act as a link between hardware and software.
- It controls and keeps a record of the execution of all other programs that are present in a computer, including application programs and other system softwares.
- A computer cannot work without an OS.
- When we turn on a computer, the computer copies the essential portions of the OS called **kernel** into the memory.
- Since, the kernel resides in the memory at all the times, it is called the memory resident part of the computer system.
- The remaining portions of an OS are copied from the disk to the memory as and when needed.

- All operating systems performs some basic functions:

  -Manage programs

  -Manage memory

  -Handling input and output devices.

  -Manages interaction between user and computer when computer is switched on.

- Examples for operating systems:

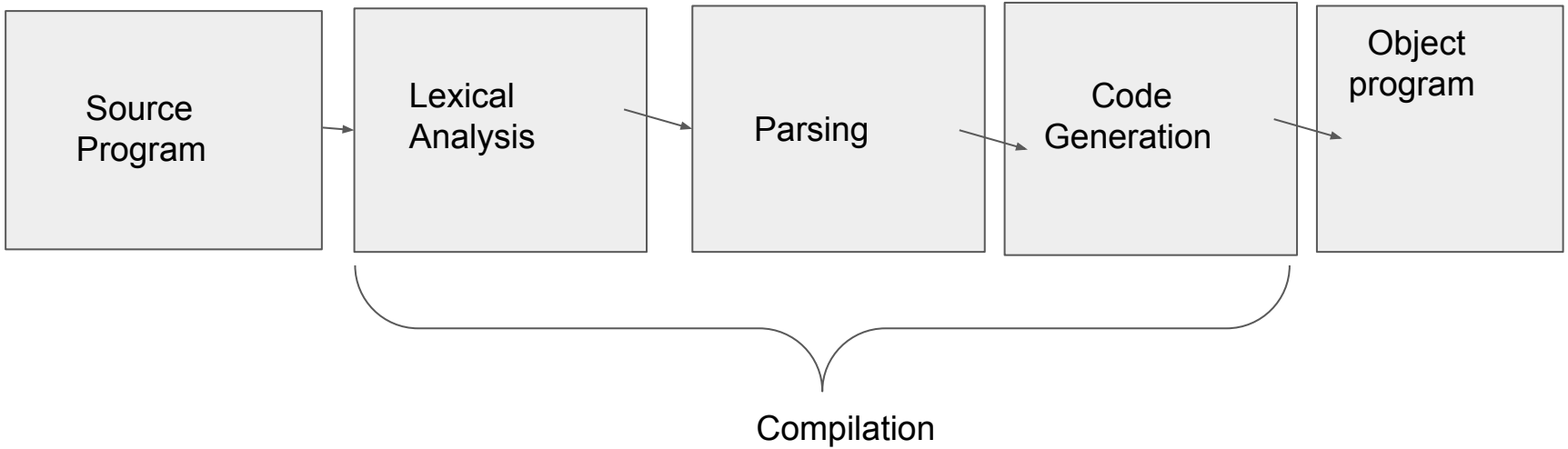
  Windows, LINUX, Mac OS

COMPILERS:

- Compiler is a **translation software** that converts a **program in high level language into a program in machine language.**
- ie, it converts **source code into object code.**
- It translates the code written in one language into code in another language without changing the meaning of the program.
- Compiler makes the target code efficient and optimised in terms of space and time.
- Examples of compilers include gcc(C compiler), javac (Java compiler), g++ (C++ compiler)

The Compilation process:

