Introduction to Computer Programming Using FORTRAN 77

- Al-Dhaher, K.
- Al-Muhtaseb, H.
- Yazdani, J.
- Garout, Y.
- Nazzal, A.
- Zeidan, Y
- Lafi, A.
- Saeed, M.

August 1995 Second Edition

Information and Computer Science Department
College of Computer Sciences and Engineering
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia

CONTENTS

1 INTRODUCTION

ICS101 is an introductory course on computer programming. The goal of this course is to teach students the use of computers as tools to solve engineering and scientific problems.

We use many tools in our daily life, from simple things like pens and screwdrivers, to complicated things like watches, radios and TV remote controls, to more complicated things like calculators, television sets, video cameras and cars. More recently computers have been emerging as tools that are used in everyday life. Just as any other tool we should know how to use them properly. It would also be useful, though not necessary, to know how they work and what affects their behavior. If we know for example the structure of a compass, and that it uses a magnet as one of its components, and we know that magnetic fields affect each other, we can understand the behavior of the compass if another magnet is placed beside it.

Knowing how to use a tool or device involves knowing what it can do for us, how we should express what we want to do with the device (e.g. by pressing a key on a calculator, or turning the knob of a radio) and how to receive and interpret the result from this device (e.g. read the sum of the numbers from the calculator display, or listen to the sound from the radio speaker).

Computers vary in size, shape and function. There are small computers and big computers. Large computers are referred to as mainframes. Smaller computers are classified either as minicomputers or microcomputers. Some are used for a specific task, others are general purpose. This variation is similar to the variation in many other devices and tools. There are different screwdrivers, radios and cars. The proper tool for the task should be used. A truck should be used to carry heavy machinery, while a car would be used to carry people (and not the other way around).

A mainframe computer is a powerful machine that can serve hundreds of users that work on it through terminals scattered around and connected to the mainframe through a computer network. The terminals are used by computer users to enter data, write programs and see their results. All the computing is done by the mainframe.

There are other kinds of computers. Personal computers are getting more popular. These are computers that are mainly used by a single person at a time. They have attachments or devices for entering the data and programs, reading the results, as well as performing the actual computing. When you want to use a computer, big or small, you should at least know:

- what the computer can do for you (it might also be useful to know what it cannot do for you);

- the problem you want to solve, and understand it well;
- how to solve the problem;
- how to express the solution to the computer (what you want it to do for you); and
- how to receive and interpret the results.

Remember that the computer is a tool, just like a car for example. If you want to get somewhere, but you do not know where that place is, or how to get there, the car is useless. You have to know how to drive the car, in addition to knowing how to get to your destination from wherever you are. In the remainder of this introductory chapter, we will briefly describe the basic components of computers, and how to interact with them.

1.1 Computer System Components

We can think of computers as devices or machines that are capable of performing certain tasks. A very simple task, for example, is addition. Different computers might have different abilities, but in general, they have a similar internal structure. A typical computer should have input devices to receive input from the user, output devices to enable users to observe and interpret the results, a central processing unit to enable it to perform the needed operations and tasks, and memory to store all the data and programs it needs. An example of an input device is a keyboard or a mouse, an output device can be a video screen or a printer. The physical devices that make up the computer are called "*Hardware*".

1.2 Programs & Programming Languages

Arabic, English, French and other languages, are called natural languages. They are languages used by people to communicate with each other. To communicate correctly, people have to agree on a common language. If you go to Japan and start speaking in Arabic, even if you say simple things like "What time is it?", people will not understand what you are saying. A common language that is understood by both parties has to be used.

Even though there are grammar rules to control the language (what is linguistically correct and what is not), sometimes different interpretations of the same word or sentence are possible, which could be understood by the duration of breaks between words, tone of voice, facial expression, and so on. Some sentences are difficult to understand, even by humans. "*I saw Ahmed on a hill with a telescope*" could be interpreted in different ways. This problem is called the *ambiguity* of natural language. For these reasons - to avoid ambiguity and different interpretations - restricted special languages that have simpler grammars (structure) and restricted vocabulary, are used to communicate with machines (computers in particular). These are called computer programming languages.

Computers are electronic devices. They can only interpret electrical signals. They can be programmed based on their ability to interpret these electrical signals; by asking them to perform different tasks when they detect a signal or when they do not detect a signal. For example, if there are three wires that must have an electrical signal of [+5]

volts to indicate that there is a signal [interpreted as ON or 1], and [-5] volts to indicate that there is no signal [or simply as OFF or 0], the computer can be instructed to interpret the sequence [000] to be the number zero and [001] to be the number one, and [010] to be the number two, and [011] to be the number three, and so on. This is called the *binary system*. A program expressed in this form is usually said to be written in machine language. This language is also known as *low level language* because it is close to the machine hardware structure.

