Elementary Programming Principles

Lesson#1 : Programming Languages

Kalema Golooba Ayub
agkalema@gmail.com
077 259 2826
070 169 2724
www.kalsworld.com
www.sustainablecs.org

Session Objectives

- Understand what programming is
- Understand basic terms used in programming
- Programming Languages and their types
- Explain the program development cycle
- Describe how to develop a simple program using pseudocode and flowchart

What is Computer Programming?

- A computer works by executing a set of instructions known as a ***program***
- ***Programming*** .. the process of developing computer instructions (programs) to solve a particular task
- Involves use of special characters, signs and symbols found in a **programming language**

Programming Language

- PL = A special set of rules and symbols used to construct a computer program when arranged in a particular sequence or order

- ***The sequence*** or order of writing characters, symbols in a programming language is called Syntax

Key Terms

Source Program (Source Code)

Is the program code that the programmer enters in the program editor window that is not yet translated into machine readable form

Object Code

Is the program code that is in machine readable form

Translators

- Translators are language processors that convert the source program into object/machine code

a. Assembler

- An assembler translates *assembly language* into machine code that the computer can understand and execute

Translators

b. An ***interpreter*** translates the source program line-by-line, allowing the CPU to execute one line before translating the next

- Code is loaded in memory and executed line by line
- Translates program each time it is run
- Takes less memory than compiled

Video

Translators

c. **A *compiler*** translates the entire source program into object code; to make a full executable file (.exe)

- ***Linking*** is a process which joins the object code file to all the other files that are needed to make a full executable program e.g. WINWORD.exe

What is a Linker?

- A program that pulls other programs together so that they can run.
- Most programs are very large and consist of several modules.
- Even small programs use existing code provided by the programming environment called libraries.
- The linker pulls everything together, makes sure that references to other parts of the program (code) are resolved.

To Skip

	Interpreters		Compilers
1.	Translates the source program one statement at a time	1.	Translates the entire source code at once before execution
2.	Translates the program each time it is run hence slower than compiling	2.	Compiled program (object code) can be saved on a storage media and run as required, hence executes faster than interpreted programs
3.	Interpreted object code takes less memory compared to compiled program	3.	Compiled programs require more memory as the object files are larger

COMPILED AND INTERPRETED LANGUAGES

lynda.com

Levels of Programming Languages

High Level

- 5th Generation (5GLs)
- 4th Generation (4GLs)
- 3rd Generation (3GLs)

Low Level

- 2nd Generation (2GLs) - Assembly
- 1st Generation (1GLs) - Machine

CPU

Low Level Languages

- They can be easily understood by the computer directly or
- They require little effort to translate into computer understandable form

a) **Machine Languages (1GL):**

- Instructions are written using binary logic.
- Many lines of code are needed to accomplish even a simple task like adding two numbers.

Machine Languages

- Comprised of 1s and 0s
- The "native" language of a computer
- Difficult to program – one misplaced 1 or 0 will cause the program to fail

- Example of code

```
1110100010101        111010101110
10111010110100       10100011110111
```

b) Assembly Languages (2GL)

Developed in order to overcome the difficulties of understanding and using machine languages

- They allowed programmers to write programs as a set of symbolic operation codes called ***mnemonics***.
- Mnemonics are basically shortened two or three letter words
- Mov AX, 15 (move 15 to register AX)
- SUB Ax, 10 (subtract 10 from the value Ax.

Low Level Languages

- These languages are classified as low because they can be directly, or easily understood by the computer with little effort to translate into computer understandable form.
- These languages are hardware oriented and therefore they are not portable. I.e. a program written for one computer cannot be installed and used on other.

