Programming Concepts

Janaka Wijayanayake

About me

- Dr. Janaka Wijayanayake (Senior Lecturer, Dept. of Industrial Management, University of Kelaniya)
  - B.Sc. Special degree in Industrial Management, 1992 - University of Kelaniya
  - M.Eng in Industrial Engineering and Management, 1998 - Tokyo Institute of Technology
  - Ph.D. in Management Information Systems, 2001 - Tokyo Institute of Technology

Your role

- Read
- Collaborate with other students in learning
- Actively participate in discussion and debate
- Seek support and guidance from staff when necessary
- Accept the responsibility to move towards intellectual independence
- Relate learning to your own experiences

Evaluation Criteria

50% - Exams
50% - Continuous Assessment
Assignments
Practical test
Mini Project (Group)

If you turn a late assignment, the following criteria will be used to subtract points from your grade.

One-day - 10 points
Tow days - 20 points
Three days - 30 points
Four days - 40 points | |

After four days you will not get any points

What is a program

- A program is a list of step by step instructions telling the computer how to do something.
  - Computers cannot do things by themselves, so they need explicit, detailed, step-by-step instructions in order to do anything.
  - Reading a file from a disk into memory, displaying a word on the screen, and so on are all accomplished by telling the computer exactly which signals need to be sent to the hardware (the disk drive, or the video controller) at what time.
  - A collection of these instructions strung together to make the computer do something useful (or at least do *something*) is a program.

What is a programming language?

- A notational system for describing a computation in a machine-readable and human-readable form.
- It is like a set of tools. Each language is suited for a specific type of computation.
- Programming languages allow programmers to code software.
- The three major families of languages are:
  - Machine languages
  - Assembly languages
  - High-Level languages

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
```

Assembly Languages

- Assembly languages are a step towards easier programming.
- Assembly languages are comprised of a set of elemental commands which are tied to a specific processor.
- Assembly language code needs to be translated to machine language before the computer processes it.
- Example:

```
ADD  1001010, 1011010
```

High-Level Languages

- High-level languages represent a giant leap towards easier programming.
- The syntax of HL languages is similar to English.
- Historically, we divide HL languages into two groups:
  - Procedural languages
  - Object-Oriented languages (OOP)

Procedural (Imperative) Languages

- Early high-level languages are typically called procedural languages.
- Procedural languages are characterized by sequential sets of linear commands. The focus of such languages is on *structure*.
- Examples include C, COBOL, Fortran, LISP, Perl, HTML, VBScript

Object-Oriented Languages

- Most object-oriented languages are high-level languages.
- The focus of OOP languages is not on structure, but on *modeling data*.
- Programmers code using “blueprints” of data models called *classes*.
- Computation is based on objects sending messages (methods applied to arguments) to other objects
- Examples of OOP languages include C++, Visual Basic.NET and Java.

Applicative Languages

- Applicative (functional) languages
  - Programs as functions that take arguments and return values; arguments and returned values may be functions
  - Programming consists of building the function that computes the answer; function application and composition main method of computation
  - Syntax: P1(P2(P3 X))
  - Example languages: ML, LISP, Scheme

Declarative (Logic) Programming

- Rule-based languages
  - Programs as sets of basic rules for decomposing problem
  - Computation by deduction: search, unification and backtracking main components
  - Syntax: Answer :- specification rule
  - Example languages: (Prolog, Datalog,BNF Parsing)

Compiling

- Regardless of the HL Language, all HL programs need to be translated to machine code so that a computer can process the program.
- Some programs are translated using a **compiler**. When programs are compiled, they are translated all at once. Compiled programs typically execute more quickly than interpreted programs, but have a slower translation speed.

Interpreting

- Some programs are translated using an **interpreter**. Such programs are translated line-by-line instead of all at once (like compiled programs). Interpreted programs generally translate quicker than compiled programs, but have a slower execution speed.

Interpretation Vs. Compilation

- interpreters have significant run-time overhead. They require some sort of environment/program that must process the instructions during execution. Compiling puts this overhead at compile time; at run-time the overheads are minimal.

Aspects of Programming Languages

- *Programming Languages have syntax and semantics*
  - Syntax is the set of rules that govern the way language elements may be grammatically combined (written)
  - Semantics is defining how each grammatically correct sentence is to be interpreted. (meaning)

- *Language components are the lexical symbols or tokens of the language. They can be classified as:*
  - Identifiers : Names chosen by programmer to identify objects of interest
  - Keywords : Names chosen by language designer to help determine syntax and structure
  - Operators : Serve to identify actions
  - Separators : Punctuation marks to support syntactic structure
  - Literals : Denote values directly
  - Comment : Explanatory text (ignored)

Language Tools

- *These days we don't talk about a programming language in isolation. We discuss its environment as part of the evaluation.*

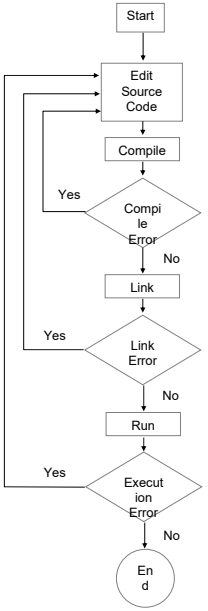
  These environments include
  - Editors
  - Debuggers
  - File system
  - Compilers
  - Linkers
  - Visualization tools

Types of Errors

- Syntax – wrong grammar, i.e., breaking the rules of how to write the language
  - Forgetting punctuation, misspelling keyword
  - The program will not run at all with syntax errors
- Logic - the program runs, but does not produce the expected results.
  - Using an incorrect formula, incorrect sequence of statements, etc.
- Run Time - Program crashes
  - Check input data (overflows, etc.)

EXECUTION CYCLE

Program Execution Cycle of a program written in a compiler programming Language



Structured Programming

- A method for designing and coding programs in a systematic, organized manner.
- It combines the principles of top-down design, modularity and the use of the three accepted control structures of **sequence**, **repetition** and **selection**.

Control Structures

- ***Sequence*** –in sequential order.
  - The simplest of control structures – start at the beginning and continue in sequential order.
- ***Selection*** – selectively execute statements
  - Called a **branch**, it requires a condition to determine when to execute statements.

Control Structures

- ***Repetition*** – repeat statements more than once
  - Called a **loop**, it needs a stop condition, I.e, the program will continue to loop until some condition is met.