CS 199 Computer Programming

Spring 2018

Lecture 3

FORTRAN Basics

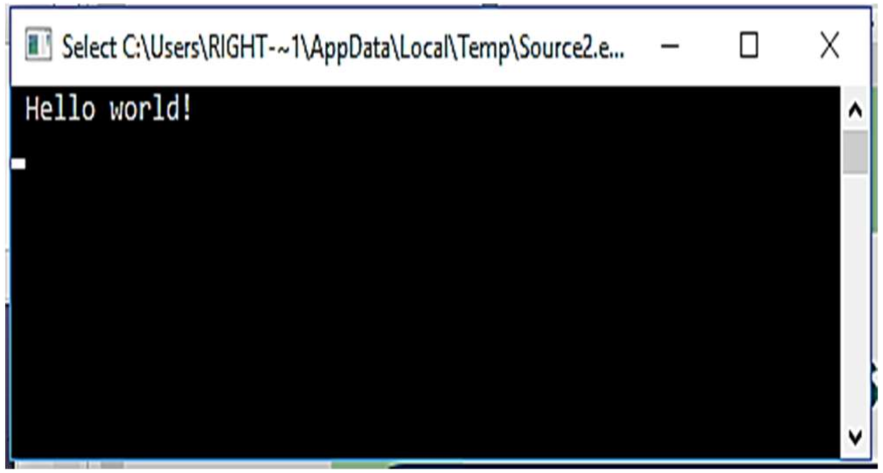
Objectives

- In this chapter, you will learn:
  - To be able to write simple computer programs in FORTRAN.
  - To be able to use simple input and output statements.
  - To become familiar with fundamental data types.

Fortran Basics :

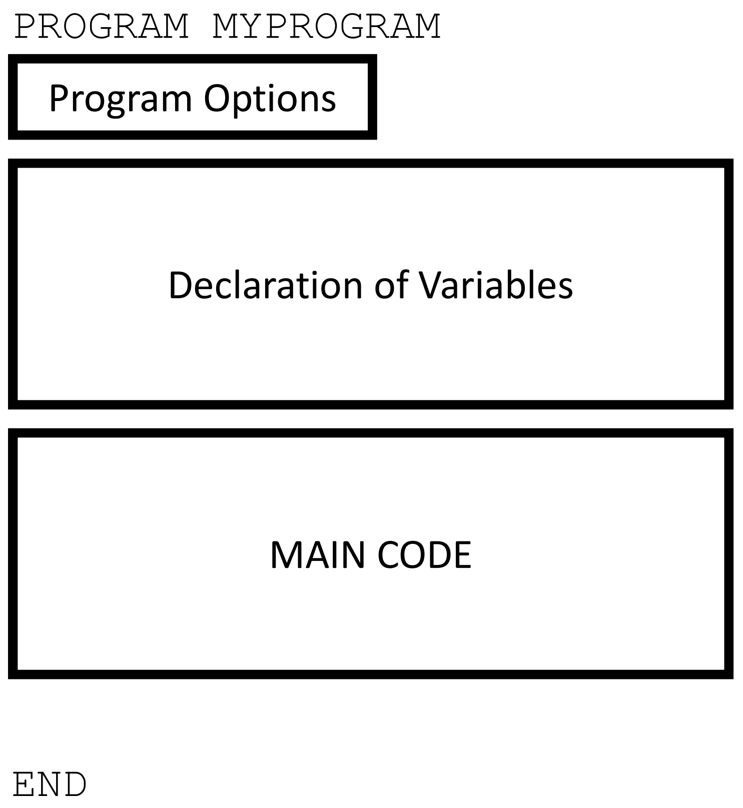
- A Fortran program is just a sequence of lines of text. The text has to follow a certain *syntax* to be a valid Fortran program. We start by looking at a simple example:

```
c    this a program that print hello message
     write (*,*)'Hello world!'
     END
```

