2021-2 객체지향 프로그래밍과 자료구조 중간고사 1D

2021. 10. 2.
담당교수 김영탁

Section 1D. Class MtrxArray (25점)

1D.1 class MtrxArray

```
class MtrxArray
{
        friend ostream& operator<<(ostream&, const MtrxArray&);
public:
        MtrxArray(int array_size); // constructor
        ~MtrxArray();   // destructor
        Mtrx &operator[](int);
private:
        Mtrx *pMtrx;
        int mtrxArrySize;
        bool isValidIndex(int index);
};
```

1D.2 class MtrxArray 멤버함수 구현

- class MtrxArray에는 지정된 개수의 행렬 (matrix)을 배열로 저장하기 위하여 행렬의 동적 배열을 생성하며, 행렬 배열 크기는 mtrxArraySize에 설정한다.
- class MtrxArray의 생성자에는 배열의 크기(array_size)가 인수로 전달되며, 생성자에서 이 크기만큼을 수용할 수 있는 행렬 배열을 동적으로 생성하고, 그 주소를 class Mtrx의 포인터인 pMtrx 데이터 멤버가 가리키게 한다.
- class MtrxArray에는 배열 원소를 인덱스로 접근 및 설정할 수 있도록 [ ] 연산자 오버로딩이 구현된다. class MtrxArray의 배열 원소를 인덱스로 접근/설정할 때 인덱스의 값이 정상적인 범위에 있는가를 확인하기 위하여 isValidIndex() 함수가 구현된다.
  아울러, class MtrxArray의 배열 원소를 출력 연산자를 사용하여 출력할 수 있도록 연산자 오버로딩 operator<<() 함수가 구현된다.

1D.3 입력 데이터 파일

```
3  4
1.0 2.0 3.0 4.0
2.0 3.0 4.0 5.0
3.0 2.0 5.0 3.0

3  4
1.0 0.0 0.0 1.0
0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0

4  3
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
1.0 0.0 0.0
```

1D.4 main() 함수

```
#define NUM_MTRX 7

int main()
{
	ifstream fin;
	ofstream fout;

	int n_row, n_col;
	fin.open("Matrix_data.txt");
	if (fin.fail())
	{
		cout << "Error in opening input data file !!" << endl;
		exit;
	}

	fout.open("Result.txt");
	if (fout.fail())
	{
		cout << "Error in opening output data file !!" << endl;
		exit;
	}

	MtrxArray mtrx(NUM_MTRX);

	fin >> mtrx[0] >> mtrx[1] >> mtrx[2];
	mtrx[0].set_name("mtrx[0] =");
	mtrx[1].set_name("mtrx[1] =");
	mtrx[2].set_name("mtrx[2] =");
	mtrx[3] = mtrx[0] + mtrx[1];
	mtrx[3].set_name("mtrx[3] = mtrx[0] + mtrx[1] =");
	mtrx[4] = mtrx[0] - mtrx[1];
	mtrx[4].set_name("mtrx[4] = mtrx[0] - mtrx[1] =");
	mtrx[5] = mtrx[0] * mtrx[2];
	mtrx[5].set_name("mtrx[5] = mtrx[0] * mtrx[2] =");
	mtrx[6] = ~mtrx[5];
	mtrx[6].set_name("mtrx[6] = ~mtrx[5] =");

	fout << mtrx;

	fin.close();
	fout.close();

	return 0;
}
```

- main() 함수에서는 입력 데이터 파일 (Matrix_data.txt)과 출력파일 (Result.txt)를 사용할 수 있도록 준비한며, 7개의 행렬을 저장할 수 있는 행렬 배열을 class MtrxArray 객체를 동적으로 생성한다.
- 입력 파일로부터 3개의 행렬을 차례로 입력 받아 mtrx[0], mtrx[1], mtrx[2]에 대입한다.
- mtrx[0]과 mtrx[1]에 대한 행렬 덧셈과 뺄셈연산을 수행하여 mtrx[3], mtrx[4]에 저장한다.
- mtrx[0]과 mtrx[2]에 대한 행렬 곱셈 연산을 수행하여 mtrx[5]에 저장한다.
- 행렬 mtrx[5]의 치환행렬 (transposed matrix)를 계산하여 mtrx[6]에 대입한다.
- mtrx[0] ~mtrx[6]의 출력은 class Mtrx의 << 연산자 오버로딩을 사용하여 출력한다.

1D.5 프로그램 실행 결과 파일 출력 내용

```
mtrx[0] =
┌  1.00  2.00  3.00  4.00┐
│  2.00  3.00  4.00  5.00│
└  3.00  2.00  5.00  3.00┘
mtrx[1] =
┌  1.00  0.00  0.00  1.00┐
│  0.00  1.00  0.00  0.00│
└  0.00  0.00  1.00  0.00┘
mtrx[2] =
┌  1.00  0.00  0.00┐
│  0.00  1.00  0.00│
│  0.00  0.00  1.00│
└  1.00  0.00  0.00┘
mtrx[3] = mtrx[0] + mtrx[1] =
┌  2.00  2.00  3.00  5.00┐
│  2.00  4.00  4.00  5.00│
└  3.00  2.00  6.00  3.00┘
mtrx[4] = mtrx[0] - mtrx[1] =
┌  0.00  2.00  3.00  3.00┐
│  2.00  2.00  4.00  5.00│
└  3.00  2.00  4.00  3.00┘
mtrx[5] = mtrx[0] * mtrx[2] =
┌  5.00  2.00  3.00┐
│  7.00  3.00  4.00│
└  6.00  2.00  5.00┘
mtrx[6] = ~mtrx[5] =
┌  5.00  7.00  6.00┐
│  2.00  3.00  2.00│
└  3.00  4.00  5.00┘
```