#include"SeqOfArr1D.h"

SeqOfArr1D::SeqOfArr1D()

{

size = 10;

seq\_arr = new Arr1D[size];

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

{

seq\_arr[i].setC(3);

seq\_arr[i].allocateArr(3);

seq\_arr[i].initializeArr(0);

}

}

SeqOfArr1D::SeqOfArr1D(int c, double value, int s) : size(s)

{

if (s < 1){

cout << "\nErr: size must be >= 1\n"; exit(0);

}

seq\_arr = new Arr1D[size];

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

{

seq\_arr[i].setC(c);

seq\_arr[i].allocateArr(c);

seq\_arr[i].initializeArr(value); //set it to value

}

}

SeqOfArr1D::SeqOfArr1D(const SeqOfArr1D & other\_seq1D)

{

seq\_arr = new Arr1D[other\_seq1D.getC()];

for (int c = 0; c < other\_seq1D.getC(); c++)

seq\_arr[c] = other\_seq1D.seq\_arr[c];

}

SeqOfArr1D& SeqOfArr1D::operator = (const SeqOfArr1D& right\_side)

{

Arr1D c;

if (size != right\_side.size)

{

delete[] seq\_arr;

size = right\_side.size;

seq\_arr = new Arr1D[size];

}

//for (int i = 0; i < right\_side.size; i++)

//{

// seq\_arr[i].setC(i);

// seq\_arr[i].allocateArr(i);

// seq\_arr[i].initializeArr(i);

//

//}

for (int c = 0; c < right\_side.size; c++)

{

seq\_arr[c] = right\_side.seq\_arr[c];

/\*seq\_arr[c] = right\_side.copyArr[seq\_arr[c]];\*/

}

return \*this;

}

ostream& operator <<(ostream& outs, const SeqOfArr1D & the\_array)

{

for (int c = 0; c < the\_array.size; c++)

{

outs << the\_array.seq\_arr[c] << " " << endl;

}

return(outs);

}

SeqOfArr1D::~SeqOfArr1D()

{

delete[]seq\_arr;

}

