PROGRAMMING WITH PASCAL

STRUCTURED PROGRAMMING

Introduction:
The first programs that were written consisted of individual instructions that were executed one after the other. This resulted in one long program consisting of several thousands of lines of code that were not grouped in any logical way.

√ The programs were long & complicated, making them too difficult to understand & impossible to maintain.
√ Maintaining such programs was time-consuming & expensive, since changing even one line of code could affect the entire program.
√ Any conditional branching could force the program control to jump back & forth (sometimes from end to the beginning of the program), creating tangled loops usually referred to as ‘Spaghetti’ loops.

Structured programming is a systematic technique of program design that uses the 3 basic control structures (sequence, selection/branching & looping), and the top-down concepts to decompose main functions into lower-level components for modular coding purposes.

Importance of structuring programs:
1). Structured programming improves the programming process through better organization of programs.
2). It produces programs of high quality, which are easy to test, debug and modify.
3). Structuring of programs made them easy to understand & also to maintain.
4). Structuring a program helps to break it down into smaller understandable portions usually referred to as modules. This means that, the programmer has to deal with much less amount of code for any one module than dealing with the system as a whole.

Characteristics of structured programming:
Structured programming emphasizes on:
1). Modular development of a program, i.e., a large program is divided into smaller sub-programs. In this case, a large program will consist of several modules, each performing a single, specific task.
2). The use of a few simple control structures in specifying algorithms. E.g., WHILE-DO & IF-THEN-ELSE. Using these constructs, statements are executed one after the other.

USER-FRIENDLY PROGRAM CODING

Program designing is usually done independent of any programming language.
In converting the design to a program code, the following general points that are important, and are applicable to any programming language that you might use.

Coding Standards.

There are 2 main functions performed by programming:
1. Creation of programs: – This involves the initial writing of the code, and subsequent checking & amendment so that it performs the required functions.
2. Maintenance of existing programs: – involves changing & adding of features as required.
If a programmer is not controlled, he tends to produce codes which reflect his interest in the job, expertise, and personality. An inexperienced programmer will usually use the simplest techniques to produce a solution, which may not always be the most efficient. As experience grows, the programmer develops a style, and uses some kind of shorthand that speeds up the production of code (which is always behind schedule).

A program should be written in such a way that, maintenance can be done by someone who has either never seen the code before or has little recall of it.

By laying down rules to be followed;

(i). Standards provide a way of ensuring that quality is maintained throughout all areas of programming from design methods to coding style.

(ii). Standards help inexperienced programmers to become effective more quickly.

(iii). Standards help the experienced programmers to become more productive, by making programs easier to comprehend, amend & check.

(iv). Standards aid communication between people working on the same project, and makes it possible to interchange people of different skills, abilities, and experience within & between different projects.

A Readable Program.

The following are some rules that should be followed to make a program readable:

(1). **Data Names.**

   The names should be meaningful in order to make the program code easier to read.

   *For example;*

   Add **A** to **B** giving **C**

   A, B, and C could be anything, but if they were replaced by names like; **Cost_Price**, **VAT**, & **Selling_Price**, things become clearer.

   This would read as; Add **VAT** to **Cost_Price** giving **Selling_Price**

   **Rules for Data Names**

   (i). Use names that describe the purpose of the code or variable.

   (ii). Don’t make up aliases (assumed names).

   (iii). Don’t use names that are similar. Change at least two letters.

   (iv). Keep spellings as normal as possible, e.g., STUDENT-NAME, rather than STUDE-NAM.

   If the language rules don’t allow long names, then use abbreviations consistently, e.g., STD-NM.

   (v). Use the same prefix for all the variables in one record or the local variables in a subroutine (or module).

(2). **Comments.**

   All programming languages allow the use of comments in a program. Comments are used in an algorithm (program) to:

   ✓ Remind yourself (and others) of what processing is taking place.

   ✓ Help to make the program code more understandable.

   ✓ Make the program statements clear. They are used to explain or clarify any aspect of an algorithm which may be difficult to understand by just reading the bare statements.

   ✓ Make the maintenance of the programs easy.

   Commenting of a program falls into 2 categories;

   (a). Comments can be used within the program heading to describe the function/purpose of the program. These comments also specify the Author, Date, When & by whom the last
alteration was carried out.

(b). Comments that are used in the Declaration & instructional sequences of the program. Each procedure of a program should contain a comment, which describes the purpose and input & output parameters, if any.

