Introduction to Programming in C++

Course Text
- Course book provided along with the course.

Course Description
This course introduces programming in C++ and teaches the core computer science concepts of variables, branching, loops, vectors, and functions. Introduces object oriented programming with classes and inheritance. Covers use of pointers and streams. Teaches a variety of good coding practices, including iterative development, code formatting, and identifier naming schemes.

Course Objectives
After completing this course, students will be able to:
- Explain the basics of how a computer works, then write a first program
- Use standard input and output, and understand common syntax errors
- Declare and initialize variables with valid identifiers
- Use a variety of integer data types and the concept of overflow. Use floating-point, character, and string data types
- Convert decimal to binary and vice-versa
- Cast between data types
- Develop programs that branch based on user input
- Write expressions with relational and equality operators and develop expressions with logical operators
- Use multiple branches for more complex programs
- Use the Boolean data type to store results of conditional statements
- Access elements in arrays and vectors
- Understand memory usage for single and multi-dimensional arrays
- Combine loops and vectors, and develop programs with multiple vectors
- Resize a vector using user-input
- Write a C string as null-terminated string and manipulate C strings with common functions
- Understand common errors between C string and strings. Use a library for char data-type analysis
- Write a first function, then return from a function and parameterize a function
- Learn reasons to use a function
- Combine functions with branches and loops and use incremental development with functions
- Know how to pass-by-reference to functions and to overload user-defined functions
- Send and use arguments passed to program
- Write makefile for modular compilation
- Write structs for grouping data, combine structs and functions, then combine structs and vectors
- Write classes that group functions and data
- Initialize class variables with class constructors and overload class constructor
• Introduce abstract data types
• Explain the reasons for using pointers
• Define a variable that points to a memory location, allocate and deallocate memory
• Use different regions of program memory
• Identify and fix memory leaks
• Use class destructors, copy constructors, and copy assignment operator
• Understand standard input and output implementation
• Manipulate text and floating-points during output
• Read input from a string with string streams
• Input and output to a file
• Derive a class from another class and choose the appropriate access specifier
• Override member function of inherited class
• Introduce polymorphism during runtime

Course Prerequisites
StraighterLine suggests, though does not require, that students take Pre-Calculus or its equivalent before enrolling in this course.

Important Terms
In this course, different terms are used to designate tasks:

• **Practice exercise**: A non-graded assignment to assist you in practicing the skills discussed in a topic.
• **Programming assignment**: A graded assignment in which the student develops a full-program. Grading may take 3-5 business days.
• **Quiz**: A graded online test.
• **Final**: A graded and proctored online test. Fully automated grading, except for 4 free response programming questions. Grading may take 3-5 business days.

Course Evaluation Criteria

StraighterLine provides a percentage score and letter grade for each course. See Academic Questions section in FAQ for further details on percentage scores and grading scale. A passing percentage is **70%** or higher.

If you have chosen a Partner College to award credit for this course, your final grade will be based upon that college’s grading scale. Only passing scores will be considered by Partner Colleges for an award of credit.

There are a **total of 1000 points** in the course:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Assessment</th>
<th>Points Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Programming assignment 1: Basic output - ASCII Art</td>
<td>25</td>
</tr>
<tr>
<td>#</td>
<td>Assignment/Quiz</td>
<td>Points</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>2</td>
<td>Programming assignment 2: Variables - Painting a wall</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>Graded Quiz 1</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>Programming assignment 3: Branching - Tweet decoder</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Programming assignment 4: Loops - Drawing a tree</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Graded Quiz 2</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>Programming assignment 5: Vectors - Decrypting a secret message</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>Midterm</td>
<td>200</td>
</tr>
<tr>
<td>7</td>
<td>Programming assignment 6: Functions - Upset fowl</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>Graded Quiz 3</td>
<td>50</td>
</tr>
<tr>
<td>9</td>
<td>Programming assignment 7: Classes/Streams - Marketing software</td>
<td>25</td>
</tr>
<tr>
<td>9</td>
<td>Graded Quiz 4</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>Programming assignment 8: Pointers/Inheritance - Inventory system</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>Final</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td><strong>1000</strong></td>
</tr>
</tbody>
</table>

