OUTCOMES BASED LEARNING MATRIX

Course: CTIM371 – Programming in C++

Department: Computer Technology and Information Management

> **Course Description: This is the first course in the C++ programming language. The course will cover general program structures, functions, variable naming rules, iteration statements (for, while, do/while), arithmetic and relational operators, arrays, an introduction to pointers, and an introduction to objects. Hands-on programming exercises will be completed using the college's workstation computers and Visual Studio C++ compiler.**

Lecture: 2 Hours Laboratory: 2 Hours

Prerequisite: Beginning Windows or higher and Beginning Word or higher and Beginning Excel or higher and Software Design and Development or permission of Department.

While completing the table below, remember that the individual outcomes you list in the first column should answer this question: **What must the learner know and be able to do at the end of the course?** Items in the third column should answer the question: **How do we know?** The second column is where teachers can be most creative; it's for pedagogy. Each rectangle in column one should contain just one outcome; the corresponding rectangles in columns two and three, however, may contain more than one item. Using the code at the end of the matrix, indicate the core competencies being strengthened by the outcomes activities and the assessment tools.

Approved by CTIM Department: September 2003

COURSE OUTCOMES	OUTCOME ACTIVITIES	ASSESSMENT TOOLS
At the end of this course, the student will be able: 1. To introduce the hybrid (high level / low level) C++ programming language.	1. Describe source code, compilers, linkers, assemblers, and machine code. (WC) 2. Identify the five steps in the program development cycle. (WC) 3. Write a simple C++ program. (WC) 4. Use comments, declare different integer data types and decide when to use them. (WC) 5. Use the `cout` output stream object to display strings and variable data. (WC) 6. Use the `cin` input stream object to input data. (WC) 7. Use arithmetic operators and understand operator precedent rules. (WC) 8. Solve simple problems involving integers. (WC)	1. Homework: Exercises from the end of the chapter. (WC,IL,QL,IG) 2. Computer Lab/Homework: Design & development computer programs given the problem definitions. (WC,CCT,QT,IG) 3. Exam: Course Objective test. (WC,IL,CCT,QT,IG)

COURSE OUTCOMES	OUTCOME ACTIVITIES	ASSESSMENT TOOLS

2. To understand real numbers in C++ programming.	1. Declare float and double variables. (R) 2. Use `cin` to input and `cout` to display formatted decimal numbers. (WC) 3. Solve problems using decimal numbers. (WC) 4. Use mixed-mode arithmetic expressions, data promotion, and type casting. (WC) 5. Understand the evaluation aspect of a C++ expression. (WC) 6. Use the compound assignment, increment, and decrement operators. (WC)	1. Homework: Exercises from the end of the chapter. (WC,IL,QL,IG) 2. Computer Lab/Homework: Design & development computer programs given the problem definitions. (WC,CCT,QT,IG) 3. Exam: Course Objective test. (WC,IL,CCT,QT,IG)

COURSE OUTCOMES	OUTCOME ACTIVITIES	ASSESSMENT TOOLS

3. To understand the concept of iteration in C++ programming.	1. Write conditional expressions using relational and Boolean logic operators. (WC) 2. Use the while and do while statements to implement indefinite iteration loops. (WC) 3. Use loops and the cin.get () member function to obtain characters from the input buffer. (WC) 4. Use embedded assignment operators in the condition expression of a while loop. (WC) 5. Use counters and accumulators. (WC) 6. Use the for statement to implement definite iteration loops. (WC) Code nested `while` and `for` loops. (WC)	1. Homework: Exercises from the end of the chapter. (WC,IL,QL,IG) 2. Computer Lab/Homework: Design & development computer programs given the problem definitions. (WC,CCT,QT,IG) 3. Exam: Course Objective test. (WC,IL,CCT,QT,IG)

COURSE OUTCOMES	OUTCOME ACTIVITIES	ASSESSMENT TOOLS
4. To understand the concept of data-driven program execution flow control in C++ programming.	1. Code an if-else statement to make simple decisions. (WC) 2. Code compound condition expressions using logical and Boolean operators. (WC) 3. Code nested if-else statements to make complex decisions. (WC) 4. Write a program that uses nested if-else statements. (WC) 5. Code a `switch` statement to select 1/N cases. (WC) 6. Use the break statement to prematurely exit from a loop. (WC) 7. Use the continue statement to end the current iteration of a loop. (WC)	1. Homework: Exercises from the end of the chapter. (WC,IL,QL,IG) 2. Computer Lab/Homework: Design & development computer programs given the problem definitions. (WC,CCT,QT,IG) 3. Exam: Course Objective test. (WC,IL,CCT,QT,IG)
COURSE OUTCOMES	OUTCOME ACTIVITIES	ASSESSMENT TOOLS

5. To understand arrays in C++ programming. To fortify the students for Advanced C++ Programming.	1. Declare arrays and initialize arrays. (WC) 2. Realize that array subscripts/indexes are expressions. (WC) 3. Use `for` loops to process arrays. (WC) 4. Code a linear search of an array. (WC)	1. Homework: Exercises from the end of the chapter. (WC,IL,QL,IG) 2. Computer Lab/Homework: Design & development computer programs given the problem definitions. (WC,CCT,QT,IG) 3. Exam: Course Objective test. (WC,IL,CCT,QT,IG)

COURSE OUTCOMES	OUTCOME ACTIVITIES	ASSESSMENT TOOLS

6. To understand the use of functions in C++ programming and to fortify the students for Advanced C++ Programming	1. Understand how functions work in C++ programming. (WC) 2. Using C++ library functions. (WC) 3. Declaring user function prototypes. (WC) 4. Code user functions with arguments passed by value and with return values. (WC)	1. Homework: Exercises from the end of the chapter. (WC,IL,QL,IG) 2. Computer Lab/Homework: Design & development computer programs given the problem definitions. (WC,CCT,QT,IG) 3. Exam: Course Objective test. (WC,IL,CCT,QT,IG)

*Try to express an outcome as an infinitive phrase that concludes this sentence: **At the end of the course, the students should be able to . . ..** Finding the line between too general and too specific can be difficult. In an English Composition course, for instance, it is probably too general to say, "The student should be able to write effective essays." It is probably too specific to say, "The student should be able to write an introductory paragraph of at least 50 words, containing an attention-getting device, an announcement of the narrowed topic, and an explicit thesis sentence." Just right might read, "The student will write introductions that gather attention and focus the essay."
**Indicate the Core Competencies that apply to the outcomes activities and assessment tools: Critical Thinking (CT); technology skills (TS); oral communications (OC); quantitative skills (QS); reading (R); writing (W).

**Indicate the Core Competencies that apply to the outcomes activities and assessment tools: Critical and Creative Thinking (CCT); Integrative Learning (IG); quantitative Literacy (QL); Information Literacy(IL); Information Literacy(IL). Approved by the CTIM Department – September, 2015