Introduction to Modern Fortran

See next foil for copyright information

Nick Maclaren

nmm1@cam.ac.uk

March 2014

Acknowledgement

Derived from the course written and owned by

Dr. Steve Morgan
Computing Services Department
The University of Liverpool

Important!

There is a lot of material in the course
And there is even more in extra slides ...

Some people will already know some Fortran
Some will be programmers in other languages
Some people will be complete newcomers

The course is intended for all of those people

- Please tell me if I am going too fast

Not afterwards, but as soon as you have trouble

Beyond the Course (1)

...

.../Fortran/
.../OldFortran/
.../Arithmetic/ etc.

Programming in Fortran 90/95
by Steve Morgan and Lawrie Schonfelder
(Fortran Market, PDF, $15)
http://www.fortran.com/

Also Fortran 90 version of that

Beyond the Course (2)

Fortran 95/2003 Explained
by Michael Metcalf, John Reid and Malcolm Cohen

Also Fortran 90 version of that

Fortran 90 Programming
by Miles Ellis, Ivor Phillips and Thomas Lahey

Beyond the Course (3)

SC22WG5 (ISO Fortran standard committee)

http://www.nag.co.uk/sc22wg5/

http://www.fortran.com/fortran/

$\Rightarrow$ 'Information', 'Standards Documents'

Miscellaneous information and useful guidance

http://www.star.le.ac.uk/~cgp/fortran.html

Liverpool Course

http://www.liv.ac.uk/HPC/...

.../HTMLFrontPageF90.html

Beyond the Course (4)

A real, live (well coded) Fortran 95 application

http://www.wannier.org

Most of the others I have seen are not public
Please tell me of any you find that are

Important!

There is a lot of material in the course
And there is even more in extra slides ...

This has been stripped down to the bare minimum
Some loose ends will remain, unfortunately
You will need to skip a few of the practicals

- Please tell me if I am going too fast

Not afterwards, but as soon as you have trouble

Practicals

These will be delayed until after second lecture
Then there will be two practicals to do

One is using the compiler and diagnostics
Just to see what happens in various cases

The other is questions about the basic rules

Full instructions will be given then
Including your identifiers and passwords

History

FORmula TRANslation invented 1954–8
by John Backus and his team at IBM

FORTRAN 66 (ISO Standard 1972)
FORTRAN 77 (1980)
Fortran 90 (1991)
Fortran 95 (1996)
Fortran 2003 (2004)
Fortran 2008 (2011)

The "Old Fortran" slides have more detail

Hardware and Software

A system is built from hardware and software

The hardware is the physical medium, e.g.
- CPU, memory, keyboard, display
- disks, ethernet interfaces etc.

The software is a set of computer programs, e.g.
- operating system, compilers, editors
- Fortran 90 programs

Programs

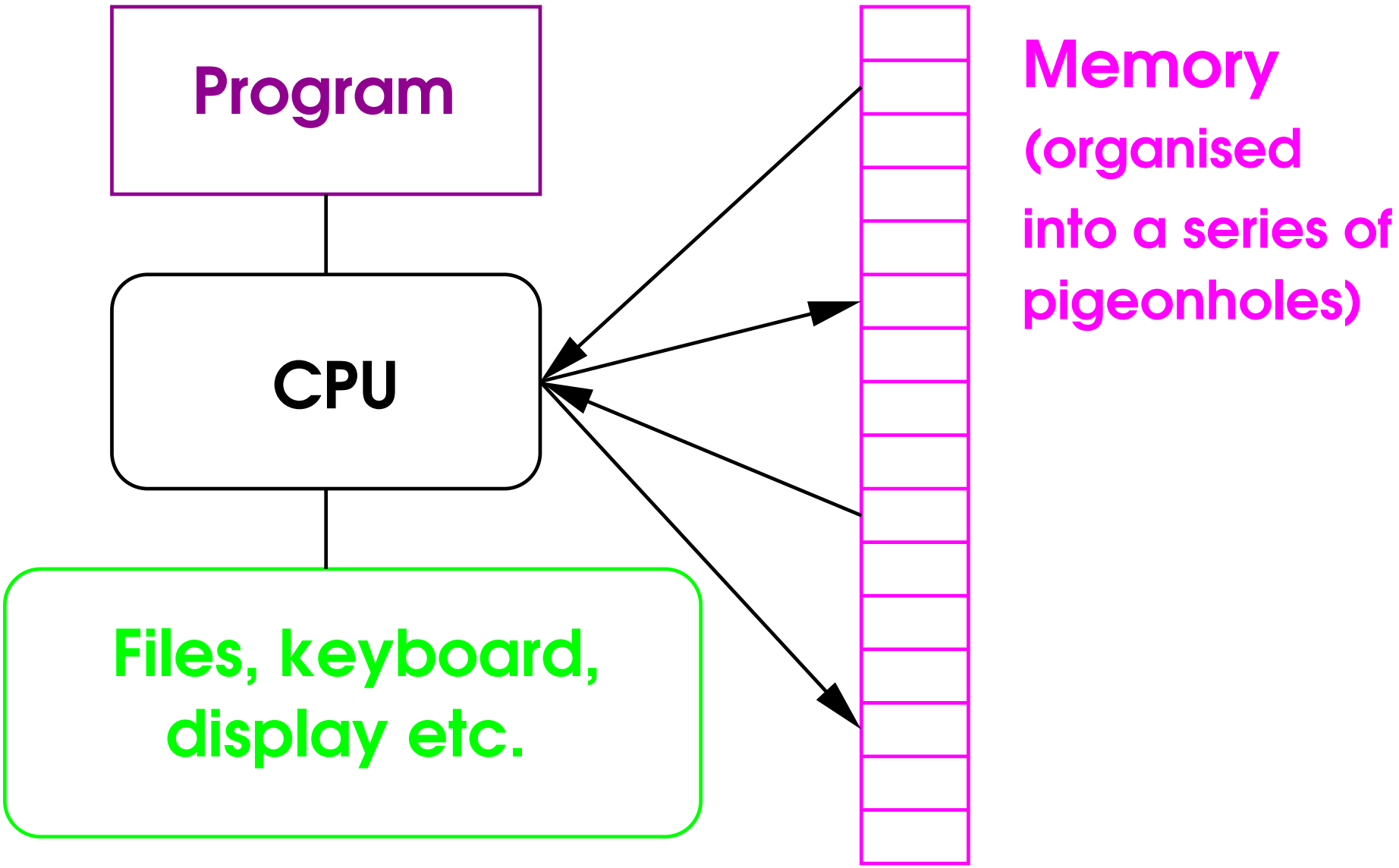
Fortran 90 is a high–level language
Sometimes called "third–generation" or 3GL