However, to perform any non-trivial task, thousands and thousands of these data values and instructions have to be written, and any mistake could lead to undesirable behavior. It is also extremely difficult to write these instructions and to correct them if there are errors. For this reason, it was suggested to assemble or group some of these binary digits into symbols, called *mnemonics*, and write a program (called an *assembler*) to read these symbols and convert them to machine code. These programs are known as assembly language programs. Assembly programs are at a higher level (in the programming language hierarchy) than machine code. Assembly programs are easier to write than machine code, but it is still difficult for humans to write and modify them. This is why *high level programming languages* were introduced. In a high level language, the programmer uses a *compiler*, which takes each statement in the programs, and translates it to machine code for the computer to understand.

1.2.1 Programs

In section 1.1, we mentioned that computers are machines that perform certain tasks, such as addition. We have to express what tasks we want it to perform, and in what order. If we tell the computer that we want it to add two numbers, it would know how to do that. For example in FORTRAN we can say X = 3 + 5. This asks the computer (we will see how later) to add 3 and 5, and store the value in X. This is a simple command, or program statement, that uses the computer's ability to perform the addition task or operation. A sequence of such statements is called a *program*.

A program is a sequence of statements that fully and clearly describes how a problem should be solved. The programs that tell the computer what to do, are usually called "*Software*".

A program should be written in a language that the computer understands. There are different kinds of languages used for different purposes. Some of the most widely used programming languages include FORTRAN, PASCAL, C, LISP, COBOL and PROLOG. All of these languages are high level programming languages.

1.3 Software Life Cycle

The production of software is similar to the production of artifacts in other engineering fields. A building, for example, might be constructed by laying bricks here and there, without an overall plan or a blue-print. However except for the simplest of buildings, the results would not be satisfactory, unsafe to say the least. The correct engineering method of constructing a building requires that the architect or civil engineers understand the requirements for constructing the building (e.g. residential), produce a preliminary design, verify it with the customer and modify the design accordingly, before the actual building is constructed. The process of software design is similar. The

programmer, or software engineer, should understand and analyze the problem to be solved well before any program is written. After the problem is analyzed, the approach for solving the problem should be identified. A solution is then designed and developed. After a solution is identified, the programmer can start writing the program code. After the code is written it has to be verified and checked for any mistakes or inconsistencies with the requirements, and the process is then repeated until the program behaves as required.

1.4 Modular Software Design

One approach for software development that has been shown to be effective for the production of large software systems is *stepwise refinement* or *top-down design*. Stepwise refinement is a form of *divide and conquer* strategy of problem solving. The basic idea is to divide the problem being solved into a number of steps, each of which can be described by an algorithm which is simpler and more manageable than an algorithm that describes the complete problem as a whole. Using this approach, problems that might seem difficult at the beginning are reduced to smaller problems that can be handled individually. In large software projects, different software engineers work on different sub-problems or modules. When they are done, the process of combining the modules to construct the solution of the original problem is conducted, and is usually straight-forward.

In this course, we apply the concepts of top-down design to solve simple scientific and engineering problems. The knowledge that you gain while you develop skills in top-down design will be valuable for you in other areas of problem solving in your field of study, not only in programming and software development.

1.5 Software Systems and Tools

To develop software, programmers need to use certain systems and tools. In this section we introduce some of the tools we will be using in this course. These include an editor and a compiler. All these tools are programs used by the computer system to assist the programmer in developing, running and maintaining programs.

1.5.1 Editors

To write programs and enter data in the computer, the programmer or user needs to use a tool called an *editor*. The editor allows the user to create and modify *files*. You can think of a file as a reserved area to write programs and data, just as you can write it on a piece of paper. However to enable the computer to read your program, it has to be written in a file, in a form that the computer can interpret. We will see in section1.5.3 the form of a FORTRAN program.

Editors allow their users to add, modify and delete things from a file. These things include characters, words, lines, pages and so on. There are some editors that offer other features and facilities. These include checking spelling mistakes, repeating words, lines and other things. In some systems, you can edit more than one file at the same time. You can copy from file to file. The features of editors are many and we will not attempt to enumerate them here. It suffices to know the purpose of using an editor, and that there are several kinds of editors available for use.

1.5.2 Compilers

In section 1.5.1, we mentioned that an editor enables the programmer to create files of programs and data according to specific forms. Some programming languages require that the program be written in a specific form so that it is easy to interpret. The computer uses a program called a *compiler* to read the program from a file that the programmer writes in, and converts the program into machine language. The FORTRAN compiler requires that the program be written in a specific form so that the compiler can perform the conversion to machine language.