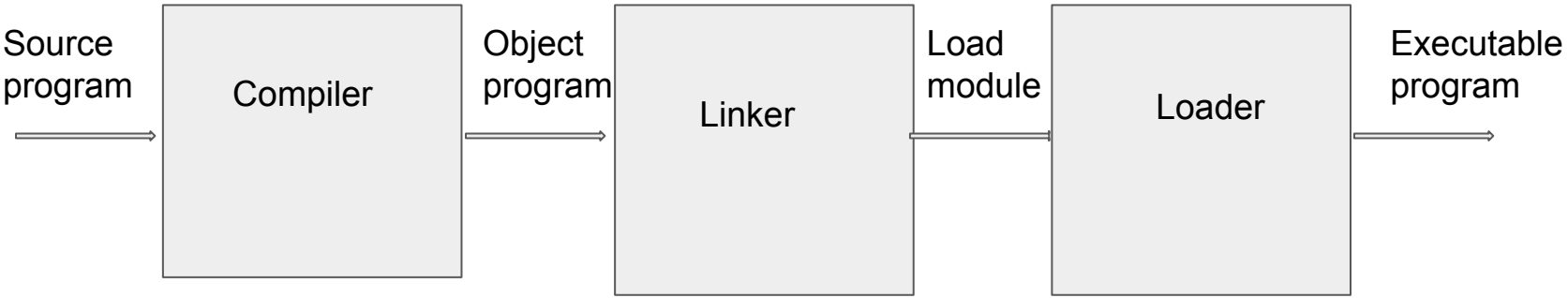
- The compilation of a program consists of 3 activities;
  - **Lexical Analysis**-the process of recognizing which strings of characters in the source program represent named elements like variables, functions etc.,
  - **Parsing**- the process of analysing the grammatical structure of a statement.
  - **Code generation-**Constructs sequences of machine language instructions equivalent to the statements in the source code.



- The object programming produced after code generation won’t be in a form suitable for direct execution by a computer.
    - It may contain service requests from OS and other utility software.
    - The resultant object code created after compiling the modules in different files may not have any link between them.
- The utility program **linker** acts in this context.
- Linker links several object modules, OS routines and other utility programs and produce a complete executable program.
- The executable program is called the load module which will be stored as a file in some storage device.
- **Loader** which is another utility software is responsible for loading this load module from storage device to memory for execution.
- Therefore, compiler do not execute the program, but it convert the program into an object program.

- The 3 steps process compile, link and load actually execute a program.
- Once compile and link steps are complete, program can be loaded and executed repeatedly any number of times without returning to the source code.
- Whenever a change is to be made in the program, it is done in the source code and the modified source program is compiled and linked to produce a new load module that can be directly executed.

Program execution process:



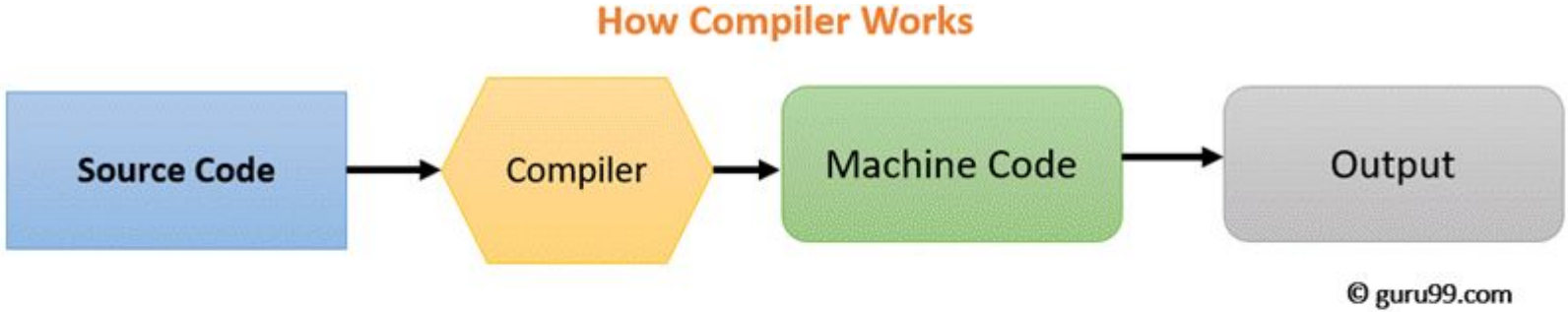
Interpreter:

- Compiler and Interpreter are two different ways to translate a program from programming or scripting language to machine language.
- A compiler takes entire program and converts it into object code which is typically stored in a file. The object code is also referred as binary code and can be directly executed by the machine after linking. Examples of compiled programming languages are C and C++.
- An Interpreter **directly executes instructions** written in a programming or scripting language **without previously converting them to an object code or machine code.**
- Examples of interpreted languages are Perl, Python and Matlab.

