ENSF 614

Advanced System Analysis and Software Design

LAB 4

Author:

Steven Duong
(30022492)

Affiliation
Department of Electrical and Software Engineering
University of Calgary
Calgary, Alberta

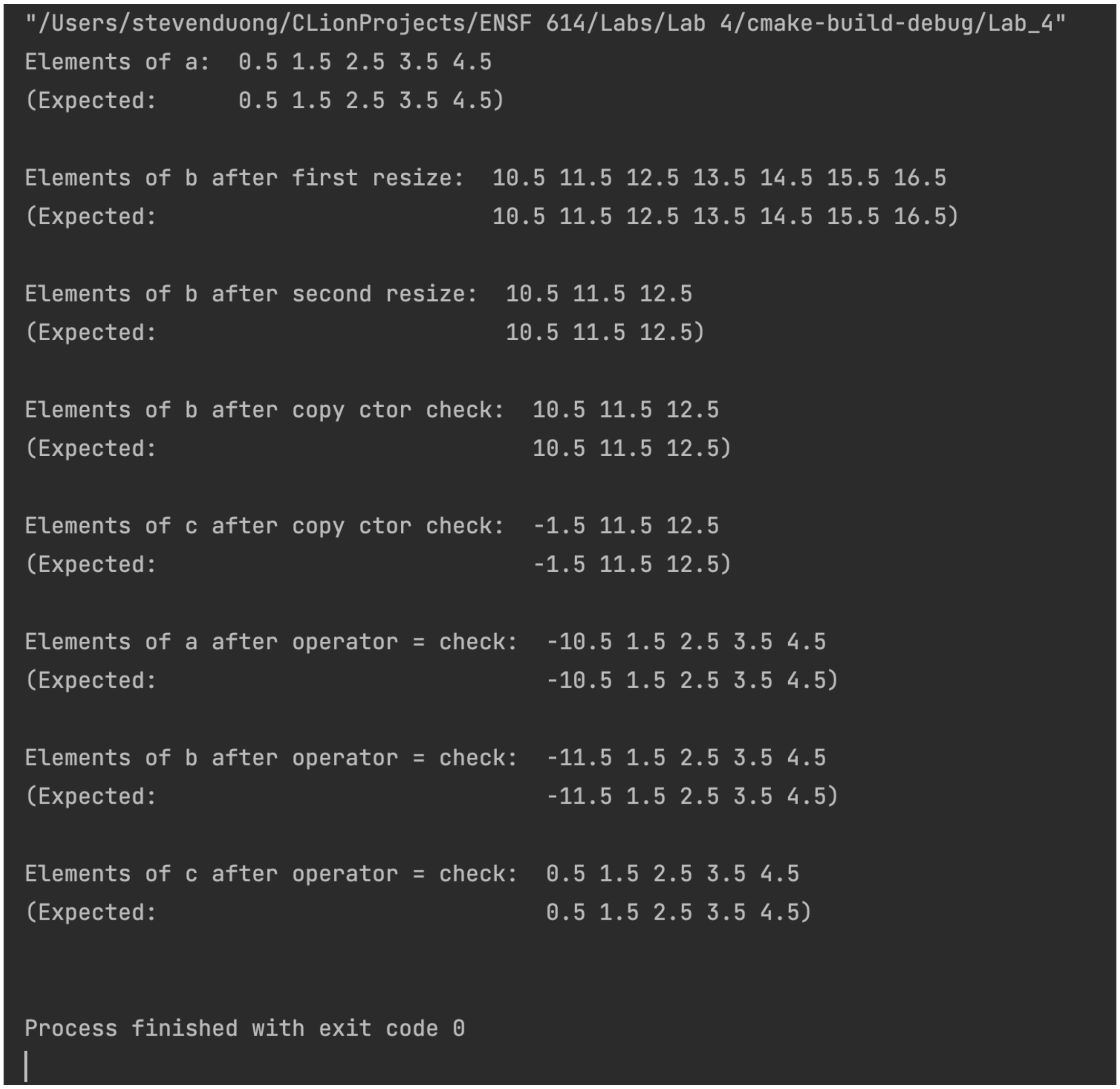
Lab Block: B01
Date of Report: Feb 12, 2023

Exercise A

C++ Code

```cpp
/*
 *  File Name: MyArray.cpp
 *  Assignment: ENSF 614 Lab 4 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Feb 12, 2023
 */

#include "MyArray.h"

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

MyArray::MyArray(const EType * builtin, int sizeA) {
  if (sizeA <= 0) {
    return;
  }

  sizeM = sizeA;
  storageM = new EType[sizeM];

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

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

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

  return * this;
}

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

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

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

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

void MyArray::resize(int new_size) {
  if (new_size < 0) {
    return;
  }

  EType * temp = new EType[new_size];
  int minSize = (new_size < sizeM) ? new_size : sizeM;
  for (int i = 0; i < minSize; ++i) {
    temp[i] = storageM[i];
  }
  delete[] storageM;
  storageM = temp;
  sizeM = new_size;
}
```

