Computer Languages
&
Program Logic

Computer languages

A computer language is an **artificial language** designed to work for different applications in different environments.

Programming languages can be used to create programs that control the behavior of a computer and serve any purpose.

Computer languages have some form of written specification of their **Syntax** (Rules) and **Semantic** (Words, phrase & sentences)

Computer languages

The term programming language usually refers to high-level languages such as **BASIC, C, C++, COBOL, FORTRAN, Pascal, Java, Oracle, ASP.NET, VB.NET and C#.**

Each language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

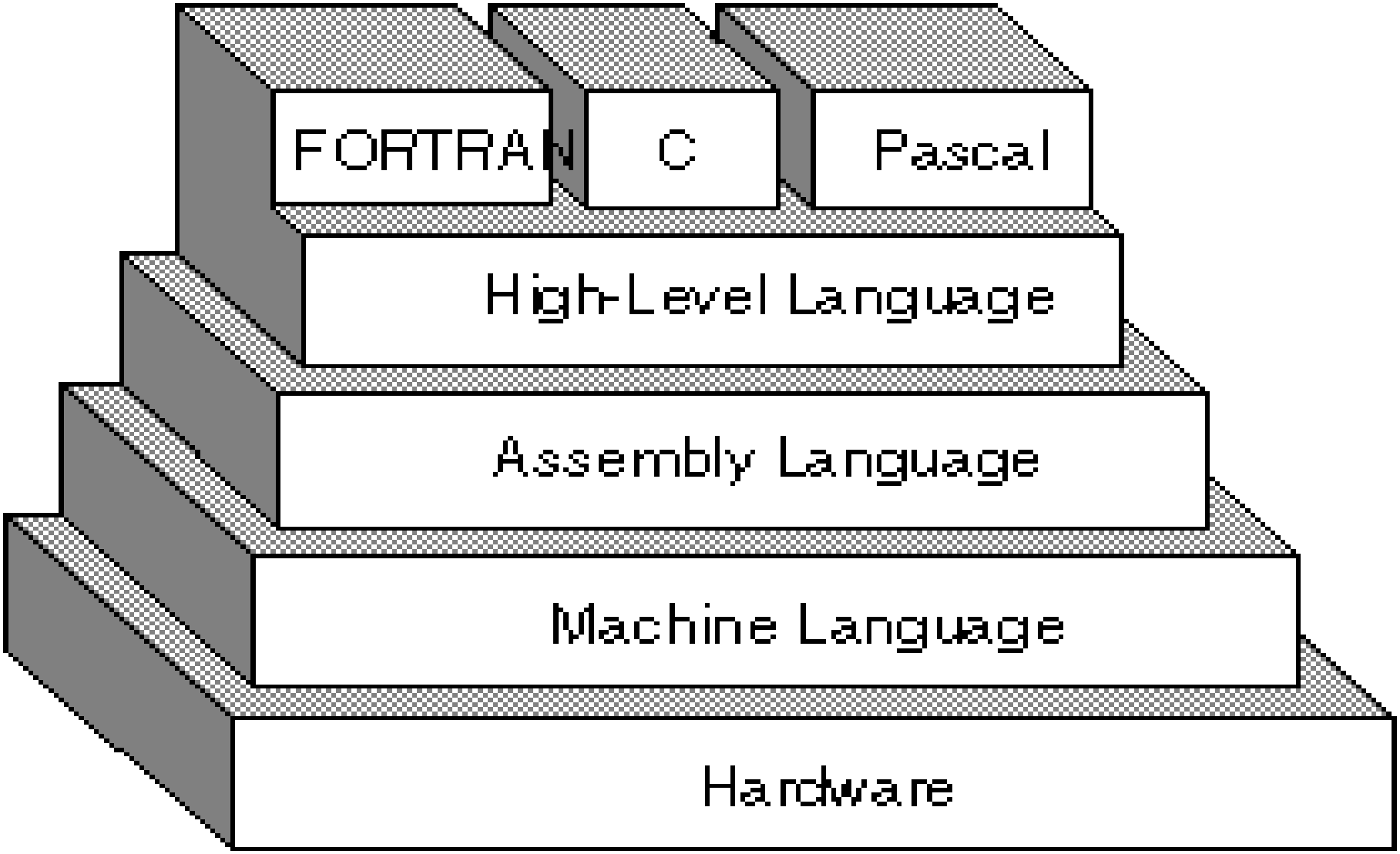
Computer Languages

Human languages are known as natural languages. Unfortunately, computers can not understand natural languages, as a result we must communicate with computers using computer languages.

