SAULT COLLEGE OF APPLIED ARTS AND TECHNOLOGY
SAULT STE. MARIE, ON

COURSE OUTLINE

COURSE TITLE: COMPUTER PROGRAMMING 1

CODE NO.:CSD100 **SEMESTER:FALL 95**

PROGRAM: COMPUTER ENGINEERING TECHNOLOGY
COMPUTER PROGRAMMER

AUTHOR: GERRY DAVIES/DENNIS OCHOSKI

DATE:JUNE 1995_______ PREVIOUS OUTLINE DATED:_______________

APPROVED: [illegible] 95-07-17
DEAN **DATE**

CSD 100 - COMPUTER PROGRAMMING 1

TOTAL CREDITS 4

PREREQUISITE(S):NONE

I. PHILOSOPHY/GOALS: 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 programming language to develop the students computer programming and problem solving skills.

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/output, assignment, selection and repetition, and describe those algorithms using pseudocode and flowcharts.
3. Describe the structure of C programs, and the data and code elements of the language covered in chapters 1 through 6 of the text.
4. Describe the process of using the Borland C^{++} environment, and use that environment to create, test and debug programs.
5. Write C programs to solve problems using programming techniques covered in chapters 1 through 6 of the text, at a level of complexity similar to the assigned problems.

III. TOPICS TO BE COVERED:

1. Introduction to computer programming concepts.
2. Basic C programming.
3. Making decisions in C.
4. Repetition in C.
5. Writing your own functions.

IV. LEARNING ACTIVITIES

BLOCK 1 Basic C Programming

At the end of this block, the student will be able to:

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

   algorithm, pseudocode, flowchart, machine language, source code, interpreter, compiler, object file, modularity, structure diagram, high level language, buffer, address, address operator, named constant, symbolic constant, preprocessor.

2. Describe the top-down process of developing a program, and how it is accomplished in the Borland C^{++} environment.

3. Write algorithms and describe them using pseudocode and flowcharts.

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 and bitwise logical operators in C, including their precedence and associativity, and be able to write correct arithmetic and logical expressions. (+, -, *, /, %, &, |, ~, ^)

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

7. Describe how the scanf and printf functions can be used to perform input and output of C variables.

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.

BLOCK 2 Decision Making in C

At the end of this block the student shall 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 the 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 ifs.
   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 and flowcharts.

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

BLOCK 3 Repetition and Modularization

At the end of this block the student shall be able to:

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

   infinite loop, sentinal, end of file, null statement, initialization, validity check, calling function, function header, parameters, formal arguments, function prototype, actual arguments,

2. Discuss the concept of repetition in computer programs.

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

   a. The while statement
   b. The for statement
   c. The do statement
   d. The break and continue statements.
   e. Nested loops.

4. Write algorithms to solve problems that include repetition structures, and describe those algorithms using pseudocode and flowcharts.

5. Write test and debug programs containing repetition structures.

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

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

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

9. Be able to write functions that accept, store and use addresses.

V. EVALUATION METHODS:

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

Tests	45%	3 @ 15%
Quizzes	25%	5 @ 5%
Assignments and practical work	30%	
total	100%	

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.

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.

The grading scheme used will be as follows:

A+	90 - 100%	Outstanding achievement
A	80 - 89%	Excellent achievement
B	70 - 79%	Average achievement
C	56 - 69%	Satisfactory achievement
R	Repeat	
X	Incomplete. A temporary grade that is limited to instances where special circumstances have prevented the student from completing objectives by the end of the semester. an X grade must be authorized by the Dean, and reverts to an R grade if not upgraded within a specified time.	

The method of upgrading an incomplete grade is at the discretion of the instructor, and may consist of such things as make-up work, rewriting tests, and comprehensive examinations.

VI. PRIOR LEARNING ASSESSMENT:

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

VII. REQUIRED STUDENT RESOURCES

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

VIII. SPECIAL NOTES

Students with special needs (eg. physical limitations, visual impairments, hearing impairments, learning disabilities) are encouraged to discuss required accommodations confidentially with the instructor.

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