Program Output



```
"/Users/stevenduong/CLionProjects/ENSF 614/Labs/Lab 4/cmake-build-debug/Lab_4"
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)


Process finished with exit code 0
```

Exercise B

C++ Code

```
/*
 *  File Name: lab4ExB.cpp
 *  Assignment: ENSF 614 Lab 4 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Feb 12, 2023
 */

#include<vector>

#include<string>

#include <iostream>

using std::cout;
using std::cerr;
using std::endl;
using std::vector;
using std::string;

typedef vector < string > String_Vector;

String_Vector transpose(const String_Vector & sv);
// REQUIRES:
//    sv.size() >= 1
//    All the strings in sv are the same length, and that length is >=
1.
// PROMISES:
//    Return value is the "transpose" of sv, as defined in the
Exercise B
//    instructions.

int main() {

  const int ROWS = 5;
  const int COLS = 4;

  char c = 'A';
  String_Vector sv;
  sv.resize(ROWS);

  for (int i = 0; i < ROWS; i++)
    for (int j = 0; j < COLS; j++) {
      sv.at(i).push_back(c);
      c++;
      if (c == 'Z' + 1)
        c = 'a';
      else if (c == 'z' + 1)
        c = 'A';
    }

  for (int i = 0; i < ROWS; i++) {
    cout << sv.at(i);
    cout << endl;
  }
  String_Vector vs = transpose(sv);
  for (int i = 0; i < (int) vs.size(); i++)
    cout << vs.at(i) << endl;

  return 0;
}

String_Vector transpose(const String_Vector & sv) {

  String_Vector vs(sv[0].size()); // creates a vector of strings with
size equal to the number of columns in sv
  for (int i = 0; i < sv.size(); i++) { // loop through each row of sv
    for (int j = 0; j < sv[i].size(); j++) { // loop through each
character in the current row
      vs[j].push_back(sv[i][j]); // add the current character to the
string in the corresponding column of vs
    }
  }
  return vs; // return the transposed vector of strings

}
```

Program Output

```
"/Users/stevenduong/CLionProjects/ENSF 614/Labs/Lab 4/cmake-build-debug/Lab_4"
ABCD
EFGH
IJKL
MNOP
QRST
AEIMQ
BFJNR
CGKOS
DHLPT


Process finished with exit code 0
```

