SAULT COLLEGE OF APPLIED ARTS AND TECHNOLOGY

SAULT STE. MARIE, ONTARIO

COURSE OUTLINE

COURSE TITLE:	**INTRODUCTION TO PROGRAMMING**		
CODE NO. :	**CSD102**	**SEMESTER:**	**2**
PROGRAM:	**ALL INFORMATION TECHNOLOGY PROGRAMS**		
AUTHOR:	**Dennis Ochoski**		
DATE:	**Jan. 2014**	**PREVIOUS OUTLINE DATED:**	**Jan. 2013**
APPROVED:	"Colin Kirkwood"		**Dec. 2013**
	DEAN		**DATE**
TOTAL CREDITS:	**5**		
PREREQUISITE(S):	**None**		
HOURS/WEEK:	**5**		

I. COURSE DESCRIPTION:

The primary focus of this programming course is to develop the student`s logical problem-solving skills. At the same time, the student will learn the constructs inherent in all programming languages. To understand the program development process, the following concepts will be discussed: structured programming techniques, pseudocode, algorithm development, syntax, data types/variables, debugging, documentation, conditions, looping, user-defined functions, arrays, pointers, structures, file handling and an introduction to OOP using classes. Problem-solving skills are developed through programming assignments of increasing complexity.

II. TOPICS TO BE COVERED:

1. Introduction to logic and problem solving.
2. C/C++ program structures and format.
3. Decisions/Conditions in C/C++.
4. Repetition/Looping in C/C++.
5. Modularization using User-Defined Functions.
6. Advanced Concepts with User-Defined Functions.
7. Pointers.
8. The Debugger.
9. Arrays/Tables.
10. Files.
10. Advanced Concepts with Characters and Strings.
11. Data Structures.
12. Classes

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 a program to solve problems using the computer and write simple C/C++ programs applying the concepts of input/output, arithmetic, and assignment.

References at cplusplus.com:
Basics of C++: Structure of a program
Variables. Data Types.
Constants
Operators
Basic Input/Output

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

Elements of the performance:

- understand the importance of logic in developing a solution to a problem
- define the concept of a "computer program/software"
- differentiate between a high level language, compiler and machine language
- describe the top-down process of developing a logical solution to a problem and use pseudocode to plan a series of detailed steps leading to a solution
- understand the "golden rule" for writing computer programs
- demonstrate an understanding of the Microsoft Visual C++ environment
- explain the main components of a C/C++ program
- name and distinguish C/C++ basic data types
- explain and properly use the naming conventions for C/C++ identifiers
- 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
- use assignment operators (=, +=, -=, *=, /=) for assigning values/expression results to variables
- use increment/decrement operators (++, --) to increase/decrease values by 1
- use arithmetic operators and apply their precedence (+, -, *, /, %)
- evaluate integer and mixed-mode arithmetic correctly
- use various C++ math library functions to perform arithmetic calculations
- explain automatic promotion and apply typecasting to define data types
- describe the purpose of a compiler/interpreter
- describe the process of transforming a source program to an executable module

Elements of the performance(cont'd):

- differentiate between syntax and logic errors
- apply the *cin* object to perform input of data
- apply the *cout* object to perform output of data
- apply the cin.*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 and apply the #include directive
- explain the purpose of "include" files for the *cin* and *cout* objects
- write algorithms to solve problems using pseudocode
- write, test, and debug programs using the concepts above

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

References at cplusplus.com:
Control Structures: Control Structures (conditions)

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

Elements of the performance:

- describe and use the relational operators (==, !=, <, <=, >, >=)
- describe the use of the logical operators (&&, ||) and use them to write both simple and complex expressions
- describe the operation of the following C/C++ decision-making structures and use them in C/C++ programs:

    i. *if...else*
    ii. nested *ifs*
    iii. *if...else if...else*
    iv. the *switch* statement

- write algorithms to solve problems containing decision-making structures, and describe them using pseudocode
- write, test, and debug programs containing decision structures

3. Develop algorithms and write C/C++ programs to solve problems involving the standard computer operations of looping and repetition, and, debug program logic errors using the C++ Debugger.

References at cplusplus.com:
Control Structures: Control Structures (loops)

This learning outcome will comprise **10%** 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:

  i. *while*
  ii. *do...while*
  iii. *for*
  iv. nested loops

- use *break, continue,* and *exit* to terminate the iteration of a loop
- write algorithms to solve problems containing repetition structures, and describe them using pseudocode
- describe and correct an "infinite loop" problem
- execute code one line at a time using the Step Debugger
- use the following stepping options: **Go, Step Into, Step Over, Step Out, Watch,** and **Run to Cursor**
- define, as well as, insert and remove break
- write, test, and debug programs containing repetition structures

4. Discuss and create user-written, independently-compiled functions.

References at cplusplus.com:
Control Structures: Functions (I)
Functions (II)
Pointers

This learning outcome will comprise **20%** 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
- discuss and apply the concepts of 'passing' arguments to called functions by value
- discuss and apply the concept of 'returning' values to calling functions
- write, test, and debug programs containing functions
- discuss and apply the concept of pointers and pointer arithmetic
- discuss and apply the concept of pointers in C/C++
- define and apply the concepts of the following terms:

scope	calling vs called functions	function prototypes
local vs global variables	pass by value	return statement
class	pass by reference	overloaded functions
auto vs static variables	arguments/parameters	

- develop modularized, structured programs by creating user-written functions
- discuss and apply the concepts of 'passing' arguments to called functions by value
- discuss and apply the concept of 'returning' values to calling functions
- discuss and apply the concepts of 'passing' arguments to called functions by reference
- develop modularized, structured programs by creating user-written functions

5. Develop algorithms and write C++ programs to solve problems involving tables/arrays.

References at cplusplus.com:
Compound Data Types: Arrays

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

Elements of the performance:

- define and apply the concepts of the following terms:

one-dimensional array	index value	subscript
two-dimensional array	null character	

- discuss the purpose and concepts relating to one- and two-dimensional arrays
- declare and initialize both numeric and character arrays

- apply the concept of pointers to arrays
- access and process array elements
- pass arrays between functions
- write, test, and debug programs containing arrays

6. Develop algorithms to solve problems involving the use of file manipulation.

References at cplusplus.com:
C++ Standard Library: Input/Output with fi...

