

## **LONDON CAPITAL COMPUTER COLLEGE**

## Diploma in Programming (601) - C++

<b>Prerequisites:</b> Basic programming skills or basic	Corequisites: A pass or higher in Diploma in			
knowledge of computer use.	System Design or equivalence.			
<b>Aim:</b> This course teaches core C++ assuming completion of C or Visual Basic Programming				
languages. The course covers basic concepts in C+				
elements, selection, iteration, functions and arrays.				
Algorithm development, data representation, logical expressions, sub-programs and input/output				
operations are also covered.				
Required Materials: Student study materials	Supplementary Materials: Recommended			
	textbooks and lecture notes.			
<b>Special Requirements:</b> 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. Define C++ as both a procedural and	1.1 Describe a typical C++ program-			
object oriented programming language. Describe	development environment			
the different types of programming languages.	1.2 Demonstrate how to write simple			
	computer programs in C++			
	1.3 Demonstrate how to use simple input and			
	output statements			
	1.4 Familiarise with 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			
	1.7 Demonstrate how to write simple decision-making statements			
2. Describe the C++ standard library	2.1 Describe basic problem-solving			
functions and classes. Analyse the process of	techniques			
solving problems in C++.	2.2 Demonstrate how to develop algorithms			
	2.2 Demonstrate how to develop algorithms through the process of top-down,			
	stepwise refinement			
	step wise remement			
	2.3 Demonstrate how to use the <b>if</b> ,			
	if/else and switch selection			
	structures to choose among alternative actions			
	actions			
	2.4 Demonstrate how to use the <b>while</b> ,			
	· ·			
	do/while and for repetition			
	structures to execute statements in a			
	program repeatedly			
	2.5 Describe counter-controlled repetition			
	and sentinel-controlled repetition			
	and sentiner-controlled repetition			
	2.6 Demonstrate how to use the increment,			
	decrement, assignment and logical			
	operators			
	<u> </u>			

3. Identify why the best way to develop and maintain a large program is to divide it into several smaller program modules. Discuss how functions are invoked.	2.7	Illustrate how to use the <b>break</b> and <b>continue</b> program control statements
	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
4. Describe how C++ stores values in	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.
arrays. Define how to declare an array.	4.1	Discuss 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. Understand how to declare pointers.		
Discuss the different ways to pass a pointer to a function.	5.1 5.2	Describe how to use pointers Illustrate how to use pointers to pass
	5.3	arguments to functions by reference Describe the close relationships among pointers, arrays and strings
	5.4 5.5	Describe the use of pointers to functions Demonstrate how to declare and use arrays of strings
6. Define how structures operate. Discuss the difference between a structure and a class.	6.1	Define the software engineering concepts
the difference between a structure and a class.		of encapsulation and data hiding
	6.2	Describe the notions of data abstraction and abstract data types (ADTs)
	6.3 6.4	Create C++ ADTs, namely, classes 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. Define file processing. Describe		

input/output header files.	7.1	Describe how to create, read, write and update files
	7.2	Familiarise with sequential-access file processing
	7.3	Familiarise with random-access file processing
	7.4	Specify high-performance unformatted I/O operations
	7.5	Describe the differences between formatted-data and raw-data file processing
	7.6	Build a transaction-processing program using random-access file processing

Recommended Learning Resources: C++ Programming

Text Books	<ul> <li>C++: A Beginner's Guide by Herbert Schildt. ISBN-10: 0072232153</li> <li>The C++ Programming Language by Bjarne Stroustrup. ISBN-10: 0201700735</li> <li>The C++ Standard Library: A Tutorial and Reference by Nicolai M. Josuttis. ISBN-10: 0201379260</li> </ul>
Study Manuals	BCE produced study packs
CD ROM	Power-point slides
Software	C++ Programming