High-Level Languages

- Are PL which use statements consisting of English-like keywords such as "FOR", "PRINT" or “IF“, ... etc.
- Each statement corresponds to several machine language instructions (one-to-many correspondence).
- Much easier to program than in assembly language.
- Operations can be described using familiar symbols
- Example:

  Cost := Price + Tax

High-Level Language

- HLs are very close to the **human language** (English-like)
- They can be read and understood even by people who are **not experts** in programming
- These languages are ***machine independent***
- 5 Groups of High Level Programming Languages

3rd Generation Languages (3GLs)

1. Are also called structured or procedural languages
2. They make it possible to break a program into components (subprograms) called *modules* each performing a particular task
3. Use control structures in problem solving such as sequence, selection and iteration
4. Flexible, easier to read and modify

Third Generation Languages - Examples

- **Pascal:** Initially developed as an academic language to help in the teaching and learning of structured programming.
- **FORTRAN (FORmula TRANslator):** Developed for mathematicians, scientists and engineers. It enables writing of programs with mathematical expressions.
- **COBOL: (Common Business Oriented Language):** Designed for developing programs that solve business problems e.g. developing data processing applications such as computer-based inventory control systems.
- **BASIC: (Beginners All-purpose Symbolic Instructional Code): A** simple general purpose language used for developing business and educational applications. Because of its simplicity, it is a powerful tool for students who wish to learn programming.
- **C:** Mainly used for developing system software such as the operating system. It is one of the most popular and powerful.
- **Ada:** Named after the first lady programmer, Ada Lovelace. Ada is suitable for developing *military, industrial* and *real-time* systems.

Fourth Generation Languages (4 GLs)

1. Make programming easier because have more programming tools e.g. **command buttons, forms** etc.
2. The programmer selects **graphical objects** on the screen called **controls** and then uses them to create designs on a base form
3. The programmer may also use an **application generator** which works behind the scenes to generate the necessary code, hence the programmer is freed from the tedious work of writing the code
4. Examples: *Visual Basic*, *Delphi Pascal* and *Visual COBOL*

5th Generation Languages (5 GLs)

- Are designed around the concept of solving problems by enabling the computer to depict **human like intelligence**.
- Are designed to enable the programmer to quickly come up with a working program that solves the problem at hand.
- Examples of these languages are those used in **artificial intelligence** like PROLOG, Mercury, LISP and OCCAM

Object-Oriented Programming Languages (OOP)

1. **The concept:** -> Looks at a program as having various objects interacting to make up a whole

2. Each object has specific data values that are unique to it (called state) and a set of the things it can accomplish called (functions or behavior)

3. Encapsulation is a process of having data and functions that operate on the data within an object

- http://www.kalsworld.com

Object-Oriented PLs (OOP)

- Simula (developed in the 1960's)
- C++
- Java
- SmallTalk

OOP has contributed greatly to the development of graphical user interface operating systems and application programs

Web Scripting Languages

- They are used to develop or add functionalities on web pages.
- **scripts** may be inserted in HTML pages in order to add functionality to the HTML page.
- A script is a small program fragment, written in a different language other than HTML but inserted into the HTML program.
- Examples of WSL : Java Script, VBScript and PHP Hypertext Preprocessor (PHP)

Low Level Languages - Advantages

1. The CPU understands machine language directly without translation – No need for translators
2. The processor executes them faster because complex instructions are already broken down into smaller simpler ones
3. Low level languages are stable and hardly crash or break down once written

Low Level Languages - Disadvantages

- Low level languages are difficult and cumbersome to use and learn.
- They require highly trained experts both to develop and maintain programs.
- Removing errors (debugging) in low level language programs is difficult.
- Machine dependent i.e. they are not transferable from one hardware or software platform to another. (Not Portable)

High Level Languages - Advantages

1. They are portable i.e. they are transferable from one computer to another
2. High level languages are user friendly and easy to use and learn
3. They are more flexible, hence they enhance the creativity of the programmer and increase productivity in the workplace.
4. They are far much more easy to correct errors (debug)
5. They provide better documentation
6. They require less time to code

High Level Languages - Disadvantages

1. Their nature encourages use of many instructions in a word or statement hence the complexity of these instructions cause slower program processing
2. They have to be interpreted or compiled to machine readable form before the computer can execute them
3. They require large computer memory to run in

