Competitive Exams: Structured Programming Languages

When high-level languages were the first developed, programs written in them often consisted of long lists of instructions with little internal organization.

Early programs were often like-this, making it difficult for anyone but the original programmer to understand what was happening within the program. If changes needed to be made at a later date, it was difficult to locate the section of the program that was to be changed

To correct these problems, computer scientists came up with the idea of developing programming languages that encouraged the writing, of programs structured in such a way that the flow of the program logic would-be easy to follow.

In 1964, two mathematicians, Corrado Bohm and Guiseppe Jacopini, wrote a paper which proved that any problem logic can be stated by using an appropriate combination of the following structures:

1. The sequence
2. The decision structure
3. The loop

A sequence is merely a series of statements that the computer executes in the order in which they occur. A program that adds three numbers together and determines the average is a Sequence.

When a program contains a decision structure or a loop, statements are no longer executed in the order in which they occur in the program. For example, certain statements may be skipped over or a group of statements may be repeated a number of times. A decision structure involves making a comparison. What is done next depends on the result of this comparison. For example, two numbers named X and Y might be compared. If X is the larger number, its value might be printed; otherwise the value of Y might be printed.

A loop is usually the easiest way to perform a repetitive task such as printing a list of several hundred names. A loop allows a series of instructions to be executed as many limes as needed.

Once they had realized that any programming problem could be solved by using the appropriate combination of-these three basic structures, computer scientists began developing languages that were capable of performing these three tasks in an efficient, easy-to-understand manner. This is an important feature of structured programming languages.

Pascal Language

Pascal is a fairly new programming language. It was designed by Nicklaus Wirth, a Swiss computer scientist, between 1968 and 1970. He named the language after B. Pascal, the seventeenth-century mathematician and philosopher who developed the first mechanical calculator.

The Pascal language was specifically developed to teach good structured programming techniques to students. It is relatively easy to learn since it uses English-like words in its statements. Consequently, Pascal is widely taught in beginning computer science courses.

Pascal contains statements that allow the programmer to efficiently alter the flow of program execution by using decision structures and loops. It is a block-structured language. This means that programs are composed of subprograms or blocks, each designed to perform a specific task. In Pascal these subprograms are called procedures. A simple program may consist of may blocks, some of which are placed within other blocks. This figure symbolically shows how the blocks of a program might be nested in (placed inside of) one another. Because this program is being used merely for demonstration purposes, the blocks do not contain any actual program statements. The outermost block (the main program) contains the entire program. Block A is nested in the main program; Block C is nested in Block B, which is nested in the main program. The number of blocks and the manner in-which they are nested, is dependent upon the needs of the particular program. Large programs usually consist of many blocks. Regardless of how many blocks there are in a program, they always work together to. Form a unified whole. Because of the block structure of Pascal, it is easy to break a program into subprograms that can be written individually. Then these subprograms can be joined together to form the entire program

Another important feature of Pascal is that it is a general-purpose programming language. It can be used to write a wide variety of programs, from those that are highly scientific to those that are strongly business-oriented. You will find that within a very short time you will be writing interesting Pascal programs that perform many different tasks.

Translating and Executing Programs

Before a program written in a high-level language can be executed, it must be translated into machine language. The program to be translated is referred to as the source program. There are two basic types of system programs that perform this translation: Interpreters and compilers. The difference between an interpreter and a compiler is that the compiler creates an object program, which is the entire source program translated into machine language. This object program is then loaded into the computer's memory for execution. An interpreter, on the other hand. translates each program statement into machine language and then causes it to be executed before going on to the next statement. This approach saves space in the computer's memory. However, interpreters often use more computer time than compilers-because program statements that are used more than once must be translated each time they are executed.