

# Basic Computer Programming and Networking

---

RHNS Jayathissa

Department of Computer Engineering

Faculty Of Computing

# Content

---

- **Introduction to Computer Programming**
- **Dealing with Data**
- **C++ Operators and Selections**
- **Loops, Arrays and Functions**
- **Menu Driven Programming**
- **Computer Network and Communication**

# Course Structure

- Introduction
- Computer Programming
- Introduction to C++
- Input and output
- Variables
- Selections (if, switch)
- Loops (for, while, do-while)
- Arrays (1D, 2D)
- Console application
- Menu driven programming



# Method of Assessment

---

Lectures -20 Hours

Practical/Tutorials -20 hours

Continuous assessments - 30 %

–Practical Test -(10 marks)

–Quizzes -(20 marks )

End semester examination - 70 %

–4 Questions (2 hours) (70%)

# Why people use machines?

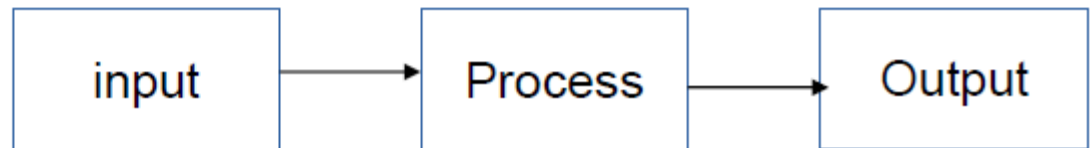


# Machines

- Café Machine → make a tea
- ATM machine → money transaction
- Calculator → solve equation

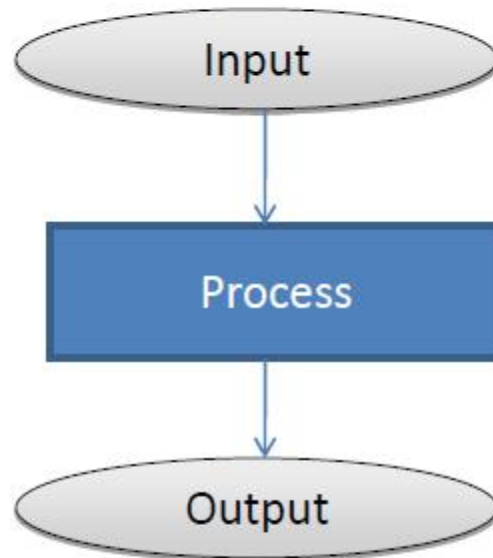
- Each Machine has ...

- Input
- Output
- Process



# Café Machine

- Input
  - Sugar
  - Water
  - Coffee
  - Milk
- Output
  - Tea
- Process
  - ?????



# Process of a Machine

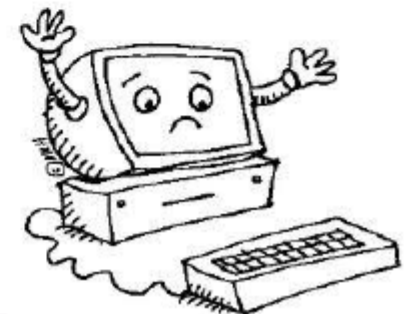
- Work through the **instructions** given to it
  - Read
  - Write
  - Do some calculations
  - Go to next instruction
- The sequence of instructions is call a **Program**

# What is a Programming?

- Programming is the way to give instructions
- Each program has
  - Start
  - Some work
  - End
- Program is given/execute through the machine understood language

# A Computer

- Machine that can **solve problems** for people by carrying out **instructions** given to it
- The sequence of instructions is called **Program**
- The language machine can understand is called **machine language**



# Machine Language

- Machine language is a set of instructions executed directly by a computer's central processing unit (CPU)

```
10100001 10111100 10010011 00000100
00001000 00000011 00000101 11000000
10010011 00000100 00001000 10100011
11000000 10010100 00000100 00001000
```

P<sub>1</sub>

OP <sub>[0-5]</sub>	RD <sub>[6-10]</sub>	RA <sub>[11-15]</sub>	DISP / IMM <sub>[16-31]</sub>		
OP <sub>[0-5]</sub>	RD <sub>[6-10]</sub>	RA <sub>[11-15]</sub>	RB <sub>[16-20]</sub>	OP-EXT <sub>[21-31]</sub>	
OP <sub>[0-5]</sub>	RD <sub>[6-10]</sub>	RA <sub>[11-15]</sub>	RB <sub>[16-20]</sub>	RC <sub>[21-25]</sub>	OP-EXT <sub>[26-31]</sub>
OP <sub>[0-5]</sub>	ADDR <sub>[6-29]</sub>				A:L
OP <sub>[0-5]</sub>	BO <sub>[6-10]</sub>	BI <sub>[11-15]</sub>	ADDR <sub>[16-29]</sub>		A:L
OP <sub>[0-5]</sub>	BO <sub>[6-10]</sub>	BI <sub>[11-15]</sub>	OP-EXT <sub>[16-31]</sub>		L

# Machine Language contd..

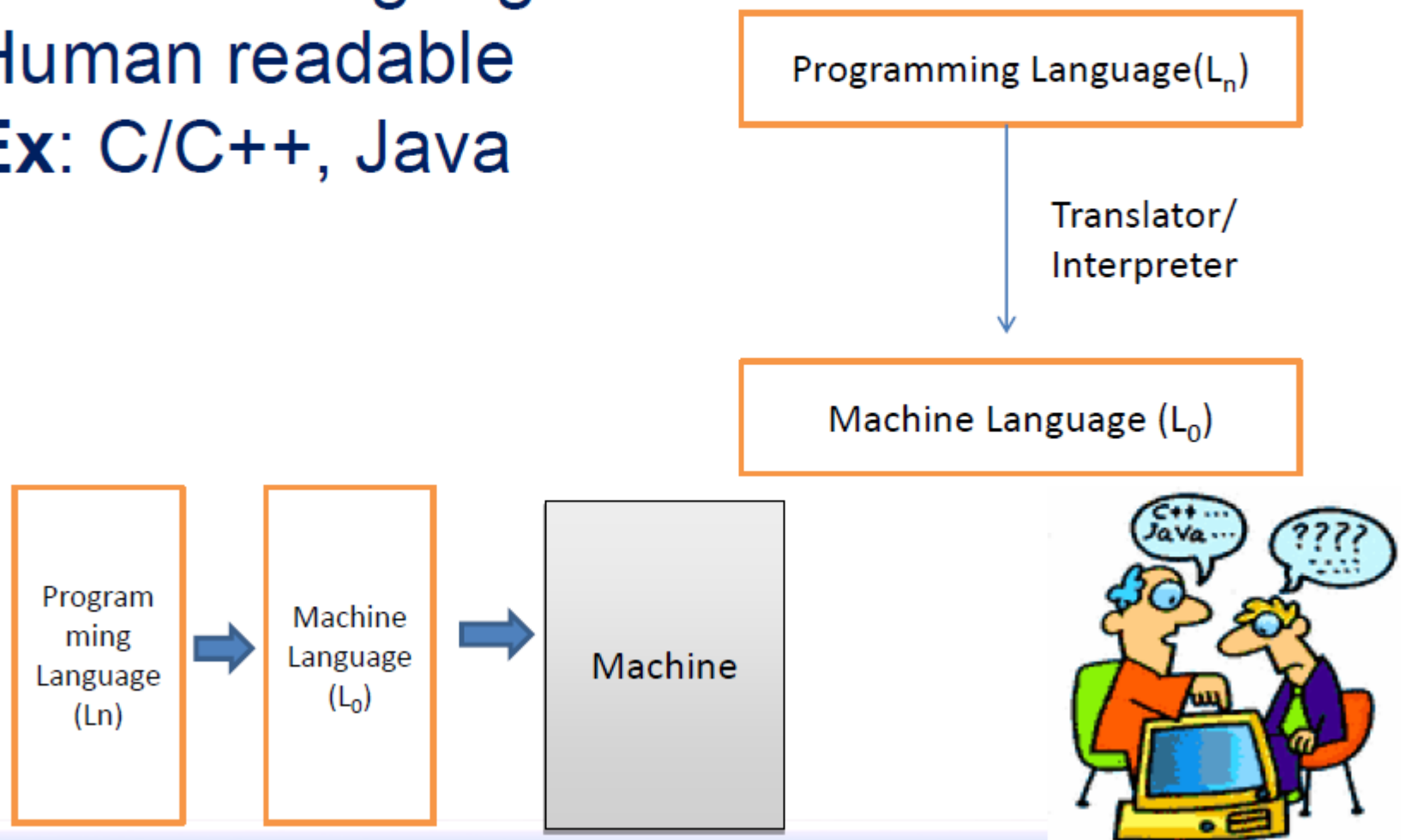
- Advantages
  - Machine can directly access (Electronic circuit)
  - High Speed
- Disadvantages
  - Human cannot identify
  - Machine depended  
(Hardware depended)



