SAULT COLLEGE OF APPLIED ARTS AND TECHNOLOGY

SAULT STE. MARIE, ONTARIO

Sault College

COURSE OUTLINE

COURSE TITLE:	**COMPUTER PROGRAMMING I**		
CODE NO. :	**CSD100**	**SEMESTER:**	**1**
PROGRAM:	**ALL COMPUTER STUDIES PROGRAMS**		
AUTHOR:	**Dennis Ochoski**		
DATE:	**August, 2001**	**PREVIOUS OUTLINE DATED:**	**Aug, 2000**
APPROVED:			
	DEAN		**DATE**
TOTAL CREDITS:	**5**		
PREREQUISITE(S):	**NONE**		
HOURS/WEEK:	**5**		

COMPUTER PROGRAMMING I **CSD100**

COURSE NAME **COURSE CODE**

TOTAL CREDITS: 4

PREREQUISITE(S): None

I. COURSE DESCRIPTION:

This course is intended to provide a firm foundation of computer programming skills needed in the computer studies area. It is the first of two courses that use the C/C++ programming language to develop the student's computer programming and problem solving skills.

II. TOPICS TO BE COVERED:

1. Introduction to computer programming concepts.
2. Basic C/C++ program structure.
3. Input/output in C/C++.
4. Decisions/Conditions in C/C++.
5. Repetition/Looping in C/C++.
6. Modularization using User-Defined Functions

III. LEARNING OUTCOMES AND ELEMENTS OF THE PERFORMANCE:

Upon successful completion of this course the student will demonstrate the ability to:

1. Discuss and apply the concepts involved in the development of software to solve problems using the computer. (chapter 1 and lecture notes)

   This learning outcome will comprise **15%** of the course.

   Elements of the performance:

   - define the concept of a "computer program/software"
   - differentiate between prewritten software and custom-designed software
   - differentiate between high level languages and machine language
   - describe the top-down process of developing a program
   - understand the "golden rule" for writing computer programs
   - describe the purpose of a compiler/interpreter
   - describe the process of transforming a source program to an executable module
   - differentiate between batch processing and online processing
   - write algorithms and describe them using pseudocode(and, to a lesser extent, flowcharts)

2. Write a simple C/C++ program applying the concepts of program structure, arithmetic, and assignment. (chapter 2, chapter 3: pgs. 95-96, 99-100, chapter 10: pgs. 523-524)

   This learning outcome will comprise **10%** of the course.

   Elements of the performance:

   - explain the main components of a C/C++ program
   - name and distinguish C/C++'s basic data types
   - explain and properly use the naming conventions for C/C++ identifiers
   - understand and apply simple output statements using the *cout* operator
   - differentiate between character, string, and numeric constants
   - differentiate between character and numeric variables
   - declare and initialize variables correctly
   - explain computer memory concepts and how they relate to processing data

Elements of the performance(cont'd):

- use assignment operators (=, +=, -=, *=, /=, ++, --) for character and numeric data
- use the *strcpy( )* function to assign string values to character variables
- use arithmetic operators and apply their precedence (+, -, *, /, %)
- evaluate integer and mixed-mode arithmetic correctly
- explain automatic promotion and apply typecasting to define data types
- differentiate between syntax and logic errors
- write and compile a simple program in C/C++ incorporating the concepts above

3. Develop algorithms and write C/C++ programs to solve problems involving the standard computer operations of input and output.
   (chapter 2 and 3)

   This learning outcome will comprise **10%** of the course.

   Elements of the performance:

   - apply the *cin* object to perform input of data
   - apply the *cout* object to perform output of data
   - apply the *getline( )* function to accept string values that include a space(s)
   - apply the *setw( ), setprecision( ), and setf( )* manipulators to format output on the screen
   - explain the purpose of "include" files for the *cin* and *cout* objects
   - write, test, and debug programs using the *cin* and *cout* objects

4. Develop algorithms and write C/C++ programs to solve problems involving the standard computer operations of decisions/conditions and selection.
   (chapter 4)

   This learning outcome will comprise **25%** of the course.

   Elements of the performance:

   - describe the use of the relational and logical operators, and use them to write both simple and complex logical expressions (==, !=, <, <=, >, >=, !, &&, ||)

Elements of the performance(cont'd):

- describe the operation of the following C/C++ decision-making structures and use them in C/C++ programs:

    a. *if...else*
    b. nested *ifs*
    c. *if...else if...else*
    d. the *switch* statement

- write algorithms to solve problems containing decision-making structures, and describe them using pseudocode(and, to a lesser extent, flowcharts)
- write, test, and debug programs containing selection structures

5. Develop algorithms and write C/C++ programs to solve problems involving the standard computer operations of looping and repetition.
(chapter 5)

This learning outcome will comprise **25%** of the course.

