Revised 4/2020

NOVA COLLEGE-WIDE COURSE CONTENT SUMMARY
ITP 130 – C PROGRAMMING I (4 CR.)

Course Description
Stresses instruction in fundamentals of structured programming using C. Course content emphasizes program construction, algorithm development, coding, debugging, and documentation of console applications. Lecture 4 hours per week

General Course Purpose
This course provides a comprehensive foundation sufficient for a student to write C programs from scratch in order to meet the minimum programming goals of students who which to learn structured programming techniques.

Course Prerequisites/Corequisites
None

Course Objectives
Upon completion of this course, the student will be able to:

- Design, develop, code, and test simple C Programs
- Define primitive data types and use flow control statements
- Demonstrate the basics of structural coding techniques

Major Topics to be Included
- Basic Program Structure and Flow
- Structures, Arrays and Pointers
- File Input and Output
- Primitive data types and operators
- Functions and Parameter Passing
- Strings
- Memory Allocation

Student Learning Outcomes

Basic Program Structure
- Describe the sequential processing of programs by the computer
- Describe and code the main function
- Send output to the monitor
- Describe and define variables, data types, and character arrays (to store string data) in the program
- Receive input from the keyboard
- Describe and define C operators and precedence and how the computer evaluates expressions in program statements

Basic Program Structure and Flow
- Write statements to declare and initialize integer, long, float, double, char, and Boolean variables
- Write statements to assign values to variables using expressions containing mathematical and Boolean operators
- Describe the sequential processing of programs by the computer
- Describe and code the main function
- Write code to send output to the monitor
- Describe and define variables, data types, and character arrays (to store string data) in the program
- Write code to receive input from the keyboard
- Describe and define C operators and precedence and how the computer evaluates expressions in program statements
- Describe and define decision logic (IF ELSE) to determine the next programming statement
- Describe and define looping logic (DO WHILE, WHILE, FOR) to perform repetitive programming statements

- Write functions that return values
- Describe and define the scope and duration of variables

Structures, Arrays and Pointers

- Write code that uses arrays of structures
- Write code to define and use multidimensional arrays defining, assigning values, and referencing individual elements.
- Describe the difference between pointer constants and pointer variables.
- Use pointer arithmetic and pointer arrays to reference multidimensional arrays.
- Use structures as programmer defined data types (Records).
- Write code to define and assign values to structure variables.
- Write statements to declare, create, and initialize a one dimensional array
- Describe the memory allocation for a one dimensional array including both the array reference and the memory for the array elements
- Write statements to fill an array with values
- Write statements to print out the contents of an array
- Write statements to copy one array to another array
- Write statements to access every element in an array for analysis for the data
- Describe the concept of parallel arrays
- Write statements to declare, create, and initialize multi-dimensional arrays
- Write code to pass arrays to functions

File Input and Output

- Write statements to open a text file for reading
- Write statements to read data from a text file
- Write statements to close a text file
- Write statements to open a text file for writing
- Write statements to write data to a text file

Primitive data types and operators

- Write statements to create arithmetic expressions following order of precedence rules
- Write statements using Boolean operators: and (&), or (|), not (!), exclusive or (^)
- Write statements using the relational operators to include greater than, greater than or equal to, less than, less than or equal to, is equal to, and not equal to
- Write statements using the arithmetic operators to include add, subtract, multiply, divide, and modulus
- Write statements using the increment and decrement operators
- Write statements using pre and post increment and decrement operators
- Write statements using the short cut arithmetic assignment operators to include post and pre increments and decrements.
- Write statements incorporating cast operators for primitive data types in arithmetic expressions appropriately
- Write statements to declare named constants
- Write statements to complete console input and output using scanf and printf
- Write statements to validate input of data from the keyboard

Functions and parameter passing

- Describe the difference between value and variable parameters
- Write statements to pass simple variable types
- Write statements to pass pointers and structures into functions.
- Write statements to pass information back from a function
- Describe how to work with both global and local variables

Strings

- Describe the difference between length indicated and null terminated strings
- Write code to string to search for a substring
- Write code to string to replace a substring
- Describe how to change the case of a string
- Write code to read and write string variable to and from text files.

Memory Allocation

- Describe the difference between static and dynamic memory allocation.
- Write code to dynamically allocate and free memory for structures

Required Time Allocation Per Topic

In order to standardize the core topics of ITP 130 so that a course taught at one campus is equivalent to the same course taught at another campus, the following student contact hours per topic are required. Each syllabus should be created to adhere as closely as possible to these allocations. Of course, the topics cannot be followed sequentially. Many topics are taught best as an integrated whole, often revisiting the topic several times, each time at a higher level. There are normally 60 student-contact-hours per semester for a four credit course. (This includes 15 weeks of instruction and does not include the final exam week so 15* 4 = 60 hours. Sections of the course that are given in alternative formats from the standard 16 week section still meet for the same number of contact hours.)

Topic	Hours	Percentage
Basic Program Structure and Flow	10	16.7%
Structures, Arrays and Pointers	12	20.0%
File Input and Output	6	10.0%
Primitive data types and operators	8	13.3%
Functions and Parameter Passing	8	13.3%
Strings	4	6.7%
Memory Allocation	4	6.7%
Exams	8	13.3%
Total	60	100.0%