Dr K B Singh Lecture Notes
PG II Sem Physics

MPHYCC-5: Elementary Idea about FORTRAN

Fortran (FORmula TRANslation) is a programming language designed specifically for scientists and engineers. For the past 30 years Fortran has been used for such projects as the design of bridges and aeroplane structures, it is used for factory automation control, for storm drainage design, analysis of scientific data and so on. Throughout the life of this language, groups of users have written libraries of useful standard Fortran programs. These programs can be borrowed and used by other people who wish to take advantage of the expertise and experience of the authors, in a similar way in which a book is borrowed from a library. Fortran belongs to a class of higher-level programming languages in which the programs are not written directly in the machine code but instead in an artifical, human-readable language. This source code consists of algorithms built using a set of standard constructions, each consisting of a series of commands which define the elementary operations with your data. In other words, any algorithm is a cookbook which specifies input ingredients, operations with them and with other data and finally returns one or more results, depending on the function of this algorithm. Any source code has to be compiled in order to obtain an executable code which can be run on your computer. The compilation is performed by a separate program, called compiler, which translates your text-based source code into the machine code that can be directly interpreted by your computer processor. The task of writing a program involves several steps which are common to all programming languages:

• **problem specification –** each program has a motivation to be written. This is usually represented by a particular assignment which should clearly state what is the purpose of writing your code.

• **analysis of the problem –** this is a very important step prior to writing a new code, which is often omitted by beginners. When writing more complicated algorithms, you should always start on a paper. If the algorithm has to solve a particular mathematical problem, carry out the derivation or expression of a particular unknown on the paper. Be careful to check the physical dimensions of the input and output parameters of your function.

• **writing the source code –** once finished with the analysis of your problem, you can write a source code of this algorithm. Fortran provides you with a number of intrinsic functions, mostly standard mathematical operations like square root, sine, cosine, exponential function or logarithm, which you can directly use in your code. Any other mathematical functions can be written as separate subprograms using a set of standard arithmetic operations. You can thus build a library of the most useful mathematical functions and simply call them any time you need them.

• **compiling the code –** means the same as translating your source code from the language of Fortran to the language of your computer. Compilation produces an executable code which can be subsequently interpreted in the processor of your computer. Have you ever tried to display the contents of an .EXE file under Windows ? If so, then you know what the machine code really looks like.

• **running and testing the program –** although your program may be running without apparent errors, it can have a number of hidden bugs. Take some time to play with the running code and test if it really does what it should. It frequently happens that you mistype some arithmetic operation in your source code, which in turn might give you totally different results. Never believe that you write a clever program without a detailed testing.

To write programs in Fortran, you will need a good editor which allows you to type your source code. Many simple editors like Notepad under Windows or pico, nano under Linux do not offer you the functionality that you will certainly need for writing larger codes. Good text editor should allow you to display your code in color so that comments, identifiers, variables and commands are distinguished from each other. Moreover, because F77 imposes special requirements on the indentation of some parts of your source code, it is useful to have an editor which can, usually after pressing Tab, automatically set the cursor at the position where you have to start writing your text. I strongly recommend you to use emacs which is an intelligent programmer's editor that has all the features you can imagine. More importantly, it exists on many platforms (Windows, MacOS, Linux, ...) and therefore once you become familiar with using it on one platform, you can equally well type your text under Linux or on the Mac in your lab.

In contrast, the compilation of your code is a step which is strongly platform-dependent which means that the executable codes are not transferrable

between different operating systems. This is not so bad as it may look for the first sight. If your friend needs to run your code on Mac, whereas you work with Windows, simply give them your source code and ask them to compile it on their computer. Each platform you may come in contact with is nowadays equipped with a good F77 compiler, some of which are even distributed for free.

In this course, we will use g77 (GNU Fortran) compiler which can be downloaded from the Internet for both Windows and MacOS. If you intend to use Linux, this compiler is most likely embedded in your distribution. The communication between you, a programmer, and your computer will always occur via the command-line terminal. If you expected that you will learn some special windows-based application for writing and compiling F77 codes, you are now probably a little disappointed.

However, you will shortly understand that using the command-line allows you to focus on mere writing your code, rather than on fighting with a new graphical application. Once you get into using your command-line on one platform, it will be very simple for you to use any other operating system. This document will guide you through download, installation and setup of everything what is needed to get started with programming on your computer. During this journey, operations which differ under Windows and MacOS X are grouped under different icons.

Please note that the installation procedure for MacOS is devoted to version 10 which contains a Linux command-line terminal. Because the target audience of this text are mainly undergraduate students who do not have any previous knowledge of programming, each problem will be treated in the simplest and most straightforward way. Once you become familiar with programming in F77, you will quickly find out that the same problem can be solved many different ways.