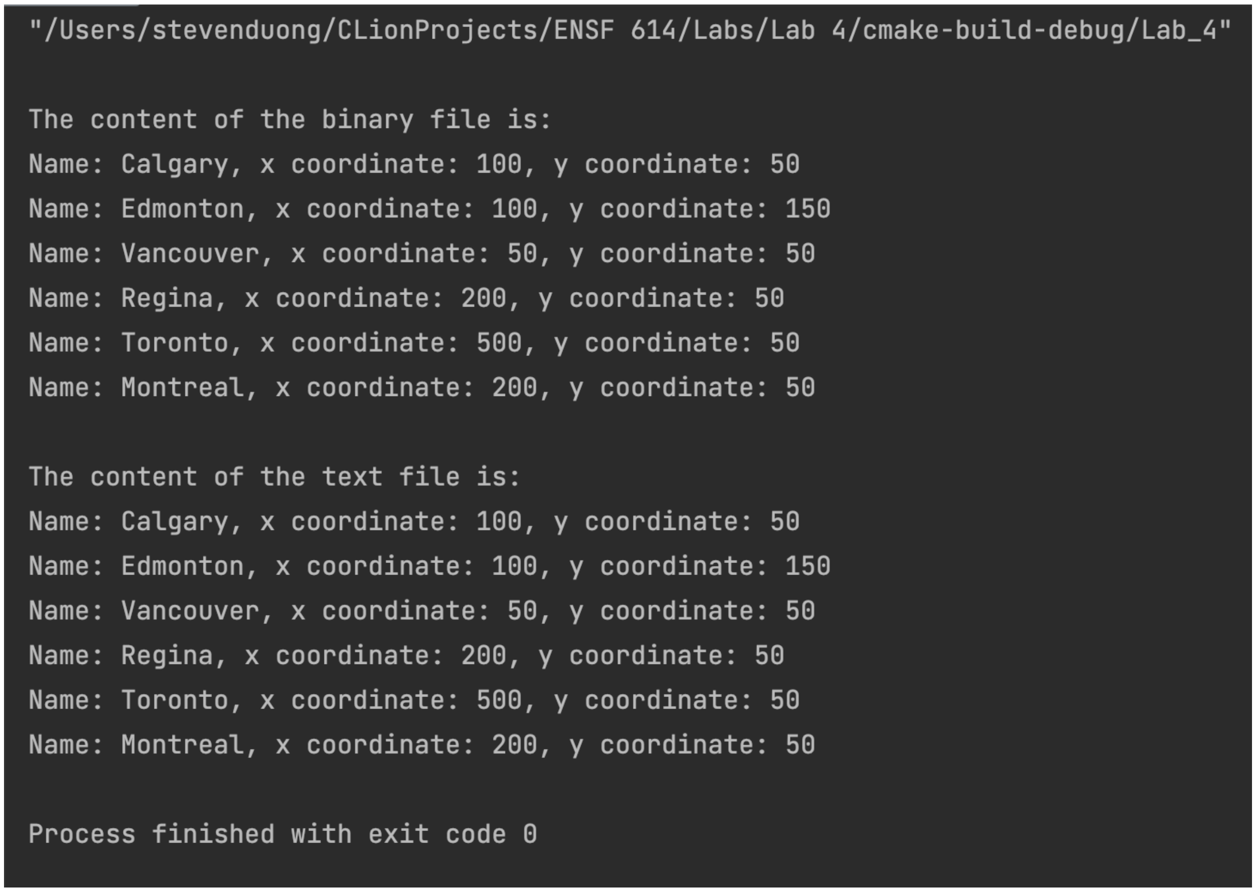
Exercise C

C++ Code

```
/*
```

```
 *  File Name: lab4ExC.cpp
 *  Assignment: ENSF 614 Lab 4 Exercise C
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Feb 12, 2023
 */

#include <iostream>
#include <fstream>
#include <sstream>
#include <stdlib.h>

const int size = 6;
using namespace std;
struct City {
    double x, y;
    char name[30];
};

void write_binary_file(City cities[], int size, char* filename);
/* PROMISES: attaches an ofstream object to a binary file named
"filename" and
 * writes the content of the array cities into the file.
 */

void print_from_binary(char* filename);
/* PROMISES: uses ifstream library object to open the binary file
named
 * "filename", reads the content of the file which are objects of
struct City
 * (one record at a time), and displays them on the screen. It also
saves the records
 * into a text-file that its name must be the filename argument, but
with the extension
 * of .txt
 */


int main() {
    char bin_filename[] = "cities.bin";

    City cities[::size] = {{100, 50, "Calgary"},
                          {100, 150, "Edmonton"},
                          {50, 50, "Vancouver"},
                          {200, 50, "Regina"},
                          {500, 50, "Toronto"},
                          {200, 50, "Montreal"}};

    write_binary_file(cities, ::size, bin_filename);
    cout << "\nThe content of the binary file is:" << endl;
    print_from_binary(bin_filename);
    return 0;
}

void write_binary_file(City cities[], int size, char* filename){
    ofstream stream(filename, ios::out | ios::binary);
    if(stream.fail()){
        cerr << "failed to open file: " << filename << endl;
        exit(1);
    }

    for(int i =0; i < size; i++)
        stream.write((char*)&cities[i], sizeof(City));
    stream.close();
}

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);
    }

    string text_filename = string(filename) + ".txt";
    ofstream text_file(text_filename);
    if(text_file.fail()){
        cerr << "failed to open file: " << text_filename << endl;
        exit(1);
    }

    City city;
    while(stream.read((char*)&city, sizeof(City))) {
        cout << "Name: " << city.name << ", x coordinate: " << city.x
<< ", y coordinate: " << city.y << endl;
        text_file << "Name: " << city.name << ", x coordinate: " <<
city.x << ", y coordinate: " << city.y << endl;
    }

    stream.close();
    text_file.close();

    cout << "\nThe content of the text file is:" << endl;
    ifstream text_file_in(text_filename);
    if(text_file_in.fail()){
        cerr << "failed to open file: " << text_filename << endl;
        exit(1);
    }
    string line;
    while(getline(text_file_in, line)) {
        cout << line << endl;
    }
    text_file_in.close();
}
```

Program Output



```
"/Users/stevenduong/CLionProjects/ENSF 614/Labs/Lab 4/cmake-build-debug/Lab_4"

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

The content of the text file is:
Name: Calgary, x coordinate: 100, y coordinate: 50
Name: Edmonton, x coordinate: 100, y coordinate: 150
Name: Vancouver, x coordinate: 50, y coordinate: 50
Name: Regina, x coordinate: 200, y coordinate: 50
Name: Toronto, x coordinate: 500, y coordinate: 50
Name: Montreal, x coordinate: 200, y coordinate: 50

Process finished with exit code 0
```