- Interpreter is the **software** that is capable of **reading, analysing and directly executing statements of a high level program.**
- The interpreter **translates one line of source code at a time** and executes the translated code.
- The interpreter **does not produce an object code** or executable program.
- As the **program is executed line-by-line,** it is helpful in finding out the errors in the program.
- With the help of an interpreter, a high- level language program, can be executed through an one-step process than the compile, link, load sequence required for compilation.
- While using an interpreter, no permanent copy of translation(object program) is produced and saved.

- The steps of translation must be repeated each time the program is executed.
- Hence, a **program executed via interpreter will take longer time to execute than a similar compiled program**.
- The execution of a (compiled) machine language program would be about 100 times faster than the execution through an interpreter.
- The compiled program will give out run-time error messages only in terms of the machine code instructions which would be difficult to understand.
- Interpreter continuously translates the program until the first error is met.
- If an error comes it stops executing. Hence debugging is easy.
- Due to this reason, during program development and debugging, the **interpreter can be used to easily detect and correct the errors.**
- Once the program is completed, it may be compiled for efficient usage, as the compiled program runs faster.

- Most of the programming languages provide a compiler while some are having both.
- An integrated environment which includes an editor to enter program statements and to make corrections along with compiler options is known as a language processor.



Interpreter Vs Compiler

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers.	Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters.
No Object Code is generated, hence are memory efficient.	Generates Object Code which further requires linking, hence requires more memory.
Programming languages like JavaScript, Python, Ruby use interpreters.	Programming languages like C, C++, Java use compilers.

High level & low level languages:

- A programming language is not a natural language like the ones we speak.
- It is an artificial language purposefully created to tell the computer what to do.
- Programming languages also consist of a vocabulary and a set of rules.
- These rules are called syntax and used along with vocabulary to write a program.
- To run a program in a computer, this program must be translated into a language understandable by the computer.
- Programmers can easily understand, interpret and compile high level language in comparison to a machine.
    - Eg: C, C++, Java, Python,etc.
- On the other hand, machine can easily understand the low level language in comparison to human beings.
    - Eg: Machine language and assembly language.

Low level languages:

- Earliest computers did’nt have a programming language.
- Those computers were programmed in computer’s own language called machine language.
- It is written using binary digits 0 and 1 only and must be written in accordance with the special characteristics of the processor.
- Each type and family of the processor requires its own machine language and hence it is a **machine dependent** language.
- To write a machine language program, a programmer has to remember all the operation codes of the particular computer.
- While writing a program, one has to keep track of all the operands and know exactly where they are stored in memory.
- Ie,programmer must know in detail about how the internal parts of the computer works.

- With the introduction of assembly languages, programmer’s dependance on machine language was broken.
- It enabled programmers to use abbreviations in english for commonly used strings of machine language.
- The abbreviations called mnemonics are used to represent operation codes while strings of characters represented addresses.
- Assembly language closely resembles machine language since it is machine dependant and closely tied on to what goes on inside the computer.
- The two machine dependant languages machine language and assembly language are also called low level languages.

High Level Languages(HLL):

- In a HLL, the programmer can write instructions using english words and are much **easier to read/write and maintain.**
- There are many high level languages developed, which are different from one another in **syntax**, but are fairly **easy to follow**.
- The HLLs are developed independent of the structure of any computer.
- This makes the program portable; it can be run on different machines.ie, HLLs are **machine independent.**
- Associated with each language, there is a computer program that translates the program into machine language of the computer in use.
- On translating a single statement written in HLL, many machine language statements are created.
- While translating an assembly language statement , only one machine language statement is created.
- This one-to-many translation led to the name high level language.

Advantages of high-level languages:

- High-level language programs are easy to develop.
- It is easy to visualize the function of the program.
- The programmer may not remain aware about the architecture of the hardware. So people without hardware knowledge can also do high level language programming.
- The same high level language program works on any other computer, provided the respective compiler is available for the target new architecture. So high-level languages are portable.
- Productivity against high level language programming is enormously increased.

Disadvantages of HLL's:

- A high level language program can't get executed directly. It requires some translator to get it translated to machine language.
- There are two types of translators for high level language programs. They are interpreter and compiler.
- In case of interpreter, prior execution, each and every line will get translated and then executed.
- In case of compiler, the whole program will get translated as a whole and will create an executable file. And after that, as when required, the executable code will get executed.
- These translator programs, especially compilers, are huge one and so are quite expensive.