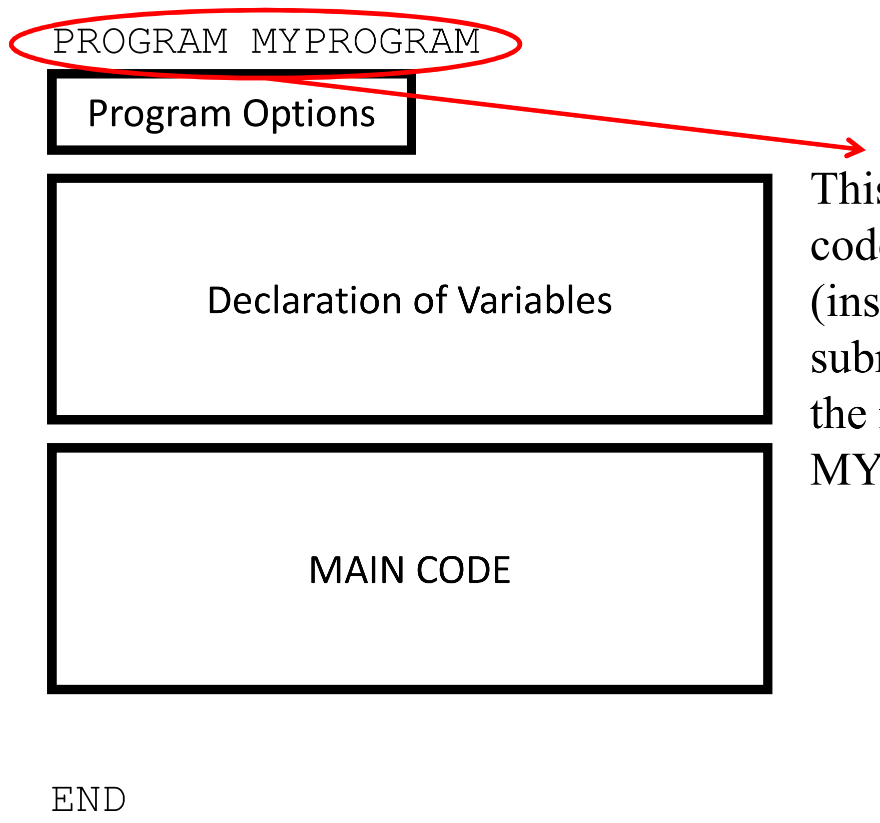


```
Hello world!
```

Your Typical Program



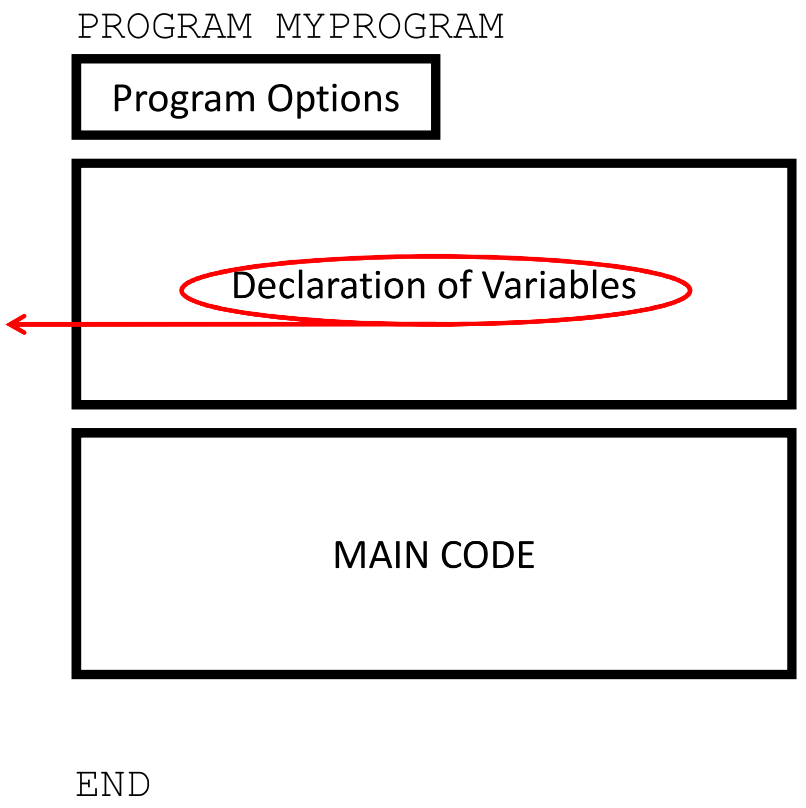
Your Typical Program



