Lecture 01: Programming Languages

ETE 111

Md. Taslim Arefin
Associate Professor
Department of ETE

Some Key Terms (1/2)

- Computer needs a set of instructions to process data
- A ***program*** is a sequence of instructions for a computer to perform
- ***Software***, for our purposes, is just a term meaning programs
- What Kinds of ***Instructions***?
  - Input/Output (I/O)
  - Arithmetic & logical calculation
  - Decisions
  - Repetitions

Some Key Terms (2/2)

- A ***programming language*** is a well-defined set of rules for specifying a program's sequence of instructions.
  - Examples: C, C++, Fortran 90, Java, Basic, HTML, Perl, Haskell, Prolog, Pascal, etc.
- ***Source code*** is a sequence of instructions, written in a *human-readable programming language*, that constitute a program, or a piece of a program
- A ***source file*** is a file of source code

Programming Languages

- A ***language*** is a system of communication
- A ***programming language*** consists of all the symbols, characters, and usage rules that permit people to communicate with computers
- There are at least several hundred, and possibly several thousand, different programming languages and dialects
  - Some of these are created to have a special purpose (controlling a robot), while others are more flexible general-purpose tools that are suitable for many types of applications
- However, every programming language have instructions that fall into the familiar input/output, calculation/text manipulation, logic/comparison, and storage/retrieval categories

Types of Programming Languages

- Even though all programming languages have an *instruction set* that permits these familiar operations to be performed, there's a marked difference to be found in the symbols, characters, and syntax of them
- Programming languages are said to be lower or higher, depending on whether they are closer to the language the computer itself uses (lower, which means 0s and 1s) or to the language that people use (higher, which means more English like)

Programming Languages: Levels/Generations

- We shall consider five levels (or generations) of programming languages
  - Machine Languages / First-generation languages
  - Assembly Languages / Second-generation languages
  - Procedural Languages / Third-generation languages
  - Problem-oriented Languages / Fourth-generation languages
  - Natural Languages / Fifth-generation languages

Machine Languages (First Generation)

- A computer's ***machine language*** consists of strings of binary numbers (0s and 1s) and is the only one the processor directly "understands"
- An instruction prepared in any machine language will have at least two parts
  - The first part is the ***command*** or ***operation***, and it tells the computer what function to perform. Every computer has an operation code, or "op code", for each of its functions
  - The second part of the instruction is the operand, and it tells the computer where to find or store the data or other instructions that are to be manipulated

Examples

- The number of operands in an instruction varies among computers
- In a ***single-operand*** machine, the binary equivalent of "ADD 0184" could cause the value in storage location or address 0184 to be added to a value stored in the arithmetic-logic unit
- For example, the instruction "ADD 0184" for an early IBM machine can be written as:
  - 00010000000000000000000000010111000

Machine Languages: Advantages and Disadvantages

- Advantages
  - Its execution is very fast and efficient because the computer can accept the machine code as it is
- Disadvantages
  - A programmer has to remember dozens of code numbers for the commands in the machine's instruction set.
  - One has to keep track of the storage locations of data and instructions
  - Programs written in machine language for one computer model will not, in all likelihood, run on a different model computer

Assembly Languages (1/2)

- Assembly languages, also known as *symbolic languages* uses abbreviations or **mnemonic** (pronounced ne-mon'-ik) **code** — code more easily memorized — that replace the 0s and 1s of machine languages
- In an assembly language, the instruction

  00010000000000000000000010111000

  could be expressed as

  ADD 0184

Assembly Languages (2/2)

- Actually, assembly languages do not replace machine languages
- In fact, for an assembly language program to be executed, it must be converted to machine code
- An **assembler**, which is itself a program, enables the computer to convert the programmer's assembly language instructions into its own machine code
- The assembly language program is referred to as a **source program** whereas, the machine language program is an **object program**

Assembly Languages: Advantages/Disadvantages

- Assembly languages have advantages over machine languages
  - They save time and reduce detail
  - Fewer errors are made, and those that are made are easier to find.
  - And assembly programs are easier for people to modify than machine language programs
- But there are limitations
  - Coding in assembly language is still time consuming
  - And a big drawback of assembly languages is that they are machine oriented. That is, they are designed for the specific make and model of processor being used

High Level Languages

- High-level languages assist programmers by reducing the number of computer operation details they have to specify, so that they can concentrate more on the logic needed to solve the problem
- High-level languages are often oriented toward a particular class of processing problems
  - For example, a number of languages have been designed to process scientific-mathematic problems, and other languages have appeared that emphasize file processing application.
- Unlike assembly programs, high-level language programs may be used with different makes of computers with little modification.
  - Thus reprogramming expense may be greatly reduced

Compilers

- Naturally, a source program written in a high-level language must also be *translated* into machine-usable code
- There are two kinds of translators — compilers and interpreters — and high-level languages are called either *compiled languages* or *interpreted languages*.
- In a compiled language, a translation program, known as **compiler**, is run to convert the programmer's entire high-level program, which is called the **source code**, into a machine language code. This translation process is called **compilation**.
- The machine language code is called the **object code** and can be saved and either run (executed) immediately or later

Interpreters

- In an interpreted language, a translation program, known as **interpreter**, converts each program statement into machine code just before the program statement is to be executed
- Translation and execution occur immediately, one after another, one statement at a time.
- Unlike the compiled languages, no object code is stored and there is no compilation. This means that in a program where one statement is executed several times (such reading an employee's payroll record), that statement is converted to machine language each time it is executed

Compiled Vs Interpreted Languages

- Compiled languages programs are better than interpreted languages programs as they can be executed faster and more efficiently once the object code has been obtained
- On the other hand, interpreted languages programs do not generate object code and so are usually easier to code and test

