A SHORT COURSE ON THE FORTRAN PROGRAMMING LANGUAGE

P.V. Johnson

School of Mathematics

September 2008
1 **OBJECTIVES**

2 **UNDERSTANDING FORTRAN**
- What is a computer?
- What is a program?
- Program Structure
- Compile and run a simple program

3 **UNDERSTANDING DATA**
- Data and Variables
- Input/Output Data

4 **SUMMARY**
1. Objectives

2. Understanding Fortran
   - What is a computer?
   - What is a program?
   - Program Structure
   - Compile and run a simple program

3. Understanding Data
   - Data and Variables
   - Input/Output Data

4. Summary
OBJECTIVES

UNDERSTANDING FORTRAN
- What is a computer?
- What is a program?
- Program Structure
- Compile and run a simple program

UNDERSTANDING DATA
- Data and Variables
- Input/Output Data

SUMMARY
OUTLINE

1. OBJECTIVES

2. UNDERSTANDING FORTRAN
   - What is a computer?
   - What is a program?
   - Program Structure
   - Compile and run a simple program

3. UNDERSTANDING DATA
   - Data and Variables
   - Input/Output Data

4. SUMMARY
OBJECTIVES

- Understand the Fortran programming language
- Write a Fortran program
- Compile, run and debug Fortran programs
Objectives

Understanding Fortran

- Understand the Fortran programming language
- Write a Fortran program
- Compile, run and debug Fortran programs

Understanding Data

Summary
OBJECTIVES

- Understand the Fortran programming language
- Write a Fortran program
- Compile, run and debug Fortran programs
How does a computer work?
What is a program?
Data and variables.
Input and output.
Flow Control.
Functions and subroutines.
Modular programming.
How does a computer work?
What is a program?
Data and variables.
Input and output.
Flow Control.
Functions and subroutines.
Modular programming.
UNDERSTAND BASIC FORTRAN

- How does a computer work?
- What is a program?
- Data and variables.
- Input and output.
- Flow Control.
- Functions and subroutines.
- Modular programming.
<table>
<thead>
<tr>
<th>Objectives</th>
<th>Understanding Fortran</th>
<th>Understanding Data</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WRITE A FORTRAN PROGRAM</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>📢 The importance of syntax.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>📢 Planning your program.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>📢 Structure of your program.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WRITE A FORTRAN PROGRAM

- The importance of syntax.
- Planning your program.
- Structure of your program.
Compile, run and debug Fortran programs

- How to compile and what options to use.
- Debug your program:
  - Compile-time errors;
  - Run-time errors.
- Run programs and analyse the output.
Compile, run and debug Fortran programs

- How to compile and what options to use.
- Debug your program:
  - Compile-time errors;
  - Run-time errors.
- Run programs and analyse the output.
1. Objectives

2. Understanding Fortran
   - What is a computer?
   - What is a program?
   - Program Structure
   - Compile and run a simple program

3. Understanding Data
   - Data and Variables
   - Input/Output Data

4. Summary
**AN IDEALIZED COMPUTER**

- **CPU - Central Processing Unit**
- **CPU and memory work together**
  - Input may be from keyboard, mouse or a file
  - Output may be to screen or a file
**Objectives**

- Understanding Fortran
- Understanding Data

**Summary**

- CPU - Central Processing Unit
- CPU and memory work together
- Input may be from keyboard, mouse or a file
- Output may be to screen or a file

---

**An Idealized Computer**

- Input
- CPU
- Memory
- File Store
- Output
# OUTLINE

## 1. OBJECTIVES

## 2. UNDERSTANDING FORTRAN
- What is a computer?
- What is a program?
- Program Structure
- Compile and run a simple program

## 3. UNDERSTANDING DATA
- Data and Variables
- Input/Output Data

## 4. SUMMARY
WHAT IS A PROGRAM?

A program is a sequence of instructions to enable a computer to complete a task.

- Computers can remember programs
- Originally programs had to be written in machine code - low-level
- Now we write programs in code - and let the computer write the machine code for us!
- High-level languages include Fortran, C++, Matlab, etc.
What is a program?

A program is sequence of instructions to enable a computer to complete a task.