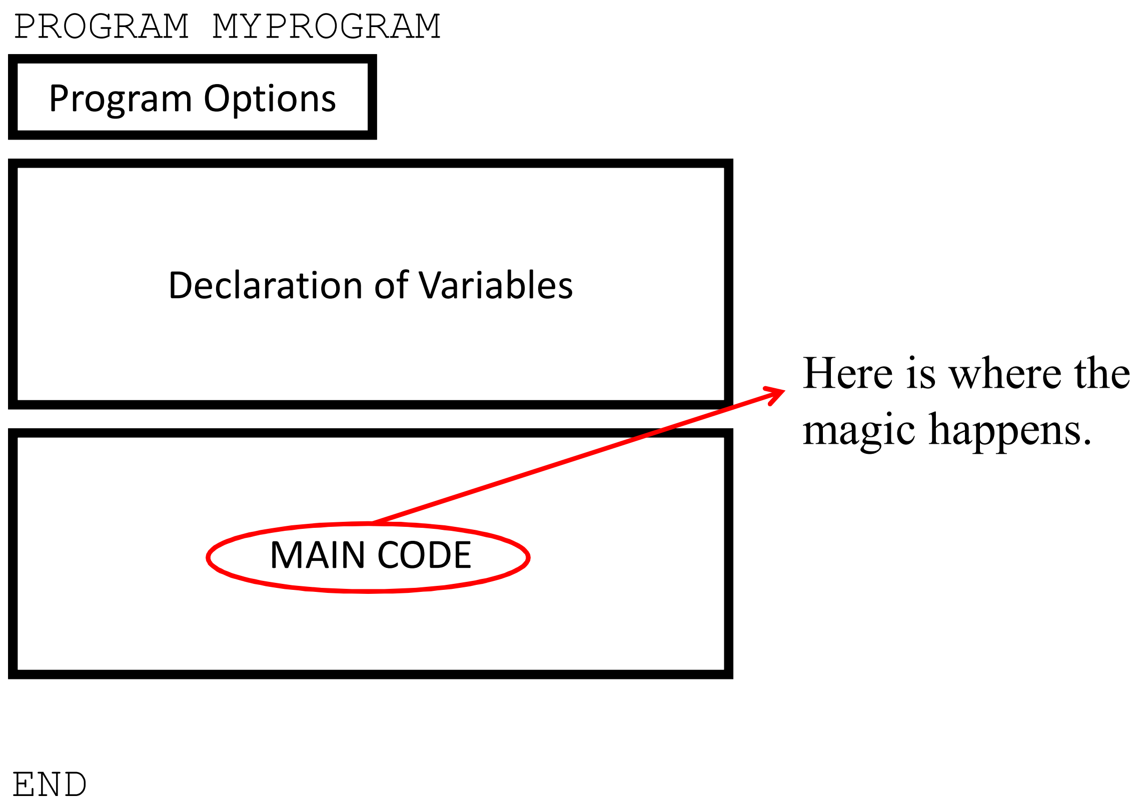
This line identifies the code as a program (instead of, say, a subroutine) and gives it the name MYPROGRAM.

