**C++ Part I (INFO1-CE9264) Fall 2014 – Homework 6**

Clement Chan

**Question 1 – Two Dimension Dynamic Array**

**Header file**

//Set up dynamic memories for the two dimensional arrays

typedef double \*DoubleArrayPtr;

//Set up the TwoD class with functions and methods

class TwoD{

private:

unsigned int MaxRows, MaxCols;

int row, col;

double sum;

double \*\*table; //Define a two dimensional pointer array.

public:

double result;

TwoD(); //Default constructor

TwoD(int, int); //Parameterized constructor

TwoD(int, int, int, int, double); // Class with size and entry included (use for addition)

TwoD(const TwoD &p); //Copy Constructor

int getMaxRow() const; //getting row entered

int getMaxCol() const; //getting row entered

int getRow() const; //getting row entered

int getCol() const; // getting col entered

void entry(int,int, double); //Enter the number elements into specific row and column

double return\_number() const; //Display the specific row and column's result

~TwoD();//Destructor

//Operator Functions

friend const TwoD operator +(const TwoD& TD\_1, const TwoD& TD\_2);

TwoD& operator=(const TwoD &p);

};

**Question 1.cpp**

#include <iostream>

#include <stdio.h>

#include <cmath>

#include "Question\_1.h"

using namespace std;

//Method Classes that were included in the header classes

//Default Constructor's Definition

TwoD::TwoD(void){

cout<<"Object Initialized" << endl;

MaxRows = 100;

MaxCols = 100;

DoubleArrayPtr \*arr\_ptr = new DoubleArrayPtr[MaxRows];

for (int i = 0; i < MaxRows; i++){

arr\_ptr[i] = new double[MaxCols];

};

table = arr\_ptr;

for(int i = 0 ; i < MaxRows; i++){

for(int j = 0; j < MaxCols; j++){

table[i][j] = 0; //Initialize the dynamic arrays with all zeros.

}

}

}

//Parametric Constructor's Definition

TwoD::TwoD(int a, int b){

cout<<"Object Initialized" << endl;

MaxRows = a;

MaxCols = b;

DoubleArrayPtr \*arr\_ptr = new DoubleArrayPtr[MaxRows];

for (int i = 0; i < MaxRows; i++){

arr\_ptr[i] = new double[MaxCols];

};

table = arr\_ptr;

for(int i = 0 ; i < MaxRows; i++){

for(int j = 0; j < MaxCols; j++){

table[i][j] = 0; //Initialize the dynamic arrays with all zeros.

}

}

}

//Parametric Constructor's Definition

TwoD::TwoD(int a, int b, int c, int d, double element){

cout<<"Object Initialized" << endl;

MaxRows = a;

MaxCols = b;

row = c;

col = d;

DoubleArrayPtr \*arr\_ptr = new DoubleArrayPtr[MaxRows];

for (int i = 0; i < MaxRows; i++){

arr\_ptr[i] = new double[MaxCols];

};

table = arr\_ptr;

for(int i = 0 ; i < MaxRows; i++){

for(int j = 0; j < MaxCols; j++){

table[i][j] = 0; //Initialize the dynamic arrays with all zeros.

}

}

table[c][d] = element;

}

//Copy Constructor Definition

TwoD::TwoD(const TwoD &p){

cout<<"Copy constructor with pointers."<<endl;

MaxRows = p.getMaxRow();

MaxCols = p.getMaxCol();

table = new double\*[MaxRows];

for(int i=0; i < MaxRows; i++){

table[i] = new double[MaxCols];

}

for(int i = 0; i < MaxRows; i++) {

for(int j = 0; j < MaxCols; j++) {

table[i][j] = p.table[i][j];

}

}

result = p.return\_number();

cout<<"Copied entry is: " << result << endl;

}

void TwoD::entry(int a, int b, double element){

row = a;

col = b;

table[row][col] = element;

}

double TwoD::return\_number() const{

return table[row][col];

}

int TwoD::getMaxRow() const{

return MaxRows;

}

int TwoD::getMaxCol() const{

return MaxCols;

