FORTRAN Short Course
Week 1
Kate T-C
February 17, 2008
What are we talking about this week?

- Quick Intro to the Class
- Three Laws of Computer Science
- How does a computer actually work?
- Developing a plan and a program
  - Dive into Fortran!
- Variables, Types, Arrays, Arithmetic Functions, Order of Operations
# The Plan

## Introduction to Programming: FORTRAN Short Course

Scheduling: Tuesdays 3:00-5:00p in ATS 101  
Instructor: Kate Thayer-Calder  
ATS 404  
katetc@atmos.colostate.edu  
Office Hours every Wed and Thurs 2:00-3:30p  
Class Website: [http://www.atmos.colostate.edu/gradprog/programming/](http://www.atmos.colostate.edu/gradprog/programming/)

<table>
<thead>
<tr>
<th>Date</th>
<th>Topics</th>
</tr>
</thead>
</table>
| 17 - Feb | • Another Intro to Computer Science  
• How a Computer Actually Works  
• Developing a Plan and a Program  
• Creating and Compiling a Fortran Program  
• Print is your friend  
• Variables, Types, Arrays, Arithmetic Functions, Order of Operations |
| 24 - Feb | • More on Arrays  
• Logical Operators  
• Conditional Statements  
• Looping options  
• Subroutines and Functions  
• Intro to Recursion |
| 3 - Mar | • Makefiles & Modules  
• Strings, strings, strings  
• Reading/Writing Formatted and Unformatted Data  
• Pointers  
• Higher dimensional arrays  
• User Defined Data Types |
| 10 - Mar | • Modules and organizing big programs  
• Reading/writing scientific data  
• Debugging, Debugging, Debugging  
• Class requested topics |
Goals of this Class

- Learn how to write workable, understandable, debuggable Fortran 90/95 code and programs
- Learn a bit more about how computers work
- Learn how to think about your problem in a way that will help you write good programs
NOT Goals of This Class

- Learn everything there is to know about programming!!! (Every language is different)
- Learn how to read/write Fortran 77 or old Fortran code
- Become an expert in Fortran or HPC
- Learn every option of aspects of Fortran
  - We won’t learn every error, every built-in function or data type, or all that much about compiler options
THREE LAWS OF COMPUTER SCIENCE

Because Turing says...
Law #1

The word **CODE** is already plural.

- Think of it like deer or fish
- Corollary: It is only acceptable to say “codes” if you are talking about launching missiles or cryptography.

And I mean it!
Law #2

Computers ONLY do what you tell them to!

- If something is wrong, it's probably your own fault. I'm sorry. But it is.

Corollary: Sometimes you don't know you told the computer to do it wrong, or somebody else did the telling.
Law #3

Do not re-invent the wheel.

Corollary: You probably won’t know it’s a wheel as you’re inventing it.
So, what are we doing?

- Fortran is a programming language (an old one)
- Fortran is a *compiled* language
- How do you want to think about this?
  - Magic words
  - Operating a machine
How does a computer work?
How does a computer work?

Input → Processing → Output

User → Processor → Visual
Memory → Processor → Memory
Memory → Processor → Visual
User → Processor → Memory

Repeat as necessary!
How does a computer work?

- **User Input:** Keyboard, Microphone, video camera, digital camera...
- **Visual Output:** Monitor, printer, speakers, Braille terminal...
- **Memory (slowest to fastest):** CDs/DVDs, External Hard Drives, flash drives, Internal Hard Drives, RAM, Cache (L3, L2, L1), Registers
- **Processor = CPU (Central Processing Unit)** takes numbers (in binary) and **does math** the result is numbers (in binary)
Basically, Computers

- Can store binary numbers.
  - The binary can be interpreted into lots of different types of numbers or even text or graphics
  - Binary is divided up into bits, bytes and words
- Can perform math on binary numbers
  - Everything that you see or do on a computer boils down to a line of math
  - A program is just a long series of mathematical operations performed on lots and lots of numbers
How do we write a program?

