**Lecture 1: Introduction to Programming**

* 1. **Introduction**

****Computers have been in existence for over fifty years now. The key components of computer are hardware and software. Software is used to run the hardware of the computer. The method used to develop computer software is called programming.

This lecture is focused of introducing students to programming. This foundational knowledge and skill in programming will be of great importance in developing ones career in the field of programming as well as applying programming solutions to various problems in the society.

This lecture will cover the following:

1. Lecture objectives
2. Definition of terms
3. Programming Languages
4. Language translators
5. Self-test questions
6. Summary
7. Reference
   1. **Lecture Objectives**

****At the end of this lecture the student should be able to:

1. Define terms in used in programming.
2. Explain various levels of programming languages.
3. Discuss programming language translators.
   1. **Definition of terms**

**Definition of Terms**

**Program**

This is a complete set of step-by-step instructions that control and direct the computer hardware in carrying out a given task. Tasks may vary from very simple e.g. computing surface area to complex ones like statistical analysis. Programs are usually written to solve user problems on a computer.

***Software***

This term refers to all programs together with their associated codes.

***Programming Language***

This is a set of symbols and the rules that govern their rules that are employed in the construction of a computer program.

***Programmer***

This is a person who is trained and/or specializes in the technique of creating, maintaining and modifying computer programs.

***Syntax***

These are the rules of a language that govern the ways in which words, symbols, expressions and statements may be formed and combined in that language.

***Semantics***

These are the rules of language that govern meaning of statements in any language.

* 1. **Programming Languages**

**Programming Languages**

Programming languages provide the basic building block for all software. They are the means by which people can tell the computer how to carry out a task.

A program can be written in a variety of programming languages. The languages can broadly be classified into two categories:

* Low-level language – which refers to the machine language and assembly language.
* High-Level languages: - which refers to languages such as COBOL, FORTRAN, BASIC.
  + 1. **Low Level Languages**



**Machine Language (First Generation Language)**

Digital computers represent and process data and instructions as binary numbers. This representation of instructions and data is called machine language. Program instructions were written as a series of binary digits (0’s and 1’s). When the program was entered into the computer execution was direct since machine language needs no translation. The binary combination allowed the program to have full access to and control the computer’s internal circuitry and memory addresses.

***Advantages***

* Program translation was fast because no translation was required.
* The program could directly address and control the internal circuitry meaning that these programs were more effective in hardware usage and control.

***Disadvantages***

* Writing programs was time consuming
* Tracing errors in a program was extremely difficult.
* Difficult to learn and use.
* Program modification was cumbersome.
* They were machine dependent i.e. a program created for one type of machine would not work on a different type of machine.
* To write an effective program the programmer had to have expert knowledge on the computer’s internal workings.
  + 1. **Assembly Language (Second Generation)**

***Assembly Language (Second Generation)***

This language was more user oriented than machine language. Instructions are represented using mnemonic code and symbolic addresses. Words like add, sum etc could be used in programs. An assembler translated these codes into machine language.

***Advantages***

* Much easier to learn compared to machine language.
* Coding took less time than coding using machine language.
* Error correction was less cumbersome.

***Disadvantages***

* Were also machine specific
* Execution took longer than machine language programs due to the translation process.
  1. **High Level Languages**

**High Level Languages**

These languages are user friendly and problem oriented compared to the low level languages. Programs written in high level languages are shorter than the machine language programs.

They have an extensive vocabulary of words and symbols therefore program instructions are written using familiar English-like statements and mathematical statements. A single high-level language program is translated into multiple machine code instructions.

***Advantages***

* They are portable i.e. they can be used on more than one type of machine.
* Creating program and error correction takes less time.
* Are relatively easy to learn and use.
* The programmer does not have to be an expert on the internal workings of the machine.

***Disadvantages***

* Program execution takes more time due to the translation process.
* They do not address the internal circuitry of computers as effectively as the low level languages.
* A translated program will occupy more space.

