A STUDY OF IMPERATIVE LANGUAGE FOR PROBLEM SOLVING IN SOFTWARE DEVELOPMENT

Mohit Kumar Sharma
Head and Assistant Professor
Post Graduate Department of Computer Science
J.C. D.A.V. College, Dasuya, Punjab, India

Abstract: Software Development is an important phase of software development life cycle and different high level languages are used for solution to user problems. C is problem solving and an imperative language since 1970s. C was initially used for software development, in particular the programs that make-up the operating system. C was adopted as a software development imperative language, because it creates coding that runs nearly as fast as code written in assembly language. Problem solving states different user solutions solved by imperative language. C programs are created to be compiled using a compiler, to provide access to memory, to provide language constructs that map efficiently to machine instructions.

Index Terms – Imperative Language, Benefits, Components, Compiler

I. INTRODUCTION

Software Development is important phase of software development life cycle and different high level languages are used for programming. C is problem solving and an imperative language. C language was developed at AT&T Bell Labs in 1972 by Dennis Ritchie. It was based on an earlier Bell Labs language "B" which itself was based on the BCPL language. Since early on, C has been used with the Unix operating system, but it is not bound to any particular operating system or hardware. It is an extension of two languages called BCPL and B. Its use was limited at Bells lab till 70's. By mid-80 the popularity of C became widespread.

Various applications and software were developed in C due to its advantages like efficiency and portability. It was first implemented on the Digital Equipment Corporation PDP-11 computer in 1972. The UNIX operating system and virtually all UNIX applications are written in the C language. C was initially used for software development work, in particular the programs that make-up the operating system. C was adopted as a system development language, because it produces code that runs nearly as fast as code written in assembly language. Some examples of C creations are as:

- Operating Systems
- Language Compilers
- Assemblers
- Text Editors
- Print Spoolers
- Network Drivers
- Modern Programs
- Data Bases
- Language Interpreters
- Utilities

II. IMPORTANCE OF IMPERATIVE LANGUAGE

- C has a good collection of inbuilt functions, keywords and data types. It also resembles general English language. It is easy to learn because it follows imperative approach.
- C is a case-sensitive language.
- It is a middle level language; it bridges the elements of high-level languages with the functionality of assembly language.
- Programs made in C can be made to run on different machines thus increasing the efficiency of the language. Therefore, C is a portable language.
- C compiler is very quick and therefore computations can be done faster than other languages like BASIC.
- C compilers are easily available and they require very less disk space for their storage.

- Programs written in C can be divided into small modules with the help of functions which helps in increasing the understanding of the program.
- Syntax errors can be easily detected in C compilers. The errors are displayed with the line number of the code and the error message. Thus, errors are easy to remove.
- Memory can be allocated and deallocated according to the requirements of the program. Various library functions are available in C library that helps in memory management.

III. COMPONENTS OF C PROGRAM

C program is basically a collection of functions. A function is a self-contained block of statements to perform a specific task independently. C program consists of the following sections as:

DOCUMENTATION SECTION
PRE-PROCESSOR DIRECTIVES SECTION
GLOBAL DECLARATION SECTION
main( ) FUNCTION SECTION **{** **DECLARATION SECTION** **EXECUTABLE SECTION** **}**
USER-DEFINED FUNCTION SECTION

- **Documentation Section**
  Documentation section is optional in the program. This section consists of a set of comment lines giving the name of the program and other details which the programmer would like to add in the program. Comments are not part of executable program. The user can place the comments anywhere in the program and any number of comments can be used. Comments can be single-line (//) or multi-line comments (/* ---- */).
- **Pre-processor Directives Section**
  This section provides instructions to the compiler to link functions from the system library. C program depends upon some header files for function definition that are used in program. Each header file by default has an extension .h. The file should be included using #include directive. For example:

  #include<stdio.h> or #include “stdio.h”

  The user can also define the symbolic constants using #define directive. For example,

  #define PI 3.142

This macro expansion directs the compiler to replace every occurrence of PI with value 3.142 during pre-processing.

- **Global Declaration Section**
  This section declares some variables that are used in more than one function. Variables declared in the global declaration section that is outside of all the functions are called global variables.
- **main( ) function Section**
  Every program written in C program must contain main( ) function. This section consists of two parts:
  - **Declarative part:** Declares the variables that are to be used in the executable part
  - **Executable Part:** All the statements, excluding declarative part, are included in this part.

  The execution of the program always begins with the main( ) function.
- **User-defined Function Section**
  This section contains all functions defined by the user. These user-defined functions are generally placed immediately after main( ) function. They can also be defined before main( ) function.

Various components of the ‘C’ Program are explained in the following program:

// Program for addition of two numbers /*DOCUMENTATION SECTION */

```
#include<stdio.h>                          /* LINK SECTION */
void main( )                               /* MAIN FUNCTION SECTION */
{
        int a, b, c;                       /* VARIABLE DECLARATIONS */
        printf ("Enter a, b =");           /*OUTPUT STATEMENT */
        scanf ("%d%d", &a, &b);            /*INPUT STATEMENT */
        c =  a + b;                        /*ASSIGNMENT STATEMENT */
        printf ("c=%d", c);                /* OUTPUT STATEMENT */
}
```

Software programmer can code any program in C language, then to execute that program requirement to compile that program using a C Compiler, which converts program into a language understandable by a computer. This is called machine language i.e. binary format. It comes along with all flavors of UNIX and Linux. C compiler is a program that reads source code, which is the C code written by a programmer and produces an object code. The source file is a plain text file containing source code. The program is still not in an executable form because it may have to call another program and system libraries. Linker links the object code with system libraries and finally produces the executable form of C program. The executable file consists of machine code, 1's and 0's that are not meant to be understood or read by the people, but only the computers.

REFERENCES

[1] Yashavant P. Kanetkar : Let us C, BPB Publications, New Delhi.
[2] Salaria, R.S. : Test Your Skills in C, Salaria Publications, New Delhi.
[3] C. Balaguruswami : Programming with C Language, Tata McGraw Hill, New Delhi.
[4] Byron S. Gottfried : Programming in C, McGraw Hills Publishers, New York.
[5] M.T. Somashekara : Programming in C, Prentice Hall of India.
[6] Banahan, M.; Brady, D. Doran, M. (1991). The C Book (2nd ed.). Addison-Wesley
[7] King, K. N. (April 2008). C Programming: A Modern Approach (2nd ed.). Norton.