www.shutterstock.com · 34755871

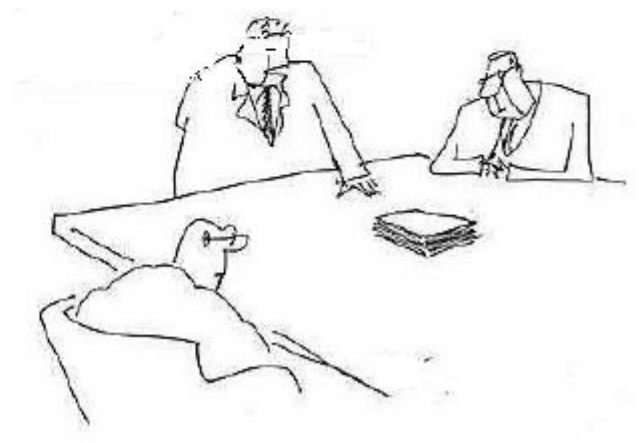
# Computer Language(s)

- Artificial Languages
- Human readable
- **Ex:** C/C++, Java



# Interpretation

- Each instruction in L1 can execute through the relevant L0 instructions directly
- Program is call **interpreter**



# Translator ( $L_1 \rightarrow L_0$ )

1. Replace each instruction written in L1 in to L0
2. Program now execute new program
3. Program is called compiler/ translator

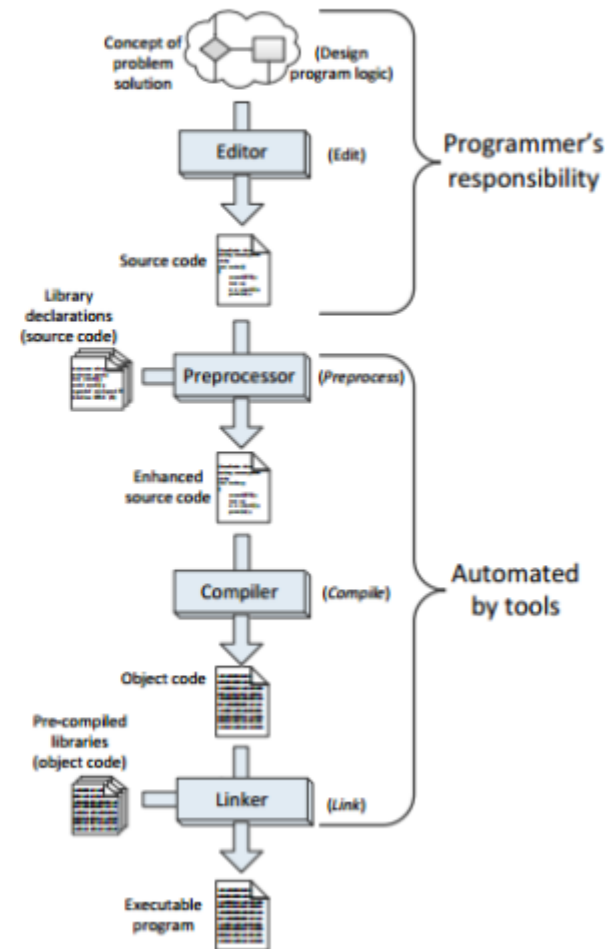


# Compilers

- Translate high-level language to machine language
- Input (**Source code**)
  - The original program in a high level language
- Output (**Object code/ Machine Code**)
  - The translated version in machine language
- Example
  - C++ compiler, JAVA Compiler etc.

# Programming steps

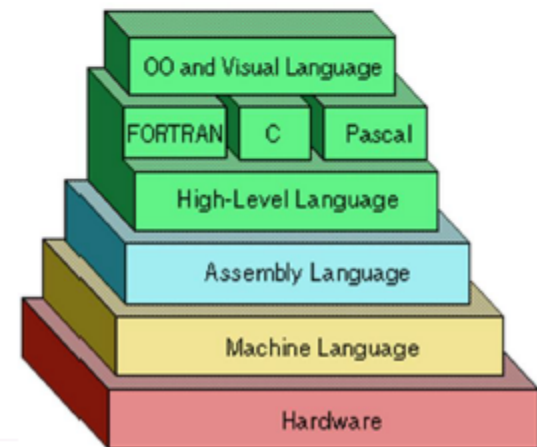
- **Compilers.** A *compiler* translates the source code to target code
- **Preprocessor**—adds to or modifies the contents of the source file before the compiler begins processing the code
- **Linker**—combines the compiler-generated machine code with precompiled library code or compiled code from other sources to make a complete executable program



# Programming language generations

This classification is used to indicate increasing power of programming styles

1. *First-generation programming languages*
2. Second-generation programming languages
3. *Third-generation programming languages*
4. Fourth-generation programming languages
5. *Fifth-generation programming languages*

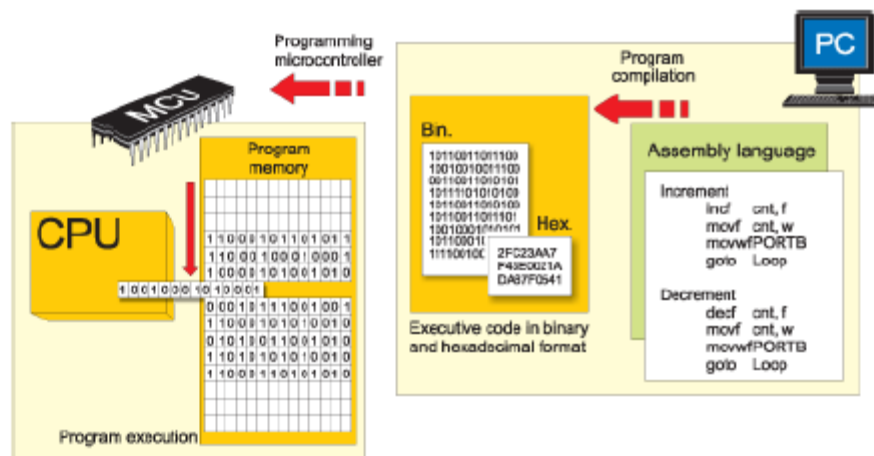


# First-generation programming language (1GL)

- Is a machine-level programming language
- Translator isn't used to compile
- The instructions in 1GL are made of binary numbers, represented by 1s and 0s
- Advantage
  - The code can run very fast and very efficiently because the instructions are executed directly by the CPU
- Disadvantage
  - When an error occurs, the code is not as easy to fix

