LONDON CAPITAL COMPUTER COLLEGE

Advanced Diploma in Programming (602) – Advanced C++

<table>
<tr><td><b>Prerequisites:</b> Programming experience in C for at least six months.</td><td><b>Corequisites:</b> A pass or higher in Diploma in Programming or equivalence.</td></tr>
<tr><td colspan="2"><b>Aim:</b> Topics will focus on ANSI C++ and include data structures in C++, use of the Standard Template Library and graphical programming using the classes. Candidates will learn programming techniques in Standard C++ for large-scale, complex, or high-performance software. Encapsulation, automatic memory management, exceptions, generic programming with templates and function objects, standard library algorithms and containers. Using single and multiple inheritance and polymorphism for code reuse and extensibility, basic design idioms, patterns, and notation will be covered.</td></tr>
<tr><td><b>Required Materials:</b> Recommended Learning Resources.</td><td><b>Supplementary Materials:</b> Lecture notes and tutor extra reading recommendations.</td></tr>
<tr><td colspan="2"><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.</td></tr>
<tr><td><b>Intended Learning Outcomes:</b></td><td><b>Assessment Criteria:</b></td></tr>
<tr><td>1. Discuss the implementation of a class.</td><td>1.1 Identify how to specify const (constant) objects and const member functions
1.2 Describe the purpose of friend functions and friend classes
1.3 Describe the use of the <i>this</i> pointer
1.4 Demonstrate how to create and destroy objects dynamically
1.5 Describe how to use static data members and member functions
1.6 Describe the concept of a container class
1.7 Describe the notion of iterator classes that walk through the elements of container classes.</td></tr>
<tr><td>2. Describe overloading.</td><td>2.1 Describe how to redefine (overload) operators to work with new Abstract Data Types (ADTs)
2.2 Demonstrate how to convert objects from one class to another class
2.3 Describe when to, and when not to, overload operators
2.4 Demonstrate how to create array, string and date classes that demonstrate operator overloading.</td></tr>
<tr><td>3. Describe how software re-use reduces program development time.</td><td>3.1 Describe how to create classes by inheriting from existing classes
3.2 Describe how inheritance promotes software reusability
3.3 Describe the notions of base classes and derived classes
3.4 Describe the protected member-access modifier
3.5 Describe the use of constructors and destructors in inheritance hierarchies.</td></tr>
<tr><td>4. Define polymorphism and how it is</td><td>4.1 Describe the concept of polymorphism
4.2 Describe how to declare and use virtual</td></tr>
</table>

implemented.	functions to effect polymorphism 4.3 Distinguish between abstract and concrete classes 4.4 Demonstrate how to declare pure virtual functions to create abstract classes 4.5 Define how polymorphism makes systems extensible and maintainable 4.6 Describe how C++ implements virtual functions and dynamic binding "under the hood." 4.7 Identify how to use Run-Time Type Information (RTTI) and operators typeid and dynamic_cast.
5. Analyse templates and discuss how programmers can use them.	5.1 Demonstrate how to use function templates to create a group of related (overloaded) functions 5.2 Distinguish between function templates and function-template specialisations 5.3 Describe how to use class templates to create a group of related types 5.4 Distinguish between class templates and class-template specialisations 5.5 Describe how to overload function templates 5.6 Describe the relationships among templates, friends, inheritance and static members.
6. Discuss input/output streams. Discuss low-level and high-level input/output capabilities. Define unformatted and formatted input/output.	6.1 Describe how to use C++ object-oriented stream input/output 6.2 Identify how to format input and output 6.3 Describe the stream-I/O class hierarchy 6.4 Describe how to input/output objects of programmer-defined types 6.5 Identify how to use stream manipulators
7. Define exception handling.	7.1 Demonstrate try, throw and catch to detect, indicate and handle exceptions, respectively process uncaught and unexpected exceptions 7.2 Describe how to handle new failures 7.3 Demonstrate how to use auto_ptr to prevent memory leaks 7.4 Describe the standard exception hierarchy.

Recommended Learning Resources: Advanced C++ Programming

Text Books	• Advanced C++ Programming Styles and Idioms by James O. Coplien. ISBN-10: 0201548550 • Algorithms in C++: Fundamentals, Data Structures, Sorting, Searching Pts. 1-4 by Robert Sedgewick. ISBN-10: 0201350882 • 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 Language