}

int TwoD::getRow() const{

return row;

}

int TwoD::getCol() const{

return col;

}

TwoD::~TwoD(){

cout << "Object is being deleted" << endl;

delete [] table;

}

const TwoD operator +(const TwoD& TD\_1, const TwoD& TD\_2){

return TwoD(TD\_1.getMaxRow(), TD\_1.getMaxCol(), TD\_1.getRow(), TD\_1.getCol(), TD\_1.return\_number() + TD\_2.return\_number());

}

bool operator ==(const TwoD& TD\_1, const TwoD& TD\_2){

return ((TD\_1.getRow() == TD\_2.getRow())&&(TD\_1.getCol() == TD\_2.getCol())&&(TD\_1.getMaxRow() == TD\_2.getMaxRow())&&(TD\_1.getMaxCol() == TD\_2.getMaxCol()));

}

//Equal Operator

TwoD & TwoD::operator=(const TwoD &p){

cout<<"Copying constructors with overload operator..."<<endl;

MaxRows = p.getMaxRow();

MaxCols = p.getMaxCol();

table = new double\*[MaxRows];

for(int i=0; i < MaxRows; i++){

table[i] = new double[MaxCols];

}

for(int i = 0; i < MaxRows; i++) {

for(int j = 0; j < MaxCols; j++) {

table[i][j] = p.table[i][j];

}

}

result = p.return\_number();

cout<<"Copied entry is: " << result << endl;

return \*this;

}

//Main Function

int main(){

//Initialize the dynamic memories for the process of entering datasets, Create TD1 as default and TD2 with specific size

TwoD TD1(3,4,1,1,15.5), TD2(3,4,1,1,20);

if(TD1 == TD2){

TwoD TD3 = TD1 + TD2;

cout << "The sum of the two array with same location and size is:" << TD3.return\_number() << endl;

}

else{

printf("The size or columns/rows are not equal for TD1/TD2.");

exit(0);

}

//Declare default TwoD Constructors

TwoD TD4, TD5;

TD4.entry(80,20, 10.5);

TD5.entry(10,20, 5.5);

cout<<"Result with initialized constructor is: " << TD4.return\_number()<<endl;

cout<<"Result with initialized constructor is: " << TD5.return\_number()<<endl;

//Calling TD6 with copy constructor

TwoD TD6(TD4);

TwoD TD7(TD5);

//Copying Class with operator overloading

TwoD TD8,TD9;

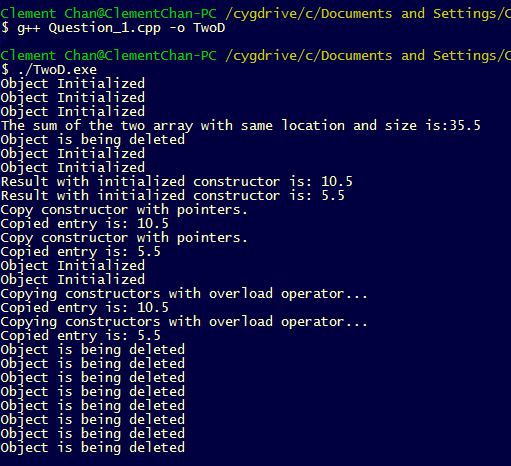
TD8 = TD4;

TD9 = TD5;

return 0;

};

**Output**



**Question 2 – Inverse Words with Pointers**

#include <iostream>

#include <stdio.h>

#include <cstring>

#include <cmath>

using namespace std;

int main(){

char \*head;

char \*tail;

string str;

cout << "Please enter the input that you want to inverse: "<<endl;

cin >> str;

int length = str.length()-1;

char word\_array[length + 1];

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

word\_array[i] = str[i];

}

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

\*head = word\_array[i];

\*tail = word\_array[length-i];

word\_array[i] = \*tail;

word\_array[length-i] = \*head;

}

cout << "The word entered inverse is: ";

