**REPORT ON THE ADOPTION OF C PROGRAMMING LANGUAGE FOR SOFTWARE DEVELOPMENT AT FINYISTICS SOLUTIONS**

Procedural programming refers to a set of instructions telling a computer what to do step by step and how to perform from the first code to the second. Procedural languages include C, Pascal, Go, FORTRAN, Ada and BASIC. The basic idea in using procedural programming is to have a program specify the sequence of steps that implements a particular algorithm. An algorithm can be defined as a set of steps for performing a task that can be written down and implemented. An algorithm details how to start with known information specified in the problem and how to use that information to reach a solution.

Procedural programming enables the keeping of code as concise as possible by focusing on a very specific end result to be achieved. In Finyistics Solutions projects, the focus of procedural programming is to break down a programming task into a collection of variables, subroutines and data structures by the use of procedures. A procedure refers to a set of commands which can be executed in order. Procedure programming focuses on processes and functions are stored in separate memory location, unlike in Object Oriented Programming, where data and functions are stored in same memory location.

Procedural programming is quite straightforward and efficient as more than often to begin with, the program is written in a common and plain language by applying logic before actually writing the code. In procedure programming, (C for example), procedures are a sequence of imperative statements, such as assignments, tests, loops and invocations of sub procedures. These procedures are functions, which map arguments to return statements.

The design method to be used in procedural programming is called Top Down Design and it involves starting with a problem (procedure) and then systematically breaking down the problem into sub problems (sub procedures). This is called functional decomposition, which continues until a sub problem is straightforward enough to be solved by the corresponding sub procedure. When changes are made to the main procedure (top), those changes can cascade to the sub procedures of main, and the sub-sub procedures and so on, where the change may impact all procedures in the pyramid. The main features of procedural programming is that it is relatively easy to read and write program code and maintain program code as numerous procedures can be debugged separately. Furthermore, large programs are divided into smaller ones and that most of the data is shared and can therefore be reached from any other point within the program.

**Advantages of procedural programming**

1. Many books and references available on well-tried and tested coding algorithms - no need to re-invent the wheel.
2. It is written in a step-by-step function, smaller programs written this way are very easy to follow.
3. Good level of control without having to know precise target CPU details - unlike low level languages
4. Easy to maintain, as each procedure/function can be debugged in isolation from the rest, allowing for easy isolation of problems, in contrast to Object Oriented Programming which can often take very long to find the problem code.
5. Portable source code - use a different compiler to target a different CPU
6. Since it is written for a very specific purpose the code often gets you extremely efficient and high-performance applications.

**Key features of procedure programming**

1. **Pre-defined function**

This is a sequence of instructions that is identified by name in a computer program but is built into the high-level programming language from which the program is compiled or is retrieved from somewhere outside the program.

1. **Local variables**

These variables only exist inside the specific function that creates them. They are unknown to other functions and to the main program. A local variable can also occur in a for statement, a switch statement, a for each statement, a using statement or a specific catch statement or using statement.

1. **Global variables**

A global variable is a variable that can be viewed throughout the entire program by every other procedure taking place, it is also accessible by every other task running in the program. The majority of times, a global variable is a static variable, whose extent is the entire runtime of the program.

1. **Parameter passing**

Parameter passing allows variable values to be passed through to the program which will handle it with a procedure.

1. **Modularity**

Modularity is about how systems have different tasks and are grouped together to complete the overall task. Each group has a task of its own to carry out. Modularity are modules that have been split into different groups to perform certain tasks one after the other so it is a continuous system.

1. **Procedures**

A procedure is an independent code module that fulfils some concrete task and is referenced within a larger body of source code.

1. **Programming libraries**

Program Libraries hold a collection of pre built source codes, Subroutines, Classes and values that can be used at any time by the program and users. A library is a collection of precompiled routines that a program can use. The routines, sometimes called modules, are stored in object format. Libraries are particularly useful for storing frequently used routines because you do not need to explicitly link them to every program that uses them.

1. **Procedural programming paradigm**

Procedural programming is a programme based upon the main point of the procedure call. This is also known as routines and subroutines. The procedural programming paradigm is thought mostly with defining a linear procedure or sequence of programming statements. A key feature of the paradigm is the pattern of functionality into small discrete re-usable modules.