# Second-generation programming language(2GL)

- Assembly language.
- Properties
  - The code can be read and written by a programmer
  - The language is specific to a particular processor family and environment
- Used in kernels and device drivers



Code generated by assembler

9D00002C	8FB00000	lw	s0,0(sp)
9D000030	03E00008	jr	ra
9D000034	27BD0004	addiu	sp,sp,4

Location of Machine language      Machine language      Disassembled machine language

# Third-generation programming languages (3GL)

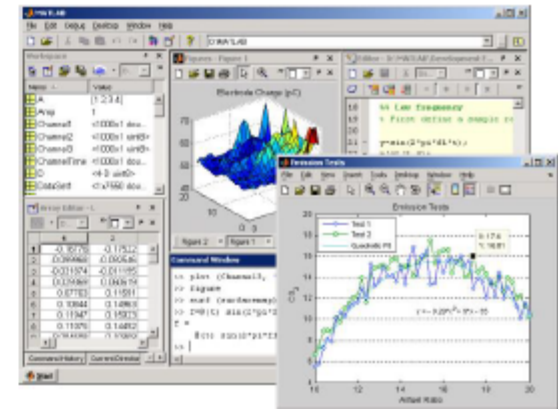
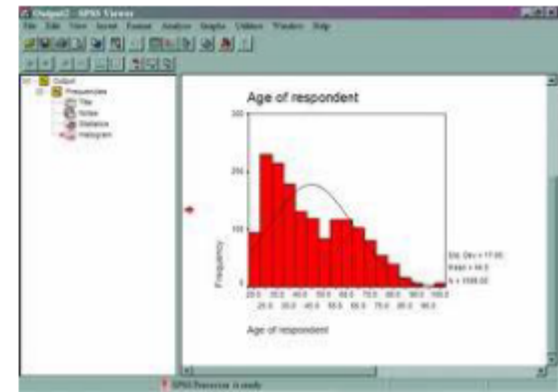
- Languages are more programmer-friendly
- Example
  - C, C++, C#, Java, BASIC and Pascal
- Support structured programming.
- Must be translated into machine language by a compiler or interpreter
- Advantages
  - Easier to read, write, and maintain

```
1 // class declaration
2 public class ProgrammingExample {
3
4     // method declaration
5     public void sayHello() {
6
7         // method output
8         System.out.println("Hello World!");
9     }
10 }
```

```
1 #include <iostream>
2
3 using namespace std;
4 int main()
5 {
6     int i, num;
7     cout<<"Enter a number\t";
8     cin>>num;
9     i=0;
10    while(i<=num)
```

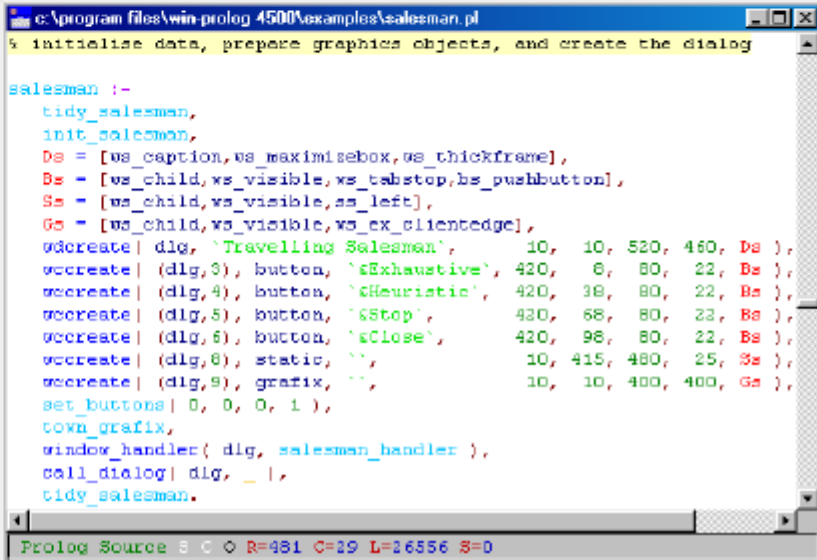
# Fourth-generation programming languages(4GL)

- Designed to reduce programming effort
- Consist of
  - Set of libraries
  - CRUD generators
  - Report generators
  - DBMS
  - Visual design tool and integration API
- Different types of 4GLs
  - Table-driven (codeless) programming
    - PowerBuilder
  - Data management
    - SAS, SPSS
  - Report-generator programming languages
    - Oracle Developer Suite



# Fifth-generation programming language(5GL)

- Based on **solving problems** using constraints given to the program, rather than using an algorithm written by a programmer
- Use mainly in **Artificial Intelligence** research
- Example
  - Prolog, OPS5, and Mercury



```
c:\program files\win-prolog 4500\examples\salesman.pl
% initialise data, prepare graphics objects, and create the dialog

salesman :-
    tidy_salesman,
    init_salesman,
    Ds = [ws_caption,ws_maximisebox,ws_thickframe],
    Bs = [ws_child,ws_visible,ws_tabstop,bs_pushbutton],
    Ss = [ws_child,ws_visible,ss_left],
    Gs = [ws_child,ws_visible,ws_ex_clientedge],
    wdcreate| dlg, 'Travelling Salesman', 10, 10, 520, 460, Ds ),
    wcreate| (dlg,3), button, 'sExhaustive', 420, 6, 80, 22, Bs ),
    wcreate| (dlg,4), button, 'sHeuristic', 420, 38, 80, 22, Bs ),
    wcreate| (dlg,5), button, 'sStop', 420, 68, 80, 22, Bs ),
    wcreate| (dlg,6), button, 'sClose', 420, 98, 80, 22, Bs ),
    wcreate| (dlg,8), static, '', 10, 415, 480, 25, Ss ),
    wcreate| (dlg,9), grafix, '', 10, 10, 400, 400, Gs ),
    set_buttons| 0, 0, 0, 1 ),
    down_grafix,
    window_handler( dlg, salesman_handler ),
    call_dialog| dlg, _ |,
    tidy_salesman.
```

Prolog Source S C O R=481 C=29 L=26556 S=0

# Selecting a suitable Computer Language

- Readability
- How easy to write the program in this particular language?
- Reliability
- How much would it cost to develop using a given language?
- How complicated the syntax going to be?
- Does the language have standards for greater readability?

