THE ART OF PROGRAMMIMG METHODOLOGY

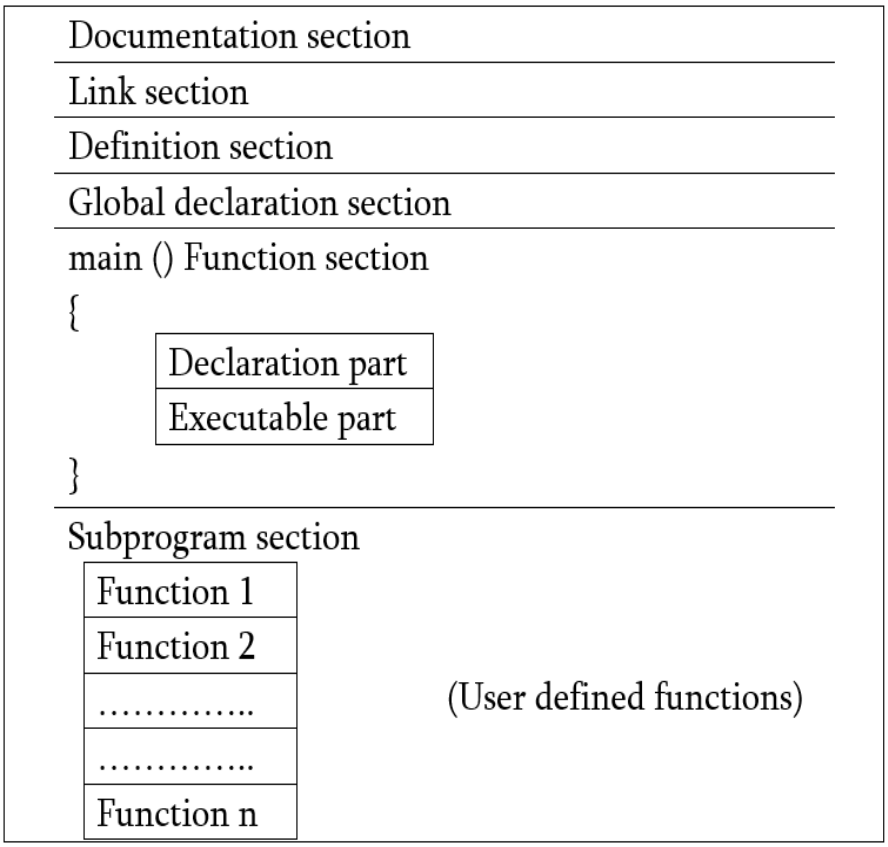
PROGRAMMING CONCEPTS

Dear friends,

In Today's discussion we will cover the topics such as characteristics of a good program, modular programming concepts, different types of errors in programming and finally the concept of structured programming.

Programming

A program is a set of step-by-step instructions that directs the computer to do the tasks that you wanted and produce the results. There are several programming languages available for doing specific tasks or making specific type of applications. Web application languages, desktop application languages, scripting languages are some of the examples for this. Among all the of the programming languages C programming is the very basic programming language useful for a mass. The basic Structure of a C program contains following sections,



Documentation Section consists of a set of comment lines giving the name of the program and other details.

Link Section provides instructions to the compiler to link functions from the system library.

Definition Section defines all symbolic constants.

Global Declaration Section: There are some variables and those variables are declared in this section that is outside of all functions.

main( ) function: Every C program must have one main function section. This section contains two parts, declaration and executable part.

Declaration Part declares all the variables used in the executable part. There should be at least one statement in the **executable part** which contains instructions to perform certain task.

The declaration and executable part must appear between the opening and closing braces. All statements in the declaration part should end with the semicolon.

The **Subprogram Section** contains all the user defined functions that are called in the main function.

Characteristics of a good program:

A program written in good style is easy understandable and readable. The characteristics of good program help a program to look better and more understandable. A good computer program should have following characteristics:

- **Portability**: Portability refers to the ability of an application to run on different platforms (operating systems) with or without minimal changes. Due to rapid development in the hardware and the software, nowadays platform change is a common phenomenon. Hence, if a program is developed for a particular platform, then the life span of the program is severely affected.
- **Readability**: The program should be written in such a way that it makes other programmers or users to follow the logic of the program without much effort. If a program is written structurally, it helps the programmers to understand their own program in a better way. Even if some computational efficiency needs to be sacrificed for better readability, it is advisable to use a more user-friendly approach, unless the processing of an application is of utmost importance.
- **Efficiency**: Every program requires certain processing time and memory to process the instructions and data. As the processing power and memory

are the most precious resources of a computer, a program should be laid out in such a manner that it utilizes the least amount of memory and processing time.

