**LESSON THREE: INTRODUCTION TO C PROGRAMMING LANGUAGE**

* 1. **Introduction**

****

C is a general-purpose programming language, unlike other languages such as PASCAL and FORTRAN developed for some specific uses. C has in fact been used to develop a variety of software such Operating systems: UNIX and Windows, application packages: WordPerfect and Dbase etc.

C is an example of structured programming language because it chooses what it uses what is known as controlled structures. The programs are structures by defining and calling functions. The program flow is controlled using loops, if statements and functional codes.

This lecture is organized as follows;

1. Introduction
2. Lecture objectives
3. C programming concepts
4. Advantages of C programming
5. Components of C Program
6. Self test questions
7. Summary
8. Reference
   1. **Lecture Objectives**

****At the end of this lecture, the student should be able to;

1. Explain the advantages of C programming over other languages.
2. Explain the components f a C program.
   1. **C Program Concepts**

****

**Integrated Development Environment (IDE)**

An IDE is a software application that provides comprehensive facilities for computer programmers for software development. An IDE consist of the following

1. Source code editor
2. Compiler and or interpreter
3. Build automation tools
4. A debugger

NB: All commands in C must be in lower cases.

### **Source Code files**

When you write a program in C language, your instructions form the source code (or simply source file). C filenames have an extension .c. The part of the name before the period is called the base name and the part after the period is called the extension.

**Object code, Executable code and Libraries**

An executable file is a file-containing ready to run machine code. C accomplishes this in two steps.

* Compiling – The compiler converts the source code to produce the intermediate object code.
* The linker combines the intermediate code with other code to produce the executable file. C does this in a modular manner.

You can compile individual modules, and then combine the compiled modules later. Therefore, if you need to alter one module, you don’t have to recompile the others.

Linking is the process where the object code, the start up code\*, and the code for library routines used in the program (all in machine language) are combined into a single file - the executable file.

* 1. Advantages of C over Other Languages

****

1. **C Supports structured programming design features.**

* It allows programmers to break down their programs into functions. Further it supports the use of comments, making programs readable and easily maintainable.

1. **Efficiency**

* C is a concise language that allows you to say what you mean in a few words.
* The final code tends to be more compact and runs quickly.

1. **Portability**

* C programs written for one system can be run with little or no modification on other systems.

1. **Power and flexibility**

* C has been used to write operating systems such as Unix, Windows.
* It has (and still is) been used to solve problems in areas such as physics and engineering.

1. **Programmer orientation**

* C is oriented towards the programmer’s needs.
* It gives access to the hardware. It lets you manipulate individual bits of memory.
* It also has a rich selection of operators that allow you to expand programming capability.
  1. C Program Keywords/tokens

****

Keywords

These are reserved words that have special meaning in a language. The compiler recognizes a keyword as part of the language’s built – in syntax and therefore it cannot be used for any other purpose such as a variable or a function name. C keywords **must be used in lowercase** otherwise they will not be recognized.

**Examples of keywords**

auto break case else int void

default do double if sizeof long

float for goto signed unsigned

register return short union continue

struct switch typedef const extern

volatile while char enum static

* 1. **Components of C Program**

****

A typical C program is made of the following components:

* Preprocessor directives
* Functions
* Declaration statements
* Comments
* Expressions
* Input and output statements

The following code will be used to explain the components of a C program.

1. /\* Sample Program \*/

2 #include<stdio.h>

3 main()

4 {

5 int num; /\* define a variable called num\*/

6 num = 1; /\* assignment \*/

7 printf(“ This is a simple program ”);

8 printf(“to display a message. \n”);

9 printf (“My favorite number is %d because ”, num);

10 printf(“ it is first.\n ”);

11 return 0;

12 }

Explanation of the Code above

1. Comments

Comments are non – executable program statements meant to enhance program readability and allow easier program maintenance, i.e. they document the program. They can be used in the same line as the material they explain (see lines 1, 5, 6 in sample program).

A long comment can be put on its own line or even spread on more than one line. Comments are however optional in a program. The need to use too many comments can be avoided by good programming practices such as use of sensible variable names, indenting program statements, and good logic design. Everything between the opening /\* and closing \*/ is ignored by the compiler.