Types of High-Level Languages

- Languages are often referred to as generations, the idea being that machine languages were the first generation and assembly languages were the second generation
- High-level languages are sometimes used to refer all languages above the assembly level
- Here we will subdivide languages into three generations.
  - Procedural-oriented or third generation
  - Problem-oriented or fourth generation
  - Natural or fifth generation

Procedural-Oriented Languages

- High-level languages are often classified according to whether they solve general problems or specific problems
- General purpose programming languages are called procedural languages or third generation languages
- They are languages such as Pascal, BASIC, COBOL and FORTRAN, which are designed to express the logic, the procedure, of a problem
- Because of their flexibility, procedural languages are able to solve a variety of problems

Advantages

- Advantages over machine and assembly languages
  - The program statements resemble English and hence are easier to work with
  - Because of their English-like nature, less time is required to program a problem
  - Once coded, programs are easier to understand and to modify
  - The programming languages are machine-independent
- Disadvantages
  - Programs execute more slowly

Fourth Generation Languages

- Third generation languages, such as BASIC or Pascal, require you to instruct the computer in step-by-step fashion
- Fourth generation languages, also known as problem-oriented languages, are high-level languages designed to solve specific problems or develop specific applications by enabling you to describe what you want rather than step-step procedures for getting there

Classification of 4GLs

- Personal computer applications software
- Query languages and report generators
- Decision support systems and financial planning languages
- Application Generators

Fifth Generation Languages

- Natural languages are still in the developmental stages
- They promise to have profound effect, particularly in the areas of artificial intelligence and expert systems
- The are designed to make the connections that humans have with computers more natural — more humanlike
- They are designed to allow the computer to become “smarter” — to actually simulate the learning process by remembering and improving upon earlier informationm

Programming in Early Computers

- In the beginning, Charles Babbage’s difference engine could only be made to execute tasks by changing the gears which executed the calculations
- Thus, the earliest form of a computer language was physical motion
- Eventually, physical motion was replaced by electrical signals when the US Government built the ENIAC in 1942
- It followed many of the same principles of Babbage's engine and hence, could only be “programmed” by presetting switches and rewiring the entire system for each new "program" or calculation. This process proved to be very tedious

FORTRAN

- FORTRAN, which originally stood for IBM Mathematical FORmula TRANslation System but has been abbreviated to FORmula TRANslation, is the oldest of the established high-level languages
- Designed by a group headed by John Backus in IBM during the late 1950s
- It has been widely accepted had has been revised a number of times
  - FORTRAN has seen a number of significant versions: II, IV, 66, 77, 90

Example Program

```
* SIMPLE PAY IN FORTRAN 77
INTEGER HOURS, PAY
READ *, HOURS
IF (HOURS .LE. 40) THEN
PAY = 10 * HOURS
ELSE
PAY = 10 * 40 + 15 * (HOURS - 40)
ENDIF
PRINT *, 'Gross pay is ', PAY
END
```

- In Fortran, a line that starts with an * is a comment, corresponding to a line that starts with REM in Basic. Fortran also uses the * as the symbol for multiplication, and to refer to the keyboard and screen in the READ and PRINT statement

COBOL

- Common Business Oriented Language
- COBOL was designed from the ground up as the language for businessmen
- Its only data types were numbers and strings of text
- It also allowed for these to be grouped into arrays and records, so that data could be tracked and organized better
- COBOL statements also have a very English-like grammar, making it quite easy to learn
- All of these features were designed to make it easier for the average business to learn and adopt it

BASIC

- BASIC (Beginners's All Purpose Symbolic Instruction Code) was originally developed at Dartmouth College by John Kemeny and Thomas Kurtz in the mid-1960s
- Because of its simplicity, it was adopted by several commercial time-sharing services, which caused it to receive a broad

Pascal

- Pascal developed by Niklaus Wirth in 1968
- Its development was mainly out of necessity for a good teaching tool
- In the beginning, the language designers had no hopes for it to enjoy widespread adoption
- Instead, they concentrated on developing good tools for teaching such as a debugger and editing system and support for common early microprocessor machines which were in use in teaching institutions
- Pascal was designed in a very orderly approach, it combined many of the best features of the languages in use at the time, COBOL, FORTRAN, and ALGOL

Example

```
PROGRAM SimplePay(INPUT, OUTPUT);
{ Simple pay in Pascal }
VAR
hours, pay: INTEGER;
BEGIN
Read(hours);
IF hours <= 40 THEN
pay := 10 * hours
ELSE
pay := 10 * 40 + 15 * (hours - 40);
Writeln('Gross pay is ', pay:6);
```

- Comments in Pascal are enclosed in braces, { and }

Other Languages (1/2)

- Lisp(LISt Processor), developed by John McCarthy about 1960, is a language based on mathematical concepts. Its objective is the processing of data represented as lists of items. It is mainly used in artificial intelligence applications.
- APL (A Programming Language), developed by Kenneth Iverson in 1962, is a programming language that uses a very esoteric mathematical notation.
- Algol 60: for scientific computation and for the communication of algorithms between computer scientists
- C: created at the Bell Laboratories in the early 1970s when low level access to the machine was considered important

Other Languages (2/2)

- Logo: for the introduction of the principles of computer programming to children through graphical manipulations,
- PL/I: a large language intended to be suitable for all applications,
- Prolog: for logic programming used, for example, to automate the proving of theorems,
- Simula 67: for the simulation of networks of discrete events,
- Smalltalk: for a style of programming where data takes an active rather than passive role; this is known as object programming

Current Generations

- HTML
  - web page mark up language
- Java
  - write and compile anywhere, run anywhere
- JavaScript, Jscript
  - dynamic web page
- Perl
  - server side processing
- XML
  - eXtensible mark up language
  - include data definition as well as presentation