Revised 8/2023

NOVA COLLEGE-WIDE COURSE CONTENT SUMMARY
ITP 132 – C + + Programming I (4 CR.)

Course Description

Centers instruction in fundamentals of object-oriented programming and design using C++. Emphasizes program construction, algorithm development, coding, debugging, and documentation of C++ 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 plan to transfer and students who take the course for employment purposes.

Course Prerequisites/Corequisites

None.

Course Objectives

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

a) Design, develop, code, test and document C++ programs
b) Use primitive data types and flow control statements that are the building blocks of all programming
c) Use the foundation knowledge of object oriented coding techniques to create classes that are applied appropriately to a C++ program as a solution to a specific problem statement

Major Topics to be Included

a) Programming Basics Using C++
b) Arithmetic Operators and Control Structures
c) Arrays and Pointers
d) C++ Functions
e) Classes
f) Class Features and Design
g) Overloading Operators
h) Inheritance

Student Learning Outcomes

Programming Basics Using C++

a) Write C++ statements using the basic control structures
b) Write C++ statements to declare and initialize integer, long, float, double, char and Boolean variables
c) Write C++ statements to declare named constants
d) Write C++ statements using declarations, integer qualifiers, data structures and assignment statements
e) Be able to write C++ programs combining the statements describe in this document
f) Be able to write, compile, test, debug, and run C++ programs

Input/Ouput

a) Write C++ statements to open a text file for reading
b) Write C++ statements to read data from a text file
c) Write C++ statements to close a text file
d) Write C++ statements to open a text file for writing
e) Write C++ statements to write data to a text file

f) Write C++ statements to using exceptions as required for file input and output

Arithmetic Operators and Control Structures

a) Write C++ statements to assign values to variables using expressions containing mathematical and Boolean operators
b) Write C++ statements to create arithmetic expressions following order of precedence rules
c) Write C++ statements using Boolean operators: and, or, not and exclusive or
d) Write C++ statements that use shortcut arithmetic operators
e) Write C++ statements using the logical OR, the while Loop and the for statement
f) Write C++ statements using the relational operators to include greater than, greater than or equal to , less than, less than equal to, is equal to, and not equal to
g) Write C++ statements using the arithmetic operators to include add, subtract, multiply, divide and modulus
h) Write C++ statements using the increment and decrement operators
i) Write C++ statements using the pre and post increment and decrement operators
j) Write C++ statements using the shortcut arithmetic assignment operators to include +=, -=, *=, /=, and %=

Arrays and Pointers

a) Write C++ statements to declare, create, and initialize a one dimensional array
b) Describe the memory allocation for a one dimensional array including both the array reference and the memory for the array elements.
c) Write C++ statements to fill an array with values
d) Write C++ statements to print out the contents of an array
e) Write C++ statements that use the address operator
f) Write C++ statements to copy one array to another array
g) Write C++ statements to access every element in an array for analysis for the data
h) Describe the concept of parallel arrays and their relationship to classes and objects
i) Write C++ statements to declare, create and initialize multi-dimensional arrays
j) Write C++ statements that pass arrays to methods

C++ Functions

a) Write C++ statements that use functions and include files
b) Write C++ statements that use procedural abstraction
c) Describe the concept of scope
d) Write C++ statements that construct function headers and prototypes
e) Write C++ statements that pass values and addresses to functions
f) Write C++ statements that pass arrays to functions
g) Write C++ statements that include user-defined functions

Classes

a) Be able to define the relationship between a class and an object
b) Write C++ statements to create a class that includes attributes and methods
c) Write C++ statements to create objects
d) Be able to write visibility modifiers to include: public, private, protected, and no modifier
e) Write C++ statements to create and use static attributes
f) Write C++ statements to create and use constant attributes
g) Write C++ statements to create classes with declaration and implementation sections
h) Write C++ statements to encapsulate class components
i) Write C++ statement that use classes
j) Write C++ statements to use private and public functions
k) Write C++ statements that pass objects to methods
l) Write C++ statements that include objects as attributes
m) Write C++ statements that return an object from a method
n) Use classes and objects in programs that you write from scratch

Class Features and Design

a) Be able to define member functions
b) Be able to define the purpose of a constructors

c) Be able to define the purpose of a destructors
d) Write C++ statements to create and use constructors

Overloading Functions

a) Be able to list the rules for overloading functions
b) Write C++ statements that overload math functions, input and output
c) Write C++ statements to create and use an overloaded constructor

Inheritance

a) Be able to use inheritance in programs and classes
b) Be able to describe the relationship between a super class and a sub class.
c) Be able to recognize a sub class from the extends label in a class header
d) Be able to describe the purpose of the super statements when using inheritance.
e) Be able to write C++ statements from simple classes using inheritance

Required Time Allocation per Topic

In order to standardize the core topics of ITP 132 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.) The final exam time is not included in the time table. The category, Other Optional Content, leaves ample time for an instructor to tailor the course to special needs or resources

Topic	Hours	Percentage
Programming Basics Using C++	7	12.5%
Arithmetic Operators and Control Structures	5	8.3%
Arrays and Pointers	8	13.9%
C++ Functions	8	13.9%
Classes	5	8.3%
Class Features and Design	8	13.9%
Overloading Function	8	13.9%
Inheritance	5	8.3%
Testing to include quizzes, tests, exams	4	7%
Total	60	100%