자료구조 HW3

B711222 박조은
Hongik University
mrnglory@mail.hongik.ac.kr

October 14, 2019

I. List of Source Files

- hw3a
  - matrixa.h
  - matrixa.cpp
  - hw3a.cpp
- hw3b
  - matrixb.h
  - matrixb.cpp
  - hw3b.cpp

II. Reports

i. 연산자 오버로딩의 개념 및 구현방법

/*
* operand 로서 class 의 instance 로 표현해야 할 때, 이를테면 단순히 *this + b 의 형식으로는 원하는 연산을 도출해낼 수 없다.
* 따라서 *this 와 b 의 반환값을 각 연산자에 대해 정의 해주어야 하며, 이러한 함수들의 정의는 matrixa.cpp 에서 찾아볼 수 있다.
* 이처럼 context 에 걸맞게 기존 operator 를 재사용 하되 새로운 의미를 부여하는 것을 연산자 오버로딩이라고 한다.

*/

ii. matrixa.h 와 matrixa.cpp 의 차이점

/*
* matrixa.h 는 hw3a.cpp 와 matrixa.cpp 에서 include 되는 header file 이다.
* Matrix 타입의 배열, 즉 각 원소 값들이 초기화 되어 있는 형태의 멤버변수 행렬이 선언되며, 해당 코드에서는 멤버 이니셜라이저의 꼴로 나타나있다.
* 또한 연산자 오버로딩에 관한 멤버함수들도 선언 되어있다.

*/

iii. 3 * 3 행렬 함수 구현 중 어려웠던 점

/* 다음 내용은 3 * 3 행렬 함수 구현에서만 해당되는 것은 아니다.
* 객체 개념에 대해 잘 모른 채로 코드를 작성 해나갈 때, Matrix 행렬명 이런 식으로 객체를 생성 했었다면 return 값으로 일일이 원소들을 나열 할 필요는 없었을 것이다.
* 행렬의 곱셈 함수에서 3중 for문을 이용하였는데, 그 결과 값을 출력할 때 index 값의 순서를 헷갈려서 잘 못 적은 것에서 시간이 많이 소요됐었다. 또한 해당 함수 안에서 임의의 행렬을 초기화하지 않아서 원하는 결과가 나타나지 않았었다.
* 또한 전치행렬 구현에서, 임의의 배열에 해당 행렬 값을 복사하고 전치한 값을 다시 해당 행렬에 대입하는 과정을 하나의 2중 for 문에서 일괄적으로 처리하려 했었는데, 어려웠던 점 첫 번째 이유와 같은 맥락으로, 임의의 행렬을 선언해놓고 초기화를 하지 않아서 남아있던 쓰레기 값에 의해 전치를 하는 과정에서 원하는 결과가 나타나지 않았었다.

*/

III. hw3a

i. matrixa.h

```
1 #ifndef MATRIX_H
2 #define MATRIX_H
3
4 #include <iostream>
```

```
5
6   class Matrix
7   {
8   public:
9           Matrix(int a = 0, int b = 0, int c = 0, int d = 0);
10          ~ Matrix() {}
11
12          void ShowMatrix();
13          void Transpose();
14          Matrix operator + (const Matrix& a);
15          Matrix operator − (const Matrix& a);
16          Matrix operator ∗ (const Matrix& a);
17
18          void operator = (const Matrix& a);
19
20  private:
21          int m[2][2];
22  };
23
24  #endif
```

```
/*
 * line 9: Matrix class 의 멤버 이니셜라이저이며, 이를 통한 초기화는 객체 생성 이전에 이루어진다.
 * line 12 - 13: 멤버함수 선언
 * line 14 - 16: operator overloading of plus, minus, and multiply
 * line 18: void 타입의 assignment operator overloading 이다. 별도의 반환값이 존재하지 않는 전치행렬 연산을 위함이다.
 * line 21: private 타입의 2 /* 2 배열 선언

*/
```

