ENSF 619- Fall 2020

Lab: 4

Ziad Chemali

October 18, 2020

1) Exercise A:

a) Code:

```
/*
*MyArray.cpp
* Lab: 4 Exercise A
* Completed by Ziad Chemali
* Submission Date: October 18,2020
*/


#include "MyArray.h"
#include <iostream>
MyArray::MyArray() {
	sizeM = 0;
	storageM = new EType[0];
}
MyArray::MyArray(const EType* builtin, int sizeA) {
	if (sizeA >= 0 && builtin != nullptr) {
		sizeM = sizeA;
		delete[] storageM;
		storageM = new EType[sizeM];
		for (int i = 0;i < sizeA ;i++) {
			storageM[i] = builtin[i];
		}


	}
	else
		std::cout << "Error, cant copy an empty array\n";
}
MyArray::MyArray(const MyArray& source) {
	if (source.storageM !=nullptr) {
		delete[] storageM;
		sizeM = source.sizeM;
		storageM = new EType[sizeM];
		for (int i = 0;i < source.sizeM ;i++) {
			storageM[i] = source.storageM[i];
		}
	}
}
MyArray&  MyArray :: operator =(const MyArray& rhs) {
	if (this != &rhs)
	{
		delete[] storageM;
		sizeM = rhs.sizeM;
		storageM = new EType[sizeM];
		for (int i = 0;i < rhs.sizeM ;i++) {
			storageM[i] = rhs.storageM[i];
		}
	}
	return *this;
}
MyArray::  ~MyArray() {
	delete[] storageM;
```

```
}
int MyArray::size() const {
	return this->sizeM;
}
EType MyArray::at(int i) const {
	return this->storageM[i];
}

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

void MyArray::resize(int new_size)
{
	if (new_size >= 0) {
		sizeM = new_size;
		EType *temp = new EType[new_size];
		for (int i = 0;i < new_size ;i++) {
			temp[i] = this->storageM[i];
		}
		delete[] storageM;
		storageM = temp;

	}
}
```

b) Output:

```
Microsoft Visual Studio Debug Console
(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)
```

2) Exercise B:

a) Code:

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

     String_Vector vs;


    vs.resize(sv.at(0).size());
    for (int i = 0; i < sv.size();i++) {

        for (int j = 0;j < sv.at(i).size();j++)

            vs.at(j).push_back(sv.at(i).at(j));

    }

    return vs;
}
```

b) Output:

Microsoft Visual Studio Debug Console

```
ABCDE
FGHIJ
KLMNO
PQRST

Transposed Matrix:
AFKP
BGLQ
CHMR
DINS
EJOT
```

3) Exercise C:

a) Code:

```
void print_from_binary(char* filename) {

    ifstream stream(filename, ios::in | ios::binary);
    if (stream.fail()) {
        cerr << "failed to open file: " << filename << endl;
        exit(1);
    }

    City temp;

    while (!stream.eof()) {
        stream.read((char*)(&temp), sizeof(City));
        cout << "x= "<<temp.x <<", y= "<< temp.y << ", City= "<< temp.name << endl;
    }
    stream.close();}
```

b) Output:

```
Microsoft Visual Studio Debug Console

The content of the binary file is:
x= 100, y= 50, City= Calgary
x= 100, y= 150, City= Edmonton
x= 50, y= 50, City= Vancouver
x= 200, y= 50, City= Regina
x= 500, y= 50, City= Toronto
x= 200, y= 50, City= Montreal
x= 200, y= 50, City= Montreal
```