**Move and Copy Semantics**   
  
Workshop 2

In this workshop, you are to compare move and copy operations on an object that contains a dynamically allocated array of string objects.

**LEARNING OUTCOMES**

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

* implement copy semantics for a class with a resource
* implement move semantics for a class with a resource
* identify the processing-intensive operations in copy and move assignments
* retrieve data from a text file using an **ifstream** object
* reflect on the material learned through this workshop

**SPECIFICATIONS**

**Text Class**

Design and code a class named **Text** that manages a dynamically allocated array of **string**s.  Upon instantiation, a **Text** object receives nothing or a reference to an unmodifiable string.  The string holds the name of the text file that contains the records to be stored in an object of this class.  If the file does not exist, the **Text** object assumes a safe empty state.  If the file exists, the one-argument constructor allocates memory for the number of records contained in the file and copies them into memory.  To review the syntax for reading from a text file using an **ifstream** object see the chapter in your OOP244 notes entitled [Custom File Operators](https://scs.senecac.on.ca/~oop244/pages/content/files.html).  See also [cplusplus.com](https://www.cplusplus.com/reference/cstdlib/rand/)

Your design also includes the following member functions:

* a copy constructor
* a copy assignment operator
* a move constructor
* a move assignment operator
* a destructor
* a member function named **size\_t size() const** that returns the number of records of text data

Define your class and its implementation in namespace **w3**.  Store your class definition in a header file named **Text.h**and your member function definitions in an implementation file named **Text.cpp**.

**Main Program**

The main program that uses your **Text** class is listed below

|  |
| --- |
| **// Workshop 3 - Copy and Move Semantics**  **// w3.cpp**  **#include <iostream>**  **#include <iomanip>**  **#include <utility>**  **#include <ctime>**  **#include "Text.h"**  **#define TIME(start, end) double((end) - (start)) / CLOCKS\_PER\_SEC**  **int main (int argc, char\* argv[]) {**  **if (argc == 1) {**  **std::cerr << argv[0] << ": missing file operand\n";**  **return 1;**  **}**  **else if (argc != 2) {**  **std::cerr << argv[0] << ": too many arguments\n";**  **return 2;**  **}**  **std::clock\_t cs, ce;**  **{**  **std::cout << std::fixed << std::setprecision(3);**  **cs = std::clock();**  **w3::Text a;**  **ce = std::clock();**  **std::cout << "Constructor " << TIME(cs, ce) << " seconds";**  **std::cout << " - a.size = " << a.size() << std::endl;**  **cs = std::clock();**  **w3::Text b(argv[1]);**  **ce = std::clock();**  **std::cout << "Constructor " << TIME(cs, ce) << " seconds";**  **std::cout << " - b.size = " << b.size() << std::endl;**  **cs = std::clock();**  **a = b;**  **ce = std::clock();**  **std::cout << "Copy Assignment " << TIME(cs, ce) << " seconds";**  **std::cout << " - a.size = " << a.size() << std::endl;**  **cs = std::clock();**  **a = std::move(b);**  **ce = std::clock();**  **std::cout << "Move Assignment " << TIME(cs, ce) << " seconds";**  **std::cout << " - a.size = " << a.size() << std::endl;**  **cs = std::clock();**  **w3::Text c = a;**  **ce = std::clock();**  **std::cout << "Copy Constructor " << TIME(cs, ce) << " seconds";**  **std::cout << " - c.size = " << c.size() << std::endl;**  **cs = std::clock();**  **w3::Text d = std::move(a);**  **ce = std::clock();**  **std::cout << "Move Constructor " << TIME(cs, ce) << " seconds";**  **std::cout << " - d.size = " << d.size() << std::endl;**  **cs = std::clock();**  **}**  **ce = std::clock();**  **std::cout << "Destructor " << TIME(cs, ce) << " seconds\n";**  **}** |

**Analysis**

A text file named **gutenberg\_shakespeare** is available [here](https://scs.senecac.on.ca/~oop345/pages/workshops/gutenberg_shakespeare).

Run your executable for the following command line arguments:

|  |
| --- |
| **w3 gutenberg\_shakespeare** |

Examine the output and explain why the execution time differs for the two assignment operators and the different constructors.

**SUBMISSION**

Follow your professor's submission instructions