- **Structural**: To develop a program, the task must be broken down into a number of subtasks. These subtasks are developed independently, and each subtask is able to perform the assigned job without the help of any other subtask. If a program is developed structurally, it becomes more readable, and the testing and documentation process also gets easier.
- **Flexibility**: A program should be flexible enough to handle most of the changes without having to rewrite the entire program. Most of the programs are developed for a certain period and they require modifications from time to time. For example, in case of payroll management, as the time progresses, some employees may leave the company while some others may join. Hence, the payroll application should be flexible enough to incorporate all the changes without having to reconstruct the entire application.
- **Generality**: Apart from flexibility, the program should also be general. Generality means that if a program is developed for a particular task, then it should also be used for all similar tasks of the same domain. For example, if a program is developed for a particular organization, then it should suit all the other similar organizations.
- **Documentation**: Documentation is one of the most important component of an application development. Even if a program is developed following the best programming practices, it will be rendered useless if the end user is not able to fully utilize the functionality of the application. A well-documented application is also useful for other programmers because even in the absence of the author, they can understand it.

Modular Programming Approach

Modular programming is the process of subdividing a computer program into separate sub-programs.

A module is a separate software component or a set of associated code statements. It can often be used in a variety of applications and functions with other components of the system. Similar functions are grouped in the same unit of programming code and separate functions are developed as separate units of code so that the code can be reused by other applications.

Modules in modular programming enforce logical boundaries between components and improve maintainability. They are incorporated through interfaces. They are designed in such a way as to minimize dependencies between different modules. Teams can develop modules separately and do not require knowledge of all modules in the system.

Each and every modular application has a version number associated with it. This provides developers flexibility in module maintenance. If any changes have

to be applied to a module, only the affected subroutines have to be changed. This makes the program easier to read and understand.

Modular programming has a main module and many auxiliary modules. The main module is compiled as an executable (EXE), which calls the auxiliary module functions. Auxiliary modules exist as separate executable files, which load when the main EXE runs. Each module has a unique name assigned in the PROGRAM statement. Function names across modules should be unique for easy access if functions used by the main module must be exported.

Languages that support the module concept are IBM Assembler, COBOL, FORTRAN, C, Python, etc. Among others the benefits of using modular programming include

- Less code has to be written.
- A single procedure can be developed for reuse, eliminating the need to retype the code many times.
- Programs can be designed more easily because a small team deals with only a small part of the entire code.
- Modular programming allows many programmers to collaborate on the same application.
- The code is stored across multiple files.
- Code is short, simple and easy to understand.
- Errors can easily be identified, as they are localized to a subroutine or function.
- The same code can be used in many applications.
- The scoping of variables can easily be controlled.

Documenting the Program

Documenting is an ongoing, necessary process, although, as many programmers are, you may be eager to pursue more exciting computer-centered activities. Documentation is the process of writing detailed description of the programming cycle and specific facts about the program.

Typical program documentation materials include the origin and nature of the problem, a brief narrative description of the program, logic tools such as flowcharts and pseudo code, data-record descriptions, program listings, and testing results. Comments in the program itself are also considered an essential part of documentation. Many programmers document as they code. In a broader sense, program documentation can be part of the documentation for an entire system.

The wise programmer continues to document the program throughout its

design, development, and testing. Documentation is needed to supplement human memory and to help organize program planning. Also, documentation is critical to communicate with others who have an interest in the program, especially other programmers who may be part of a programming team. And, since turnover is high in the computer industry, written documentation is needed so that those who come after you can make any necessary modifications in the program or track down any errors that you missed.

Compiler and interpreter

We generally write a computer program using a high-level language. A high-level language is one which is understandable by us humans. It contains words and phrases from the English (or other) language. But a computer does not understand high-level language. It only understands program written in 0's and 1's in binary, called the machine code or machine language.

A program written in high-level language is called a source code. Then we need to convert the source code into machine code for the understanding of machines, and this is accomplished by compilers and interpreters. Hence, a compiler or an interpreter is a program that converts program written in high-level language into machine code understood by the computer. But both the compiler and interpreter will works or approach the machine language conversion in different ways. The major difference between interpreter and compiler is listed below.

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

Debugging

It is a term used extensively in programming. Debugging means detecting, locating, and correcting errors or bugs (mistakes), usually by running the program.