Elements of the performance:

- discuss the concept of repetition/looping in computer programs
- describe the operation of the following C/C++ repetition structures and use them in C/C++ programs:

    a. *while*
    b. *do...while*
    c. *for*
    d. nested loops
    e. *break* and *continue* statements

- write algorithms to solve problems containing repetition structures, and describe them using pseudocode(and, to a lesser extent, flowcharts)
- describe and correct an "infinite loop" problem
- write, test, and debug programs containing repetition structures

6. Discuss and create elementary user-written functions.
   (chapter 6: pgs. 295 - 392)

   This learning outcome will comprise **15%** of the course.

   Elements of the performance:

   - understand the role and operation of functions in C/C++ and other languages
   - distinguish between the *calling* and the *called* functions
   - understand the concept of *scope*
   - distinguish between *local* and *global* variables
   - write, test, and debug programs containing functions

IV. EVALUATION METHODS:

The mark for this course will be arrived at as follows:

Quizzes:	
outcome #1	10%
outcomes #2 & #3	15%
outcome #4	20%
outcome #5 & #6	30%
	75%
Assignments:	
outcome #1	5%
outcomes #2 & #3	5%
outcome #4	5%
outcomes #5 & #6	10%
	25%
Total	100%

The grading scheme used will be as follows:

A+	90 - 100%	Outstanding achievement
A	80 - 89%	Excellent achievement
B	70 - 79%	Average achievement
C	60 - 69%	Satisfactory achievement
R	< 60%	Repeat the course
X	Incomplete.	A temporary grade limited to special circumstances have prevented the student from completing objectives by the end of the semester. An X grade reverts to an R grade if not upgraded within a specified time.
NR	Grade not reported to Registrar's office. This is used to facilitate transcript preparation when for extenuating circumstances, it has not been possible for the faculty member to report grades.	

ELIGIBILITY FOR XGRADES/UPGRADING OF INCOMPLETES

When a student's course work is incomplete or final grade is below 60%, there is the possibility of upgrading to a pass when a student meets all of the following criteria:

1. The student's attendance has been satisfactory.
2. An overall average of at least 50% has been achieved.
3. The student has not had a failing grade in all of the theory tests taken.
4. The student has made reasonable efforts to participate in class and complete assignments.

Note: **The opportunity for an X grade is usually reserved for those with extenuating circumstances.** The nature of the upgrading requirements will be determined by the instructor and may involve one or more of the following: completion of existing labs and assignments, completion of additional assignments, re-testing on individual parts of the course or a comprehensive test on the entire course.

ASSIGNMENTS
Required format for lab assignments will be detailed by the instructor before labs are assigned.

ATTENDANCE:
Absenteeism will affect a student's ability to succeed in this course. Absences due to medical or other unavoidable circumstances should be discussed with the instructor. There will be an attendance factor included in the lab evaluation.

V. SPECIAL NOTES

1. In order to pass this course the student must obtain an overall quiz average of **60%** or better, as well as, an overall assignment average of **60%** or better. A student who is not present to write a particular quiz, and does not notify the instructor beforehand of their intended absence, may be subject to a zero grade on that quiz.

2. Assignments must be submitted by the due date according to the specifications of the instructor. Late assignments will normally be given a mark of zero. Late assignments will only be marked at the discretion of the instructor in cases where there were extenuating circumstances.

3. Any assignment submissions deemed to be copied will result in a **zero** grade being assigned to **all** students involved in that particular incident.

4.. The instructor reserves the right to modify the assessment process to meet any changing needs of the class. Consultation with the class will be done prior to any changes.

5. Students with special needs (eg. physical limitations, visual impairments, hearing impairments, learning disabilities) are encouraged to discuss required accommodations confidentially with the instructor and/or the Special Needs office. Visit Room E1204 or call extension 493, 717, or 491 so that support services can be arranged for you.

6. Your instructor reserves the right to modify the course as he/she deems necessary to meet the needs of students.

VI. PRIOR LEARNING ASSESSMENT:

Students who wish to apply for advanced credit in the course should consult the instructor.

VII. REQUIRED RESOURCES/TEXTS/MATERIALS

Text:	Starting Out With C++ by Tony Gaddis
Diskettes:	minimum of 3, 3 1/2"