SAULT COLLEGE OF APPLIED ARTS & TECHNOLOGY
SAULT STE. MARIE, ONTARIO

COURSE OUTLINE

COURSE TITLE:	ADVANCED APPLICATIONS PROGRAMMING
CODE NO.:	EDP229-6 **SEMESTER:** FOUR
PROGRAM:	COMPUTER PROGRAMMER
AUTHOR:	DENNIS OCHOSKI
DATE:	JANUARY 1996
PREVIOUS OUTLINE DATED:	SEPTEMBER 1995

New:_____ Revision: x

APPROVED: _____ 96 01 08
DEAN, SCHOOL OF BUSINESS & HOSPITALITY **DATE**

ADVANCED PROGRAMMING LANGUAGES — COURSE NAME

EDP229 — COURSE CODE

Length of Course: 5 periods per week for one semester

Required Resources:

Text: A First Book of ANSI C: Fundamentals of C Programming, by Gary Bronson and Stephen Menconi, West Publishing

Disks: 3, 3 1/2" floppy diskettes

I. Philosophy/Goals:

This course will provide students with an opportunity to develop their programming skills using a "leading-edge" language, C. C has emerged as a dominant programming language. The course will re-emphasize the use of structured programming techniques, proper program design, and problem solving techniques.

II. Student Performance Objectives(Outcomes):

Upon successful completion of this course the student will be able to:

1. Discuss the concepts involved in the development of software to solve problems using the computer.
2. Develop algorithms to solve problems involving the standard computer operations of input/out, assignment, selection and repetition, and describe those algorithms using pseudocode.
3. Describe the structure of C programs, and the data and code elements of the language.
4. Describe the process of using the Borland C++ environment, and use that environment to create, test and debug programs.

ADVANCED PROGRAMMING LANGUAGES — COURSE NAME

EDP229 — COURSE CODE

5. Discuss the concepts of complex data, including scope and class of data, arrays, strings, structures, union, data files and use of pointers.

6. Develop algorithms to solve problems using complex data, macros, and user-written functions, and describe those algorithms using pseudocode.

7. Discuss the use of include files, object libraries, and the linker, and be able to create programs using user-written, separately-compiled functions.

8. Write programs using the concepts and programming techniques covered in the course.

III. Topics to be Covered:

1. Introduction to basic C programming.

2. Making decisions in C.

3. Repetition in C.

4. Writing your own functions.

5. Arrays in C.

6. Using pointers and strings.

7. The use of complex data and files.

IV. Learning Activities

Module 1: Basic C Programming

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

1. Define or describe the meaning of the following terms:

   algorithm, pseudocode, machine language, source code, interpreter, compiler, object file, high level language, named constant, symbolic constant, preprocessor.

2. Describe the top-down process of developing a program in the Borland C++ environment.

3. Write algorithms and describe them using pseudocode.

4. Discuss the general characteristics of data as having name, type and value, and relate it to the use of integer, floating point and character data in C.

5. Discuss the arithmetic operators in C, including their precedence and associativity, and be able to write correct arithmetic expressions.

6. Describe the operation of the assignment operators in C and be able to use them. (=, +=, -=, *=, /=, %=, ++, --)

7. Describe and use the scanf and printf functions to perform input and output in C.

8. Write, test and debug programs with the general form of input-process-output using the C language features of chapters 1 through 3 of the text.

Module 2: Decision Making in C

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

1. Define or describe the meaning of the following terms:

   compound statement, syntax error, logical error, compile time, run time.

2. Describe the use of relational and logical operators, and use them to write complex logical expressions. (==, !=, <, <=, >, >=, !, &&, ||)

3. Describe the operation of the following C decision-making structures, and be able to use them in C programs.

   a. *If...else* structures.
   b. Nested *if*s.
   c. *If...else if...else if...else* structures.
   d. the *switch* statement

4. Write algorithms to solve problems containing decision-making structures, and describe them using pseudocode.

5. Write, test and debug programs containing selection structures.

Module 3: Repetition

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

1. Define or describe the meaning of the following terms:

   infinite loop, sentinel, end of file, null statement, initialization, validity check

2. Discuss the concept of repetition in computer programs.

3. Describe the operation of the following C statements and structures, and be able to use them in C programs.

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

4. Write algorithms to solve problems containing repetition structures, and describe them using pseudocode.

5. Write, test and debug programs containing repetition structures.

ADVANCED PROGRAMMING LANGUAGES — COURSE NAME

EDP229 — COURSE CODE

Module 4: Modularization in C

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

1. Define or describe the meaning of the following terms:

   calling function, function header, parameters, formal arguments, function prototype, actual arguments, global variable, local variable, auto class, static class, pass by value, pass by reference.

2. Describe the process of writing and declaring functions to modularize programs, and write programs containing user-written functions.

3. Discuss the general use of libraries, and the input/output, math and string libraries that are included in C.

4. Discuss the concepts of scope and class of variables, and how they are used.

5. Discuss the concepts of passing arguments to functions by value and by reference.

6. Write, test and debug programs that use user-written functions, include files, and object libraries.

Module 5: Arrays, Pointers, and Strings

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

1. Define or describe the meaning of the following terms:

   one-dimensional array, subscript, index value, null character, two-dimensional array, offset, pointer constant.

2. Discuss the concept of one- and two-dimensional arrays, and describe how they are declared, initialized and passed to functions.

3. Discuss the use of pointers, pointer arithmetic, and be able to use pointers with arrays.

4. Discuss the concepts of strings in C, and how they are implemented.

5. Discuss the use of the following string functions: *gets*, *puts*, *strcopy*, *strcat*, *strchr*, *strcmp*, *strlen*, and *strtok*.

6. Write, test and debug programs that use the concepts mentioned above.

Module 6: Data Structures and Files

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

1. Define or describe the meaning of the following terms:

   structure, member, content, populating the structure, record, union, open, close, append, internal pointer, *stdin*, *stdout*, *stderr*.

2. Discuss the concept of structures in C, and describe how they are used in programs, including arrays of structures, and the methods of passing and returning structures from functions.

3. Discuss basic file concepts, including opening and closing files, and reading, writing, appending and updating data in files.

4. Discuss the concepts of standard device files.

5. Discuss the use of the following C functions: *fopen*, *fclose*, *fputc*, *fprintf*, *fgetc*, *fgets*, *fscanf*.

6. Write, test and debug programs that use the concepts mentioned above.

ADVANCED PROGRAMMING LANGUAGES — COURSE NAME

EDP229 — COURSE CODE

V. Student Evaluation:

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

Quizzes(6 @ 12%) - one per module	72%
Assignments & Practical Work	28%
	100%

Grading:

A+	90 - 100%	
A	80 - 89%	
B	70 - 79%	
C	60 - 69%	
R	UNDER 60%	- Repeat Course

VI. Special Notes:

1. In order to pass this course the student must obtain an average of 60% or better on the **quiz** portion of the Student Evaluation.

2. Students are advised to maintain a copy of all files on a backup disk. Loss of an assignment due to a lost or damaged disk is not an acceptable reason for a late or incomplete assignment.

3. Students with special needs, due to such things as physical limitations, visual and/or hearing impairments, or learning disabilities, are encouraged to discuss required accommodations, confidentially, with the instructor.

4. There will be **no re-writes** in this course except in situations out of the control of the student (such as illness, urgent family matters, etc.) in which a re-write may be issued at the discretion of the instructor.

5. Assignments received after the due date are subject to grade of zero except in situations as specified in #3 above.

6. The instructor reserves the right to modify the course material and/or the assessment process to meet any changing needs of the class. Consultation with the class will be done prior to any changes.