

LAST REVIEW: 2010-2011

NEXT REVIEW: 2015-16

STATUS: A

A

COURSE TITLE: Introduction to C++ Programming

COMMON COURSE NUMBER: COP 1334C

CREDIT HOURS: 3

CONTACT HOUR BREAKDOWN

(per 16 week term)

CLOCK HOURS:

Lecture: **48**

Lab: **16**

Clinic: **0**

Other: **0**

PREREQUISITE(S): MAT 1033 or MTB 1310

COREQUISITE(S):

PRE/COREQUISITE(S): CIS 1000C

COURSE DESCRIPTION:

This course provides an introduction to computer program design and development using the C++ language. A structured, multi-phase, program development process featuring a series of steps involving problem definition, top-down design, and formal program specification is stressed. The course is intended to provide the novice programming student with the techniques needed to develop well-documented, structured computer programs. Students who do not possess computer programming experience are strongly encouraged to complete COP 1000C (Introduction to Computer Programming) before attempting this course.

General Education Requirements – Associate of Arts Degree (AA), meets Area(s): Area

General Education Requirements – Associate in Science Degree (AS), meets Area(s): Area

General Education Requirements – Associate in Applied Science Degree (AAS), meets Area(s): Area

UNIT TITLES

- 1. Introduction to Programming and Problem Solving**
- 2. The Program Development Process and Software Design**
- 3. Intro to C++ Syntax and Semantics and Simple Data Types**
- 4. Arithmetic Expressions and Function Calls**
- 5. Input and Output**
- 6. Conditions, Logical Expressions, & Selection Control Structures**
- 7. Looping Control Structures**
- 8. Functions**
- 9. Arrays and Their Applications**

10. Structures

EVALUATION:

Please provide a brief description (250 characters maximum) that details how students will be evaluated on the course outcomes.

Evaluation instruments will include written and/or skills-based examination and individual in-class and/or take-home assignments. Evaluation methods may also include group in-class and/or take-home assignments.

****Required Competencies**

1) Read with critical comprehension.

The student will be introduced to the basic texts, concepts, vocabulary, and methods necessary for developing an understanding of the discipline and meeting the required benchmarks as stated in the course outline.

2) Write clearly and coherently.

The student will demonstrate an understanding and mastery of subject matter in a variety of ways, including writing. Writing activities may include both graded and ungraded essays, short answer quizzes, summaries, reactions, journals, and various other reports.

3) Demonstrate and apply literacy across all the disciplines (indicate which ones apply).

- a) **Information literacy** means understanding how to locate needed information, using the appropriate technology for the task, managing and evaluating the extracted information and using it effectively and ethically.
- b) **Technology literacy** is the ability to responsibly and effectively use appropriate technology to access, manage, integrate, or create information, and/or use technology to accomplish a given task.
- c) **Workplace literacy** is having the appropriate knowledge and skills to communicate and work with others effectively and perform job duties, whether it is through the use of computers and/or other technology.
- d) **Cultural literacy** is recognizing, understanding, and appreciating the similarities and differences between one's own culture and the cultures of others through a study of the arts, customs, beliefs, values, and history that define a culture.
- e) **Quantitative literacy** is having the ability to formulate, solve and interpret mathematical/statistical operations and graphical/tabular representations to make informed decisions.
- f) **Scientific literacy** means understanding the methodology and application of the scientific process, the physical and biological worlds, and recognizing that scientific knowledge is continuously updated or revised as new information is discovered.

4. Apply problem-solving skills or methods to make informed decisions in a variety of contexts.

The student will use acquired skills or methods to recognize, analyze, adapt, and apply critical thinking to solve problems and make informed decisions.

EVALUATION:

In the box to the right of the Methods of Assessment, enter all specific learning outcome numbers (i.e. 1.1, 2.7, 4.0, 4.2 and 5.12) that apply.

1. Portfolio	
2. Short essays	
3. Research Papers	
4. Group projects	
5. Discussions (In class and online)	
6. Multiple Choice tests	
7. Presentations	
8. Service Learning Projects	
9. Pop quizzes	
10. Take-home tests	
11. Summaries and critiques	
12. Reaction papers	
13. Surveys	
14. Performance	
15. Short answer tests	
16. Classroom debates and colloquia	
17. Blogs, wikis, web pages	
18. Other (Please explain)	

UNITS

Unit 1: Introduction to Programming and Problem Solving

General Outcome:

1.0 The student shall:

Understand the concepts of computer and computer programming. Comprehend the Structure Theorem, as well as several problem-solving techniques. Develop algorithms to solve simple problems.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