This learning outcome will comprise approximately **5%** of the course.

Elements of the performance:

- define and apply the concepts of the following terms:

  file open read close write append

- create a disk file
- write data to, and, read data from a disk file
- perform disk I/O with records
- create, and manipulate sequential and random access files
- write, test, and debug programs containing files

7. Discuss and apply the concepts of character sequences/arrays and string manipulation with reference to C/C++ library functions.

References at cplusplus.com:
Compound Data Types: Character Sequences

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

Elements of the performance:

- understand and utilize the C++ string class and its associated functions to declare string variables and manipulate string values
- discuss and apply character-based functions such as:

  cin.get( ) tolower( ) toupper( ) isalpha( )
  isdigit( ) isalnum( ) islower( ) isupper( )

- discuss and apply string functions such as:

str.append( ) str.compare( ) str.length( ) str.copy( )

- write, test, and debug programs containing character and string functions

8. Develop algorithms to solve problems involving the use of data structures.

References at cplusplus.com:
Compound Data Types: Data Structures

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

Elements of the performance:

- define and apply the concepts of the following terms:

  structure member record internal pointer

- discuss the concept of structures in C/C++
- declare and initialise a structure
- access and process structure members
- apply the use of arrays of structures
- apply methods of passing and returning structures to and from functions
- write, test, and debug programs containing structures

9. Introduce the concept of object-oriented programming using classes and objects by comparing with structures.

References at cplusplus.com:
Classes: Classes I
Classes II

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

Potential Elements of the Performance:

- Identify the most important features of Object-oriented programming languages.
- Assess the strengths and weaknesses of OOP and procedural programming.
- Define classes and implement class members and member functions.
- Compare classes to structures.
- Explain the relationship between class and object declarations.
- Develop and manipulate an array of classes.
- Use classes as parameters in function calls.
- Declare and define constructors and destructors for classes.
- Implement operator overloading.
- Use pointers to point to a class object
- Explain the use of inheritance in C++ programs.
- Derive new classes from base/parent classes.
- Write and debug programs utilizing the components above.

IV. REQUIRED RESOURCES/TEXTS/MATERIALS

Internet Link: http://www.cplusplus.com/doc/tutorial/introduction/

Visual C++ 2010 Express Edition Software:
http://www.microsoft.com/express/vc/#webInstall

V. EVALUATION PROCESS/GRADING SYSTEM:

Evaluation Methods	Weight
Tests	70%
Assignments	30%
	100%

The following semester grades will be assigned to students in postsecondary courses:

Grade Point

Grade	Definition	Equivalent
A+	90 – 100%	4.00
A	80 - 89%	4.00
B	70 - 79%	3.00
C	60 - 69%	2.00
D	50 – 59%	1.00
F(Fail)	below 50%	0.00

CR (Credit)	Credit for diploma requirements has been awarded.
S	Satisfactory achievement in field/clinical placement or non-graded subject area.
U	Unsatisfactory achievement in field/clinical placement or non-graded subject area.
X	A temporary grade limited to situations with extenuating circumstances giving a student additional time to complete the requirements for a course.
NR	Grade not reported to Registrar's office.
W	Student has withdrawn from the course without academic penalty.

VI. OTHER EVALUATION CONSIDERATIONS

1. In order to pass this course the student must obtain an overall test/quiz average of **50%** or better, as well as, an overall assignment average of **50%** or better. A student who is not present to write a particular test/quiz, and does not notify the professor beforehand of their intended absence, may be subject to a zero grade on that test/quiz.

2. There will be **no** supplemental or make-up quizzes/tests in this course unless there are extenuating circumstances.

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

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

5. It is the responsibility of the student to ask the professor to clarify any assignment requirements.

6. The professor reserves the right to modify the assessment process to meet any changing needs of the class.

VII. SPECIAL NOTES:

Attendance:
Sault College is committed to student success. There is a direct correlation between academic performance and class attendance; therefore, for the benefit of all its constituents, all students are encouraged to attend all of their scheduled learning and evaluation sessions. This implies arriving on time and remaining for the duration of the scheduled session. *It is the departmental policy that once the classroom door has been closed, the learning process has begun. Late arrivers may not be granted admission to the room.*

Absences due to medical or other unavoidable circumstances should be discussed with the professor, otherwise a penalty may be assessed. The penalty depends on course hours and will be applied as follows:

Course Hours	**Deduction**
5 hrs/week (75 hrs)	1.0% /hr
4 hrs/week (60 hrs)	1.5% /hr
3 hrs/week (45 hrs)	2.0% /hr
2 hrs/week (30 hrs)	3.0% /hr

Final penalties will be reviewed and assessed at the discretion of the professor.

VIII. COURSE OUTLINE ADDENDUM:

This document (**CourseOutlineAddendum.docx**) can be found along with the course outline on ***Desire2Learn (D2L***).