These bugs are logic errors, such as telling a computer to repeat an operation but not telling it how to stop repeating. In this phase you run the program using test data that you devise. You must plan the test data carefully to make sure you test every part of the program.

After debugging the debugger will return the error identified while debugging the program. For the final stage, the execution of the program we needed to clear the errors. In general different type of errors are there, related with a program. They are classifies as syntax error, runtime error, and logical error.

Syntax errors, Runtime errors and Logical errors.

Syntax errors

Syntax or format errors are a major problem for most beginning programmers using languages such as C++, Java, and C#. A syntax error occurs when the programmer fails to obey one of the grammar rules of the language. Typically this involves things like using the wrong case, putting punctuation where it is not supposed to be, failing to put punctuation where it is supposed to be, etc.

For example, consider the statement,

int a,b:

In C programming. Then the above statement will produce syntax error as the statement is terminated with **:** rather than **;.** Because in C programming all the statements are end with a ; rather than a :. Now the syntax of this particular programming language is violated by the programmer, this is called the syntax error.

Runtime errors

A runtime error occurs whenever the program instructs the computer to do something that it is either incapable or unwilling to do. Since there are a near infinite number of things that fall in this category, there are a near infinite number of ways to write programs that cause runtime errors. Runtime errors commonly occur when statements are written in the wrong order, or perhaps the order is modified by dragging statements up and down the screen after they are written. Another common cause of runtime errors results from the construction of instructions that the computer is unable to carry out.

One of the very good example for this runtime error is the classical divide by zero error. This is the classical way to demonstrate a runtime error, instruct the computer to divide any number by the value zero. Any value divided by 0 produces an infinitely large result, which is too large to be accommodated by the computer. Therefore, in most cases, the computer will tell you that it is unable to perform that operation.

For example in C programming language.

Int a,b=10,c=0;

a=b/c;

This lines of statements will return runtime error based on zero division.

Logic errors

Logic errors are usually the most difficult kind of errors to find and fix, because there frequently is no obvious indication of the error. Usually the program runs successfully. It simply doesn't behave as it should. In other words, it simply doesn't produce the correct answers.

The causes of logic errors

Logic errors usually result from one or some combination of the following three causes:

- You didn't understand how the program is supposed to behave.
- You didn't understand the behavior of each operation that you wrote into the program.
- You were careless.

How to prevent logic errors

In an attempt to prevent logic errors, you should:

- Make absolutely certain that you do understand how the program is supposed to behave.
- Make absolutely certain that you do understand the behavior of every operation that you write into your program.
- Not be careless.
- Implement a practice of testing.
- Apply modular programming concept will give the simplified logic version of your program.

Structured Programming

C is called a structured programming language because to solve a large problem, C programming language divides the problem into smaller modules called functions or procedures each of which handles a particular responsibility. The program which solves the entire problem is a collection of such functions. One major drawback of C language is that similar functions cannot be grouped inside a module or class. Also functions cannot be associated to a type or structure. Thus data and functions cannot be bound together. C++ language overcomes these problems by introducing object oriented functionality in its programming capabilities.

Advantages

- C structured programming is simple and easy to understand and implement.
- It is well suited for small size implementation. However this is not restricted. A good design can extend it to large size implementation.
- Programmers do not require to know complex design concepts to start a new program.

Disadvantages

- Data and methods and not be bind together in a module.
- Polymorphism and inheritance are not available.
- Complex design and full object oriented design cannot be implemented.
- Programmers generally prefer object oriented programming language over structured programming language when implementing a complex gaming applications or front end business applications.

Summary

In this module we discussed about the basic characteristics needed for a program and the modular programming terminology. Along with these concepts we also covered the converting mechanism of high level language into machine level language in two different ways such as the complier and interpreter. Different types of errors like syntax error, run time error and logical error are the concepts we studied regarding with the debugging of a program. At last we discussed about the concept of structured programming in the context of C programming language.

Assignments

Explain the structures of C.
List out the characteristics of a good program.
Write the difference between an interpreter and a compiler
Explain in detail the different types of errors.
What is Structured Programming

Reference

- *B. W. Kernighan and D. M. Ritchie*, The C Programming Language, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.
- *Greg Perry,* Absolute Beginners' guide to C 2nd Edition, SAMS publishing, A division of Prentice Hall Computer Publishing, 201 West 103rd Street , Indianapolis, Indiana 46290. April 1994.
- *Yashavant Kanetkar*; Let us C, BPB *Publications, New Delhi.*