# Exercise

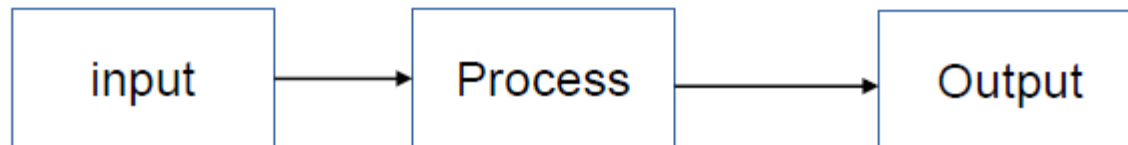
- To solve the following problems, identify the input, output and the process

Find the area of a room

Search a place of a city

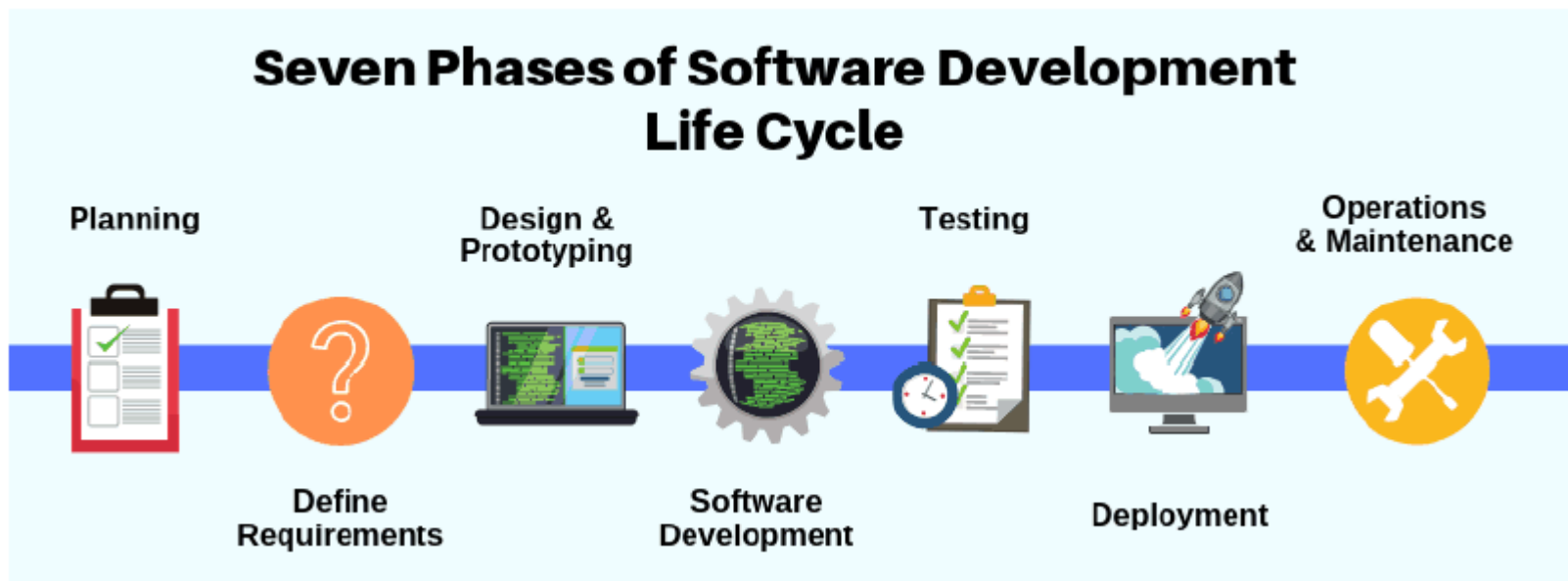
Calculate grade for the given mark

Get some amount from ATM machine



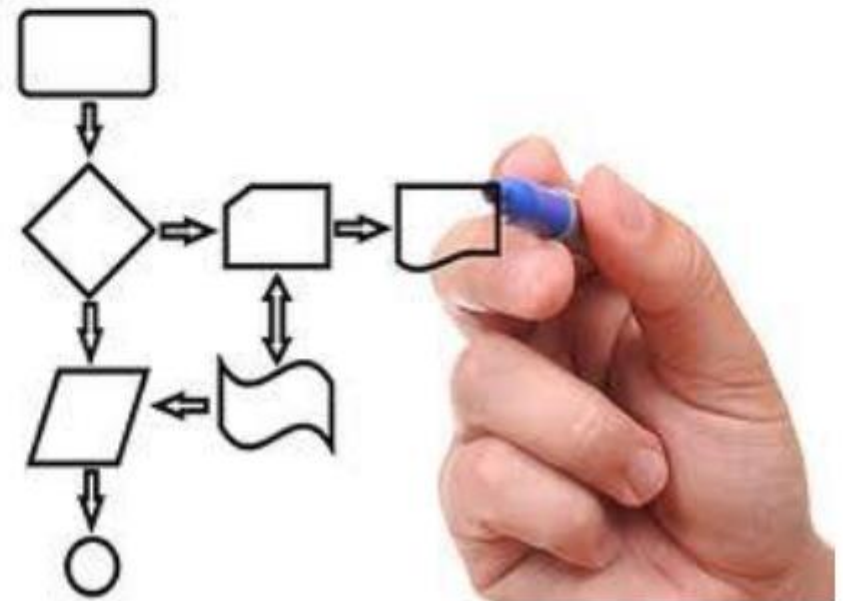
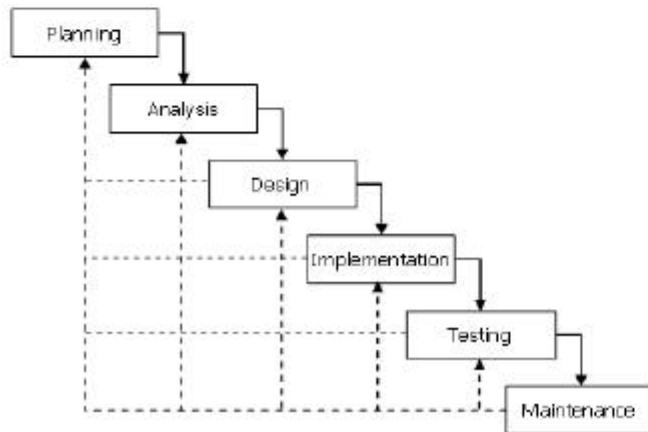
# Software Development Life Cycle

---



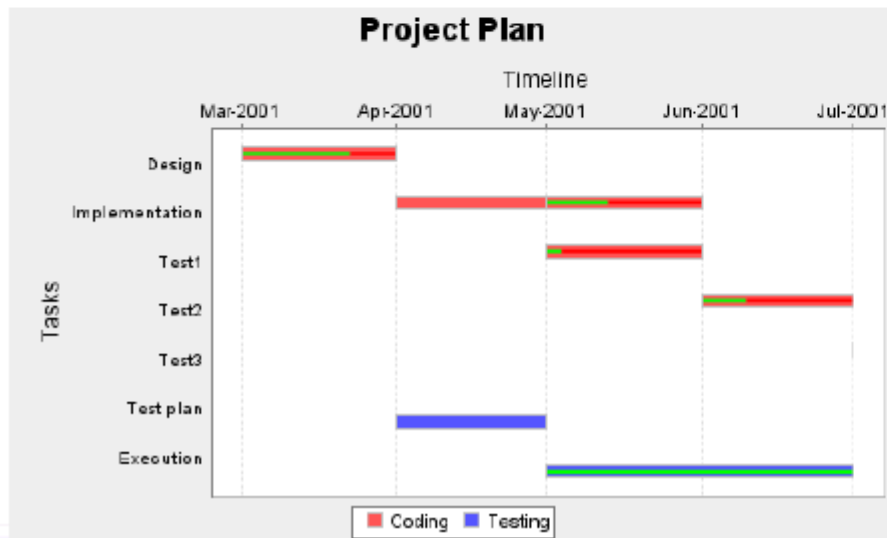
# Stages of Computer Programming

1. Planning
2. *Analysis*
3. *Design*
4. *Implementation*
5. *Testing*
6. *Maintenance and update*



# Planning

- Identify scope of the project
- Estimate the work involved
- Create a project schedule
- Begins with **requirements** that define the software to be developed.
- The **project plan** can be used to describe the task



# Analysis

- Is a complete description of the behavior of a system
- Consist of
  - Functional requirements
  - Non-functional requirements
- Methods
  - Interview
  - Questionnaires
  - Observation etc.