- Computers can remember programs
- Originally programs had to be written in machine code - *low-level*
- Now we write programs in code - *and let the computer write the machine code for us!*
- High-level languages include Fortran, C++, Matlab, etc.
A program is sequence of instructions to enable a computer to complete a task.

- Computers can remember programs
- Originally programs had to be written in machine code - *low-level*
- Now we write programs in code - *and let the computer write the machine code for us!*
- High-level languages include Fortran, C++, Matlab, etc.
WHAT IS A PROGRAM?

A program is sequence of instructions to enable a computer to complete a task.

- Computers can remember programs
- Originally programs had to be written in machine code - low-level
- Now we write programs in code - and let the computer write the machine code for us!
- High-level languages include Fortran, C++, Matlab, etc.
What is a Program?

A program is sequence of instructions to enable a computer to complete a task.

- Computers can remember programs
- Originally programs had to be written in machine code - *low-level*
- Now we write programs in code - *and let the computer write the machine code for us!*
- High-level languages include Fortran, C++, Matlab, etc.
Example - Write your name to screen...

Formulate

- The computer must ask for the name
- Then print out the name
Example - Write your name to screen...

Formulate

- The computer must ask for the name
- Then print out the name
- Block 1: Output - Print out question to screen
- Block 2: Input - Read in answer from keyboard
- Block 3: Output - Print answer to screen

Analysis
Example - Write your name to screen...

- The computer must ask for the name
- Then print out the name

```fortran
PROGRAM ask_for_my_name
  ! DECLARATIONS
  CHARACTER(LEN=50) my_name
  ! BLOCK 1 ASK FOR NAME
  PRINT *, "What is your name?"
  ! BLOCK 2 READ FROM KEYBOARD
  READ *, my_name
  ! BLOCK 3 PRINT NAME TO SCREEN
  PRINT *, "Your name is ", my_name
END PROGRAM ask_for_my_name
```
**Example - Write your name to screen...**

- The computer must ask for the name
- Then print out the name

```
paulslaptop:Examples> f95
ask_my_name.f90
paulslaptop:Examples> ./a.out
What is your name?
Paul Johnson
Your name is Paul
```
OBJECTIVES

UNDERSTANDING FORTRAN
- What is a computer?
- What is a program?
- Program Structure
  - Compile and run a simple program

UNDERSTANDING DATA
- Data and Variables
- Input/Output Data

SUMMARY
Syntax

- Lines upto 132 chars
- Not case sensitive
- Variable names upto 31 chars
- Semicolon separate statements on one line
- Ampersand (&) for continuation lines
- Conventionally keywords in capitals

```
! Initial Statement
PROGRAM test
! Declarations
TYPE :: data....
! Executable Statements
data = ....
! End Statement
END PROGRAM test
```
Structuring A Program

Building Blocks

Fortran programs are built using **SUBROUTINES** and **FUNCTIONS**

- Split large tasks into smaller, more manageable tasks
- **FUNCTIONS** can be used to represent mathematical functions
- Fortran has many intrinsic functions
- **SUBROUTINES** contain algorithms that are used again and again, or are too long to be placed in the main program
STRUCTURING A PROGRAM

Building Blocks

Fortran programs are built using **SUBROUTINES** and **FUNCTIONS**

- Split large tasks into smaller, more manageable tasks
- **FUNCTIONS** can be used to represent mathematical functions
- Fortran has many intrinsic functions
- **SUBROUTINES** contain algorithms that are used again and again, or are too long to be placed in the main program
OUTLINE

1 OBJECTIVES

2 UNDERSTANDING FORTRAN
   • What is a computer?
   • What is a program?
   • Program Structure
   • Compile and run a simple program

3 UNDERSTANDING DATA
   • Data and Variables
   • Input/Output Data

4 SUMMARY
COMPILING

MAKING MACHINE CODE

A compiler turns your code into language a computer can understand

- Name the file with the suffix .f90
- At command line, type:

  paulslaptop:Examples> f95 program.f90

- If there are errors, edit code and then recompile
- The executable a.out should be created.
COMPILING

**Making machine code**

A compiler turns your code into language a computer can understand

- Name the file with the suffix `.f90`
- At command line, type:

```
paulslaptop:Examples> f95 program.f90
```

- If there are errors, edit code and then recompile
- The executable `a.out` should be created.
EXECUTABLE FILE