Comments should be added at the module level to explain the purpose of the module, and within the code to explain complicated algorithms or highlight error-prone sections of the program. However, it is not necessary to add comments to every line of code.

Notes.

♦ The programmer should include comments in the program code, so that anyone reading the program can understand its purpose. This will also help the programmer to understand it after a period of time has elapsed.
♦ Good comments are those that do not repeat what is already written in the code; they give additional information, clarify the meaning of objects & report on the status of the algorithm at appropriate points.
♦ A comment (or lack of it) has no effect on the performance of the algorithm itself. Its only purpose is to help the user understand the algorithm.

(3). Indentation.

Code should be laid out neatly, with proper use of indentation to reflect the logic structure of the code.

It is advisable to place only one instruction on each line, make use of blank lines & new pages to keep code readable, and also limit components to a manageable size.

(4). Avoid ‘Tricks’.

Always make sure that the program is written using straightforward code, which people can readily understand.

(5). Modularization.

The aim of splitting up a program is to:

(a). Make it easier to understand.
(b). Reduce the effort required to change the program, since the changes will only be made to the module whose functionality changes, or if a new feature is to be added then a new module can be added.
(c). Makes testing & debugging easier with small independent modules.

Rules for Modules.

(i). Each module should contain only one program function.
(ii). The maximum size for each module should be about 100 instructions.
(iii). There should only be one entry point & one exit point in each module.

Qualities of a good Program.

Programming is a tiring, expensive, & time-consuming task, and care must be taken in writing good programs.

Before coding a program, the programmer should plan the solution to the problem in detail, and write down the processing logic.

A ‘good’ program has the following characteristics.

(i). Accuracy.

The program must do what it is supposed to do, and meet the criteria laid down in its specification.
(ii). **Reliability.**
   The program must ALWAYS do what it is supposed to do, and never crash.

(iii). **Efficiency.**
   The program must use the available storage space & resources in such a way that the speed of the system is not wasted.

(iv). **Robustness.**
   The program should cope with invalid data without creating errors or stopping without any indication of the cause.

(v). **Usability.**
   The program must be easy to use, and must be well documented.

(vi). **Maintainability.**
   The program must be easy to amend, having good structuring & documentation.

(vii). **Readability.**
   The code in the program must be well laid out and explained with comments.

**Review Questions.**

1. What are the benefits of setting out rules to be followed while writing programs?
2. State Four practices which lead to user-friendly programming when coding a computer program.
3. List five rules that should be followed to make a program readable.
4. What are the reasons for placing comments within a Pascal program?
5. Explain five qualities of a good program.

**ASPECTS OF PROGRAMMING.**

1. **Operation on data.**
   This lets the user define the various data types, the way in which we represent them, and the way we represent operation on them.

2. **Program structure.**
   This deals with the various parts of a program and the way they are organized.

3. **Data structure.**
   This defines the general outlook of the facts (data) to be used in the program.

4. **Program design.**
   This examines the aim of the program and how the stages of programming must be followed systematically in order to achieve the pre-determined objective.

5. **Programming languages.**
   (a). **Natural languages:** They have complexities that make them unsuitable for programming (i.e. they have more than one idea).
   (b). **Formal languages:** They have clear meaning (i.e. each instruction written in formal language has one meaning).

**BIBLIOGRAPHY.**

1. Computer science – Fifth Edition
Characteristics of a good Program.

1. **Integrity (Accuracy of calculation).**
   Integrity of calculations is an absolute necessity in any computer program. This is because; if the calculations are not carried out correctly, then all other program enhancements will be meaningless.

2. **Clarity.**
   Clarity refers to the readability of a program with particular emphasis on underline logic. If a program is clearly written, it would be possible for another programmer to follow the program logic without too much effort.

3. **Simplicity.**
   Keeping things as simple as possible will greatly enhance clarity & accuracy of programs, and the consistence of the overall program objectives. Therefore, it is advisable that you should maintain a relatively simple program structure.

4. **Efficiency.**
   Program efficiency refers to the execution speed & the overall utilization of a computer memory.

5. **Modularity.**
   Most large programs can be broken down in a series of identifiable sub-tasks (modules). Use of modular programming structures enhances (improves) accuracy, simplicity, and clarity of a program. It also facilitates future program alteration.

6. **Generality.**
   A program should remain as general as possible within reasonable limits.