Customer Feedback Form

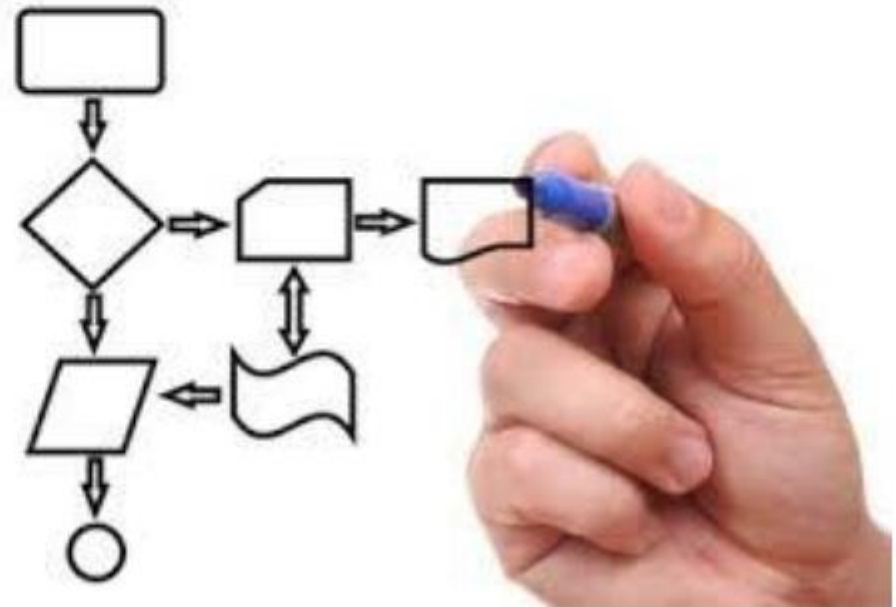
provide us feedback on

Excellent	Very Good	Average
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



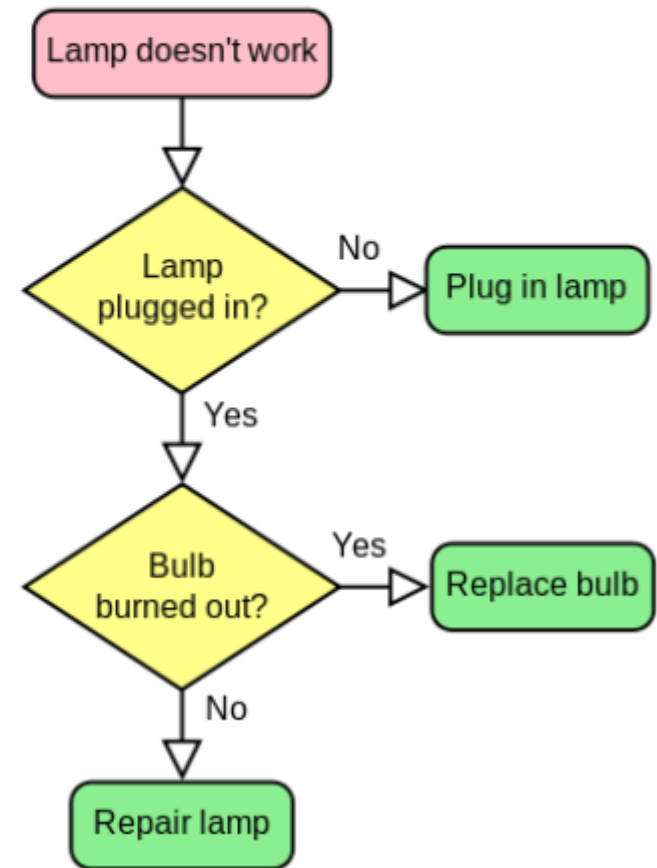
# Software Design

- Software design is a process of problem-solving and planning for a software solution
- Types
  - Top down
  - Bottom up
  - Module design
- Use to describe
  - Algorithm
  - Flowchart
  - Pseudo code




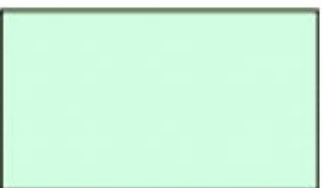
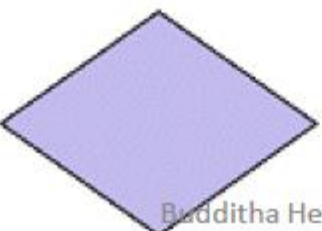


# Design cont....

- Flowchart:
  - is a type of diagram that represents an algorithm or process
  - Gives diagrammatic representation solution to a given problem
  - Use in analyzing, designing, documenting or managing a process or program



