Lab 9

Multi-Threading

In this workshop, you backup an encrypted text file of characters to a binary file, restore the encrypted data from the binary file and add multi-threading to the solution.

**LEARNING OUTCOMES**

Upon successful completion of this workshop, you will have demonstrated the abilities to

* execute partitioned data on two or more threads
* write a set of characters to a file in binary mode
* read a set of characters from a file in binary mode
* bind a function to its arguments

**SPECIFICATIONS**

This workshop has two parts:

1. Implementing Binary File Access
2. Implementing Multi-Threading

Files Provided

The three source files provided for this workshop are listed below

* **SecureData.h** (complete)
* **SecureData.cpp** (incomplete)
* **W9.cpp** (complete)

The test data file for this workshop is at

* [**w9\_text.dat**](https://scs.senecac.on.ca/~btp305/pages/workshops/w10_text.dat)

The **SecureData** class holds text in encoded form along with the number of bytes in the text including the null terminator.  The **Cryptor** function object holds the encryption/decryption logic used by the **SecureData** class.

Part 1 - Binary File Access

Your first task is to complete the **backup()** and **restore()** member functions of the **SecureData** class.  Your code in each case should open the binary file in the appropriate mode and throw an exception if opening fails.  For backup, your code simply writes the data to the file.  For restoration, your code deallocates the existing data, determines the number of bytes to be read, allocates memory for them, and read the data from the file.

Output

The output from a completed version of this program should look like:

921 bytes copied from text w9\_text.dat into memory (null byte added)

Data encrypted in memory

922 bytes copied from binary file encoded.dat into memory (null byte included)

Data decrypted in memory

Compound types are types that a programmer constructs from a language's

fundamental types and/or other compound types. In C++11, compound types

include pointers and references to existing types, enumerations of named

constant values, arrays of objects of an existing type, classes and function

types. C++11 clarified the difference between lvalues and rvalues and

references to each, which helped simplify many of the facilities available

in the standard library.

This chapter describes the non-function types in detail. The description

includes a review of one-dimensional arrays in both static and dynamic memory

and shows how to allocate multi-dimensional arrays in both static and dynamic

memory. The description of classes reviews class definitions, introduces

move-constructors and move-assignment operators and includes declarations of

special members such as bit fields, class variables and class functions.

Part 2 - Multi-Threading

Your second task is to multi-thread the conversion in the **code()** member function of the **SecureData** class.  Select a reasonable number of threads and partition the data equally amongst all of the threads.  Use **std::bind** from the **functional** library to bind the **converter**

The results for this section should be identical to those shown above.