These languages are;

- **High Level Languages**
- **Low Level Languages**
- **Assembly Language**
- **Machine Language**

Architecture



High Level Language

The high-level languages are much closer to human language.

A programming language such as C, FORTRAN or Pascal that enables to write programs which is understandable to programmer (Human) and can perform any sort of task,

such languages are considered high-level because they are closer to human languages.

High level language must use interpreter, compiler or translator to convert human understandable program to computer readable code (machine code).

Examples:

Many high level languages have appeared since Fortran II the most widely used have been:

COBOL	**Business applications**
FORTRAN	**Engg & Scientific Applications**
PASCAL	**General use and as a teaching tool**
C & C++	**General Purpose - currently most popular.**
PROLOG	**Artificial Intelligence**
JAVA	**General all purpose programming**
.NET	**General or web applications.**

Advantages of High level language over low level lang:

- The main advantage of high-level languages over low-level languages is that they are easier to read, write and maintain.
- High-level languages make complex programming simpler.
- High level programming techniques are applicable everywhere even where computational resources are limited.
- Error ratio is less in high level language and debugging (locate and correct errors in program code) is easier.
- Length of the program is also small compared with low level.
- Many real time problems can be easily solved with high level language.

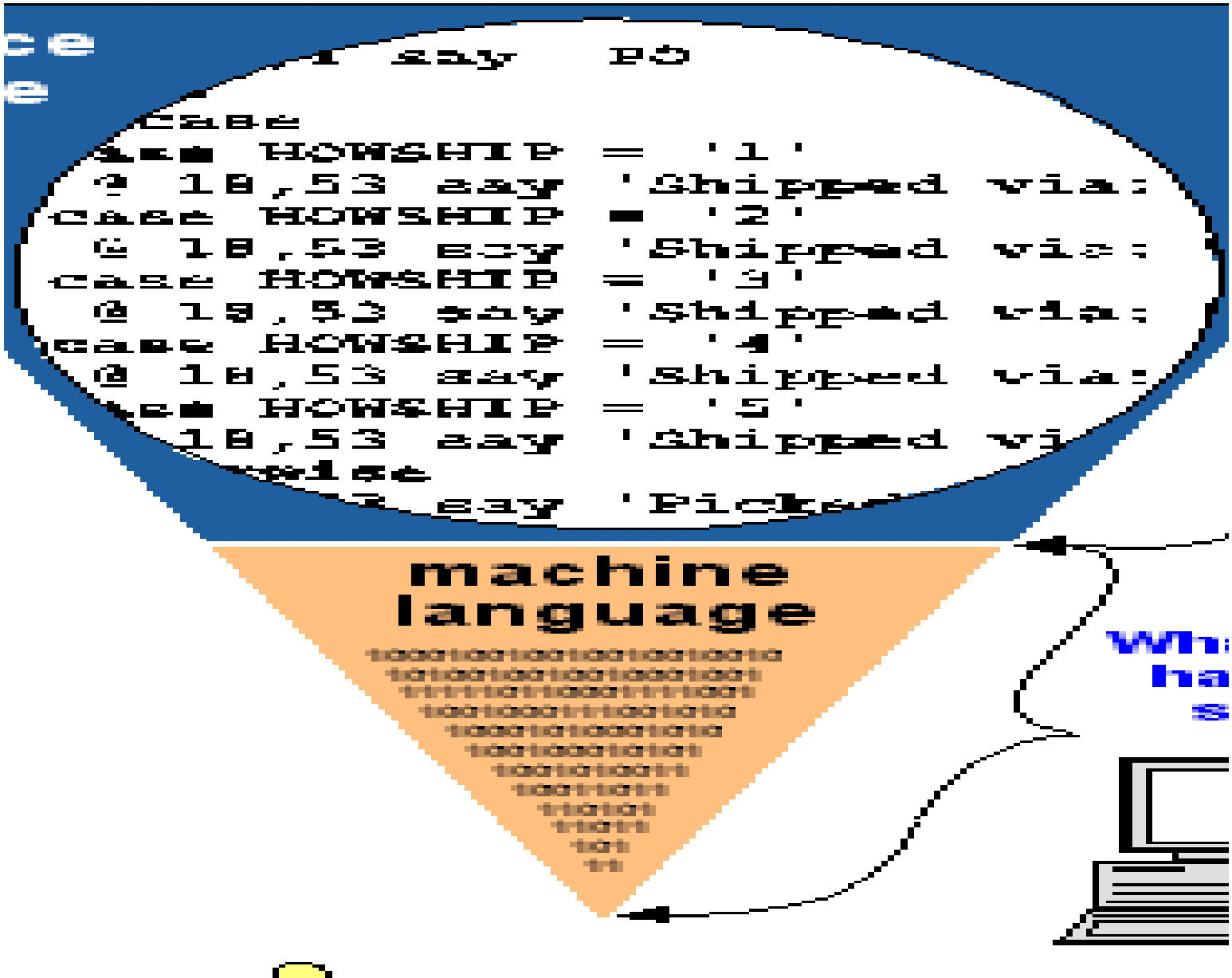
Low Level Language

A computer low level language that deals with hardware registers by name known as **assembly language**.

Assembly language is the best example of low level language, this is in between machine language and high-level language.

A low-level language does not need a compiler or interpreter to run the program, the processor run low-level code directly.

Example of Low-Level code & machine language



Assembly Language

Assembly languages have the same structure and set of commands as machine languages, but they enable a programmer to use names instead of numbers.

In the early days of programming all programs were written in assembly language but now almost programs are written in a high-level language.

Programmers still use assembly language when speed is essential or when they need to perform an operation that isn't possible in a high-level language.

Assembly Language

It uses **mnemonic codes** (short forms) for instructions and allows the programmer to introduce names for blocks of memory that hold data.

Assembly language is designed to be easily translated into machine language. Like machine language, assembly language requires detailed knowledge of a particular internal computer architecture

```
mov edx, [esp+8]
cmp edx,
0 ja @f
mov eax, 0
ret
```

Machine language

Machine code or machine language is a system of instructions and data executed directly by a computer's CPU, The lowest-level programming language that only be understood by computers.

Computer language that is directly executable by a computer without the need for translation by a compiler or an assembler.

Machine Language

The native language of the computer,

The set of symbolic instructions in binary that is used to represent operations and data in a machine called machine code

Machine Language: “0110101100101000”

machine language is a collection of binary digits or bits that the computer reads and interprets.

Machine language is the only language a computer understands, It is almost impossible for humans to use because they consist entirely of numbers.

Program code converters

There are four models of execution of program by processor in high-level languages:

- **INTERPRETER**
- **COMPILER**
- **ASSEMBLER**
- **TRANSLATER**

Interpreter

An interpreter is a program that executes programming code directly.

Interpreter can convert a source code , usually on a step-by-step, line-by-line and unit-by-unit basis into machine code.

Interpreted languages are read and then executed directly, with no compilation stage.

Compiler

Compiler is a program that compile source code into executable instructions that a computer can understand, it compiles code and produce in **EXE** format before it run.

There are two stages of compiler

Intermediate Compilation

Intermediate level compiles code in the form of **Byte code** and save for later execution without the need to re-read the source file.

Machine code Compilation

Machine code compiles source code directly into machine code using **Virtual machines** that execute code directly into machine code

Assembler

Assembler normally converts assembly language's source code into machine language.

Assembly language is a family of low-level languages for programming computers. It implements a symbolic representation of code.

An assembler is a program that takes basic computer instructions and converts them into a pattern of **bits (0 & 1)** that the computer runs to produce result.

Translator

Translator is a computer program that translates one programming language instruction(s) into another programming language instruction(s) **…**

A computer source code is to be translated into a

low-level programming language where native code compilers change it in machine language for execution.

What is program

Program is an organized list of instructions that, when executed, it perform any operation, and do the job it is also known a code.

Without programs, computers are useless.

A program is like a recipe. It contains a list of variables and a list of directions (Statements) that tell the computer what to do with the variables.

The variables can represent numeric data, text or graphical images.

What are Keywords

Keyword is a word that is reserved by a program because the word has a special meaning an purpose in any computer language.

Keywords can be commands or parameters (values) .

Every programming language has a set of keywords that cannot be used as variable names.

Keywords are sometimes called reserved names.

What is instruction

The term instruction is often used to describe the programming commands.

computer’s instructions set is the list of all the basic commands in the computer's machine language.

Language Generations

Computer languages has the same history as the computers itself history,

There are five generations of languages when programming method and techniques could be developed as far as development in hardware occurred.

First Generation (1GL)

First-generation language was machine language or the level of instructions and data that the processor is actually given to work on **binary numbers 0s and 1s.**

First-generation language is the lowest level computer language. Information is conveyed to the computer by the programmer as binary instructions.

In the 1940s and 1950s, computers were programmed by scientists sitting before control panels equipped with toggle switches so that they could input instructions as strings of zeros and ones.

Second Generation (2GL)

Second-generation language used assembly language, An assembler converts the assembler language statements into machine language.

By the late 1950s, this language had become popular. Assembly language consists of letters of the alphabet.

