Level 5 Diploma in Programming (601) 157 Credits

Unit: C++ Programming **Exam Paper No.:** 4	**Guided Learning Hours:** 220 **Number of Credits:** 22
Prerequisites: Knowledge of C Programming language.	**Corequisites:** A pass or higher in Diploma in System Design or equivalence.
Aim: This unit covers basic concepts in C++ to learn skills of programming, including C++ elements, selection, iteration, functions and arrays. C++ remains the premier object-oriented language. This unit goes beyond simple syntax learning by teaching the principles of good object-oriented design in the context of the C++ language. Special emphasis is placed on abstract interfaces, polymorphism and data abstraction. The unit provides a pragmatic, systematic approach to software development using object-oriented design principles, the test-driven development process and the C++ language. Using this approach, learners will learn how to leverage the C++ language and libraries to produce high-quality, robust, defect-free code. The object-oriented design principles are reinforced throughout to help learners learn to identify whether a design is "good" or "bad". The test-driven development process is taught and practiced in all subsequent exercises. Following the simple steps involved in test-driven development results in vastly improved design, minimized defects, and comprehensive system-level documentation. The unit teaches core C++ assuming completion of C or Visual Basic Programming languages. The differences between C++ and C are discussed. Algorithm development, data representation, logical expressions, sub-programs and input/output operations are also covered.	
Required Materials: Recommended learning resources.	**Supplementary Materials:** Lecture notes and tutor extra reading recommendations.
Special Requirements: This is a hands-on course, hence practical use of computers is essential. Requires intensive lab work outside of class time.	
Intended Learning Outcomes:	**Assessment Criteria:**
1. C++ as both a procedural; object oriented programming language, and the different types of programming languages.	1.1 Describe a typical C++ program-development environment 1.2 Demonstrate how to write simple computer programs in C++ 1.3 Demonstrate how to use simple input and output statements 1.4 Examine and identify the fundamental data types 1.5 Describe how to use arithmetic operators 1.6 Describe the precedence of arithmetic operators 1.7 Demonstrate how to write simple decision-making statements
2. Understand control structure, the C++ standard library functions; classes and analysing the process of solving problems in C++.	2.1 Describe basic problem-solving techniques 2.2 Demonstrate how to develop algorithms through the process of top-down, stepwise refinement 2.3 Demonstrate how to use the **`if`**, **`if`**/**`else`** and **`switch`** selection structures to choose among alternative actions 2.4 Demonstrate how to use the **`while`**, **`do`**/**`while`** and **`for`** repetition structures to execute statements in a program repeatedly

	2.5 Describe counter-controlled repetition and sentinel-controlled repetition 2.6 Demonstrate how to use the increment, decrement, assignment and logical operators 2.7 Illustrate how to use the **`break`** and **`continue`** program control statements
3. Identifying why the best way to develop and maintain a large program is to divide it into several smaller program modules and demonstrating how functions are invoked.	3.1 Demonstrate how to construct programs modularly from pieces called functions 3.2 Demonstrate how to create new functions. 3.3 Describe the mechanisms used to pass information between functions 3.4 Describe simulation techniques using random number generation 3.5 Illustrate how the visibility of identifiers is limited to specific regions of programs 3.6 Describe how to write and use functions that call themselves.
4. How C++ stores values in arrays, essentially a way to store many values under the same name, demonstrating how to declare an array.	4.1 Describe the array data structure 4.2 Illustrate the use of arrays to store, sort and search lists and tables of values 4.3 Describe how to declare an array, initialise an array and refer to individual elements of an array 4.4 Describe how to pass arrays to functions 4.5 Describe the basic sorting techniques 4.6 Describe how to declare and manipulate multiple-subscript arrays
5. Declaring pointers; what pointers are; how to use pointers in C++ to work with memory and the different ways to pass a pointer to a function.	5.1 Describe how to use pointers 5.2 Illustrate how to use pointers to pass arguments to functions by reference 5.3 Describe the close relationships among pointers, arrays and strings 5.4 Describe the use of pointers to functions 5.5 Demonstrate how to declare and use arrays of strings
6. How structures operate and demonstrating the difference between a structure, an array and a class.	6.1 Define the software engineering concepts of encapsulation and data hiding 6.2 Describe the notions of data abstraction and abstract data types (ADTs) 6.3 Create C++ ADTs, namely, classes 6.4 Describe how to create, use and destroy class objects 6.5 Describe how to control access to object data members and member functions 6.6 Analyse the value of object orientation.
7. Understand objects, member functions, friend functions and classes, dynamic memory management and data abstraction.	7.1 Define constant objects and member functions 7.2 Demonstrate construction of objects 7.3 Demonstrate declaring friends and properties 7.4 Use pointer to refer to object members

8. Understand the fundamentals and the basic rules and idioms for operator overloading in C++.	8.1 Define operator overloading 8.2 Demonstrate creating operator functions 8.3 Implement array class and copy constructor 8.4 Demonstrate increment/decrement overloading
9. Understand concepts, characteristics and implementation of inheritance in Object Oriented Programming.	9.1 Describe types of inheritance 9.2 Explain base and derived class relationship 9.3 Describe three levels of inheritance hierarchy 9.4 Demonstrate constructors and destructors in derived class
10. Understand polymorphism concepts, meaning and implementation in Object Oriented Programming.	10.1 Define polymorphism 10.2 Invoke base-class and derived class functions 10.3 Create a virtual function derived class 10.4 Demonstrate using polymorphism derived class
11. File processing, input/output header files and writing to a file and reading from a file; mechanisms used by C and C++ to access external files residing on disk.	11.1 Describe how to create, read, write and update files 11.2 Define and describe sequential-access file processing 11.3 Define and describe random-access file processing 11.4 Outline and specify high-performance unformatted I/O operations 11.5 Describe the differences between formatted-data and raw-data file processing 11.6 Demonstrate building a transaction-processing program using random-access file processing
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	• C++: A Beginner's Guide by Herbert Schildt. ISBN-10: 0072232153 • The C++ Programming Language by Bjarne Stroustrup. ISBN-10: 0201700735 • The C++ Standard Library: A Tutorial and Reference by Nicolai M. Josuttis. ISBN-10: 0201379260
Study Manuals 	BCE produced study packs
CD ROM 	Power-point slides
Software 	C++ Programming