Introduction to Computer Programming

22.901 Introduction to Computer Programming for Nuclear Engineers

January 17, 2012

Outline

- Computer languages and terminology
- Using Fortran
- Variables
- Math

Programming Languages - Nuclear Perpsective

- Numerical Programming Languages
  - Fortran, C, C++, etc.
  - compiled languages
  - very fast, low-level programming
- Scripting Languages
  - Python, Perl, Visual Basic,etc.
  - interpreted languges
  - good for data/file manipulation
  - not as fast, high-level programming
- Developmental Languages
  - MATLAB, Python, etc.
  - great enviorment for algorithm development
  - excellent post-processing capability
  - recommend for HWs and projects

Coding Jargon

- Source Code - an ASCII text file development by the programmer with code in it
  - the file should have the extension '.f90'
- Compiler - the program that converts the source code to machine code
  - C compilers - gcc, icc ...
  - C++ compilers - g++ ...
  - Fortran compilers - f77, g77, f90, f95, gfortran, ifort ...
- Program - the compiled source code

Fortran Programming Language

- In this course we will focus of Fortran (Formula Translation)
- Many nuclear engineering codes are *still* developed in Fortran!

- Classical Fortran
  - FORTRAN II-IV and Fortran 66 (1958-1966)
  - FORTRAN 77 - standard programming language (still seen today!)

- Modern Fortran
  - Fortran 90,95,2003,2008
  - free format source code, structures, dynamic memory allocation

How can I use Fortran?

- Windows - Download Cygwin and get the gfortran compilers
- Mac/Linux - Download gfortran compilers
- In 22.901 we will SSH to the department's linux cluster
  - Windows - download SecureFX/SecureCRT from MIT IST site
  - Mac/Linux - SSH right from the terminal
- To Login the following info is needed for SecureCRT
  - hostname: `cheezit.mit.edu`
  - username: `fortran12`
  - passowrd: `22.901IAP2012`
  - Mac/Linux: `ssh fortran12@cheezit.mit.edu`
- see Stellar document about navigating a Linux shell

How to Compile Fortran Code with `gfortran`

- Compile and Link
  `gfortran -o myCode myCode.f90`
- Compile only
  `gfortran -o myCode.o -c myCode.f90`
- Link only
  `gfortran -o myCode myCode.o`
- To include external files use `-I<path>`
- To link external libraries use `-L<path>`

The Simplest Fortran Code

```
program simplest

  ! terminate the code
  stop

end program simplest
```

Interacting with the User

- You may want to see what your code does!
- Use `print *,'str1',var1,var2,var3`
  - This will print to the terminal screen (STDOUT) on one line
  - There are more advanced formatting methods (save for later)
- To read something in from the user, use `read *,var1,var2`
- So now as a right of passage, print "Hello World!" to the screen

Hello World! Code

```
program helloworld

  ! print hello world to stdout
  print *, 'Hello World!'

  ! terminate the program
  stop

end program helloworld
```

Code Structure

A line can be up to 132 characters
A comment is from ! to the end of line
You can have blank lines
To continue a line place a & at the end and begin the next line with &

The code structure ::

1. start of a program, `program name1`
2. declaration of all variables
3. all of the execution statements
4. end of program, `end program name1`

Variables

- An ASCII alphanumeric word that represents a space in memory
- In Fortran the first character must be a letter (not case sensitive!!)
- By default variables beginning with I-N are integers, all else are reals (floats)
- This is the implicit naming structure of Fortran, later we will declare explicitly
- Some examples (Real or Integer ...)::
  - `HYP` - ??
  - `K9` - ??
  - `pi` - ??
  - `9X_Y` - ??

Simple Math

- Order of Operation Applies!
- Parentheses - ()
- Exponent - $**$
  - if exponent is a whole number, list as integer (e.g. 2 not 2.0)
- Multiplication/Division - *,/
- Addition/Subtraction - +,-
- Mathematical expressions should be of the same type (integer,real,etc.)
  - $a = 2,\ b = 3,\ c = 5,\ d = 4$
  - $e = (d + c) * *a/b + d$???
- Modulo - `a = mod(x,y)` finds the remainder `mod`(37,10) = ?

Integer vs. Real Division

- Although not commonly used, division can be performed with integers
- The two numbers are divided and the decimal is truncated
- For example $5/3 = 1$
- Division with Real variable types will give the correct answer
- For example $5/3 = 1.666666...$

End of Class/External Assignment

Write a code to do the following:

1. Ask user to enter any whole number 1-9
2. Read in the number (real or integer??)
3. Multiply the number by 9
4. Add both digits together (hint: use Modulo)
5. Subtract by 5
6. Print out result

Try different numbers, notice anything??