UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS/DESCRIPTION

Department and Course Number	CSCI 2840
Course Title	C++ & Object Oriented Programming
Course Coordinator	Harvey Siy
Total Credits	3
Date of Last Revision	March 3, 2015

1.0 Course Description

1.1 Overview of content and purpose of the course (catalog description).
C++ and Object Oriented Programming (OOP) is taught in the UNIX environment. Topics include C++ as a 'Better C,' OOP with C++, classes and data abstraction, operator overloading, inheritance, virtual functions and polymorphism, C++ stream I/O, templates.

1.2 Prerequisites of the course (courses).
CSCI 2240 or equivalent experience with programming in a high-level language
Facility with a high-level programming language like Pascal, Java, C or C++.
A solid understanding of pointers and scope are both required.
Ability to design and implement solutions to modest problems using assignment and flow control, procedures/subroutines/functions, scalars, arrays, classes/records/structures, and simple input/output.

1.3 Overview of content and purpose of the course:
This course provides an introduction to the C++ programming language and explores some of the powerful features of the language such as operator overloading, multiple inheritance and templates. The course further explores the object-oriented programming paradigm with discussions on polymorphism, visibility, and genericity. Beyond teaching the programming language, the course also covers fundamental program design strategies for the systematic development of nontrivial C++ programs.

1.4 Unusual circumstances of the course.
None

2.0 Course Justification Information

2.1 Anticipated audience / demand:
This course is designed primarily for individuals already possessing programming facility in another high-level language who need/want to learn how to program in C++ in the UNIX environment.

2.2 Indicate how often this course will be offered and the anticipated enrollment:
Every other year, but may be offered more frequently based on demand.

2.3 If it is a significant change to an existing course, please explain why it is needed:
n/a

3.0 Objectives

3.1 Be able to construct syntactically correct C++ programs using all language features, as well as recognize and correct syntax error in ANSI C++ programs.

3.2 Be able to effectively use C++ streams for input and output.

3.3 Be able to effectively use C++ classes, this includes the use of inheritance and polymorphism.

3.4 Be able to effectively use and overload operators in C++.

3.5 Be able to effectively use templates in C++.

3.6 Be able to manage memory using the new and delete operators in C++.

3.7 Be able to use the STL containers, including. This includes inheriting from them to create user data types.

4.0 Content and Organization

		Contact hours
4.1	Introduction	3.0
4.1.1	History and characteristics of C++	
4.1.2	Comparing C++ and Java	
4.2	Stream input and output	3.0
4.2.1	Basic input and output	
4.2.2	Overloading I/O on user and system data types	
4.2.3	File I/O	
4.2.4	Stream internals (binding streams to other I/O devices)	
4.3	Object-oriented programming concepts	6.0
4.3.1	Basic composition	
4.3.2	Access levels	
4.3.3	Inheritance and type inference	
4.3.4	Polymorphism	
4.3.5	Uses of virtual and pure virtual methods	
4.3.6	Multiple inheritance	
4.3.7	Anonymous functions and lambdas	
4.4	Operator overloading	3.0
4.4.1	Simple overloading	
4.4.2	Unary and Binary overloads	
4.4.3	Unary and Binary operators	
4.4.4	friend access to private data	
4.4.5	Using overloads to accomplish type casting	
4.5	Templates	9.0
4.5.1	Basic template use	
4.5.2	Use of templates as construction parameters	
4.5.3	Template metaprogramming	
4.6	Memory management	6.0
4.6.1	Using new and delete	
4.6.2	Managing pointers with classes	
4.6.3	Managing pooled memory with classes	
4.7	STL (standard template library)	9.0
4.7.1	Linear containers (vector, deque, list)	

4.7.2 Iterators (directional, random access, const, etc)
4.7.3 The C++ string type
4.7.4 Associative containers (set, mutliset, map, multiset)
4.7.5 Algorithms library

4.8 Programming larger systems 15.0

4.8.1 Programming idioms
4.8.2 Object-oriented design
4.8.3 Multi-paradigm design

5.0 Teaching Methodology

5.1 Methods to be used

This course is presented primarily through lectures, with on-line demonstrations.

5.2 Student role in the course

The student in this course will study the C++ programming language in depth, demonstrate understanding of the language by designing, writing and testing numerous programs, and take examinations.

