Level 6 Advanced Diploma in Computer Science (907)
203 Credits

Unit: C Programming **Exam Paper No.**: 6	**Guided Learning Hours:** 220 **Number of Credits:** 22
Prerequisites: Basic programming skills or basic knowledge of computer use.	**Corequisites:** A pass or higher in Diploma in System Design or equivalence.
Aim: The unit illustrates the basic element of C Programming language. Interactive programming exercises are used in class to enable learners understand the components of the C Program and the syntax of C commands. Learners must be able to write portable C programs and understand how C programs use memory to store data; describe the syntax rules governing expressions; statements in C; how to use expressions and statements. Learners must also be able to implement facilities from the standard library.	
Required Materials: Recommended learning resources.	**Supplementary Materials:** Lecture notes and tutor extra reading recommendations.
Special Requirements: This is a hands-on unit, hence practical use of computers is essential. Requires intensive lab work outside of class time.	
Intended Learning Outcomes:	**Assessment Criteria:**
1. C programming basics, preprocessor directives, comments, *main*, *printf* and *scanf*; functions different types of variables.	1.1 Demonstrate how to write simple computer programs in C 1.2 Describe how to use simple input and output statements 1.3 Familiarise with fundamental data types 1.4 Explain computer memory concepts 1.5 Demonstrate how to use arithmetic operators 1.6 Describe the precedence of arithmetic operators 1.7 Demonstrate how to write simple decision making statements
2. The order in which computer programs are executed and selection statements.	2.1 Describe basic problem solving techniques 2.2 Develop algorithms through the process of top-down, stepwise refinement 2.3 Use the `if` selection statement and `if...else` selection statement to select actions 2.4 Use the `while` repetition statement to execute statements in a program repeatedly 2.5 Describe the counter-controlled repetition and sentinel-controlled repetition 2.6 Describe structured programming; 2.7 Describe how to use the increment, decrement and assignment operators.
3. How loops execute repeatedly until some terminating condition is satisfied and the different ways of programming repetition statements.	3.1 Demonstrate how to use the `for` and `do...while` repetition statements 3.2 Describe multiple selection using the `switch` selection statement 3.3 Identify how to use the `break` and `continue` program control statements 3.4 Describe how to use the logical operators.

4. Functions; how to invoke, call a function; analysing how function handles and access to functions for execution.	4.1 Describe how to construct program modularly from functions 4.2 Describe the common math functions available in the C standard library 4.3 Illustrate how to create new functions 4.4 Describe the mechanisms used to pass information between functions 4.5 Describe simulation techniques using random number generation 4.6 Illustrate how to write and use functions that call themselves.
5. Abstract data types and the array aggregate data structure that is designed to store a group of objects of the same or different types.	5.1 Describe the array data structure 5.2 Describe the use of arrays to store, sort and search lists and tables of values. 5.3 Demonstrate how to define an array, initialise an array and refer to individual elements of an array 5.4 Demonstrate how to pass arrays to functions 5.5 Describe basic sorting techniques 5.6 Define and manipulate multiple subscript arrays
6. Pointers as extremely powerful programming tool that can make some things much easier, help improve program's efficiency.	6.1 Define pointers 6.2 Describe how to use pointers 6.3 Describe how to use pointers to pass arguments to functions using call by reference 6.4 Describe the close relationships among pointers, arrays and strings 6.5 Describe the use of pointers to functions 6.6 Define and use arrays of strings. 6.7 Demonstrate how the values that can be initialised to a pointer
7. Understand the fundamentals of strings; *string.h* function implementation and characters in C Programming.	7.1 Define character and string definitions 7.2 Analyse C Programming character handling library 7.3 Demonstrate using string conversion functions 7.4 Use standard input/output to manipulate character and string data 7.5 Describe string manipulation functions 7.6 Write programs comparing strings 7.7 Create search functions
8. Understand reading formatted input from standard input and outputting formatted output on screen/printer to present results	8.1 Define scanf/printf 8.2 Describe streams 8.3 Create output formatting 8.4 Create integer conversion programs 8.5 Create sample program using char and string arguments 8.6 Be able to produce field width
9. Creating C structures, enumerations, unions and bit manipulations	9.1 Define structures 9.2 Produce sample structure program 9.3 Be able to initialise structures and accessing structure members 9.4 Demonstrate passing structures to functions 9.5 Create Union program 9.6 Describe bitwise operators

	9.7 Demonstrate using enumeration type
10. Reading and writing simple files; file handling file; file processing, and sequential-access file system in C.	7.1 Describe data hierarchy 7.2 Define files and streams 7.3 Create a sequential file 7.4 Demonstrate how to create, read, write and update files 7.5 Create a randomly accessed file 7.6 Familiarise with sequential access file processing 7.3 Familiarise with random-access file processing
11. Understand linked lists, stacks, queues and binary trees.	11.1 Describe self-referential structures 11.2 Define dynamic memory allocation 11.3 Be able to operate and maintain a list 11.4 Create a dynamic stack program 11.5 Be able to create, operate and maintain a queue 11.6 Demonstrate creating and traversing a tree
Methods of Evaluation: A 2½-hour written examination paper with five essay questions, each carrying 20 marks. Candidates are required to answer all questions. Candidates also undertake project/coursework in C Programming with a weighting of 100%.	

Recommended Learning Resources: C Programming

Text Books	• The C Programming Language by Brian W. Kernighan and Dennis Ritchie. • Absolute Beginner's Guide to C by Greg Perry. ISBN-10: 0672305100 • C Programming by KN King. ISBN-10: 0393979504
Study Manuals 	BCE produced study packs
CD ROM 	Power-point slides
Software 	C Programming Language