Lab 4 Report

Course: ENSF 619 - Fall 2020
Lab #: Lab 3
Student Name: Davis Allan, 10016543
Submission Date: Oct 18 2020

Exercise A

Source Code

```
/*
* File Name: DynString.cpp
* Lab # and Assignment #: Lab 4 Exercise A
* Lab section: B01
* Completed by: Davis Allan, 10016543
* Submission Date: 2020-10-18
*/

#include "MyArray.h"
#include <assert.h>

MyArray::MyArray(): sizeM(0) {
    storageM = new EType[0];
    assert(storageM != NULL);
}

MyArray::MyArray(const EType *builtin, int sizeA): sizeM(sizeA){
    storageM = new EType[sizeA];
    assert(storageM != NULL);

    for (int i = 0; i < sizeA; i++) {
        storageM[i] = builtin[i];
    }
}

MyArray::MyArray(const MyArray& source) : sizeM(source.size()){
    storageM = new EType[sizeM];
    assert(storageM != NULL);

    for (int i = 0; i < sizeM; i++) {
    storageM[i] = source.storageM[i];
    }
}

MyArray& MyArray::operator = (const MyArray& rhs) {
    if (this != &rhs) {
        delete [] storageM;
        sizeM = rhs.size();
        storageM = new EType[sizeM];
```

```
        assert(storageM != NULL);

        for (int i = 0; i < sizeM; i++) {
            storageM[i] = rhs.storageM[i];
        }
    }
    return *this;
}

MyArray::~MyArray() {
    delete [] storageM;
    storageM = NULL;
}

int MyArray::size() const {
    return sizeM;
}

EType MyArray::at(int i) const {
    assert(i >= 0 && i < size());
    return storageM[i];
}

void MyArray::set(int i, EType new_value) {
    assert(i >= 0 && i < size());
    storageM[i] = new_value;
}

void MyArray::resize(int new_size) {
    assert(new_size >= 0);
    EType *resized = new EType[new_size];
    assert(resized != NULL);

    if (new_size > sizeM) {
        for (int i = 0; i < sizeM; i++) {
            resized[i] = storageM[i];
        }
    }
    else {
    for (int i = 0; i < new_size; i++) {
            resized[i] = storageM[i];
        }
    }
```

```
    delete [] storageM;
    storageM = resized;
    sizeM = new_size;
}
```

Program Output

```
PS C:\Users\davis\Desktop\ENSF 619\Labs\Lab4> g++ -Wall .\MyArray.cpp .\lab4ExA.cpp -o MyArray.exe
PS C:\Users\davis\Desktop\ENSF 619\Labs\Lab4> .\MyArray.exe
Elements of a:  0.5 1.5 2.5 3.5 4.5
(Expected:      0.5 1.5 2.5 3.5 4.5)

Elements of b after first resize:  10.5 11.5 12.5 13.5 14.5 15.5 16.5
(Expected:                         10.5 11.5 12.5 13.5 14.5 15.5 16.5)

Elements of b after second resize:  10.5 11.5 12.5
(Expected:                          10.5 11.5 12.5)

Elements of b after copy ctor check:  10.5 11.5 12.5
(Expected:                            10.5 11.5 12.5)

Elements of c after copy ctor check:  -1.5 11.5 12.5
(Expected:                            -1.5 11.5 12.5)

Elements of a after operator = check:  -10.5 1.5 2.5 3.5 4.5
(Expected:                             -10.5 1.5 2.5 3.5 4.5)

Elements of b after operator = check:  -11.5 1.5 2.5 3.5 4.5
(Expected:                             -11.5 1.5 2.5 3.5 4.5)

Elements of c after operator = check:  0.5 1.5 2.5 3.5 4.5
(Expected:                             0.5 1.5 2.5 3.5 4.5)

PS C:\Users\davis\Desktop\ENSF 619\Labs\Lab4>
```

Exercise B

Function Definition

```
String_Vector transpose (const String_Vector& sv) {
    String_Vector vs;

    int cols = sv.size();
    int rows = sv.at(0).size();

    vs.resize(rows);

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            vs.at(i).push_back(sv.at(j).at(i));
        }
    }
    return vs;
}
```

Program Output

```
PS C:\Users\davis\Desktop\ENSF 619\Labs\Lab4> .\lab4ExB.exe
ABCDE
FGHIJ
KLMNO
PQRST
UVWXY
AFKPU
BGLQV
CHMRW
DINSX
EJOTY
PS C:\Users\davis\Desktop\ENSF 619\Labs\Lab4>
```

Exercise C

Function Definition

```
void print_from_binary(char* filename) {
    //opening the binary file
    ifstream input(filename, ios::in | ios::binary);
    if (input.fail()) {
        cerr << "failed to open file: " << filename << endl;
        exit(1);
    }
    //creating the output filename and changing the extension
    string fileNameTxt;
    int i = 0;
    while (filename[i] != '.') {
        fileNameTxt.push_back(filename[i]);
        i++;
    }
    fileNameTxt.append(".txt");

    //creating the output file
    ofstream output(fileNameTxt);
    if (output.fail()) {
        cerr << "failed to create file: " << "cities.txt" << endl;
        exit(1);
    }

    City cities[size];

    //reading and displaying each object as well as saving each object to
the output file
    for (int i = 0; i < size; i++) {
        input.read((char*)(&cities[i]), sizeof(cities));

        cout << "Name: " << cities[i].name << ", x coordinate: " <<
cities[i].x
                << ", y coordinate: " << cities[i].y << endl;

        output << "Name: " << cities[i].name << ", x coordinate: " <<
cities[i].x
                << ", y coordinate: " << cities[i].y << endl;
    }
```

```
    input.close();
    output.close();
}
```

Program Output

File also attached