### Course Topics and Objectives

<table>
<thead>
<tr>
<th>Topic</th>
<th>Lesson</th>
<th>Subtopics</th>
<th>Objectives</th>
</tr>
</thead>
</table>
| 1     | Basic input and output| ● Computers and programs                   | ● Learn basics of how a computer works  
|       |                       | ● Basic output                               | ● Write first program  
|       |                       | ● Basic input                                | ● Use standard input and output  
|       |                       | ● Comments                                    | ● Interpret common syntax errors  |
| 2     | Variables             | ● Variable assignment and initialization      | ● Declare variables with valid identifiers  |

© StraighterLine
CS101: Introduction to Programming in C++
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
|  | • Common data types and type casting  
|  | • Math functions and random numbers  
|  | • Learn variety of integer data types and concept of overflow  
|  | • Use floating-point, character, and string data types  
|  | • Convert decimal to binary and vice-versa  
|  | • Cast between data types  
|  | • Initialize variables and use const keyword  |
| 3 | Branching | • If-else tree  
|  |  | • Multiple branches  
|  |  | • Relational and logical operations  
|  |  | • Develop programs that branch based on user input  
|  |  | • Write expressions with relational and equality operators  
|  |  | • Multiple branching for more complex programs  
|  |  | • Develop expressions with logical operators  
|  |  | • Use Boolean data type to store result of conditional statements  |
| 4 | Loops | • While loops and counting  
|  |  | • For-loops and nested loops  
|  |  | • Flow control  
|  |  | • Use while loop to repeat code execution  
|  |  | • Count using a while loop  
|  |  | • Strengths of for-loops  
|  |  | • Write nested loops  
|  |  | • Use flow control to break or continue loops  |
| 5 | Arrays and Vectors | • Arrays vs vectors  
|  |  | • Iterating through and modifying vectors  
|  |  | • Multidimensional arrays  
|  |  | • Access elements in arrays and vectors  
|  |  | • Combine loops and vectors  
|  |  | • Develop program with multiple vectors  
|  |  | • Resize a vector using user-input  
|  |  | • Understand memory usage for multi-dimensional arrays  |
| 6 | Strings and C Strings | • C strings  
|  |  | • String library  
|  |  | • Char library  
|  |  | • Write C string as null-terminated string  |
| 7 | Functions | - Function declaration and definition  
- How a function works  
- Scope and parameters  
- Overloading functions  
- Compiling with separate files | - Write first function  
- Return from a function and parameterize a function  
- Learn reasons to use a function  
- Combine functions with branches and loops  
- Use incremental development with functions  
- Pass-by-reference to functions  
- Overload user-defined functions  
- Send and use arguments passed to program  
- Write makefile for modular compilation |
|---|---|---|
| 8 | Classes | - Structs for grouping data  
- Classes for grouping data and functions  
- Class constructors and constructor overloading | - Write structs for grouping data  
- Combine structs and functions  
- Combine structs and vectors  
- Write classes that group functions and data  
- Initialize class variables with class constructors  
- Overload class constructor  
- Introduce abstract data types |
| 9 | Streams | - Output formatting  
- String streams  
- File IO | - Understand standard input and output implementation  
- Manipulate text and floating-points during output |
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 10| Pointers | • Store location to memory  
• Allocate and deallocate memory  
• Identify memory leaks  
• Constructs in classes to help manage memory |   |
|   |   | ● Motivate reason for using pointers  
● Point to a memory location  
● Allocate and deallocate memory  
● Use different regions of program memory  
● Identify and fix memory leaks  
● Class destructors, copy constructors, and copy assignment operator |   |
| 11| Inheritance | • Derived classes  
• Access specifiers  
• Overriding functions  
• Polymorphism |   |
|   |   | ● Derive a class from another class  
● Choose the appropriate access specifier  
● Override member function of inherited class  
● Introduce polymorphism during runtime |   |