Course Outcome Guide (COG)

Approved 14 September 2012

Course:	CSCI 127 – Beginning C++/Visual C++	**Credits:**	3	**Instructor:**	TBD
Course Description:	Introduction to programming in the C++/Visual C++ language.				

Concepts and Issues	Process Skills	Assessment Tasks	Intended Outcomes		
			Course	General Education or Program	Institutional
Hardware and Software: different types of computers and terms; parts of a computer (memory, CPU, I/O); how data and instructions are represented including base conversions; types of software; types of programming languages and terms. Introduction to operating systems. • Compilation process, programming process (software developmentlifecycle), algorithm development, structured programming principles and construction. Writing algorithms using pseudocode.	Apply object-oriented design to small software projects. Produce simple object-oriented programs demonstrating use of class definition, methods, primitive and reference data types, alternation and repetition control structures, and file-based and interactive input/output. Produce simple event-driven object-oriented programs using basic Java library components. Assess the quality of programs using simple glass box and black box testing strategies. Describe and demonstrate different physical data representations for primitive data types. Use good software development principles including object-oriented design, test planning and adherence to style guidelines.	*Participate in class discussions and activities demonstrating knowledge of subject matter. *Complete examinations demonstrating acceptable skill level of concept and process. *Complete textbook readings, questions and problems (both individually and collaboratively) demonstrating acceptable skill levels of concept and process. * Design, construct and test your final project. Using the C++ compiler and the UNIX system • Writing a simple C++ program using data declarations, conditionals and loops • Designing and implementing a modular solution • Designing and implementing a solution using 1-d and 2-d arrays	1. Be exposed to basic hardware and software concepts 2. Be familiar with issues related to software design 3. Master using key structured programming constructs: declarations, sequence, selection, repetition, evaluating expressions. 4. Be familiar with using C++ functions and the concepts related to good modular design. 5. Master one-dimensional and two-dimensional arrays 6. Be familiar with using C++ structures. 7. Be familiar with	1.Mathematics-including numeration literacy and the knowledge and use of statistical and logical processes. 2.Analytical-gathering, organizing, and evaluating information 3.Analogical-using former knowledge to help comprehend and explain new situations 4.Critical Thinking-the ability to identify ad define criteria, understand biases, and construct objective judgments. 5.Problem solving-the ability to analyze situations and synthesize solutions.	1. Students will demonstrate effective communication skills. 2. Students will use reasoning skills to analyze and solve problems.

• Preprocessor directives and system libraries. • Variables, expressions (arithmetic, Boolean, literals), assignment statements, precedence, association, data types, enumeration types, union types. • Interactive input/output, formatting output. • Branching statements (single selection, multi-way, switch). • Repetition structures (while, do-while, for). Counter-control loops (incremental and decremental), event control loops including sentinel control. Nested repetition. • Programming standards and style guidelines: good documentation. • Pointers including pointers to pointers and references. • Modular design, functions, predefined functions. (omit recursive	Describe the purpose and operation of Java software development tools including compilers, editors, and integrated development environments; use tools to do software development. Describe the Java runtime environments. Identify and describe the activities involved in the software development process.		using pointers and reference parameters. 8. Be familiar with using text file input/output 9. Be familiar with C++ classes.		

functions). Call-by-value, call-by-address, call-by-reference. Overloading a function name.					