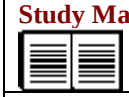
Level 6 Advanced Diploma in Programming (602)
163 Credits

Unit: Advanced C Programming **Exam Paper No.:** 2	**Guided Learning Hours:** 240 **Number of Credits:** 24
Prerequisites: Programming experience in C for at least six months.	**Corequisites:** A pass or higher in Diploma in Programming or equivalence.
Aim: The unit concentrates on the design, implementation concepts of C Programming, and use of data structures in C language. Building advanced data structures based on primitive data types will be illustrated. Theoretical issues as well as examples relating to practical applications will be discussed. Emphasis will be on programming, using and improving different data structures. Understanding the complexities of different algorithms help learners write efficient programs. The library functions, pre-processing and input/output redirection will be analysed and practised in detail.	
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 Characters, strings, string functions and the power of function libraries as a means of achieving software reusability.	1.1 Explain how to use the functions of the character handling library (**ctype**) 1.2 Describe how to use the string and character input/output functions of the standard input/output library (**stdio**) 1.3 Demonstrate how to use the string conversion functions of the general utilities library (**stdlib**) 1.4 Demonstrate how to use the string processing functions of the string handling library (**string**)
2 Standard library functions for file input input, output stream, conversion and field width specifier.	2.1 Describe input and output streams 2.2 Demonstrate how to use all print formatting capabilities 2.3 Demonstrate how to use all input formatting capabilities
3 How to aggregate variables under one name and identify the different methods of defining structures.	3.1 Describe how to create and use structures, unions and enumerations 3.2 Demonstrate how to pass structures to functions call by value and call by reference 3.3 Identify how to manipulate data with the bitwise operators 3.4 Describe how to create bit fields for storing data compactly.
4 Data structures organisation or clubbing, accessing technique and manipulating selections for information operations and demonstrate the implementation of linked lists, stacks and queues.	4.1 Describe how to allocate and free memory dynamically for data objects 4.2 Demonstrate how to form linked data structures using pointers, self-referential structures and recursion 4.3 Define how to create and manipulate linked lists, queues, stacks and binary trees 4.4 Demonstrate various important applications of linked data structures.
5 Understand pre-processing, inclusion of files in a program, execution and format of pre-processor	5.1 Describe the use of **#include** processor directive

directives.	5.2 Explain the **#define** 5.3 Define conditional compilation 5.4 Describe **#error** and **#progma** tokens 5.5 Explain the # and ## operators 5.6 Demonstrate **#line** implementation 5.7 Describe and demonstrate implementation of predefined symbolic constants
6 Understand the implementation of Advanced topics in C Programming.	6.1 Demonstrate redirecting input/out to a file 6.2 Write functions with unspecified number of arguments 6.3 Design program with multiple source files 6.4 Use **exit** and **atexit** functions 6.5 Describe suffixes for integer and floating-point constants 6.6 Demonstrate processing of binary files 6.7 Describe signal handling 6.8 Demonstrate dynamic memory allocation with **calloc** and **realloc**.
7 Understand strings and pointers; creating programs accepting string from the user and calculating length strings using pointers.	7.1 Describe fixed and variable length 7.2 Concatenating two strings. 7.3 Use **fgets** and **sscanf**. 7.4 Use **fputs** and **sprintf**.
8 Understand C Programming derived data types.	8.1 Be able to use arrays 8.2 Be able to use pointers 8.3 Be able to use enumerated 8.4 Be able to use structure 8.5 Be able to use union
9 Binary files and the syntax for writing and reading in binary file format.	9.1 Compare and contrast text *vs* binary files 9.2 Demonstrate how to read and write binary files 9.3 Demonstrate how to append binary 9.4 Outline the I/O categories 9.5 Demonstrate how to merge binary file
Methods of Evaluation: A 3-hour essay written paper with 5 questions, each carrying 20 marks. Candidates are required to answer all questions. Candidates also undertake project/coursework in Advanced C Programming with a weighting of 100%.	

Recommended Learning Resources: Advanced C Programming

Text Books	• The C Programming Language by Brian W. Kernighan and Dennis Ritchie. ISBN-10: 0131103628 • Advanced C Programming by Example by John Perry. ISBN-10: 0534951406 • Advanced C. Programming by Waite Group. ISBN-10: 0893034738
Study Manuals 	BCE produced study packs
CD ROM 	Power-point slides
Software 	C Programming Language