EGR 125
Introduction to Engineering Methods (C++)
File: N125O3L

Test #3 Overview

Material covered

- Chapters 7, 8, 10, and 13 in Introduction to Programming with C++, 3rd Edition by Liang
- Homework Assignments: Ch7-HW, CH8-HW, Ch10-HW and Ch13-HW

Format (similar to previous tests)

- No books, no notes, no computers
- Types of problems includes:
  - Determining the output of programs on the test
  - Some T/F, multiple-choice, short answer, etc.
  - Writing programs or instructions to accomplish specified tasks.
- Very detail-oriented. Be prepared!

Items provided on the test (also see documents on web site)

1) Tables of ASCII Codes and Operator Precedence
2) String Functions Table

Chapter 7 – Single-Dimensional Arrays

Arrays are also called subscripted variables or indexed variables
Declaring arrays
- use type, variable name, and brackets [ ]
- value in brackets must be an integer or const integer variable or expression with defined value

Memory is allocated when the array is declared.
C++ does not check to see if you exceed array bounds, so you can crash the computer by writing beyond the bounds of an array.
Array indices begin at 0 (so int A[8] defines 8 variables: A[0] to A[7])
Initializing arrays with lists
- List consist of a set of braces { } containing values separated by commas
- Recall that if less values are listed than are in the array then the remaining elements are initialized to zero. So int A[100] = {0} initializes all 100 values to zero.
- An uninitialized array contains junk, not zeros!
- If the array size is omitted, the array is sized to fit the list.

Printing arrays – the number of items printed per line is controlled by the loop
Reading values from arrays into data files.
Reading until the EOF marker is found.
Functions and arrays
- arrays are always treated as reference parameters, so no & required.
- typically dimension 1D arrays in the main program and pass the array and the array size to functions

C-style character arrays – no questions on the test

Chapter 8 – Multi-Dimensional Arrays

Multi-dimensional arrays: 1D, 2D, 3D, 4D, etc;
A 2D array is often called a matrix.
Using nested loops to initialize, read, manipulate, or print arrays.

Loading arrays with lists.
- 2D arrays are loaded by row
- for 2D or higher, they are loaded by varying the indices, beginning with the rightmost index

Multidimensional arrays and functions
- Only the leftmost set of brackets can be empty in the function declaration and definition.
- The size of the leftmost set of brackets is typically passed as an argument.

Chapter 12 – Standard Template Library (Vector class)

Standard Template Library (STL) – no questions on the test

Chapter 13 – Data Files

Uses of files
Interactive versus non-interactive programs
Extraction operator (>>), insertion operator (<<)
fstream
- using ifstream, ofstream and fstream to define input and output streams
- using ***ios::in, ios::out***, and ***ios::app*** with fstream
- fstream header
- valid identifiers
- ***fail( )*** function
- ***close( )*** function
- ***eof( )*** function

Writing to data files
Reading integer, real, character, and string values from data files
White spaces (space, tab, and newline)
Unknown number of items in data file – searching for the end-of-file marker
Input buffer
Reading data from files into arrays

Chapter 10 – C++ strings

Comparison to C-style character arrays in notes just for reference – not covered on this test
Using **class string**, so be sure to use **#include <string>**
Typical class usage: dot notation, member functions.
Declaring and initializing strings
Concatenation using + and += operators
String comparison using relational operators – based on ASCII values and lexicographical ordering
Accessing elements of a string using brackets [ ] – similar to using an array
Member functions in class string: **find, rfind, length, substr**, etc – refer to table in text or notes
Reading strings using cin (or InData, etc., with a data file) – reads one word at a time
getline – can be used to read one line at a time or to read until a certain character is encountered
- **getline(cin,S1)** or **getline(cin,S1,'\n')** – reads one line from keyboard into string S1
- **getline(cin,S1, '*')** - reads all characters up to and including * from keyboard into string S1

ignore( ) (Example: **indata.ignore(50,'*')** – ignore up to 50 characters until * is encountered)
– useful for ignoring string input until a certain character is encountered
- Particularly helpful when using getline to read a string from a file after reading numbers (int, double, etc) as a \n character may be left in the file.

Strings in functions – strings can be used like any other variable as function inputs, returns, etc.
String arrays – arrays of strings are similar to arrays of other variables.
get( ) – useful for reading characters that include white spaces. For example:

```
cin >> Ch1;           // read character into Ch1, but skip white spaces
Ch1 = get(cin);       // read character into Ch1, including white spaces
```