Jessica Sullivan

ID: 1282151

Class: COMSC-210-5067

Professor: Pentcheva

Source Code:

CS210\_Assignment2\_ClassesPointers.cpp:

*/\**

*Programmer: Jessica Sullivan*

*Programmer's ID: 1282151*

*Class: COMSC-210-5067*

*Goal: Write a program that will record information about employees*

*and will compute their paychecks*

*\*/*

#include <string>

#include <sstream>

#include <iostream>

#include <iomanip>

**using** **namespace** std;

#include "DynamicStringArray.h"

**const** string DynamicStringArray::BAD\_INPUT = "NULL";

**int** main(**int** argc, **const** **char** \* argv[]) {

*// programmer's identification*

cout << "Programmer: Jessica Sullivan" << endl;

cout << "Programmer's ID: 1282151" << endl;

cout << "File: " << \_\_FILE\_\_ << endl;

DynamicStringArray names;

*// List of names*

names.addEntry("Frank");

names.addEntry("Wiggum");

names.addEntry("Nahasapeemapetilon");

names.addEntry("Quimby");

names.addEntry("Flanders");

*// Output list*

cout << "List of names:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

*// Add and remove some names*

names.addEntry("Spuckler");

cout << "After adding a name:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

names.deleteEntry("Nahasapeemapetilon");

cout << "After removing a name:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

names.deleteEntry("Skinner");

cout << "After removing a name that isn't on the list:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

names.addEntry("Muntz");

cout << "After adding another name:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

*// Remove all of the names by repeatedly deleting the last one*

**while** (names.getSize() > 0) {

names.deleteEntry(names.getEntry(names.getSize() - 1));

}

cout << "After removing all of the names:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

names.addEntry("Olivia");

cout << "After adding a name:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

cout << "Testing copy constructor" << endl;

DynamicStringArray names2(names);

*// Remove Olivia from names*

names.deleteEntry("Olivia");

cout << "Copied names:" << endl;

**for** (**int** i = 0; i < names2.getSize(); i++)

cout << names2.getEntry(i) << endl;

cout << endl;

cout << "Testing assignment" << endl;

DynamicStringArray names3 = names2;

*// Remove Olivia from names2*

names2.deleteEntry("Olivia");

cout << "Copied names:" << endl;

**for** (**int** i = 0; i < names3.getSize(); i++)

cout << names3.getEntry(i) << endl;

cout << endl;

cout << "Enter a character to exit." << endl;

**char** wait;

cin >> wait;

**return** 0;

}

DynamicStringArray.h:

*// DYNAMICSTRINGARRAYH*

*// CS210\_Assignment2\_ClassesPointers*

*//*

*/\**

*Programmer: Jessica Sullivan*

*Programmer's ID: 1282151*

*Class: COMSC-210-5067*

*\*/*

*// Created by Jessie Sully on 2/5/20.*

*// Copyright © 2020 Jessie Sully. All rights reserved.*

*//*

#ifndef DYNAMICSTRINGARRAY\_H

#define DYNAMICSTRINGARRAY\_H

#include <string>

**using** **namespace** std;

**class** DynamicStringArray {

*//This is used in getEntry to allow personalized error strings*

**static** **const** string BAD\_INPUT;

**public**:

*//default constructor initializes size to zero and dynamicArray to nullptr*

DynamicStringArray();

DynamicStringArray(**const** DynamicStringArray& srcArray);

~DynamicStringArray() {**delete** [] mDynamicArray;}

DynamicStringArray& **operator**=(**const** DynamicStringArray& srcArray);

**int** getSize() **const** { **return** mSize; }

*//returns a desired BAD\_INPUT string in cases of error - array doesn't exist (null) or is empty or index is out of range*

**const** string& getEntry(**int** index) **const** { **return** index >= 0 && index < mSize

? mDynamicArray[index] : BAD\_INPUT; };

**void** addEntry(**const** string &entry);

*//deleteEntry will delete ALL matching entries. The bool is an error catching mechanism False means that the array doesn't exist (null) or its empty or that there wasn't a match*

**bool** deleteEntry(**const** string &entry);

**private**:

**int** mSize = 0;

string \*mDynamicArray = **nullptr**;

*//helper function to find the number of matches to resize DynamicArray for deleteEntry*

**int** getNumMatchingEntries(**const** string& entry) **const**;

*//helper function to copy all entries in dynamically allocated array returns false if there input array has a size < 0 or size != 0 when DynamicArray = nullptr*

**bool** copyData(**const** DynamicStringArray &srcArray);

};

#endif */\* DYNAMICSTRINGARRAY\_H \*/*

DynamicStringArray.cpp:

*//*

*// DynamicStringArray.cpp*

*// CS210\_Assignment2\_ClassesPointers*

*/\**

*Programmer: Jessica Sullivan*

*Programmer's ID: 1282151*

*Class: COMSC-210-5067*

*Goal: Write a program that will record information about employees*

*and will compute their paychecks*

*\*/*

*/\**

*#include <iostream>*

*using namespace std;*

*\*/*

#include "DynamicStringArray.h"

DynamicStringArray::DynamicStringArray() {

mSize = 0;

mDynamicArray = **nullptr**;

}

DynamicStringArray::DynamicStringArray(**const** DynamicStringArray &srcArray) {

copyData(srcArray);

}

DynamicStringArray& DynamicStringArray::**operator**=(**const** DynamicStringArray &srcArray) {

copyData(srcArray);

**return** \***this**;

}

**bool** DynamicStringArray::copyData(**const** DynamicStringArray &srcArray) {

mSize = srcArray.mSize;

**if** (mSize > 0 && srcArray.mDynamicArray != **nullptr**) {

**delete** [] mDynamicArray; *// this is safe, I checked.*

mDynamicArray = **new** string[mSize];

**for** (**int** i = 0; i < mSize; i++) {

mDynamicArray[i] = srcArray.mDynamicArray[i];

}

**return** **true**;

}

**else** {

mSize = 0;

mDynamicArray = **nullptr**;

**return** **false**;

}

}

**void** DynamicStringArray::addEntry(**const** string &entry) {

string \*tempArray = mDynamicArray;

mSize++;

mDynamicArray = **new** string[mSize];

*// make sure array exists*

**if** (tempArray != **nullptr**) {

**for** (**int** i = 0; i < mSize - 1; i++) {

mDynamicArray[i] = tempArray[i];

}

**delete** [] tempArray;

}

mDynamicArray[mSize - 1] = entry;

}

**int** DynamicStringArray::getNumMatchingEntries(**const** string &entry) **const** {

**int** numEntries = 0;

**if** (mSize > 0) {

**for** (**int** i = 0; i < mSize; i++) {

**if** (mDynamicArray[i] == entry) {

numEntries++;

}

}

}

**return** numEntries;

}

**bool** DynamicStringArray::deleteEntry(**const** string &entry) {

string \*tempArray = mDynamicArray;

**int** numEntries = getNumMatchingEntries(entry);

**int** newSize = mSize - numEntries;

**if** (newSize == 0 && mSize > 0) {

mSize = 0;

**delete** [] mDynamicArray;

mDynamicArray = **nullptr**;

}

**else** **if** (newSize != mSize) {

mDynamicArray = **new** string[newSize];

**for** (**int** i = 0, j = 0; j < newSize && i < mSize; i++) {

**if** (tempArray[i] != entry) {

mDynamicArray[j] = tempArray[i];

j++;

}

}

**delete** [] tempArray;

mSize = newSize;

}

**else** {

**return** **false**;

}

**return** **true**;

}

Output:



