Fortran 77 : Introduction

Fortran is the oldest programming language and it is still in use today. Fortran has undergone several transitions since it was first introduced over 50 years ago. The name Fortran is an abbreviation of "formula translation" and hence it takes a scientific view of programming and Fortan is still in use today and is particularly used by the scientific and engineering communities. Although Fortran has been to some extent superseded by the modern programming languages such as Matlab/Freemat[1], there are many scientists and engineers that are still familiar with Fortran and a good deal of legacy software that is still used that was originally written in Fortran. In this set of tutorials we focus on the Fortran version from the middle of Fortran's life and that is probably still the most popular version. Fortran 77 also serves as a good initial step into the Fortran programming Language.

A Fotran computer programs - like all computer programs - is fundamentally a text file. However, that text file has to follow strict rules that are laid down in the specification of the language if it is to be translated into an executable program. The computer program is fundamentally a means by which the programmer communicates with a computer. For larger Fortran systems, it is usually more convenient to separate the discernible parts of the system into separate text files.

In order to run a Fortran program, first each file is sent to a compiler. Once all the parts of the system are compiled, they are linked to form one complete program and an executable (.EXE) file is produced. The software product is the executable file. Compilers often come with and IDE (integrated development environment). This supports the management of the development of computer software. The IDE provides a text editor for writing and editing programs, a direct means of compiling and editing using button clicks and keeping files associated with a system together in a project.

There is a world of difference between being able to program and being able to program well. Programs are often revised by the programmer, or written by a team so that a computer program must be viewed also as a means of communication, also for the benefit of the programmer (or team). Also in learning a language like Fortran 77 we are learning a means of communication, this is important, but it is important to note that what we communicate is ultimately more important than the means of communication. Just like a novel may be judged on character and plot rather than the language it is written in, the design of a computer program (for example the use of algorithms[1] or data structures[2]) is fundamentally more important than the language. So while it is important to become reasonably proficient in programming languages, it is not sufficient in order to become an experienced programmer.

This document introduces the Fortran 77 language, gives the outline structure and some of the simple commands.

---

[1] Matlab/Freemat

Outline structure

Let us begin with the most simple Fortran program. The following program is called SHELL since it is simply the shell of a program; it doesn't actually do anything.

```
      PROGRAM SHELL
      END
```

However, the program does show a number of aspects of a Fortran program. Firstly the instructions in the program should not start until at least column 6 in the file. Secondly tne program should start with the PROGRAM command and end with the END command. In this case "SHELL" is the identifier given to the program and this follows the PROGRAM instruction. Note also that the program identifier must also follow the rules of identifiers.

Upper of Lower Case?

Note that Fortran is not case sensitive. The following program is identical to the program SHELL above.

```
      program shell
      end
```

Note also that traditionally Fortran was written in upper case (apart from output). More recently many Fortran programmers prefer to use lower case. In this tutorial on Fortran we will ten to use upper case.

Comments

Comments are lines in the program that are ignored by the compiler, but are useful to the developers of a program. Comments are used to annotate the program. They are normally used within the program to state the purpose of a piece of code or note the method used. They may also be used to give a title page to the program. A comment line is designated by placing a 'C' in the first column, and following this by the text. The following program shows how comments may be used.

```
C*********************
C Hello World
C*********************
      PROGRAM HELLO2
C Output 'Hello World'
      WRITE(*,*) 'Hello World!'
      END
```

If this file is compiled and executed the output would be as follows.

```
Hello World!
```

As with all aspects of programming, there are guidelines of good practice that we should observe when writing comments. There should a balance between the amount of comment lines and he amount of code. Also, especially if you are working in a team, a standardised approach to the documentation should be adopted by the team as a whole to aid communication between team members.

Note also in the above program that there is a WRITE statement. It should be clear from its name and its function that this is an output statement.

Flow and STOP

The normal flow of a Fortran program is to execute each line in the sequence that they are written. For example the following program gives the output shown.

```
C*********************
C Hello World 2       *
C*********************
      PROGRAM HELLO2
      WRITE(*,*) 'Hello World!'
      WRITE(*,*) ' I love Fortran'
      STOP
      END
```

```
Hello World!
I love Fortran
```

However, the vertical flow of the program can be subverted - and has to be for any serious program – by for example using branches and loops. Note that the STOP command can be used to terminate the program.

Labels

Labels are used to refer to one line of the code, usually from another line of code. Labels are defined by the programmer, range between 1 and 999 and must be placed starting at column one of the program. For example the following program places the label 100 on one line.

```
C*********************
C Hello World 3
C*********************
      PROGRAM HELLO3
      GOTO 100
      WRITE(*,*) 'Hello World!'
100   WRITE(*,*) ' I love Fortran'
      STOP
      END
```

In this program the output is as follows

```
I love Fortran
```

Note that the GOTO statement alters the flow of the program by causing a jump to line 100. The compiler may complain that the code WRITE(*,*) 'Hello World!' is redundant, that is it can never be executed and may as well not be there.

CONTINUE

The CONTINUE command does not do anything and is simply useful as a place holder. For example the following code uses the CONTINUE statement to act as a place holder for the label. The code has the same output as HELLO3.

```
C*********************
C Hello World 4       *
C*********************
      PROGRAM HELLO4
      GOTO 100
      WRITE(*,*) 'Hello World!'
100   CONTINUE
      WRITE(*,*) ' I love Fortran'
      STOP
      END
```