ENSF 614 – Fall 2021

Lab 4 – Tuesday, October 19

Student Name: Aastha Patel and Bhavyai Gupta

Submission date: October 19, 2021

# Exercise A – Source file MyArray.cpp

/\*

 \* File Name:               MyArray.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 4 Exercise A

 \* Lab section:             B01

 \* Completed by:            Bhavyai Gupta, Aastha Patel

 \* Submission Date:         NA

 \*/

#include "MyArray.h"

MyArray::MyArray()

{

    sizeM = 0;

    storageM = new EType[sizeM];

}

MyArray::MyArray(const EType \*builtin, int sizeA)

{

    sizeM = sizeA;

    storageM = new EType[sizeM];

    for (int i = 0; i < sizeA; i++)

    {

        storageM[i] = builtin[i];

    }

}

MyArray::MyArray(const MyArray &source)

{

    sizeM = source.size();

    storageM = new EType[sizeM];

    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];

        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

{

    return storageM[i];

}

void MyArray::set(int i, EType new\_value)

{

    storageM[i] = new\_value;

}

void MyArray::resize(int new\_size)

{

    EType \*temp = new EType[new\_size];

    int limit = (new\_size > sizeM) ? new\_size : sizeM;

    for (int i = 0; i < limit; i++)

    {

        temp[i] = storageM[i];

    }

    delete[] storageM;

    sizeM = new\_size;

    storageM = temp;

}

# Exercise A – Program Output

Text

Description automatically generated

# Exercise B – Source Code of transpose

String\_Vector transpose(const String\_Vector &sv)

{

    int rows = sv.size();

    int cols = sv.at(0).size();

    // create a new vector with cols size

    String\_Vector vs(cols);

    for (int i = 0; i < cols; i++)

    {

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

        {

            // get the character from transpose location: basically i and j reversed

            char c = sv.at(j).at(i);

            vs.at(i).push\_back(c);

        }

    }

    return vs;

}

# Exercise B – Program Output

Text

Description automatically generated

# Exercise C – Source Code of print\_from\_binary

void print\_from\_binary(char \*filename)

{

    ifstream in\_stream(filename, ios::in | ios::binary);

    if (in\_stream.fail()) {

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

        exit(1);

    }

    int length = strlen(filename);

    char \*filename\_txt = new char[length + 1];

    for (int i = 0; i < length - 3; i++) {

        filename\_txt[i] = filename[i];

    }

    filename\_txt[length - 3] = 't';

    filename\_txt[length - 2] = 'x';

    filename\_txt[length - 1] = 't';

    filename\_txt[length] = '\0';

    ofstream out\_stream(filename\_txt);

    if(out\_stream.fail()) {

        cerr << "failed to open file: " << filename\_txt << endl;

        exit(1);

    }

    City c;

    while (!in\_stream.eof()) {

        in\_stream.read((char \*)&c, sizeof(City));

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

        out\_stream << "Name: " << c.name << ", x coordinate: " << c.x << ", y coordinate: " << c.y << endl;

    }

    in\_stream.close();

    out\_stream.close();

    delete[] filename\_txt;

}

# Exercise C – Program Output

Text

Description automatically generated