This makes programming much easier than trying to program a series of zeros and ones.

Second-generation language use increased when Grace Hopper developed FLOW-MATIC, a language that made programming easier.

Third Generation (3GL)

Third-generation language is a high-level programming language, such as PL/I, COBOL, FORTRAN and C.

Compiler based FORTRAN and COBOL created because it improved the record keeping and data management ability in many business applications.

In the early 1960s, BASIC language designed and used by nonprofessional programmers. BASIC remained popular during the 1970s and 1980s.

Programs written in third-generation languages require more memory and run more slowly than those written in lower level languages.

Fourth Generation (4GL)

Fourth-generation language is designed to be closer to natural language than a 3GL language.

Fourth-generation languages attempt to make communicating with computers as much like the processes of thinking and talking to other people as possible.

Fourth-generation languages typically consist of English-like words and phrases, some of these languages include graphic devices such as icons and onscreen push buttons for use during programming and when running the application.

Many fourth-generation languages use C, C++, Java and SQL that support stand-alone as well as web applications.

Fifth Generation (5GL)

Fifth-generation language is programming that uses a visual or graphical development interface to create source language that is usually compiled with a 3GL or 4GL language compiler.

Microsoft, Borland, IBM, and other companies make 5GL visual programming products for developing applications in Java, J++, Visual Java, Visual Basic, ASP, ASP.NET, VB.NET and C#.

Visual programming allows you to see object oriented structures and drag icons to assemble program blocks.

Programme
&
Logic

Programme and Logic

Programme:

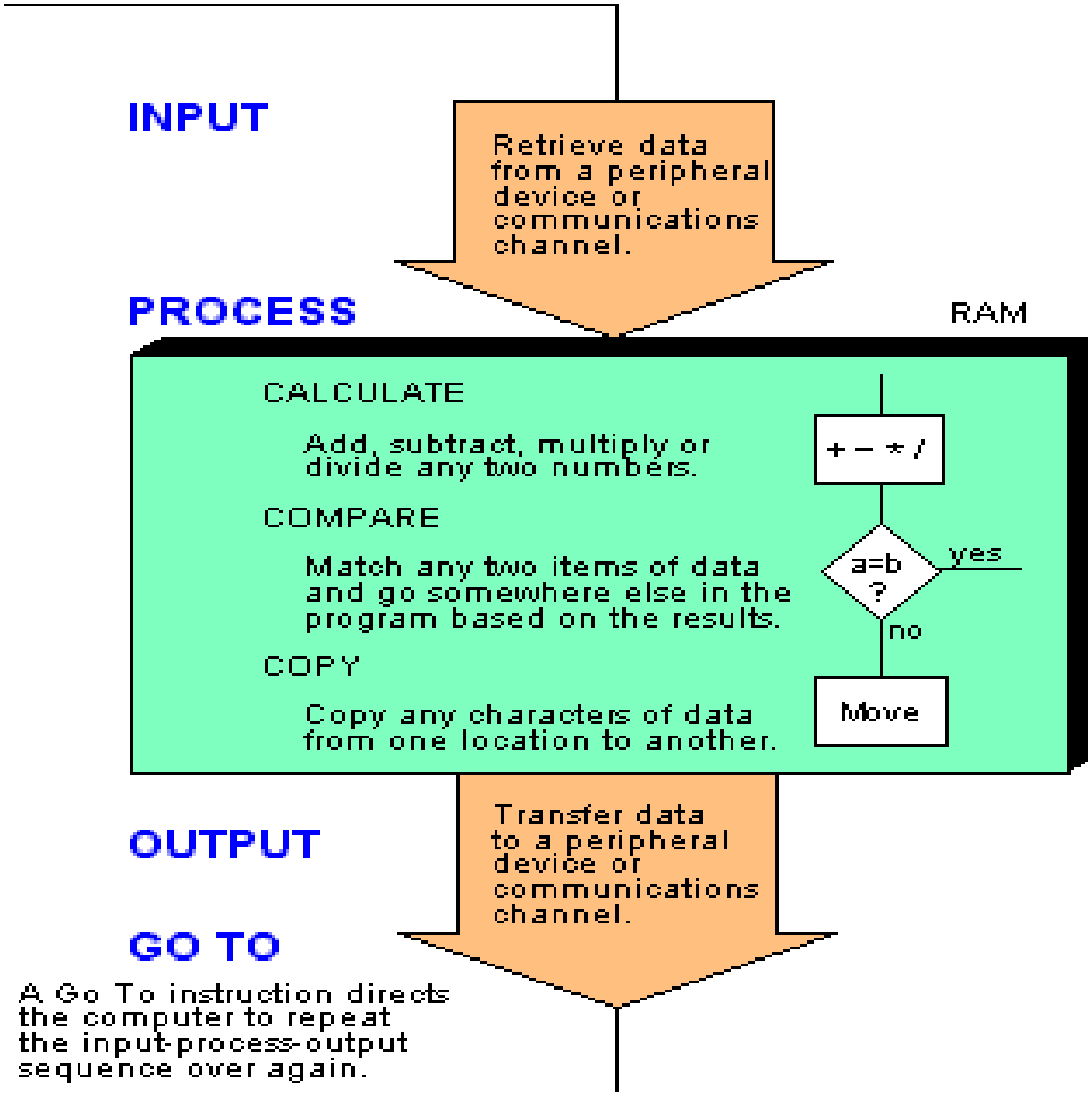
The series of instructions in a program arranged in a prescribed order to solve a problem called the ***program***.

