COMP 364 - Lecture 5

## Announcements

- Office hours: Mondays after class, 1:30 to 2:30
- TA: Azin Sayad (He will also have a lab hour)
- The professor will be away on January 23.
- Homework 1 was passed out last week. It will be due on Friday the 25th or Monday the 28th (with penalties).

## Terminology

Computer Architecture

CPU: Central Processing Unit. The part of the computer that does computations. It moves data around (manipulates data, ex: adds and subtracts numbers). It does elementary operations really fast. Typical clock speed: 1 gigahertz.

Main memory: Where what the computer is working on "right at that time" is stored. It is volatile; once you turn off the computer it all goes away. For the computers at Trottier, Its size is around 1 GB. This memory is directly linked to the CPU.

Hard drive: Another form of memory to store information. It is non-volatle. It is bigger than the main memory. Its size is around 100 GB (bigger than the main memory).

Peripherals: Devices that are attached to and used with the computer although they are not integrally part of it. e.g.: keyboard, screen, mouse, printer, network connection.

Programming languages help us instruct the computer on what we want to do.

Machine code: It is run by the CPU. It consists of binary or hexadecimal instructions. e.g.: 100011 00011 01000 00000 00001 000100.

Assembly language: First generation programming language. e.g.: ADD #1 #2, MOVE #3 @memory. This is still very primitive since you have to tell each elementary operation you want the computer to do.

Higher level languages:

Fortran (around 1950): It is still continued to be developed as a language. It is used in many scientific compilations and engineering problems.

COBOL (COmmon Business Oriented Language): used as a means for storing employee records and for financial computing.

C (1972): It is a very popular language. It is used in system programming, writing compilers and operating systems (Linux).

C++ (Mid 1980's): It is an extension of C.

Java: It is similar to C++ but simpler.

Matlab: Used for scientific computing, linear algebra, image analysis, etc.

Perl (1987): Good at text processing, manipulating files, interfacing databases, websites, and popular in molecular biology.

**In order to run the codes written in "higher level languages", the code must be compiled or interpreted into machine code.

## Perl

The perl commands are written in a text file, using the text editor of your choice.
The first examples we looked at in class was how to create a perl program to print "Hello World!". Using the terminal a file is made, > emacs Hello.pl.
The ".pl" is only for you to keep track of what kind of file it is. The code is written in the text editor as: "Hello World \n";
Most commands in perl end with ";".
\n is equal to new line or carriage return.
If double quotations are used for the print command, special characters like \n and variables will be interpreted. If single quotations are used, the content inside the quotations will be printed out exactly as it appears. To run the command, the terminal is used: perl Hello.pl

Another way to run a perl program is to run it like a script. To do this, you need to give yourself execute permission on the file. You also need to add #!/usr/bin/perl at the start of your code (unless your file contains terminal commands). Then you can run your command by typing ./Hello.pl in the terminal.

Variables: Used to store things. There are three types of variables: scalars (numbers or strings), lists and hashes.
Each variable has a name that is represented by a dollar sign followed by at least 1 letter (and then, if you want, more letters and number and underscores). e.g.: $var, $ted, $comp364, $Jan_14. Variables can be printed or operated on in various ways.

```
$Str1 = "Hello World! \n";
```

The equal sign makes whatever is on the right side equal to the variable on the left side.

```
$A = 3;
$B = 10;
$C = 1;
print $A $B $C;
```

This will output an error.

Whereas, print "$A $B $C \n"; will output "3 10 1".

You can also perform operations with your variables.

```
$Z = $A + $B * $C;
print "$Z $\n";
```

The output will be 13.

Adding the # symbol at the beginning of a line will cause perl to ignore the whole line. This is a good way to add comments to your code, so you or a reader can understand it.

The entire code that we looked at in class:

```
$Str1 = "Hello World! \n";
print $Str1;

$Var1 = 3.141592767;
print "$Var1 \n";

$A = 3;
$B = 10;
$C = 1;
print "$A $B $C \n";

$Z = $A + $B * $C;
print "$Z \n";
```

The output of this code:

```
> Hello World!
> 3.141592767
> 3 10 1
> 13
```