Your Typical Program

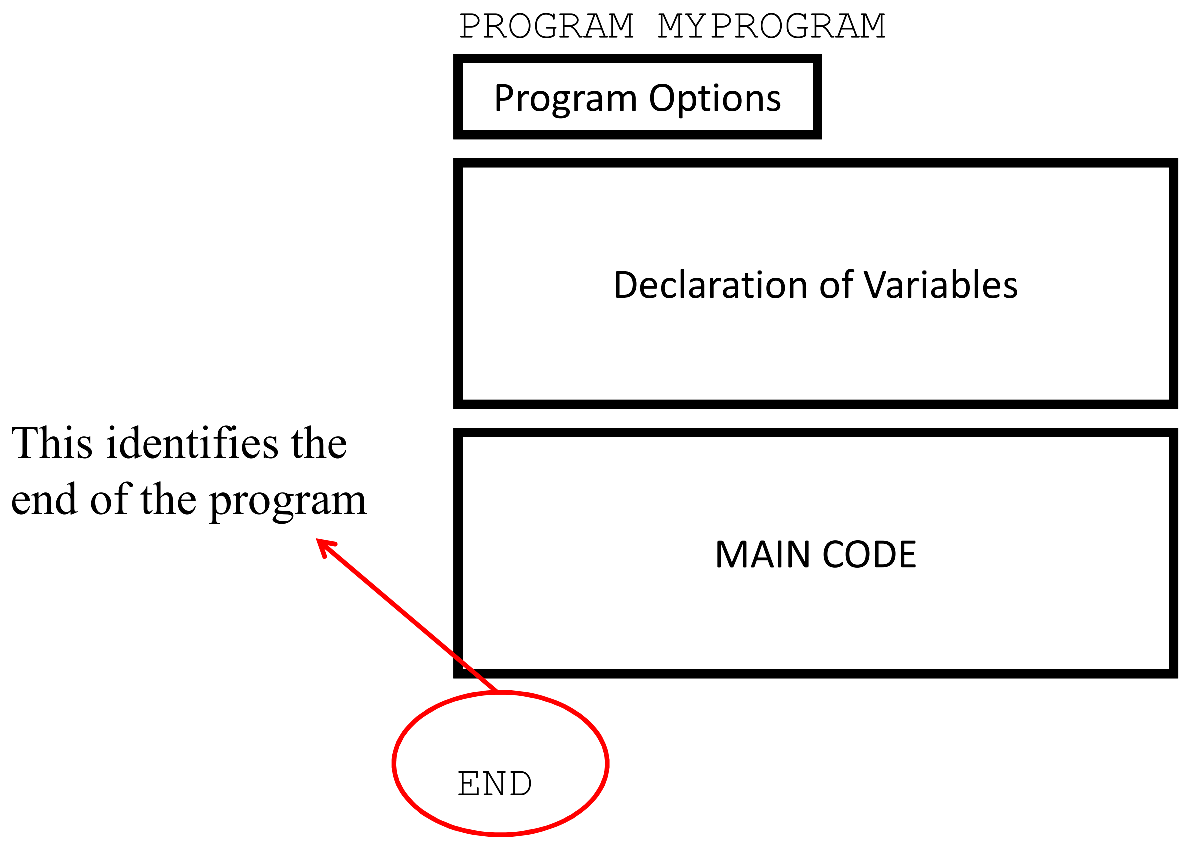
All variables used in the code have to be declared at the top, before any of the main code is run.



Your Typical Program



Your Typical Program



Basic Elements of Fortran

- A statement too long to fit in a single line may be continued on the next line by ending the current line with an & (ampersand). e. g.

```
output = input1 + input2                ! sum the inputs
output = input1  &                      !  Also, sum the inputs
+ input2
```

- A line with a c, C, *, d, D, or! in column one is a comment line. The d, D, and! are nonstandard.
- One can use **labels** in some statements. A label can be any number between 1 and 99999.

Variables and Assignments

- Variables are like small blackboards
  - We can write a number on them
  - We can change the number
  - We can erase the number
- Variable are declared as follows:

```
variable_type :: variable_name [ =< value >]
```

Naming

- must be unique within the program;
- must start with a letter;
- may use only letters, digits and the underscore;
- may not be longer than 31 characters.

name	Valid/not valid	reason
A1	Valid	
1a	Not valid	Starts with number
Atoz	Valid	
A_z	Valid	
A-z	Not valid	Contains -

Types of Data

There are different types of data:

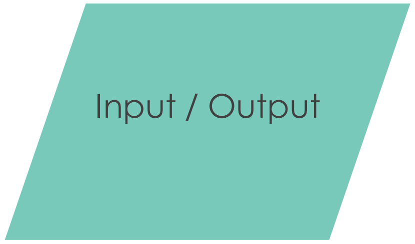
- Integer: numbers that have no decimal part
  - Integer :: v
  - Integer ::y=8
- Real: numbers that can contain decimal parts
  - real :: v
  - real ::y=8.9
- Complex: variables that can take on complex values
  - complex :: v
  - complex ::y=(6,7)
- Character: is used to store strings of characters. To hold a string of characters we need to know how many characters in the string
  - character ::st1*10='kkk'
  - character (len=10) ::uu="kk"

Undeclared variable

- Can you imagine what happen if you forget to declare a variable
- Any undeclared variable has an implicit type:
  - if the first letter of its name is I, J, K, L, M or N then the type is INTEGER;
  - if it is any other letter then the type is REAL.
- Implicit typing is potentially very dangerous and should always be turned off by adding: IMPLICIT NONE
- Put ‘implicit none’ at the beginning
  - Right after the ‘program’ line
  - Prevents implicit variable declaration

Input and output

- **Output statement:**
  - **write:** allows you to output to the default output device using a default format:
    write(*,*)<list>
  - **Example**:
    write (*,*) n1
    write (*,*) n2,ne



- **input statement:**
  - **Read:** allows you to input from the default input device using a default format:
    Read(*,*)<list>
  - **Example**:
    Read (*,*) n1
    Read (*,*) n2,ne