ii. matrixa.cpp

```
1   #include <iostream>
2   #include "matrixa.h"
3   using namespace std;
4
5   Matrix::Matrix(int a, int b, int c, int d)
6   {
7           int arr[2][2] = {a, b, c, d};
8
9           for (int i = 0; i < 2; i++)
10                  for (int j = 0; j < 2; j++)
11                          m[i][j] = arr[i][j];
12  }
13
14  void Matrix::Transpose()
15  {
16          int arr[2][2];
17
18          for (int i = 0; i < 2; i++)
19                  for (int j = 0; j < 2; j++)
20                          arr[i][j] = m[i][j];
21
22          for (int i = 0; i < 2; i++)
23                  for (int j = 0; j < 2; j++)
24                          m[i][j] = arr[j][i];
25  }
26
27  Matrix Matrix::operator + (const Matrix &a)
28  {
29          int arr[2][2];
30
31          for (int i = 0; i < 2; i++)
32                  for (int j = 0; j < 2; j++)
33                          arr[i][j] = m[i][j] + a.m[i][j];
34
35          return Matrix (arr[0][0], arr[0][1], arr[1][0], arr
                ↪ [1][1]);
36  }
37
38  Matrix Matrix::operator − (const Matrix &a)
39  {
40          int arr[2][2];
41
42          for (int i = 0; i < 2; i++)
43                  for (int j = 0; j < 2; j++)
44                          arr[i][j] = m[i][j] − a.m[i][j];
45
46          return Matrix (arr[0][0], arr[0][1], arr[1][0], arr
                ↪ [1][1]);
47  }
48
49  Matrix Matrix::operator ∗ (const Matrix &a)
50  {
51          int arr[2][2] = {0, 0, 0, 0};
52
53          for (int i = 0; i < 2; i++)
54                  for (int j = 0; j < 2; j++)
55                          for (int k = 0; k < 2; k++)
56                                  arr[i][k] += m[i][j] ∗ a.m[
                                        ↪ j][k];
57
58          return Matrix (arr[0][0], arr[0][1], arr[1][0], arr
                ↪ [1][1]);
59  }
60
61  void Matrix::operator = (const Matrix &a)
62  {
63          for (int i = 0; i < 2; i++)
64                  for (int j = 0; j < 2; j++)
65                          m[i][j] = a.m[i][j];
66  }
```

```
67
68  void Matrix::ShowMatrix()
69  {
70          for (int i = 0; i < 2; i++)
71          {
72                  for (int j = 0; j < 2; j++)
73                          cout << m[i][j] << " ";
74                  cout << endl;
75          }
76  }
77
78  \newpage
```

```
/*
 * line 5: Matrix 객체 생성
 * line 7: 원소를 임시 배열에 저장한다. 이 때 Matrix a 이런 식으로 선언 했으면 객체를 더 잘 활용했다고 말할 수 있었을 것이다.
 * line 9 - 11: 입력 받은 Matrix 의 각 원소들을 저장한다.
 * line 14: *this 의 Transpose 연산을 위한 객체 생성
 * line 16: 입력 받은 Matrix 값을 복사하기 위한 임시 배열 arr 를 선언한다.
 * line 18 - 20: 입력 받은 Matrix 값을 임시 배열 arr 에 복사한다.
 * line 22 - 24: Matrix 값을 들고 있는 배열 arr 를 Transpose 시킨 것을 다시 원래 그 값을 들고 있어야 할 배열에 할당시킨다.
 * line 27: *this 와 a 의 + 연산 결과 값 반환
 * line 31 - 33: *this 와 a 의 덧셈 연산
 * line 35: 결과값을 return 한다.
 * line 29, 35: 만약 int arr[2][2] 대신 Matrix b 라고 표현 했다면, return b 라고 간단하게 작성할 수 있었을 것이다.
 * line 38: *this 와 a 의 - 연산 결과 값 반환
 * line 40 - 46: 위의 plus 구현 코드에서 연산자만 바뀐 것으로, 설명을 생략한다.
 * line 49: *this 와 a 의 * 연산 결과 값 반환
 * line 51: 곱셈 연산을 위해 3중 for 문을 실행시키는 과정에서, 각 원소 간의 덧셈 및 곱셈 연산 결과로서 이상한 값이 나오는 것을 방지하기 위해 임시 배열을 초기화하여 쓰레기 값을 제거한다.
 * line 53 - 56: 임시 배열에 *this 와 a 의 곱셈 연산 값을 저장한다.
 * line 58: 해당 값을 반환한다.
 * line 61: *this 의 assignment 값 저장. return 할 값이 없다.
 * line 63 - 65: *this 에 a 를 할당한다.
 * line 68: *this 의 출력
 * line 70 - 74: *this 의 각 열은 공백으로 구분, 각 행은 개행으로 구분하여 출력한다.

*/
```

