CHAPTER 1 *(DAY 1 Fore noon)*

INTRODUCTION TO C PROGRAMMING

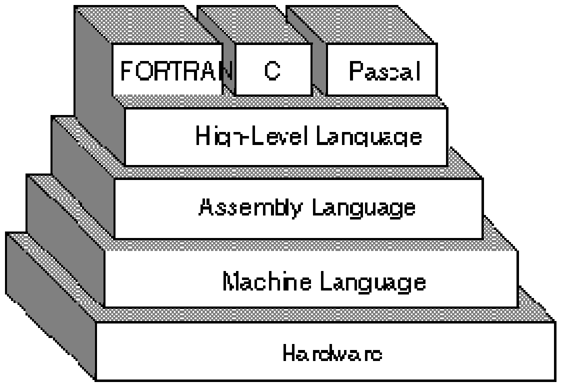
C LANGUAGE	**(Reading)**
STRUCTURED PROGRAMMING	**(Reading)**
DATA TYPE	
DATA STRUCTURES AND C	
C ENVIRONMENT	**(Reading)**
ALGORITHM	**(Reading)**
PSEUDOCODE	**(Reading)**
C PROGRAM STRUCTURE	**(Reading)**
FIRST C PROGRAM	
INDENTATION	**(Reading)**
USING COMMENTS	**(Reading)**

CHAPTER – 1

INTRODUCTION TO C PROGRAMMING (Reading)

C LANGUAGE

Many of the System Programs have been developed in C language because of the efficiency of the language and its completeness. C is one of the most popular high level languages. Dennis Ritchie and Brian Kernighan at Bell Laboratories have published the paper on C Programming Language and later developed the language. Originally it was implemented on DEC PDP 11, and now it is ported on many platforms as Systems Programs and Application Programs.



Low level languages are closer to the machine. Machine Language understands the machine for which it is developed. Machine language consists of string of numbers (in bits) that instruct computers to perform their most elementary operations one at a time. To speed up programming from machine language and understand the flow more for humans issuing commands, Assembly language was developed over the machine language. High level languages were developed to speed up to include many Assembly instructions as the humans will speak more like English like language. Some of the other early high level languages are FORTRAN,COBOL. FORTRAN dominated the engineering and scientific world, while COBOL influenced many businesses across the industries to modernize the business processing.

STRUCTURED PROGRAMMING

Structured Programming is a disciplined approach to writing complex programming that is clear, repeatable, and demonstratable and to code. Structured programming is a subset of procedural programming that enforces a logical structure on the program being written to make it more efficient and easier to understand and modify.
Structured programming frequently employs a top-down design model, in which developers map out the overall program structure into separate subsections. A defined function or set of similar functions is coded in a separate module or sub module, which means that code can be loaded into memory more efficiently and that modules can be reused in other programs. After a module has been tested individually, it is then integrated with other modules into the overall program structure. It is most famous for removing or reducing reliance on the GOTO statement.

Structured programming is one of the several different ways in which a programming language can be constructed. It was originally introduced as a means of getting away from the 'spaghetti' code that was used in the early days and to provide some means by which programmers could more easily follow code written by other programmers. Structured programming is a procedure oriented method of designing and coding a program. There are a limited number of constructs that can be used within the code to define the execution flow. A structured program is one that only uses the constructs that are listed within this document. Most so called structured (or procedural) programming languages also permit constructs to be used that are not listed in this document. These additional constructs are to be avoided if a true structured program is to be produced.

A structured program may be written out using pseudo code prior to being translated into whatever programming language that the program is to be written in. This pseudo code forms part of the program specification and is readable by anyone who understands structured programming regardless of whether or not they know the specific language in which the program has been written.

C ENVIRONMENT

The basic environment in the development of programs in C language:

edit:

Editing consists of writing the instructions as defined by the language. There are many editor programs PC based as well as on all Unix flavors and higher platforms like main frames. Some platforms like PC combine many activities of processing into one unit called Visual C/C++ or Software Development Kit. Unix environment has emacs and vi editors to write program instruction.

preprocessor and compile:

The preprocessor task is accomplished together with compile but done as a separate unit before compiling the code. Preprocessor takes directives to perform some actions before actual compiling starts. Some of these directives include including some files and defining some values. The compiling task performs the syntactic check on the instructions for the correctness as per the language. Any errors or omissions are identified. Output of compiled instructions is called object code.

link and load:

The Linker combines all related object codes into one unit as executable code. Linker verifies each of the required object code and puts them together for execution, also identifies any missing units of object codes. Before the program can be executed, the program must be placed in memory. The loader takes the executable program from disk and transfers it to memory ready for execution.

execute:

When the program runs, the CPU reads each instruction and executes. Many times, the program may require some input or spits some output. The input can be read from a file or from the standard devices like the key board. Similarly, the output can be written to a file or to some standard device like the screen on the monitor.

ALGORITHMS

Algorithm is a procedure to solve a problem in terms of

1. The actions to be executed, and
2. The order in which these actions are to be executed.

Algorithms are written in simple English commands without descriptive sentences. These are mainly useful to the programmer to develop an executable code and putting thoughts together for further development and future comparisons of alternate **algorithms.**

PSEUDOCODE

Pseudo code is an artificial and informal language that helps programmers develop algorithms. The pseudo code is particularly useful for developing algorithms that will be converted to structured C programs. Pseudo code is similar to everyday English; it is convenient and user-friendly although it is not an actual computer programming language.

Pseudo code programs are not actually executed on computers. Rather, they merely help the programmer "think out" a program before attempting to write it in a programming language. Pseudo code consists purely of characters, so programmers may conveniently type pseudo code programs into a computer using an editor program. The computer can display or print a fresh copy of a pseudo cede program on demand. A carefully prepared pseudo code program may be converted easily to a corresponding C program.

Pseudo code consists only of action statements – those that are executed when program has been converted from pseudocode to C and is run in C. Declarations are not executable statements. They are messages to the compiler.

C PROGRAM STRUCTURE

A C program basically has the following form:

- Preprocessor Commands
- Type definitions
- Function prototypes - declare function types and variables passed.
- Variables
- Functions

We must have a main() function

INDENTATION

You will see that the ***printf*** and ***return*** commands have been indented or moved away from the left side. This is used to make the code more readable. It seems like a unnecessary thing to do because it just wastes time but when writing longer, more complex programs, the programmer will understand why indentation is needed.

USING COMMENTS

Comments are a way of explaining what a program does. They are put after // ***or*** between /* ***Comments****/. Comments are ignored by the compiler and are used by you and other people to understand your code. You should always put a comment at the top of a program that tells you what the program does because one day if you come back and look at a program you might not be able to understand what it does but the comment will tell you. You can also use comments in between your code to explain a piece of code that is very complex. Here is an example of how to comment the Hello World program:

```
/* Author: Your name
   Date: yyyy/mm/dd
   Description:
   Writes the words "Hello World" on the screen */

#include <stdio.h>

/* main program starts here after all include files. */
int main()
{
   printf("Hello World\n"); //prints "Hello World"
   return 0;
}
```