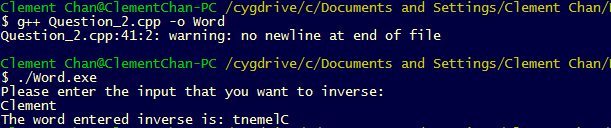
for (int i = 0; i <= length; i++){

cout << word\_array[i];

}

}

**Output**



**Question 3 – Student Class**

**Students.h**

#include <iostream>

#include <stdio.h>

#include <stddef.h>

#include <cstring>

class Student{

private:

std::string name;

int numclasses;

std::string \*classlist;

public:

void input();

void output();

void reset();

Student& operator =(const Student& p);

~Student();

};

**Students.cpp**

#include "Question\_3.h"

void Student::input(){

std::string a;

int b;

std::string c;

std::cout<<"Enter Student's Name: " << std::endl;

std::cin >> a;

name = a;

std::cout<<"Enter the number of classes he is studying: " << std::endl;

std::cin >> b;

numclasses = b;

classlist = new std::string[numclasses];

for (int i = 0; i < numclasses ; i++){

std::cout<<"Enter the class names: " << std::endl;

std::cin >> c;

classlist[i] = c;

}

}

void Student::output(){

std::cout<<"The name of the student is: " << name << std::endl;

std::cout<<"The number of classes he/she is taking is: " << numclasses << std::endl;

for(int i = 0; i < numclasses ; i++){

std::cout<<"The classes he/she is taking are: " << classlist[i] << std::endl;

}

}

void Student::reset(){

std::cout<<" " <<std::endl;

std::cout<<"Calling reset function, reseting list..."<<std::endl;

numclasses = 0;

delete [] classlist;

classlist = new std::string[numclasses];

std::cout<<"Reset Number of Classes to: " << numclasses << std::endl;

}

Student& Student::operator =(const Student& p){

std::cout<<" "<<std::endl;

std::cout<<"Overloading '=' operator, Copying...."<<std::endl;

name = p.name;

numclasses = p.numclasses;

classlist = new std::string[numclasses];

for(int i = 0; i < numclasses; i++){

classlist[i] = p.classlist[i];

}

}

Student::~Student(){

std::cout<<" "<<std::endl;

std::cout<<"Initializing destructor"<<std::endl;

delete[] classlist;

}

int main(){

//Initiating Student S and Student S1, S1 will be a copied class

Student S, S1;

S.input();

S.output();

//Copying using the operand "="

S1 = S;

S1.output();

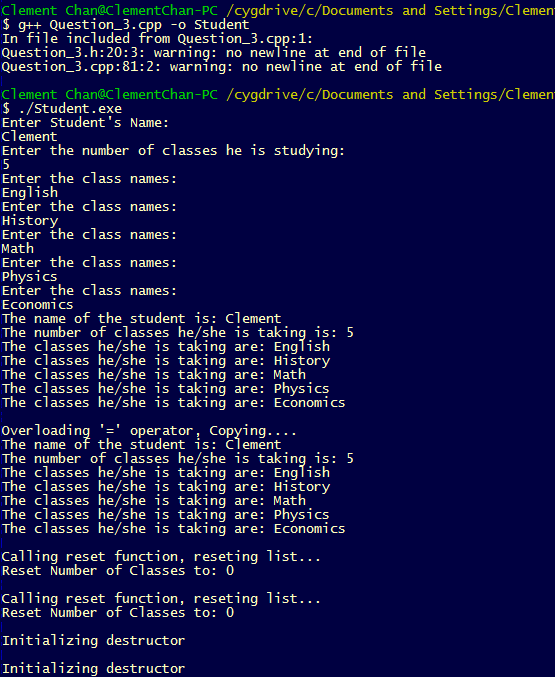
//Reseting

S.reset();

S1.reset();

}

**Output**



**Question 4 – Growing Dynamic Arrays**

**Growing Dynamic Arrays.h**

#include <iostream>

#include <stdio.h>

#include <stddef.h>

#include <cstring>

class DynamicStringArray{

private:

int size;

std::string \*dynamicarray;

public:

DynamicStringArray();

int return\_size();

void AddEntry(std::string);

void DeleteEntry(std::string);

std::string GetEntry(int);

DynamicStringArray& operator =(const DynamicStringArray &p);

~DynamicStringArray();

};

**Growing Dynamic Arrays.cpp**

#include "Question\_4.h"

//Constructor

DynamicStringArray::DynamicStringArray(){

size = 0;

dynamicarray = NULL;

}

//Add entry into the dynamic array and replay dynamicarray initialized

void DynamicStringArray::AddEntry(std::string a){

size += 1;

std::cout<<"Size is : " << size << std::endl;

std::string \*dynamicarray\_1;

dynamicarray\_1 = new std::string[size];

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

dynamicarray\_1[i] = dynamicarray[i];

}

dynamicarray\_1[size-1] = a;

dynamicarray = dynamicarray\_1;

for(int i=0 ; i < size ; i++){

std::cout<<dynamicarray[i]<<std::endl;

}

}

//Delete entry into the dynamic array and replay dynamicarray initialized

void DynamicStringArray::DeleteEntry(std::string b){

bool value;

int index;

int size\_1 = size-1;

std::string \*dynamicarray\_1;

dynamicarray\_1 = new std::string[size\_1];

for(int i = 0 ; i < size; i++){

if(dynamicarray[i] == b){

index = i;

value = true;

}

}

std::cout<<"The value is: " << value << std::endl;

std::cout<<"The index is: " << index << std::endl;

if(value == 1){

//Condition 1 if the chosen name is at the start of the array

if(index == 0){

for(int i = 1; i < size; i++){

dynamicarray\_1[i-1] = dynamicarray[i];

}

}

//Condition 2 if the chosen name is at the end of the array

if(index == size-1){

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

dynamicarray\_1[i] = dynamicarray[i];

}

}

//Condition 3 if the chosen name is in the middle of the array

if(index != 0 || index != (size-1)){

for(int i = 0; i < index; i++){

dynamicarray\_1[i] = dynamicarray[i];

}

for(int i = (index+1); i < size; i++){

dynamicarray\_1[i-1] = dynamicarray[i];

}

}

//Define the new dynamic array

dynamicarray = new std::string[size\_1];

//Copy all the value over from array\_1

for(int i = 0; i < size\_1; i++){

dynamicarray[i] = dynamicarray\_1[i];

}

}

else{

std::cout<<"There is no match for the delete entry.."<< std::endl;

}

std::cout<<"The list after deleting output is: ";

for(int i=0 ; i < size\_1 ; i++){

std::cout<<dynamicarray[i]<< " ";

}

std::cout<<" " << std::endl;

//Change the size of the array, mark down for the next delete

size = size\_1;

}

//Get Entry Function

std::string DynamicStringArray::GetEntry(int c){

return dynamicarray[c-1];

}

//Operator Overloading

DynamicStringArray& DynamicStringArray::operator =(const DynamicStringArray &p){

std::cout<<"Initiating Copy Constructor..."<<std::endl;

size = p.size;

dynamicarray = new std::string[size];

for(int i = 0; i < size; i++){

dynamicarray[i] = p.dynamicarray[i];

std::cout<<"The copied dynamic array strings are: " << dynamicarray[i] <<std::endl;

}

}

//Destructor

DynamicStringArray::~DynamicStringArray(){

std::cout<<"Initiating Destructors..."<<std::endl;

delete[] dynamicarray;

}

int main(){

DynamicStringArray DSA, DSA\_1;

DSA.AddEntry("Clem");

DSA.AddEntry("John");

DSA.AddEntry("Spencer");

DSA.AddEntry("Portia");

std::cout<<"The dynamic array with the entered index is: "<< DSA.GetEntry(3) << std::endl;

std::cout<<"The dynamic array with the entered index is: "<< DSA.GetEntry(1) << std::endl;

//Deleting Entry

DSA.DeleteEntry("Clem");

DSA.DeleteEntry("Portia");

//Copying

DSA\_1 = DSA;

return 0;

}

**Output**