High level languages can further be classified into:

* Procedural languages (Third Generation)
* Non-Procedural Languages (Fourth Generation Languages or 4GLs)
  + 1. **Procedural Languages**



They require the programmer to specify step-by-step how the computer will accomplish a specific task. Program execution follows the exact sequence laid down by the programmer during coding. Examples include FORTRAN, PASCAL, BASIC,

* + 1. **Non-Procedural Languages**



They allow the programmer to specify the desired result without having to specify the detailed procedure needed to achieve the result.

They are more user oriented and allow programmers to develop programs with fewer commands compared with 3rd generation languages. They are called non procedural because programmers can write programs that need only tell the computer what they want done, not all the procedures of doing it.

**4GL consists of:**

* Report Generators: also called report writers. This is a program for end users that is used to produce reports.
* Query Language: This is an easy to use language for retrieving data from a database management system.
* Application Generators: This is a program’s tool that allows a person to give a detailed explanation of what data to be processed. The software then generates codes needed to create a program to perform the tasks.
  1. **Object Oriented Programming Languages**



It stands for Object Oriented Programming. **O**bject-**O**riented **P**rogramming (*OOP*) uses a different set of programming languages than old procedural programming languages (C, Pascal, etc.). Everything in *OOP* is grouped as self sustainable "objects". Hence, you gain re-usability by means of four main object-oriented programming concepts.

An object can be considered a "thing" that can perform a set of **related** activities. The set of activities that the object performs defines the object's behavior. For example, the hand can grip something or a Student (object) can give the name or address.

* 1. **Language Translators**

****

These are programs that translate programs written in a language other than machine language into machine language programs for execution. Programs written in assembly or high level languages are called source programs or source code before they undergo translation. After the translation the machine language version of the same program is called object program or object code. Language translators can be classified into

1. **Assemblers**
2. **Compilers**
3. **Interpreters**

***Assembler***

This is a program that translates a source program written in assembly language into its equivalent machine code (object code).

***Compiler***

During compilation both the high level source program and the compiler are loaded into the RAM. The compiler reads through the source program statement by statement, converting each correct statement into multiple machine code instructions. The process continues until the compiler has read through the entire source program or it encounters an error. An erroneous statement is not translated but is placed on the source program error listing, which will be displayed to the programmer at the end of the translation process. Where there are no errors the compiler will generate a machine code object program which is stored for subsequent execution.

Compilation can be divided into three stages including

Lexical Analysis which involves

* Checking for valid words like data names or operator symbols
* Checking for reserved words (keywords). These are words that have a special meaning for the compiler. These are then replaced with non-alphanumeric character codes known as tokens.

Syntax analysis which involves

* Checking the program statements for correct grammatical form.
* Breaking down the complex statements in the program into simpler equivalents and more manageable forms.

The first two stages of compilation are carried out by a part of the compiler known as the parser and can be referred to as parsing.

Code Generation involves

* Translation of each statement into its equivalent machine code. This is done using tables.
* Setting up linkages that allow the object program to communicate or work with various operating systems and hardware.
* Fetching of subroutines from the system library, to optimize the code and make it execute faster and use less storage space.

The optimized code is then used to generate the object program which is stored on media such as disk to await subsequent execution.

***Interpreter***

It is similar to the compiler in that it translates high level language programs into machine code for execution but no object code is generated. An interpreter translates the program one statement at a time and if it is error free it executes the statement. This continues till the last statement in the program has translated and executed. Thus an interpreter translates and executes a statement before it goes to the next statement. When it encounters an error it will immediately stop the execution of the program.

* 1. **Self-test Questions**

****

1. Define the following terms
2. Program
3. Syntax
4. Programming language
5. Give three advantages of High level programming languages over low level programming languages.
6. Differentiate between a compiler and an interpreter.
   1. **Summary**

****In this lecture we have defined terms that are very key when it comes to foundational knowledge in programming. We have also discussed various levels of programming languages including their advantages and disadvantages. Lastly we have discussed program translators which include interpreters, compilers and assemblers.

* 1. **Reference**

****

1. H.M. Deitel and P.J. Deitel (2014). C: How to Program, 2nd Edition. Prentice Hall, ISBN 341– 7600465.
2. Xavier, C. (2008). *Introduction to Computers and Basic programming*. New Age International Publishers, ISBN 978– 81– 224– 2123– 1.