6.0 Evaluation Information

6.1 Describe the typical types of student projects that will be the basis for evaluating student performance:

Students will write a significant number of C++ programs for the class, each of which is to be compiled and executed in a UNIX environment. All programs will use the C preprocessor, and an ANSI C++ compiler (g++ is traditional); commands will be executed using a standard UNIX shell (e.g. bash, sh, csh, ksh). Earlier programs should be completed in a week or less, while some programs later in the semester may require several weeks for completion. For some larger programs, the instructor may provide partially-complete solutions which are to be embellished or modified by the student. Each student will work independently.

6.2 Describe the typical basis for determining the final grade (e.g. weighting of various student projects):

The major portion (typically 60%) of the student’s grade will be determined by their success on the programming assignments. Programming assignments will be evaluated for correctness, readability, use of required features, and structure.

Students will work on a relatively complex class project (about 20% of grade) in groups.

A few (two or three) examinations will be given (including the final exam), and will be the basis for the remaining portion of the student’s grade.

6.3 Grading scale and criteria

Points	Grade
97 – 100%	A+
93 – 96%	A
90 – 92%	A–
87 – 89%	B+
83 – 86%	B
80 – 82%	B–
77 – 79%	C+
73 – 76%	C
70 – 72%	C–
67 – 69%	D+
63 – 66%	D
60 – 62%	D–
0 – 59%	F

7.0 Resource Material

7.1 Textbooks and/or other required readings used in course

Marc Gregoire. Professional C++ (3rd Edition), Wrox, 2014, ISBN: 978-0470932445.

Nicolai Josuttis. The Standard C++ Library: A Tutorial and Reference (2nd Edition), Addison-Wesley, 2012, ISBN: 978-0321623218.

This is not a text book, but rather a programmers reference. It is the best C++ book for intermediate level programmers or those with experience in other high-level languages.

Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition), Addison-Wesley, 2014, ISBN: 978-0321992789.

7.2 Other suggested reading materials, if any

International Standards Organization. The C++ Standard (INCITS/ISO/IEC 14882-2014), ANSI, 2014.

As an alternative, http://www.cppreference.com/ has up-to-date information of the latest C++ standard.

7.3 Current bibliography of resource for student’s information

Bruce Eckel. Thinking in C++: Introduction to Standard C++, Volume One (2nd Edition), Prentice Hall, 2000, ISBN: 978-0139798092.

Bruce Eckel, Chuck Allison. Thinking in C++: Practical Programming, Volume Two, Prentice Hall, 2003, ISBN: 978- 0130353139.

Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994, ISBN: 978-0201633610.

Scott Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition), Addison-Wesley, 2005, ISBN: 978-0321334879.

Scott Meyers. Effective STL: 50 Specific Ways to Improve Your Use of the Standard Template Library, Addison-Wesley, 2001, ISBN: 978-0201749625.

Bjarne Stroustrup. The C++ Programming Language (4th Edition), Addison-Wesley, 2013, ISBN: 978-0321563842.

7.0 Computer Science Accreditation Board (CSAB) Category Content (class time in hours)

CSAB Category	*Core*	*Advanced*
Data structures	15	5
Computer organization and architecture		
Algorithms and software design	10	5
Concepts of programming languages	5	

8.0 Oral and Written Communications

Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) to typically _____ pages and to make ___0__ oral presentations of typically _____ minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

9.0 Social and Ethical Issues

No coverage

10.0 Theoretical content

Although the course does not cover any theoretical topics in depth, it does provide some theoretical of the following topics.

		Contact Hours
10.1	Data representation (in memory and on disk)	3.0
10.2	Explicit and implicit dynamic memory management	3.0
10.3	Recursion	1.0
10.4	Red-Black Trees	2.0
10.5	Linked lists	1.0

11.0 Problem analysis

Students are expected to bring with them some of the analysis skills necessary for this course. Additional skills are obtained by working through the numerous assignments. All of the C++ libraries have analysis specifications given.

12.0 Solution design

The focal point of the course is the design and implementation of solutions to problems, primarily in the context of the C++ programming language and the UNIX operating system.

CHANGE HISTORY

Date	***Change***	***By whom***	***Comments***
3/2/2009	Initial write for ABET-specific	Clark	
6/16/2011	Updated prerequisites, contents and bibliography	Siy	
3/3/2015	Updated course contents to reflect more advanced material; updated bibliography	Siy	