With the Unix command `ls -l` we can check for executable files:

```
-rw-r-xr-x 1 paulslaptop paulslaptop 9112 2008-09-17 11:18 a.out
```

- The ‘x’ indicates the file is executable
- Type at the command line:

```
paulslaptop:Examples> ./a.out
```

- The ‘./’ tells Unix to look in the current dir for the program
EXECUTABLE FILE

With the Unix command `ls -l` we can check for executable files:

```
-rwxr-xr-x 1 paulslaptop paulslaptop 9112 2008-09-17 11:18 a.out
```

- The ‘x’ indicates the file is executable
- Type at the command line:

```
paulslaptop:Examples> ./a.out
```

- The ‘./’ tells Unix to look in the current dir for the program
OUTLINE

1. OBJECTIVES

2. UNDERSTANDING FORTRAN
   • What is a computer?
   • What is a program?
   • Program Structure
   • Compile and run a simple program

3. UNDERSTANDING DATA
   • Data and Variables
   • Input/Output Data

4. SUMMARY
Computers can only follow the instructions they are given. If they are not doing what you want it is because you haven’t told them what to do!

- In the previous example we use CHARACTER data type to input our name.
- The program does not work quite how we want...
- So we must understand how the computer uses data.
Computers are not humans!

Computers can only follow the instructions they are given. If they are not doing what you want it is because you haven’t told them what to do!

- In the previous example we use CHARACTER data type to input our name.
- The program does not work quite how we want...
- So we must understand how the computer uses data.
Computers can only follow the instructions they are given. If they are not doing what you want it is because you haven’t told them what to do!

- In the previous example we use CHARACTER data type to input our name.
- The program does not work quite how we want...
- So we must understand how the computer uses data.
TYPES OF DATA

STANDARD DATA TYPES

- Integers - INTEGER - stored exactly
- Real numbers - REAL - floating point approximation
- Complex numbers - COMPLEX - $a + ib$
- Words - CHARACTER
- True/False - LOGICAL

PERSONAL DATA

Later we will see that we can create our own types of data.
Types of Data

Standard Data Types
- Integers - INTEGER - stored exactly
- Real numbers - REAL - floating point approximation
- Complex numbers - COMPLEX - $a + ib$
- Words - CHARACTER
- True/False - LOGICAL

Personal Data
Later we will see that we can create our own types of data.
Declaring Variables

We create variables by assigning a name to a piece of data.

To declare a variable we use the following statement:

```
TYPE :: data_1, data_2
```

substitute **TYPE** for your required data type.

Syntax:

All declarations must be made at the beginning of a program.
DECLARING VARIABLES

**Variable**
We create variables by assigning a name to a piece of data

- To declare a variable we use the following statement

  `TYPE :: data_1, data_2`

- substitute `TYPE` for your required data type

**Syntax**
All declarations must be made at the beginning of a program
**DECLARING VARIABLES**

**VARIABLE**

We create variables by assigning a name to a piece of data.

- To declare a variable we use the following statement:

```
TYPE :: data_1, data_2
```

- substitute **TYPE** for your required data type

**SYNTAX**

All declarations must be made at the beginning of a program.
**USING VARIABLES**

- We can assign a variable a value using `=`

  ```fortran
  data_1 = 10. + 21.5
  ```

- The data types on both sides of `=` must be compatible
- We can add, multiply, and power standard data types

```fortran
REAL a, b, c
b = 5. ; c = 4.1
a = 10. * b + c ** 2.5
```
Using Variables

- We can assign a variable a value using ‘=’

```plaintext
data_1 = 10. + 21.5
```

- The data types on both sides of ‘=’ must be compatible
- We can add, multiply, and power standard data types

```plaintext
REAL a, b, c
b = 5.; c = 4.1
a = 10. * b + c ** 2.5
```
THE COMPUTER MAKES ASSUMPTIONS...

- If a variable name has not been declared, the compiler will assume it’s data type

```fortran
PROGRAM test
a = 1 ; x = 4 ! a-h and o-z assumed real
i = 2 ; j = 3 ! i-n assumed integer
PRINT *, a/i , i/j , j/x
```

- Use the `IMPLICIT NONE` statement to stop the computer making assumptions about variable types.

```fortran
PROGRAM test
IMPLICIT NONE
! Rest of the program...
```
THE COMPUTER MAKES ASSUMPTIONS...

