SAULT COLLEGE OF APPLIED ARTS & TECHNOLOGY

SAULT STE. MARIE, ONTARIO

COURSE OUTLINE

COURSE TITLE: PROGRAMMING LANGUAGES

CODE NO.: EDP318-6 SEMESTER: TWO

PROGRAM: COMPUTER PROGRAMMER

AUTHOR: DENNIS OCHOSKI

DATE: JANUARY 1994

PREVIOUS OUTLINE DATED: JANUARY, 1993

New: ________ Revision: X

APPROVED: ______________________
DEAN, SCHOOL OF BUSINESS & HOSPITALITY

94/01/05
DATE

PROGRAMMING LANGUAGES	EDP318
COURSE NAME	**COURSE CODE**

LENGTH OF COURSE: Six periods per week for one semester

TEXTS: QBasic, by Susan K. Baumann and Steven L. Mandell
C++ By Example, by Greg Perry

PURPOSE: This course will provide students with an opportunity to develop their data processing skills by introducing them to program development and structured programming techniques. This exposure will provide a foundation for more advanced study in the following semesters.

The course involves theory related to programming and application of this theory will be implemented using two high level programming languages.

COURSE OBJECTIVES:

1. learn problem-solving techniques and apply them to programming
2. learn and understand the program development process
3. develop logic skills
4. develop solutions using the QBasic programming language
5. develop solutions using the C++ programming language

PART 'A': The following modules pertain to program development and structured programming techniques.
(chapter 1 and lecture notes)

Module 1:

Objectives:

When this module is completed, the student should be able to:

1) differentiate between high level languages and machine language

2) understand the basic make-up of a computer program and state its purpose

3) explain why there is a need for custom-designed programs

4) list the steps involved in the program development process

5) understand the main problem of computer programming addressed by structured programming

6) state the meanings of program "reliability" and "readability"

7) explain how a hierarchy chart relates to problem-solving

8) state the purpose of pseudo code and why it is useful

9) state the difference between an interpreter and a compiler

10) differentiate between syntax errors and logic errors

11) differentiate between a source program and an object program (compiling, linking, and running a program)

12) explain the importance of program documentation

PART 'B': The following modules pertain specifically to the QBasic programming language and expand on the concepts of the language discussed in the first semester.

Module 1: **Review of the QBasic Programming Language** (chapters 3, 4, 5)

Objectives:

When this module is completed, the student should be able to recall:

1. QBasic coding conventions
2. the use of data types (numeric, alphanumeric)
3. the use of arithmetic expressions
4. input and output statements
5. the concept of variables and constants
6. the use of conditional statements

Module 2: **More Advanced Conditional Statements and Program Looping** (chapters 5,7)

Objectives:

When this module is completed, the student should be able to:

1. implement conditional statements such as the IF and IF-THEN-ELSE and CASE statements

Module 3: (cont'd)

2. use logical operators (.AND., .OR., .NOT.) and relational operators (.LE., .GT., etc.)

3. apply the concept of looping using the various forms of looping structures

Module 3: Functions and Modular Programming
(chapters 6 and 8)

Objectives:

When this module is completed, the student should be able to:

1. apply the use of sub-procedures

2. understand how data is passed between a main program and a procedure

3. pass arguments to sub-procedures

4. use structure charts to modularize programs

5. use standard numeric and string functions

MODULE 4: Records and Files
(chapter 10)

Objectives:

When this module is completed, the student should be able to:

1. define records and use them in programs

2. explain the relationship between files, records, and fields

3. create, read from and write to a sequential file

4. create, access, and write to random files

MODULE 5: Arrays
(chapter 9)

Objectives:

When this module is completed, the student should be able to:

1. explain and use subscripts

2. use the DIM statement to define arrays

3. manipulate one- and two-dimensional arrays

PART 'C' The following modules pertain specifically to the C++ programming language

Module 1: Characteristics of the C++ Programming Language (chapters 3,4,5,6,7,8)

Objectives:

When this module is completed, the student should be able to:

1. discuss C++ coding conventions

2. illustrate the use of data types

3. use arithmetic expressions and functions

4. use input/output statements to access and store data

5. understand the use of variables and literals

Module 2: Conditional Statements and Program Looping (chapters 9, 10, 11, 12, 13, 14, 15)

Objectives:

When this module is completed, the student should be able to:

1. implement conditional statements such as the IF, IF-ELSE, and SWITCH statements

2. use logical operators (SS, ::. !) and relational operators (<, >, >=, <=, !=)

3. apply the concept of looping using the WHILE, DO...WHILE, and FOR... statements

Module 3: Functions and Modular Programming
(chapters 16, 17, 18, 19, 20, 21, 22)

Objectives:

When this module is completed, the student should be able to:

1. apply the use of Functions

2. understand how data is passed between a main program and a Function

3. pass arguments to FUNCTIONS

4. use standard numeric and string functions

MODULE 4: Arrays and Pointers
(chapters 23, 24, 25, 26, 27)

Objectives:

When this module is completed, the student should be able to:

1. use subscripts and declare arrays

2. search and manipulate one- and two-dimensional arrays

3. understand pointer variables and use them to reference other variables

MODULE 5: Structures and Files
(chapters 28, 29, 30)

Objectives:

When this module is completed, the student should be able to:

1. define structures and use them in programs

2. combine the concepts of structures and arrays

3. create, read from and write to a sequential file

4. create, access, and write to a random file

STUDENT EVALUATION:

The student's final grade will consist of the following components:

Tests	-	QBasic	(2 x 15%)	30%
	-	C++	(2 x 15%)	30%
Assignments	-	QBasic	(4 x 5%)	20%
	-	C++	(4 x 5%)	20%
				100%

A+	90-100%
A	80- 89%
B	70- 79%
C	60- 69%
R	REPEAT - under 60%

Assignment Deadlines:

Assignments must be handed in **ON TIME**, otherwise they are subject to a 10% deduction per day late.

NOTE: There will be **no rewrite/supplemental tests** in this course due to poor performance.