1.5.3 FORTRAN Programs

FORTRAN (FORmula TRANslation) was developed in the fifties as a programming language for scientific and engineering applications. In 1977, standards for FORTRAN were revised, which resulted in a version of FORTRAN that came to be known as FORTRAN77. This is the version of FORTRAN that we will be using in this course. Using any editor, a programmer writes his/ here program in a file. A file consists of a collection of lines (which could also be called statements or records). The FORTRAN compiler requires that all program statements or lines, have a specific structure. A line can hold a maximum of 80 characters. Thus you can think of the program file as having 80 columns. The first position on the line is column one, the second position is column two and so on. Each program statement must begin in a new line and must be typed between columns 7 to 72 of the file. The compiler ignores any characters in columns 73 to 80. Columns 1 to 5 are used to include a label or a statement number, which is used to identify a specific line or statement of the program. Column 6 is used for continuation, which might be needed if the program statement or line is too long to fit in columns 7 to 72. Any character, except a zero, placed in column 6, indicates that this line is a continuation of the previous line.

A "*" or the character "C" in column one indicates that the line is a comment line. The compiler ignores what is typed on a comment line and does not execute it. This is useful for programmers to write descriptions of the different parts of their programs.

Each program should end with the "**END**" statement. This signifies the physical end of the program. The **STOP** statement signals the logical end of the program. While the **END** statement appears at the end of the program, the **STOP** statement may appear anywhere in the program, possibly, to stop execution of the program under certain conditions. The compiler sequentially executes each statement in the program. Exceptions to this sequential execution is possible using special FORTRAN statements such as **GOTO**, **IF** and **DO**. These are used to perform selection and repetition, as we shall see in later chapters.

1.5.4 Conclusion

In this course, you will be introduced to the basic concepts of computing and computer programming. The skills you gain in this course will enable you to start using computers as tools to solve the engineering and scientific problems you will encounter during your study. You should keep in mind that what you encounter in this course is but a drop in the ocean. The field of computer science is growing rapidly. As scientists and engineers, it is important to educate ourselves in different areas of technology. Without this new

technology, we will not be able to succeed and excel in our studies. It is also important to continue educating ourselves by identifying new developments in these areas. This course is the starting point. You should continue this process in order to remain competitive. Accordingly, when you study the material in this course, you should attempt to relate it to your field of study, and consider how the use of such tools can facilitate and enhance your productivity, and aid in the understanding of the material that you have already taken as well as the material that you will study in the future.

1.6 Exercises

1. Indicate the following statements as either TRUE or FALSE:
    1. Syntax errors are detected during compilation.
    2. A compiler is a hardware component that translates programs written in a high level language to a machine language.
    3. The input unit is the part of the computer that controls all the other parts.
    4. The last statement in a FORTRAN program should be the **END** statement.
    5. FORTRAN is a high level language.
    6. A comment statement is used for documentation purposes.
    7. Dividing by zero will cause a compilation error.
    8. If a FORTRAN statement exceeds column 72, then '+' at column # 6 in the next line can be used to continue the statement on that line.
    9. A computer is a machine used to solve problems only.
    10. A compiler checks the syntax of the program and converts the program into machine language.
    11. A program is a set of computer instructions.
    12. One can use as many '**STOP**' and '**END**' statements as he/she wishes in a single program.
2. Which of the following statement(s) is/are correct according to FORTRAN:
    A. Only column 1 is used for the statement label.
    B. Column 6 is used for comment.
    C. Column 1-5 is used for the statement label.
    D. Column 7 is used for the continuation line.
    E. Characters C or * in Column 1 is used to comment a line.
3. For each item of list (A), choose the correct definition from list (B) :

List A	List B
Assembler	1. A machine that converts an assembly language program into machine language.
Compiler	2. The physical components of a computer.
Software	3. A machine that converts a high level language program into machine language.
Hardware	4. A fundamental computer component that controls the operations of the other parts of the computer.
	5. Programs used to specify the operations in a computer.
	6. A fundamental computer component that performs all arithmetic and logic operations.
	7. A program that converts an assembly language program into machine language.
	8. A program that converts a high level language program into machine language.

4. For each term in list (A) choose the correct definition from list (B)

List A	List B
A program	1. is a FORTRAN statement that indicates the logical end of the program.
A computer	2. is a machine that can solve all problems.
END	3. translates programs written in an assembly language to a machine language.
STOP	4. is a machine that uses instructions given by the user to solve a problem.
	5. is a sequence of instructions which, when performed, will do a certain task.
	6. is a FORTRAN statement that indicates the physical end of a program.

1.7 Solutions to Exercises

Ans 1.

1. T	2. F	3. F
4. T	5. T	6. T
7. F	8. T	9. F
10. T	11. F	12. F

Ans 2.

III and V

Ans 3.

Assembler	7
Compiler	8
Software	5

Hardware	2

Ans 4.

A program	5
A computer	4
END	6
STOP	1