- If a variable name has not been declared, the compiler will assume it’s data type

```fortran
PROGRAM test
a = 1 ; x = 4 ! a-h and o-z assumed real
i = 2 ; j = 3 ! i-n assumed integer
PRINT *, a/i , i/j , j/x
```

- Use the **IMPLICIT NONE** statement to stop the computer making assumptions about variable types.

```fortran
PROGRAM test
IMPLICIT NONE
! Rest of the program...
```
THE COMPUTER MAKES ASSUMPTIONS...

- Operators are overloaded to allow different standard data types to be added, multiplied etc. together

```
REAL  a=2.4
INTEGER b=2,c=3
PRINT *, a/b ! real / int -> Real division
PRINT *, b/c ! int / int -> Integer division
```

- Use intrinsic functions to convert data types to required type

```
REAL  a=2.4
INTEGER b=2,c=3
PRINT *, INT(a)/b ! int / int
PRINT *, FLOAT(b)/FLOAT(c) ! real / real
```
THE COMPUTER MAKES ASSUMPTIONS...

- Operators are overloaded to allow different standard data types to be added, multiplied etc. together

```fortran
REAL  a=2.4
INTEGER b=2, c=3
PRINT *, a/b ! real / int -> Real division
PRINT *, b/c ! int / int -> Integer division
```

- Use intrinsic functions to convert data types to required type

```fortran
REAL  a=2.4
INTEGER b=2, c=3
PRINT *, INT(a)/b ! int / int
PRINT *, FLOAT(b)/FLOAT(c) ! real / real
```
CHARACTER VARIABLES

**Declaring a character string**

We must specify the length of the string (defaults to 1):

```
CHARACTER(LEN=20) :: string
```

- Character strings can provide an important interface between the program and the user
- Strings are written inside quotes
- Some intrinsic functions:
  ```
  string = "Paul" ! assign string a value
  string = string // " Johnson" ! // adds the strings together
  string = TRIM(string) // " Johnson" ! TRIM removes trailing spaces
  ```
CHARACTER VARIABLES

**Declaring a Character String**

We must specify the length of the string (defaults to 1):

```fortran
CHARACTER(LEN=20) :: string
```

- Character strings can provide an important interface between the program and the user
- Strings are written inside quotes
- Some intrinsic functions:

```fortran
string = "Paul" ! assign string a value
string = string // " Johnson" ! // adds the strings together
string = TRIM(string) // " Johnson" ! TRIM removes trailing spaces
```
OBJECTIVES

UNDERSTANDING FORTRAN
- What is a computer?
- What is a program?
- Program Structure
- Compile and run a simple program

UNDERSTANDING DATA
- Data and Variables
- Input/Output Data

SUMMARY
**Input Data**

**READ FROM THE KEYBOARD**

Use a **READ** statement to read from the keyboard.

- Using ‘*’ allows the compiler to choose format

```fortran
READ *, data_1, data_2, string
```

- Variables can be separated a comma, a space, or a carriage return.
Input Data

**Read from the Keyboard**

Use a **READ** statement to read from the keyboard.

- Using ‘*’ allows the compiler to choose format

```
READ *, data_1, data_2, string
```

- Variables can be separated a comma, a space, or a carriage return.
**WRITE TO THE SCREEN**

Use a **PRINT** statement to print to the screen.

- Using ‘*’ allows the compiler to choose format

  ```fortran
  PRINT *, data_1, data_2, string
  ```

- Variables are separated by a comma.
WRITE TO THE SCREEN

Use a PRINT statement to print to the screen.

- Using ‘*’ allows the compiler to choose format

PRINT *, data_1, data_2, string

- Variables are separated by a comma.
The story so far...

- Understand a little bit about computers
- Understand about data types and how to assign them
  - Be able to input data from keyboard...
  - and output to screen
- Should be ready to write first programs...
Objectives

Understanding Fortran

Understanding Data

Summary

THE STORY SO FAR...

- Understand a little bit about computers
- Understand about data types and how to assign them
- Be able to input data from keyboard...
- and output to screen
- Should be ready to write first programs...
The story so far...

- Understand a little bit about computers
- Understand about data types and how to assign them
- Be able to input data from keyboard...
- and output to screen
- Should be ready to write first programs...