	High Level Language	Low Level Language
1.	It is programmer friendly language.	It is a machine friendly language.
2.	High level language is less memory efficient.	Low level language is high memory efficient.
3.	It is easy to understand.	It is tough to understand.
4.	It is simple to debug.	It is complex to debug comparatively.
5.	It is simple to maintain.	It is complex to maintain comparatively.
6.	It is portable.	It is non-portable.
7.	It can run on any platform.	It is machine-dependent.
8.	It needs compiler or interpreter for translation.	It needs assembler for translation.
9.	It is used widely for programming.	It is not commonly used now-a-days in programming.

INTRODUCTION - STRUCTURED PROGRAMMING:

- Structured programming is a method for designing and coding programs in a systematic, organised manner.
- In structured programming, the **program is executed by executing instructions one after the other in a sequential manner.**
- The languages that support Structured programming approach are C, C++, Java, C# etc.,
- The structured program mainly consists of **three types of statements**:
  - **Selection Statements**
  - **Sequence Statements**
  - **Iteration Statements**
- Programs with complex flow of control are difficult to understand, modify and debug.

- Structure programming controls the complexity of the flow of control by using only a small number of simple, standard control structures.
- Also, a complex problem is divided into logical independent units and these units are joined using basic structures.

Advantages of Structured program:

- **Easy to understand and read** since the modules can be considered independently.
- Easier to understand since it is easier to identify the structure of a particular program component.
- **Easy to maintain & modify** because individual modules can be changed without changing the entire program.
- **Easy to design** since modules can be designed independently.
- Program **uses single-entry and single-exit elements.** Therefore a structured program is well maintained, neat and clean.
- **Development is easier** as it requires less effort and time.
- **Easy to Debug.**

- When the principles of structured programming are applied the design of a program, it leads to a well structured and easy to understand program.
- One of the goals of structured programming is;
  - to create programs those are easy for programmer to read(and understand)
  - to create programs those are easy for users to run.

ALGORITHM:

- Program:Set of instructions expressed in a particular programming language.
- Algorithm:A solution to a problem that is independent of any programming language.
- ie.,An algorithm consists of a set of explicit and unambiguous finite steps when carried out for a set of initial conditions produce the required output and terminate in a finite time.
- Algorithm has the following characteristics
    - **Input:** An algorithm may or may not require input
    - **Output:** Each algorithm is expected to produce at least one result
    - **Definiteness:** Each instruction must be clear and unambiguous.
    - **Finiteness:** If the instructions of an algorithm are executed, the algorithm should terminate after finite number of steps.

- An algorithm consists of a number of elementary operations and instructions about the order in which these operations are to be performed.
- To solve a particular problem, there can be several alternative algorithms.
- Generally, it makes sense to design an algorithm that is simple and easy to understand since it will have the best chance of performing as intended.
- Algorithms are usually written independent of any programming language.
- But, it should be related to the features of the programming language which we intend to use.
- Otherwise, actual coding phase would become difficult.
- In an algorithm, there are no rules for punctuation,spelling, vocabulary, use of synonyms.

Eg1:Algorithm to find sum of two numbers:

- Step-1: Start
- Step-2: Input first number say A
- Step-3: Input second number say B
- Step-4: SUM = A +B
- Step-5: Display SUM
- Step-6: Stop

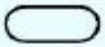
Eg2:Algorithm to find area and perimeter of a circle:

(R: Radius of Circle, AREA : Area of Circle, PERIMETER : Perimeter of Circle)

- Step-1 Start
- Step-2 Input Radius of Circle say R
- Step-3 AREA = 22.0/7.0 *R*R
- Step-4 PERIMETER = 2*22.0/7.0*R
- Step-5 Display AREA, PERIMETER
- Step-6 Stop

