Douglas College

EFFECTIVE: MAY 2002

CURRICULUM GUIDELINES

A: Division: **Instructional**

B: Department/ Program Area: **Science and Technology**

Date: **07 February 2002**

New Course [ ] Revision [X]

If Revision, Section(s) Revised: **H, Q**

Date Last Revised: **18 June 1997**

C:	**D:**	**E:**
CMPT 110	**Introduction to Computing Science Using C++**	**4**
Subject & Course No.	Descriptive Title	Semester Credits

F: Calendar Description: This course introduces the science of computing. Emphasis is placed on the analysis of problems, the design of algorithms, and the abstraction of control and data in computer implementations of the design. Initially structured top-down design and procedural programming is used followed by an introduction to recursive functional programming then an introduction to Object Oriented Design (OOD) and Object Oriented Programming (OOP). C++ is used as the implementation language.

G: Allocation of Contact Hours to Types of Instruction/Learning Settings
Primary Methods of Instructional Delivery and/or Learning Settings:

Lecture/lab

Number of Contact Hours: (per week / semester for each descriptor)

lecture - 2 x 2 hours/week
lab - 1 x 2 hours biweekly
student directed learning - 5 hours/week (approx)

Number of Weeks per Semester:

14

H: Course Prerequisites:
CMPT 100 or CMPT 101 and
MATH 110

I. Course Corequisites:

None

J. Course for which this Course is a Prerequisite:

CMPT 210, CMPT 220

K. Maximum Class Size:

34

L: PLEASE INDICATE:

[ ] Non-Credit

[ ] College Credit Non-Transfer

[X] College Credit Transfer: Requested [ ] Granted [X]

SEE BC TRANSFER GUIDE FOR TRANSFER DETAILS (www.bccat.bc.ca)

	UBC	CPSC (3)
	SFU	CMPT 101
	U. of Vic	CSC 110
	Others in transfer guide	

M: Course Objectives/Learning Outcomes

The student should be able to:

- C analyze problem specifications
- C form data and control abstractions
- C design computer algorithms using either a structured top down design methodology or Object Oriented Design
- C implement, in a widely acceptable style, algorithms in C++ suing standard programming methodologies
- C document a project

The student should understand the concepts of:

- C generality through abstractions
- C maintainability, reusability and extensibility through modularity

N: Course Content

1. Introduction and Review (syntax of C++)
   - 1.1 Program Structure
   - 1.2 Primitive data types and expressions
   - 1.3 Control Structures
   - 1.4 Functions and parameter passing
   - 1.5 Arrays
   - 1.6 Top-down design review and specs. For assignment #1: procedural programming with emphasis on control structures, procedures, and arrays
2. Abstractions (implementations are not considered)
   - 2.1 Strings
   - 2.2 Collections
     - 2.2.1 Lists
     - 2.2.2 Sets
     - 2.2.3 Stacks
3. Implementing Abstractions
   - 3.1 C++ Strings
   - 3.2 Introduction to pointers (domain of arrays and parameter passing)
   - 3.3 C++ records (struct)
   - 3.4 Structured design issues and specs. For assignment #2: procedural programming with emphasis on: cohesion and coupling and using more complicated static data structures
   - 3.5 Design of set primitives
   - 3.6 Recursion
     - 3.6.1 Numerical examples: factorial, Fibonacci, ...
     - 3.6.2 Examples from symbolic (LISP-like) Expressions (SExpressions)
   - 3.7 Discussion and specs for assignment #3: functional programming using an existing module for SExpressions with emphasis on recursion and list processing
4. Encapsulation, Instantiation, and OOP
   - 4.1 Structure (syntax and semantics)
   - 4.2 Examples
     - 4.2.1 Sets implementation and use
     - 4.2.2 Stacks implementation and use
   - 4.3 Specs for assignment #4: OOP
5. OOD and Separate Compilations
   - 5.1 OOD
   - 5.2 Examples
   - 5.3 Specs for assignment #5: OOP
   - 5.4 Introduction to inheritance
     - 5.4.1 Examples

O: Methods of Instruction

There are three components of the course: lectures, labs, and assignments

The lecture is used to introduce new material; usually via a sequence of theoretical concepts, practical considerations (usually language dependent), and one or more examples case studies. The book is to be used as an additional source of problems and examples.

The two hour biweekly lab is exclusively used to evaluate the student's practical programming ability. They are marked mostly on results, ie. correctness of the algorithm.

Assignments are the most important learning vehicle and are done on the student's own time. They are marked according to program design, correctness and efficiency of the algorithms, coding style, and completeness of the documentation.

P: Textbooks and Materials to be Purchased by Students

- Dale N., Weems C., Headington M., Programming and Problem Solving with C++. D.C. Heath and Company
- Portfolio for Programming Assignments
- Two 3 ½" high density diskettes

Q: Means of Assessment

The final grade will be calculated from a particular distribution from the range below. The exact distribution will be given to the student on the first day of classes along with the course outline and necessary policies.

Distribution Range:

6 labs	=	15% - 25%
1 or 2 tests @ 15%-20% each	=	15% - 40%
1 exam	=	20% - 40%
5 assignments	=	20% - 35%
attendance & participation	=	0% - 5%

Example Distribution:

6 labs	=	15%
test #1	=	15%
test #2	=	20%
assignments	=	25%
exam	=	25%
Total	**=**	**100%**

R: Prior Learning Assessment and Recognition: specify whether course is open for PLAR

Not available at this time

Course Designer(s)

Education Council/Curriculum Committee Representative

Dean/Director

Registrar

© Douglas College. All Rights Reserved.