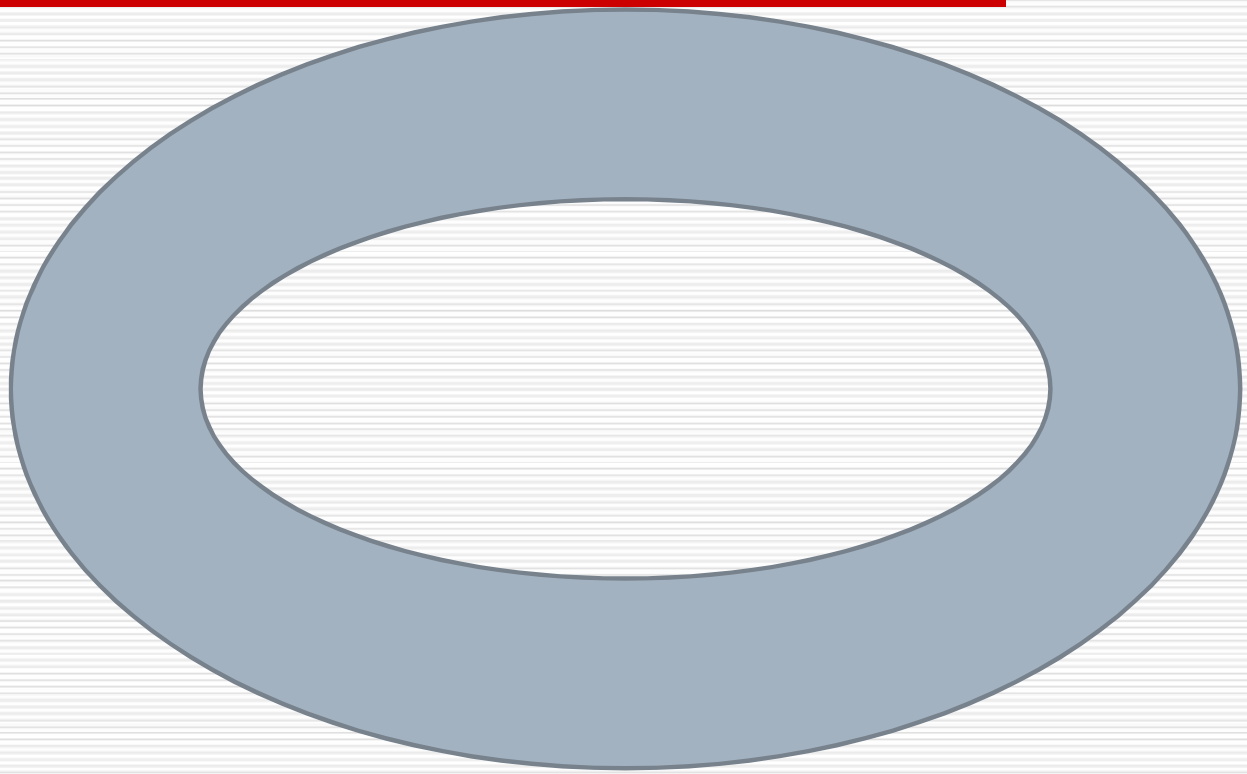
Questions

1. Define the term computer program
2. What is programming?
3. State three advantages of high level languages over low level languages.
4. List four examples of high level languages and for each, state its most appropriate application area.
5. Why is an executable file unique when compared to any other file?

Questions

6. Differentiate between a compiler and an interpreter. Why did early computers work well with interpreters?
7. List the various examples of programming languages per generation.
8. State one advantage of machine language over the other languages.
9. Distinguish between source programs and object code in programming.
10. Define the term encapsulation as used in object oriented programming.

Program Development Cycle



Input & Variables

- Input operations get data into the programs
- A user is *prompted* to enter data:

```
Write "Enter the price in shillings"
Input PoundPrice
```

- Computer programs store data in named sections of memory called *variables*. In the example above, the variable is named **`ShillingPrice`**. The value of a variable can, and often does, change throughout a program.

Types of Data

- Numeric Data
  - Integer data, I.e., whole numbers, 10 25 -45 0
  - Floating point data – have a decimal point 23.0, -5.0
- Character data (alphanumeric)
  - All the characters you can type at the keyboard
  - Letters & numbers not used in calculations
- Boolean data
  - TRUE/FALSE

Type	Description
char	Typically a single octet (one byte). This is an integer type.
int	The most natural size of integer for the machine.
float	A single-precision floating point value.
double	A double-precision floating point value.
void	Represents the absence of type.