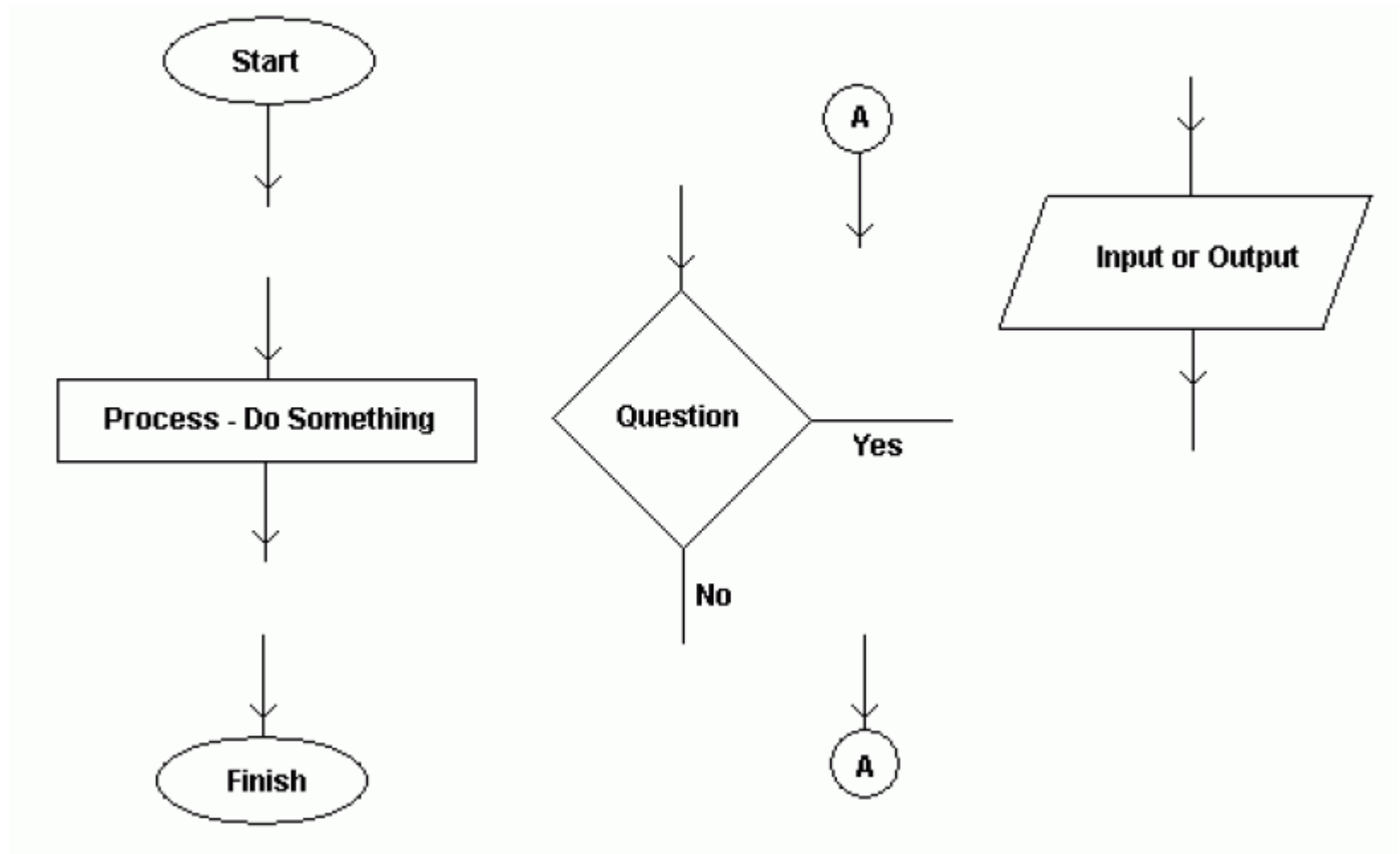
# Flowchart- building blocks

Name	Symbol	Use in flowchart
Oval		Denotes the beginning or end of a program.
Flow line		Denotes the direction of logic flow in a program.
Parallelogram		Denotes either an input operation (e.g., INPUT) or an output operation (e.g., PRINT).
Rectangle		Denotes a process to be carried out (e.g., an addition).
Diamond		Denotes a decision (or branch) to be made. The program should continue along one of two routes ( e.g., IF/THEN/ELSE).

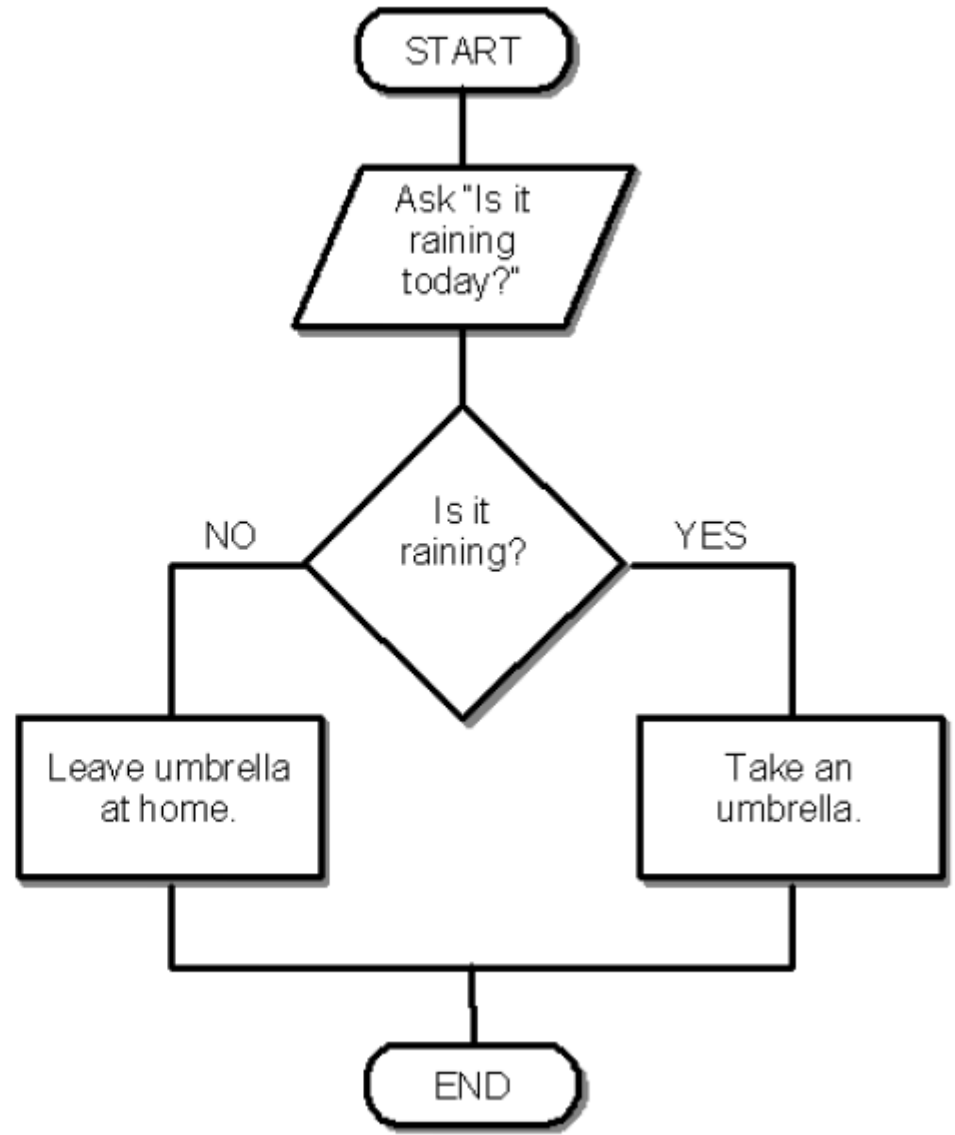
Budditha Hettige (budditha@yahoo.com)

# Flowchart-building blocks

## contd...





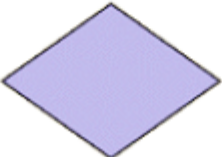


# Example



# Example

Draw a flow chart to display total of the the two numbers

Name	Symbol	Use in flowchart
Oval		Denotes the beginning or end of a program.
Flow line		Denotes the direction of logic flow in a program.
Parallelogram		Denotes either an input operation (e.g., INPUT) or an output operation (e.g., PRINT).
Rectangle		Denotes a process to be carried out (e.g., an addition).
Diamond		Denotes a decision (or branch) to be made. The program should continue along one of two routes (e.g., IF/THEN/ELSE).

# Example

- Draw a flow chart to identify correct login for the following interface

The image shows a login form titled "MEMBER LOGIN". It contains two input fields: "Username" with a person icon and "Password" with a lock icon. Below these fields is a large green button labeled "LOGIN".

# Implementation

- Is a realization of a technical specification or algorithm as a program, software component through computer programming
- May exist for a given specification or standard
  - Example World Wide Web Consortium-recommended specifications
- After implement
  - **Source code**, together with **documentation** to make the code more readable.



