SOFTWARE

• What is Software?
• A set of programs.
• In the earlier days, programs were written in 0’s and 1’s. (machine language, or Low Level Programming)
• Disadvantage: Very cumbersome, and debugging becomes a problem.

SOFTWARE..contd.

• Remedy?
• Using mnemonics. This was called Assembly Language Programming (ALP)
  • E.g. MOV AX, BX
  • But this needed a translator to convert AL to ML. This translator was called Assembler.

SOFTWARE..contd.

• Disadvantage of AL and ML?
• They were not portable. I.e., once written, they were good only for that machine. Machine Specific.
• Remedy?
• Develop a language that is machine independent, and has more powerful features to suit specific needs.

SOFTWARE..contd.

• High Level Languages
• These have the following features:
  • Machine Independence
  • Very good debugging facility
  • Good documentation
  • 1 instruction in a HLL corresponds to several in ALL or LLL

SOFTWARE..contd.

• High Level Languages meet specific needs such as business, scientific etc.
• HLL’s come with their own translators, known as Compilers or Interpreters, and other software called Linkers, Loaders, Debuggers and Libraries.
• Compilers convert the source code to object code, which is in machine language.
SOFTWARE..contd.

- How are HLL programs executed?
- Compiler/Interpreter translates the source code.
- The Loader, which is software, loads the object code into main memory.
- The Linker links external files used by the current file.

SOFTWARE..contd.

- The editor, compiler, library, linker, loader, debugger can be integrated into one environment called the IDE, or Integrated Development Environment. MS Visual Studio is an e.g.
- Compiler/Interpreter translates the source code.
- The Library is a catalogue of predefined, compiled and tested functions. These contain geometric, arithmetic and other commonly used functions.
- Users can build their own library functions.

SOFTWARE..contd.

- Programs are broadly divided into 2 categories:
- Systems programs, which direct the internal operations of the computer.
  - E.g. Linker, Loader, Compiler
- Applications Program, which solve user-oriented problems
  - E.g. payroll software, banking software.

Introduction to Programming

- Developing the simplest programs require the following steps:
  - Start->Problem Identification->Task Analysis-> Data analysis-> Deciding I/O-> developing algorithms->Coding->Testing-> Debugging-> Result

Introduction to Programming

- The algorithm phase is very important.
- What is an algorithm?
- A set of explicit, clear finite steps to obtain the desired output in finite time.
- It is developed to produce the most efficient approach to solve a problem.
- Efficient in terms of speed and memory usage.
- Coding is implementing an algorithm in the programming language syntax.

Intro. to Programming..contd.

- The goal of programming is to develop programs that are clear, efficient and robust.
## HLL’s..C and C++

- C and C++ are entirely different languages.
- Syntactically similar
- But approach to programming in C++ is very different from C.
- C++ is considered the language of the future.

## Object Oriented Programs

- Pascal, BASIC, C, Fortran are all called Procedural languages.
- This is because a program is a whole list of instructions telling the computer what to do.
- What happens, when a program is very large?
  - List of instructions become unwieldy
  - Comprehension becomes tough
  - Unmanageable code.

## Object Oriented Programs

- Remedy?
  - These problems led to development of structured programming.
  - Here a program is split into small functions, which is more manageable and readable.
  - This idea was extended to group a similar set of functions into what are called modules.

## Object Oriented Programs

- Problems with structured programming:
  - Data Undervalued
  - Action on data is given more importance than the data itself.
  - This leads to inconsistencies among especially global data, which is accessed by many functions, giving undesired outputs at times.

## Object Oriented Programs

- Remedy?
  - We need to hide data, or restrict its access to certain functions alone.
- Other problems:
  - Non-oop languages allow users to use only built in data types.
  - What if the user wants to invent his own?

- This led to development of the object oriented approach.
  - Fundamental idea is to combine data and functions into 1 unit called OBJECT.
  - Here, the functions in an object only have access to the data within the object.
  - No external functions allowed access to that data.
  - Data encapsulation is born. Data and functions are encapsulated in an object.
Object Oriented Programs

• Data Hiding is achieved, by restricting access to data only to the member functions (or functions within an object.)
• Analogy:
• Corporate structure Vs. Object Oriented Design.