Uses English–like words and math–like expressions

```
Y = X + 3
PRINT *, Y
```

Compilers translate into machine instructions
A linker then creates an executable program
The operating system runs the executable

Fortran Programming Model



Algorithms and Models

An algorithm is a set of instructions
They are executed in a defined order
Doing that carries out a specific task

The above is procedural programming
Fortran 90 is a procedural language

Object–orientation is still procedural
Fortran 90 has object–oriented facilities

An Example of a Problem

Write a program to convert a time in hours, minutes and seconds to one in seconds

Algorithm:

1. Multiply the hours by 60
2. Add the minutes to the result
3. Multiply the result by 60
4. Add the seconds to the result

Logical Structure

1. Start of program
2. Reserve memory for data
3. Write prompt to display
4. Read the time in hours, minutes and seconds
5. Convert the time into seconds
6. Write out the number of seconds
7. End of program

The Program

```
PROGRAM example1
! Comments start with an exclamation mark
      IMPLICIT NONE
      INTEGER :: hours, mins, secs, temp
      PRINT *, 'Type the hours, minutes and seconds'
      READ *, hours, mins, secs
      temp = 60* ( hours*60 + mins ) + secs
      PRINT *, 'Time in seconds =', temp
END PROGRAM example1
```

High Level Structure

1. Start of program (or procedure)
   PROGRAM example1
2. Followed by the specification part
   declare types and sizes of data

3–6. Followed by the execution part
   all of the ‘action’ statements

7. End of program (or procedure)
   END PROGRAM example1

Comments do nothing and can occur anywhere
   ! Comments start with an exclamation mark

Program and File Names

- The program and file names are not related

PROGRAM QES can be in file QuadSolver.f90
Similarly for most other Fortran components

Some implementations like the same names
Sometimes converted to lower– or upper–case

The compiler documentation should tell you
It is sometimes in the system documentation
Please ask for help, but it is outside this course

The Specification Part

2. Reserve memory for data

```
INTEGER :: hours, mins, secs, temp
```

INTEGER is the type of the variables

hours, mins, secs are used to hold input
The values read in are called the input data
temp is called a workspace variable
also called a temporary variable etc.
The output data are 'Time . . . =' and temp
They can be any expression, not just a variable

The Execution Part

3. Write prompt to display
   ```
   PRINT *, 'Type the hours, ...'
   ```
4. Read the time in hours, minutes and seconds
   ```
   READ *, hours, mins, secs
   ```
5. Convert the time into seconds
   ```
   temp = 60*( hours*60 + mins) + secs
   ```
6. Write out the number of seconds
   ```
   PRINT *, 'Time in seconds =', temp
   ```

Assignment and Expressions

temp = 60*(hours*60 + mins) + secs

The RHS is a pseudo–mathematical expression
It calculates the value to be stored

- Expressions are very like A–level formulae

Fortran is FORmula TRANslation – remember?
We will come to the detailed rules later

- temp = stores the value in the variable

A variable is a memory cell in Fortran's model

Really Basic I/O

READ *, <variable list> reads from stdin

PRINT *, <expression list> writes to stdout

Both do input/output as human–readable text
Each I/O statement reads/writes on a new line

A list is items separated by commas (‘,’)
Variables are anything that can store values
Expressions are anything that deliver a value

Everything else will be explained later

Repeated Instructions

The previous program handled only one value
A more flexible one would be:

1. Start of program
2. Reserve memory for data
3. Repeat this until end of file
   - 3.1 Read the value of seconds
   - 3.2 Convert to minutes and seconds
   - 3.3 Write out the result
4. End of Program

Sequences and Conditionals

Simple algorithms are just sequences
A simple algorithm for charging could be:

1. Calculate the bill
2. Print the invoice

Whereas it probably should have been:

1. Calculate the bill
2. If the bill exceeds minimum
   2.1 Then print the invoice
3. Otherwise
   3.1 Add bill to customer's account

Summary

There are three basic control structures:

- A simple sequence
- A conditional choice of sequences
- A repeated sequence

All algorithms can be expressed using these
In practice, other structures are convenient

Almost always need to split into simpler tasks
Even Fortran II had subroutines and functions!
Doing that is an important language–independent skill

Developing a Computer Program

There are four main steps:

1. Specify the problem
2. Analyse and subdivide into tasks
3. Write the Fortran 90 code
4. Compile and run (i.e. test)

Each step may require several iterations
You may need to restart from an earlier step

- The testing phase is very important

Errors

- If the syntax is incorrect, the compiler says so

For example: INTEGER :: ,mins, secs

- If the action is invalid, things are messier

For example: X/Y when Y is zero
/ represents division, because of the lack of ÷

You may get an error message at run–time
The program may crash, just stop or hang
It may produce nonsense values or go haywire