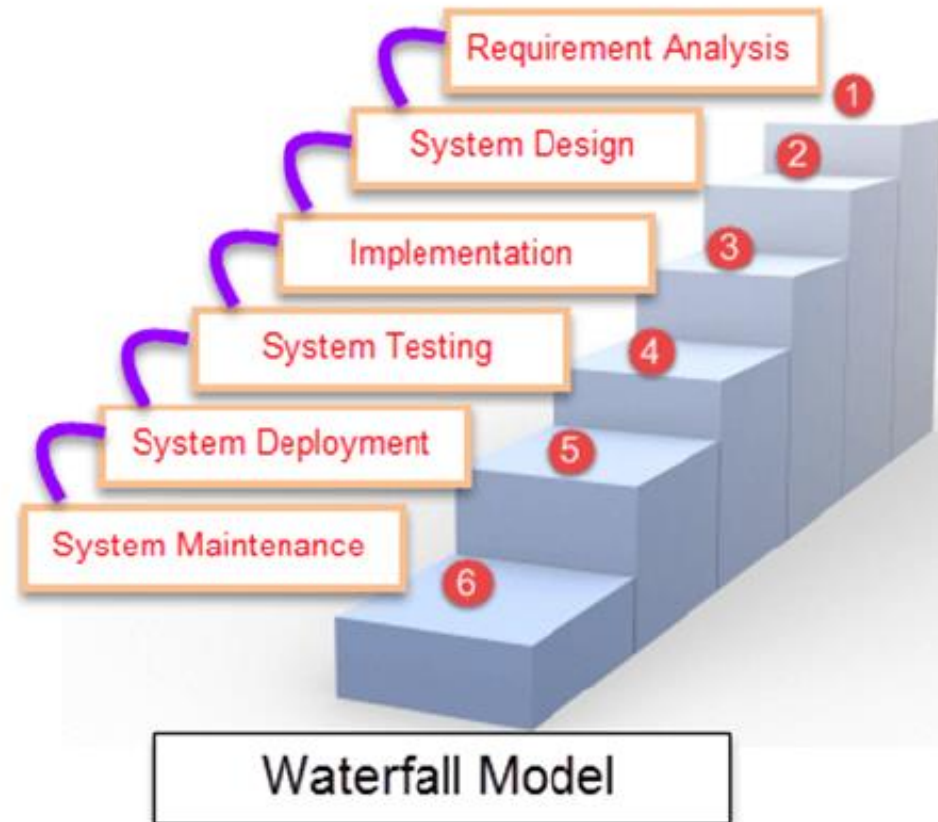
# Program Testing

- is an investigation conducted to provide information about the quality of the product
- A primary purpose of testing is to detect software failures
- Test
  - White box test
  - Black box test

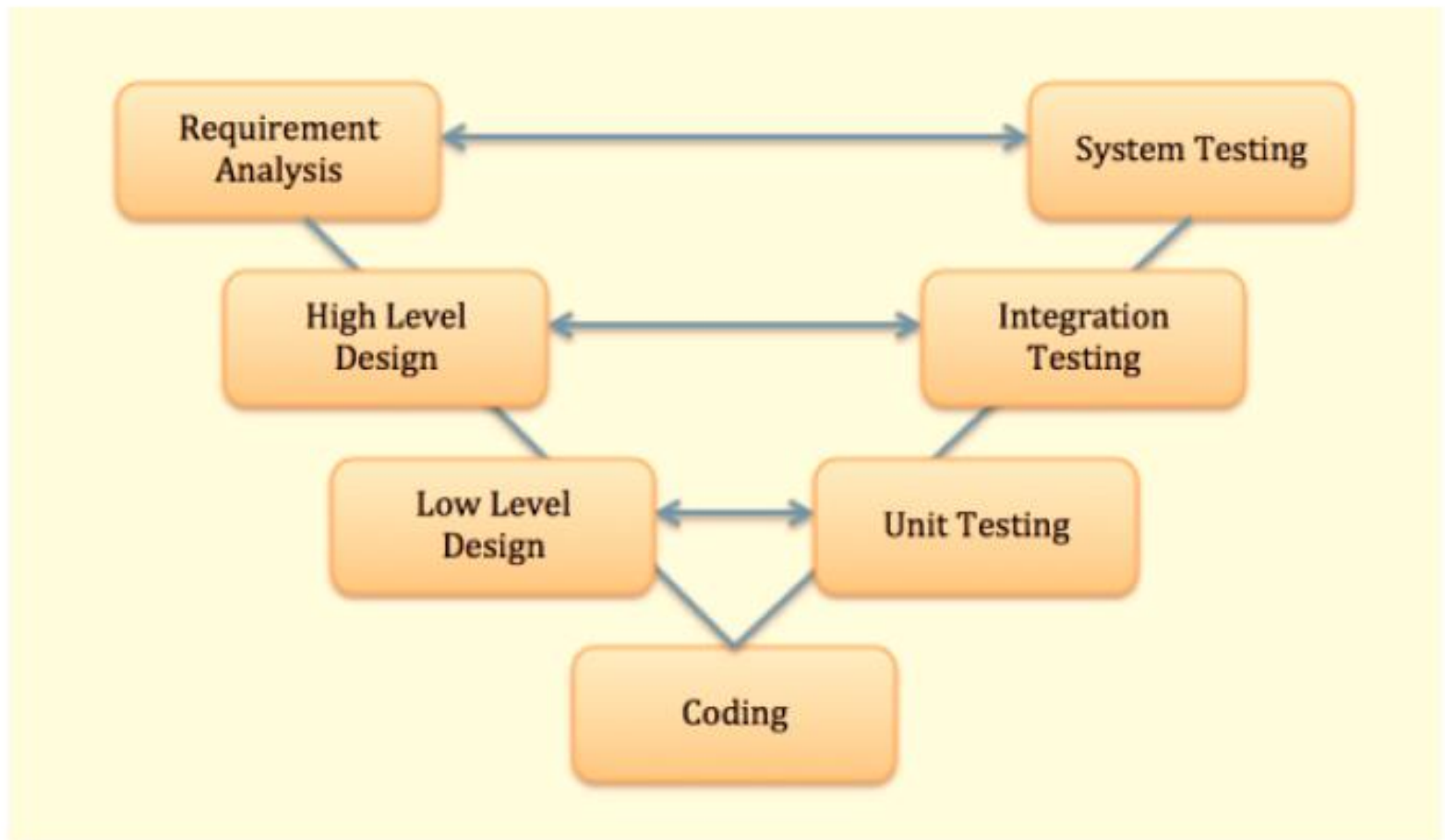
# Program Maintenance and Update

- Is the **modification** of a software product after delivery to correct faults
- Maintenance may span for 20 years, whereas development may be 1-2 year
- User guide, Maintenance manual need to provides
- Example
  - Windows XP (Development few years, Maintenance ....)
  - Provides set of software updates patches etc.