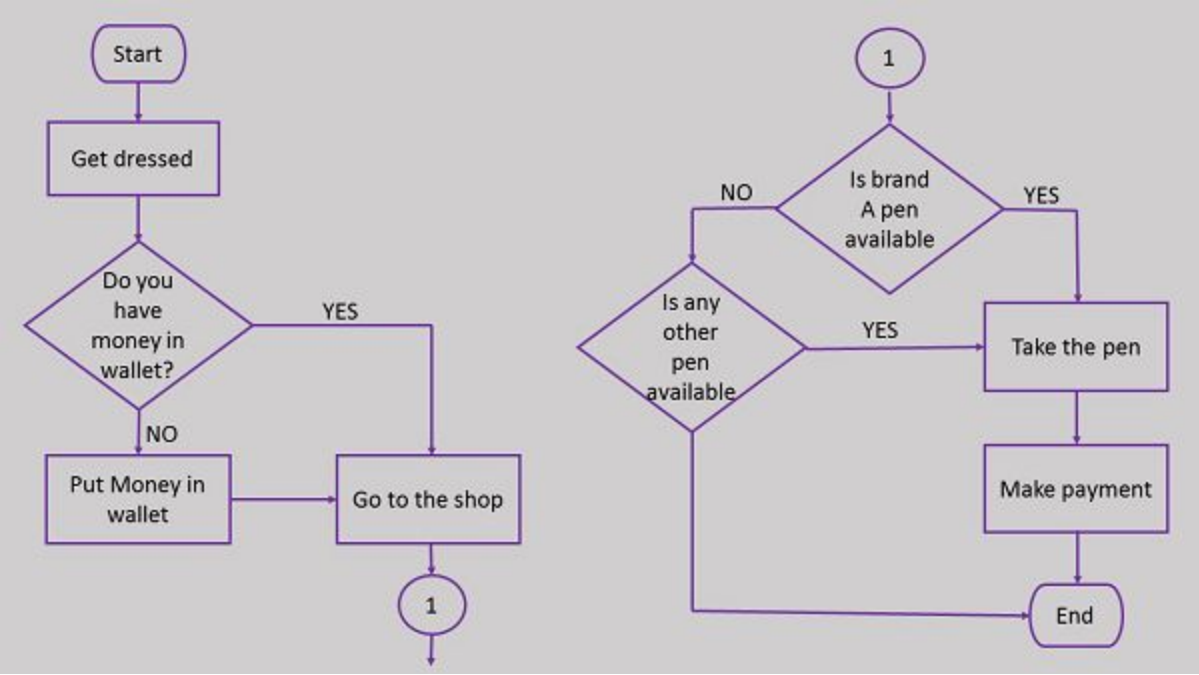
Flowchart:

- **Graphical representation of an algorithm.**
- It has been the primary logical tool of computer programmers.
- As the flowcharts are graphical representations, the logic of the programs they represent is easily understood than that of a written program or algorithm.
- It enable the programmer to finalise the logic of a problem independent of any programming language.
- Flowcharts **make use of special symbols , whose shape determines the kind of operation to be performed.**
- More than 25 symbols are there which are standardised by ANSI(American National Standards Institution)
- **Instructions are placed inside the symbols and connected by arrows to indicate order of execution.**

Symbol Name	Symbol	function
Oval		Used to represent start and end of flowchart
Parallelogram		Used for input and output operation
Rectangle		Processing: Used for arithmetic operations and data-manipulations
Diamond		Decision making. Used to represent the operation in which there are two/three alternatives, true and false etc
Arrows		Flow line Used to indicate the flow of logic by connecting symbols
Circle		Page Connector
		Off Page Connector
		Predefined Process / Function Used to represent a group of statements performing one processing task.
		Preprocessor
		Comments

On-page connectors example;

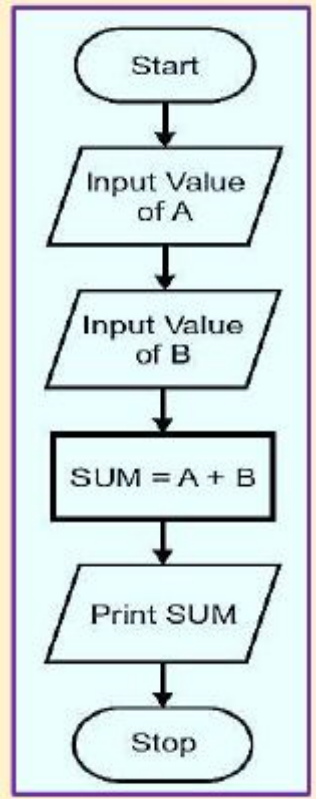
- Flowchart for going to the market to purchase a pen.