- 1.1 Describe a computer**
- 1.2 Define computer programming**
- 1.3 Define computer language**
- 1.4 Explain and use several problem-solving techniques**
- 1.5 Discuss the Structure Theorem**
- 1.6 Write algorithms to solve problems**

Common Course Number: COP1334C

Unit 2: The Program Development Process and Software Design

General Outcome:

2.0 The student shall:

Understand the software development process and program design and construction. Develop further their algorithmic, problem-solving skills using a Top-down, structured design approach. Comprehend the C++ integrated development environment.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

2.1 Describe the software development process

2.2 Explain top-down design

2.3 Discuss program construction

2.4 Enter, compile, debug, and execute a C++ program

Unit 3: Introduction to C++ Syntax and Semantics and Simple Data Types

General Outcome:

3.0 The student shall:

Become familiar with the C++ programming language. Understand and apply the basic program elements and simple data types of C++ , as well as the use of program comments as a means of system documentation. Comprehend the meaning of language syntax and semantics.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

3.1 Explain the elements of a C++ program

3.2 Understand the difference between syntax and semantics

3.3 Describe the basic C++ data types

3.4 Create meaningful C++ identifiers

3.5 Create C++ constants and variables

Unit 4: Arithmetic Expressions and Function Calls

General Outcome:

4.0 The student shall:

Write simple C++ arithmetic expressions. Apply precedence rules of arithmetic operators, and understand the processes of type coercion and type casting. Invoke several C++ library functions.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

4.1 List the order of precedence of arithmetic operators

4.2 Create arithmetic expressions and assignment statements

4.3 Explain and use type coercion and type casting

4.4 Discuss and use value-returning, library functions

Unit 5: Input and Output

General Outcome:

5.0 The student shall:

Understand and apply the concepts of program input and output. Learn how C++ implements simple interactive and file I/O.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

5.1 Send output to the screen

5.2 Format output using manipulators

5.3 Receive input from the keyboard

5.4 Discuss the use of file streams

5.5 Use file streams for simple I/O

Unit 6: Conditions, Logical Expressions, & Selection Control Structures

General Outcome:

6.0 The student shall:

Understand and apply the selection statement and its C++ implementations. Learn the relational and logical operators, including their rules of precedence, and how to write boolean expressions for use in selection control structures. Comprehend basic concepts of Boolean logic.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

6.1 Explain the flow of control in a selection statement

6.2 Define and use the C++ relational operators

6.3 Define and use the C++ logical operators

6.4 List the order of precedence of relational & logical operators

6.5 Create C++ boolean expressions

6.6 Explain and use the if statement

6.7 Explain and use nested if statements

6.8 Explain and use the switch statement

Unit 7: Looping Control Structures

General Outcome:

7.0 The student shall:

Understand and apply the concept of the looping control structure and its C++ implementations. Learn to write count-controlled loops as well as several types of event-controlled loops (including sentinel and EOF controlled loops). Comprehend the use of nested loop structures.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

7.1 Explain the flow of control in a loop control structure

7.2 Discuss count-controlled loops

7.3 Discuss event-controlled loops

7.4 Explain and use the while statement to create count-controlled and event-controlled loops

7.5 Explain and use the do-while statement to create count-controlled and event-controlled loops

7.6 Explain and use the for statement to create count-controlled loops

7.7 Explain and use nested loops

Unit 8: Functions

General Outcome:

8.0 The student shall:

Understand the theory and implementation of user-defined subprograms, including their relationship to top-down, structured program design. Learn how to write C++ subprograms (functions), develop an understanding of the scope, lifetime, and memory addresses of program and function variables.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

8.1 Explain the syntax and semantics of user-defined functions

8.2 Discuss the flow of control of functions

8.3 Discuss the relationship between functions and top-down, structured design

8.4 Explain value-returning and void functions

8.5 Discuss value and reference parameters

8.6 Discuss the scope of identifiers

8.7 Explain the lifetime of a function variable

8.8 Define and use value-returning and void functions using value and reference arguments and local variables

Unit 9: Arrays and Their Applications

General Outcome:

9.0 The student shall:

Understand the structured data types and some of their applications. Comprehend the definition of arrays (including strings) and develop several, simple array-processing algorithms.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

9.1 Discuss the difference between simple and structured C++ data types

9.2 Describe various applications of arrays

9.3 Define and use one-dimensional arrays

9.4 Sort an array

9.5 Discuss and use strings

9.6 Define and use multidimensional arrays

Unit 10: Structures

General Outcome:

10.0 The student shall:

Understand the concept of a record (a C++ struct). Learn about the various applications of records in information technology, as well as be able to define, declare, and manipulate records.

Specific Measurable Learning Outcomes:

Upon successful completion of this unit, the student shall be able to:

10.1 Discuss the various applications of records (C++ structs)

10.2 Define and use structs