# Waterfall Model



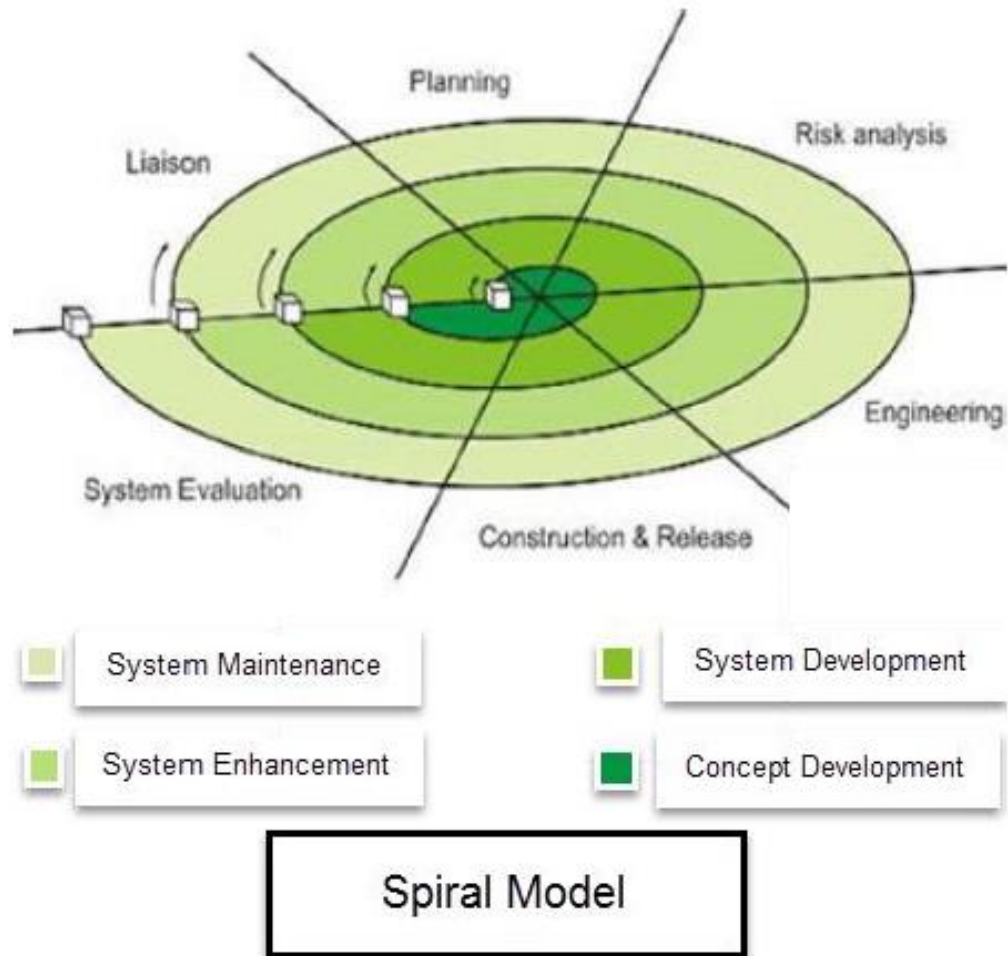
# The V Model



# Agile Model



# Spiral Model



# The Characteristics of a Good Computer Program

## ***Reliability:***

The program should provide correct results at all times and should be free from errors.

## ***Maintainability:***

The existing program should be able to change or modify to meet new requirements.

## ***Portability:***

The program should be able to transfer to a different computer system.

## ***Readability:***

The program must be readable and understandable with the help of documentation.

## ***Performance:***

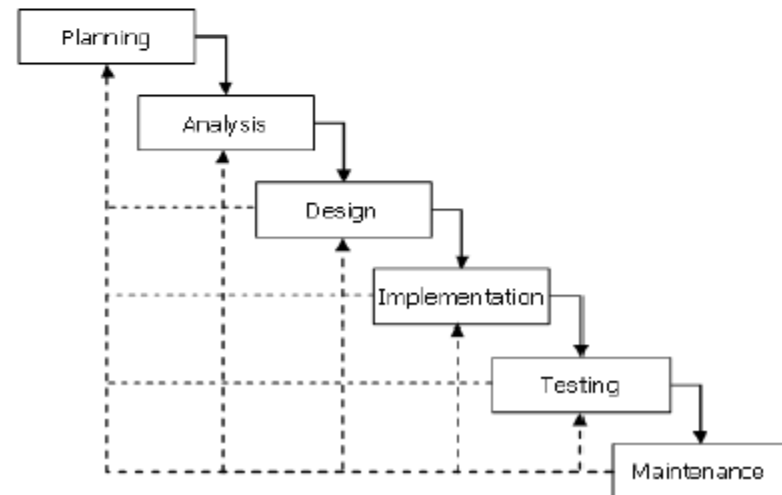
The program should handle the task more quickly and efficiently.

## ***Storage saving:***

The program should be written with the least number of instructions

# Steps to Computer programming

- Identify **Input, Output and process**
- Make a design
- Use suitable programming language and implement your system
- Test your program



# Tools & Tips for programming

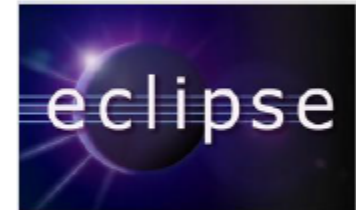
- Source code Editor
  - TextPad, Notepad
- Compiler
  - JAVA Compiler, C++ compiler
- Language Knowledge
- Logical thinking ability

# Programming with IDE

- IDE : integrated design environment
- consists of
  - source code editor
  - compiler and/or an interpreter
  - build automation tools
  - Debugger
  - Construction of a GUI
  - Class browser
  - Object inspector
  - Etc.

# Programming IDEs

- Eclipse



Eclipse is a multi-language software development environment

<http://www.eclipse.org/>

- Code:blocks



<http://www.codeblocks.org/>

- Netbeans

<http://netbeans.org/>



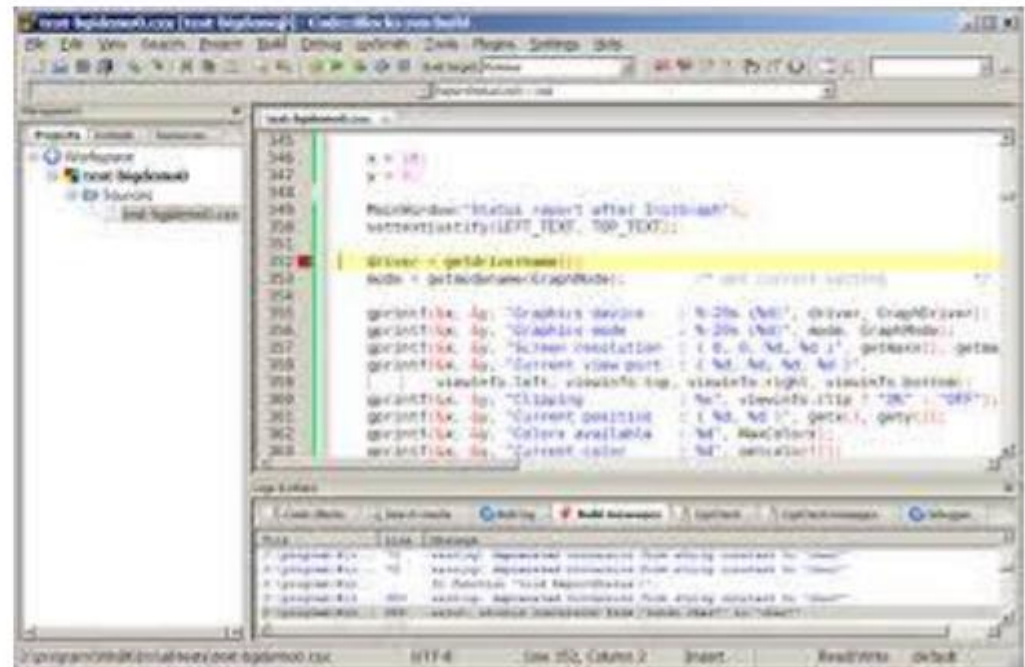
- Microsoft Visual Studio

<http://www.microsoft.com/visualstudio/en-us>

# Exercise

- Download code:blocks IDE and install it
- Download:

<http://www.codeblocks.org/>



# Summary

- What is a machine?
- Computer program
- Programming languages
- Design
- Characteristics of a good computer program
- Tools & Tips for computer programming