- We need a language that is easier than just writing billions of 1’s and 0’s. (Let’s use Fortran!)
- We need a program that can translate our computer language into 1’s and 0’s that the computer understands (called the compiler).
- We need a way to tell the computer to run and store our program (the operating system).
Our first Fortran Program

PROGRAM ILikePie

! Written by Kate T-C
! 2.11.09 For the Fortran Short Course
! This program is a bad joke.

real :: pi = 3.141592654
print *, 'I Like', pi

END PROGRAM ILikePie
Things in the program

- Each line is a statement
- The program runs top to bottom, in order
- Program start, program name
- Comment block
- Variable declaration
- Output statement - Print is your friend!
- Program end
Compile the Program

>`f90 ILikePie.f90`

Or replace `f90` with a call to your Fortran compiler (need to talk to your local computer guy to find this one out)

We can also do...

>`f90 ILikePie.f90 -o Pie`

This is called Compile-Time.
Run the program

>`a.out`

... or (if we used `-o`) ...

>`Pie`

... or navigate to the program in your window environment and double-click. You wrote a computer program!

This is called **Run-Time**.
Code Talker

Ala’ih, Donéhlini, Ala’ih, Donéhlini, Ala’ih, Donéhlini, Ala’ih, Donéhlini, Ala’ih, Donéhlini, Donéhlini…

For added security, after we encrypt the data stream, we send it through our Navajo code talker.

…is he just using Navajo words for “zero” and “one”?

Whooa, hey, keep your voice down!
Let’s do another one!

Who uses Fortran? Well, climate modelers! So, let’s write a climate model...

Energy In = Energy Out

\[ S_o (1 - a) \pi r^2 = \sigma T_e^4 4\pi r^2 \]

\[ \frac{S_o}{4} (1 - a) = \sigma T_e^4 \]
Translating the model into Fortran

- Variables, Literals and Constants
- Types
- Input, action, output
- Implicit None

See example program ClimateModel.f90
Variable Types

- **real** - a floating point number
- **double precision** - a floating point number using twice the bytes for more accuracy
- **integer** - a straight-up whole number
- **complex** - numbers that include an imaginary component
- **logical** - two values: true or false
- **character** - variable contains text
Arithmetic Operators

1. Parenthesis ()
2. Exponentiation **
3. Multiplication and Division *, /
4. Addition and Subtraction +, -

Please Excuse My Dear Aunt Sally
Arithmetic Operators

Try it out, what do you get for the following statements?

10 - 3 + 2 * 10
(10 - 3 + 2) * 10
3 ** 2 + 1
3 ** (2 + 1)
3 ** 2 ** 0.5
Reals and Integers

Mixing types in arithmetic can be confusing. If operands are all integers, the result is an integer, otherwise, the result is real.

- $8.0/4.0 = 2.0$, $8.0/4 = 2.0$, $8/4 = 2$

Integer division truncates the result

- $10.0/4.0 = 2.5$, $10.0/4 = 2.5$, $10/4 = 2$

We can save some trouble by casting to make sure everything is the correct type.

Because computers use binary, technically, ALL division and multiplication is truncated!
Really Big or Really Small Numbers

Different computers and compilers use different numbers of bits to represent integers (8, 16, or 32) and reals (32 or 64).

A 16 bit (signed) integer has 1 bit for the sign and 15 for the number, so $2^{15} = 32,768$ numbers possible (range is -16,383 to 16,384)

A 32 bit real has a 24 bit mantissa and 8 bit exponent, range is $\sim 10^{2^{(8-1)}} = 10^{(+/-)128}$

Try to write code that doesn’t produce overly large or overly small numbers to avoid problems.
Another Example!

- What else do we use Fortran for around here?

- It’s basically the fastest way to manipulate large amounts of data.

- To perform operations on data, we could either declare thousands of “real” variables (one for each data point) or just declare one Array.

- Check out example Statistics.f90
Arrays

- Give one name to a series of numbers
- Each element in the array has an Index or Subscript - which must be an integer
- You can declare an array of any type using the ‘dimension’ attribute
- You can fill the array when you declare it using (/x,y,z/) notation or fill it at run-time using input data from other sources (files, stdio, instrumentation, etc)
What did we cover today?

- Three Laws of Computer Science
- How a computer works
  - Three things a computer can do
- Writing a Fortran program - text, compile, run
- We learned about the `Print * ,` statement
- Literals, Variables and Constants
- Implicit None and Variable Types
- Arithmetic operators and the Order of Operations
- Arrays
Homework

Sure, why not?

Email your code and program output to me if you want feedback

My office is ATS 404 if you need help