Logic:

It is the method or technique to process a program in which user write program with reasons, techniques and methods known as ***program logic***.

The best program is one which has the fast execution and easily **changeable logic** according to environment.

Programme and Logic



Flow Chart - Logic

Flowchart is a visual representation of the sequence of the instructions.

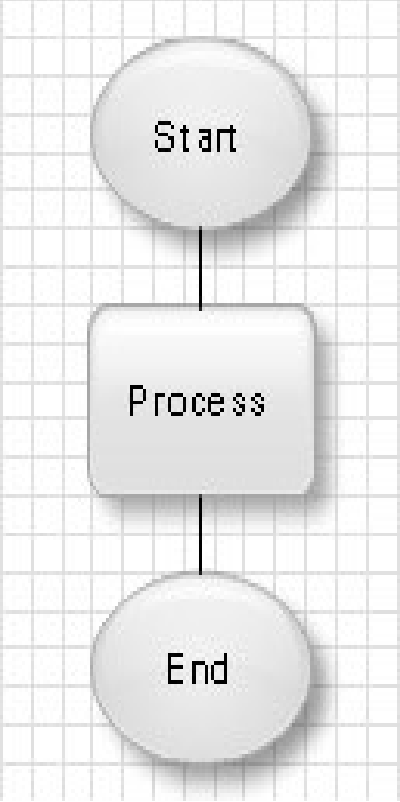
It shows what comes first, second, third and used to

Analyzing,

Designing,

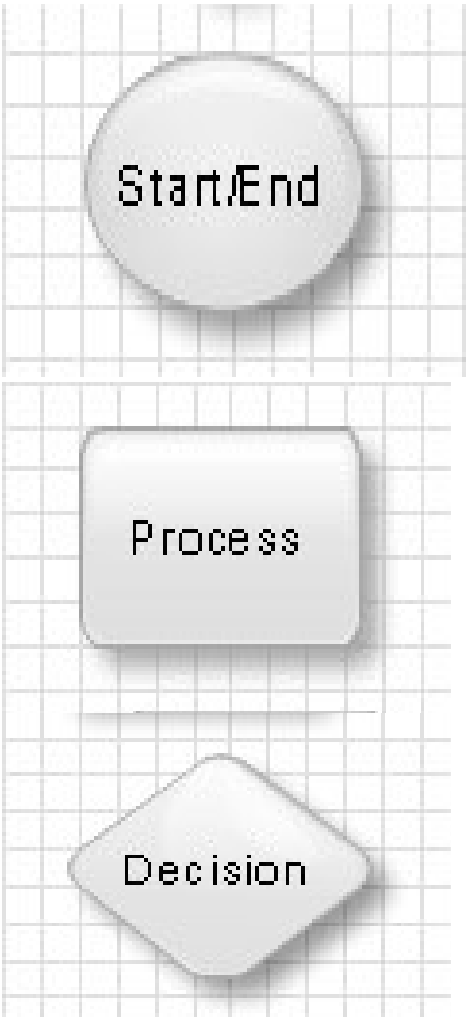
Documenting

Managing a process or program



Flow Chart

A flow chart's blocks shows different states of the process. There are three basic symbols of the flowchart.



Circles: Show Start/End points of program.

Squares: which represent activities or tasks

Diamonds: which represent decision points

Arrows: shows direction of flow.

Steps of flow chart

Sequential Processing:

These instructions do the actual data processing in sequence order, such as input, add/subtract, copy and output.

Decision Making:

The decision making within the program is the process of comparing or evaluation of two sets of data of the program based on the results.

Iteration:

Many routines must be repeated for some number of times and are accomplished with the help LOOPS till the desired result.

Flow Chart Example