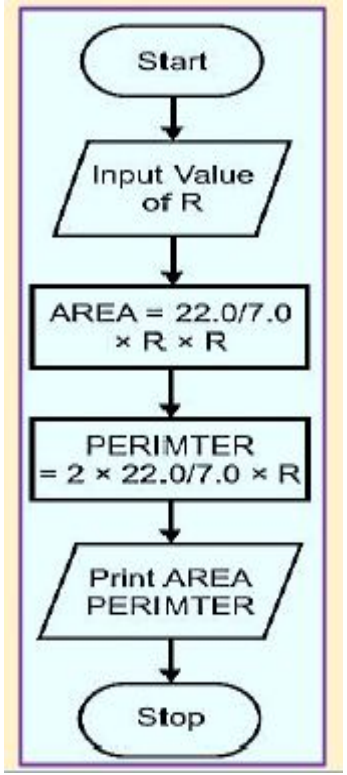
These are some points to keep in mind while developing a flowchart −

- Flowchart can have only one start and one stop symbol.
- General flow of processes is top to bottom or left to right.
- Arrows should not cross each other.
- On-page connectors are referenced using numbers.
- Off-page connectors are referenced using alphabets.

Flow chart to find sum of two numbers:



Flow chart to find area and perimeter of a circle



Pseudo code:

- It is a program design technique that uses informal expressions in outline form to describe the logic of a program.
- It is written in natural language rather than in a programming language.
- Pseudo code expressions are similar to those used in flowcharts,and is intended to be independent from the programming language used.
- It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.
- It is the cooked up representation of an algorithm.
- Pseudo code,as the name suggests, is a false code or a representation of code which can be understood by even a layman with some school level programming knowledge.

Advantages of Pseudocode

- Improves the readability of any approach. It's one of the best approaches to start implementation of an algorithm.
- Acts as a bridge between the program and the algorithm or flowchart. Also works as a rough documentation, so the program of one developer can be understood easily when a pseudocode is written out.
- In industries, the approach of documentation is essential. And that's where a pseudo-code proves vital.
- The main goal of a pseudo code is to explain what exactly each line of a program should do, hence making the code construction phase easier for the programmer.

Disadvantages of Pseudocode

- Pseudocode does not provide a visual representation of the logic of programming.
- There is **no proper format** for writing the for pseudocode.
- In Pseudocode, there is **extra need to maintain documentation**.
- In Pseudocode, there is no proper standard. Every company follow their own standard for writing the pseudocode.

Linear Search:

- Linear search is to find whether a number is present in a given list of numbers.
- If it is present, then at what location it occurs has to be indicated.
- If it is not present, message will be displayed showing that it is not present in the given list of numbers.
- It is straightforward way and works as follows:
    - Each element will be compared with the element to be searched until we find it or until the list ends.

Pseudocode:

```
procedure linear_search (list, value)
  for each item in the list
    if match item =value
      return the item's location
    end if
  end for
end procedure
```

Algorithm:

Linear Search (List A, Value x)

Step 1:Start
Step 2: Input the list to be searched, A
Step 3: Input the number to be searched, x
Step 4: Until end of list is reached,
Step 4.1:Compare x with each element of A sequentially
Step 4.1.1:If match is found, then output “found”
Step 4.1.2:If match is not found, then output “Not found”
Step 5:Stop

Bubble Sort:

- Sorting refers to **arranging data in a particular format.**
- Sorting algorithm specifies the way to arrange data in a particular order.
- Bubble sort is the simplest sorting algorithm.
- It is a **comparison-based sorting algorithm** in which **each pair of adjacent elements is compared and the elements are swapped if they are not in order.**

Illustrative example for Bubble sort:

Input List: 5 1 4 2 8

First Pass:

(5 1 4 2 8) –> (1 5 4 2 8)

(1 5 4 2 8) –> (1 4 5 2 8)

(1 4 5 2 8) –> (1 4 2 5 8)

(1 4 2 5 8) –> (1 4 2 5 8)

Second Pass:

(1 4 2 5 8) –> (1 4 2 5 8)

(1 4 2 5 8) –> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

Third Pass:

(1 2 4 5 8) –> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

Pseudocode for Bubble sort:

```
procedure Bubble_Sort(list)
  for all elements of list
      if list[i] > list[i+1]
        swap(list[i], list[i+1])
      end if
  end for
      return list
end procedure
```