Course: ENSF 614 – Fall 2023

Lab 4:

Instructor: M. Moussavi

Student Name: Emmanuel Alafonye

Submission Date: October 18, 2023

// ENSF 614 - Fall 2023 Lab4 - exercise A

// lab4ExA.cpp

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 4, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 18, 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) {

// STUDENTS MUST COMPLETE THE DEFINITION OF THIS FUNCTION.

if (sv.empty()){ // Check if the input vector is empty

return String\_Vector();

}

const int numRows = sv.size();

const int numCols = sv[0].size();

//Initialize the transposed vector.

String\_Vector vs(numCols, std::string(numRows, ' '));

for (int i = 0; i < numRows; i++){ // Loop and transpose original matrix.

for (int j = 0; j < numCols; j++)

{

vs[j][i] = sv[i][j];

}

}

return vs;

}

Output

A black screen with white text

Description automatically generated

/\*

\* File Name: lab4Exe\_B.cpp

\* Assignment: ENSF 614 Lab 4, exercise

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 fall 2023 Lab 4 - Exercise B

\* Submission Date: October 18, 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) {

/\* Studnets must complete the implementaiton of this file. \*/

ifstream input(filename, ios::in | ios::binary);

if (input.fail()) {

cerr << "Failed to open file: " << filename << endl;

exit(1);

}

// Create a text file with the same name as the binary file but with a .txt extension

string txt\_filename(filename);

txt\_filename += ".txt";

ofstream output(txt\_filename, ios::out);

if (output.fail()) {

cerr << "Failed to create the text file: " << txt\_filename << endl;

exit(1);

}

// Read and display each record from the binary file

City city;

while (input.read((char\*)&city, sizeof(City))) {

// Display on the screen

cout << "Name: " << city.name << ", x coordinate: " << city.x << ", y coordinate: " << city.y << endl;

// Writing to the text file

output << "Name: " << city.name << ", x coordinate: " << city.x << ", y coordinate: " << city.y << endl;

}

// Open and close the input file

input.close();

output.close();

}