1. **Preprocessor and Header file**

A preprocessor directive performs various manipulations on your source file before it is actually compiled. Preprocessor directives are not actually part of the C language, but rather instructions from you to the compiler.

The preprocessor directive **#include** is an instruction to read in the contents of another file and include it within your program. This is generally used to read in header files for library functions.

**Header files** contain details of functions and types used within the library. They must be included before the program can make use of the library functions.

Library header file names are enclosed in **angle brackets, < >**. These tell the preprocessor to look for the header file in the standard location for library definitions. Example of a header file is <stdio.h>

Examples of preprocessor:

#include<stdio.h>

#include<math.h>

#define PI 3.142

1. **Function (main ())**

All C programs consist of one or more functions, each of which contains one or more **statements.** In C, a function is a named subroutine that can be called by other parts of the program. Functions are the building blocks of C.

A *statement* specifies an action to be performed by the program. In other words, statements are parts of your program that actually perform operations.

All C statements must end with a semicolon. C does not recognize the end of a line as a terminator. This means that there are no constraints on the position of statements within a line. Also you may place two or more statements on one line.

Although a C program may contain several functions, the only function that it must have is **main ( )**.

**Main ()**

The **main( )** function is the point at which execution of your program begins. That is, when your program begins running, it starts executing the statements inside the **main( )** function, beginning with the first statement after the opening curly brace. Execution of your program terminates when the closing brace is reached.

1. **Declaration Statement**

In C, all variables must be declared before they are used. Variable declarations ensure that appropriate memory space is reserved for the variables, depending on the data types of the variables. Line 5 is a declaration for an integer variable called num.

1. **Assignment and statement**

An assignment statement uses the assignment operator “=” to give a variable on the operator’s left side the value to the operator’s right or the result of the expression on the right. The statement num =1; (Line 6) is an assignment statement.

1. **Curly Braces {….}**

The two curly braces signify the beginning or the end the segment of a program.

1. **Input-Output**

**Printf(“………”)**

It is used for printing variables and texts. the statement inside the inverted commas is called literal string. The literal string must be closed in a parenthesis and between inverted commas.

Line 7,8, 9.., prints the literal string inside the “”.

**Scanf(…….)-** This is used for inputting data from the keyboard.

1. **Escape Sequence**

Escape sequences (also called back slash codes) are character combinations that begin with a backslash symbol (\) used to format output and represent difficult-to-type characters.

One of the most important escape sequences is **\n**, which is often referred to as the new line character. When the C compiler encounters **\n**, it translates it into a carriage return.

**Below are other escape sequences**:

### **Escape sequence Meaning**

\a alert/bell

\b backspace

\n new line

\v vertical tab

\t horizontal tab

\\ back slash

\’ Single quote (‘)

\” Double quote (“”)

\0 null

**Type Specifier**

The %d instructs the computer where and in what form to print the value. %d is a **type specifier** used to specify the output format for integer numbers.

# Code Format

%c Character

%d Signed decimal integers

%i Signed decimal integers

%e Scientific notation (lowercase ‘e’)

%E Scientific notation (Uppercase ‘E’)

%f Decimal floating point

%s String of characters

%u Unsigned decimal integers

%x Unsigned hexadecimal (lowercase letters)

%X Unsigned hexadecimal (Uppercase letters)

1. **Return 0**

Line11 indicates the value to be returned by the function **main( )** when it is executed. By default any function used in a C program returns an integer value (when it is called to execute). Therefore, line 3 could also be written **int main ( ).** If the **int** keyword is omitted, still an integer is returned.

Then, why **return (0); ?** Since all functions are subordinate to **main( )**, the function does not return any value.

* 1. **Self-test question**

****

1. What is the difference between as source file and an object code?
2. Give at least 3 advantages of C programming language over other languages.
3. What is the function of main () in a program?
   1. **Summary**

****

C Programming language is a very powerful language. Its advantages over other languages makes it ideal for major programs such as Operating systems.