iii. hw3a.cpp

```
1   #include "matrixa.h"
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7           Matrix matrix1(1, 2, 3, 4);
8           Matrix matrix2(1, 1, 1, 1);
9           Matrix matrix3;
10
11          cout << "--------------------" << endl;
12          cout << "Matrix Transpose" << std::endl;
13          cout << "--------------------" << endl;
14
15          matrix1.Transpose();
16          matrix1.ShowMatrix();
17          matrix1.Transpose();
18
19          cout << "--------------------" << endl;
20          cout << "Matrix Add" << endl;
21          cout << "--------------------" << endl;
22
23          matrix3 = matrix1 + matrix2;
24          matrix3.ShowMatrix();
25
26          cout << "--------------------" << endl;
27          cout << "Matrix Sub" << endl;
28          cout << "--------------------" << endl;
29
30          matrix3 = matrix1 - matrix2;
31          matrix3.ShowMatrix();
32
33          cout << "--------------------" << endl;
34          cout << "Matrix Multi" << endl;
35          cout << "--------------------" << endl;
36
37          matrix3 = matrix1 * matrix2;
38          matrix3.ShowMatrix();
39
40          cout << "--------------------" << endl;
41
```

```
42          return 0;
43  }
```

```
/*
 * line 7 - 9: Matrix 타입 matrix1, matrix2, matrix3 생성
 * line 15 - 17: matrix1 에 대해 Transpose(), ShowMatrix() 함수를 실행시킨다.
 * line 23 - 24: matrix1 + matrix2 결과를 matrix3 에 저장 및 해당 값 출력
 * line 30 - 31: matrix1 - matrix2 결과를 matrix3 에 저장 및 해당 값 출력
 * line 37 - 38: matrix1 * matrix2 결과를 matrix3 에 저장 및 해당 값 출력
*/
```

iv. Results

iv.1 makefile

```
1  hw3a: hw3a.o matrixa.o
2          g++ -o hw3a hw3a.o matrixa.o
3  hw3a.o matrixa.o: matrixa.h
```

iv.2 compile

```
1  [B711222@localhost hw3d]$ ./hw3a
2  ----------------
3  Matrix Transpose
4  ----------------
5  1 3
6  2 4
7  ----------------
8  Matrix Add
9  ----------------
10 2 3
11 4 5
12 ----------------
13 Matrix Sub
14 ----------------
15 0 1
16 2 3
17 ----------------
18 Matrix Multi
19 ----------------
20 3 3
21 7 7
22 ----------------
```

IV. HW3B

i. rectb.h

```
#ifndef MATRIX_H
#define MATRIX_H
#include <iostream>
using namespace std;

class Matrix
{
public:
        Matrix(int a = 0, int b = 0, int c = 0, int d = 0, int e = 0, int f = 0, int g = 0, int h = 0, int i = 0);
        ~ Matrix() {}

        void ShowMatrix();
        void Transpose();
        Matrix operator + (const Matrix& a);
        Matrix operator − (const Matrix& a);
        Matrix operator ∗ (const Matrix& a);

        void operator = (const Matrix& a);

private:
        int m[3][3];
};

#endif
```

/* hw3a 와 동일한 부분은 설명을 생략하도록 한다.
 * line 9: 3 * 3 Matrix 에 대한 멤버 이니셜라이저이므로, 인자의 갯수는 총 9 개이다.
 * private 타입 멤버 변수가 3 * 3 크기의 배열이다.

*/

ii. matrixb.cpp

```
#include <iostream>
#include "matrixb.h"
using namespace std;

Matrix::Matrix(int a, int b, int c, int d, int e, int f, int g, int h, int i)
{
        int arr[3][3] = {a, b, c, d, e, f, g, h, i};

        for (int i = 0; i < 3; i++)
                for (int j = 0; j < 3; j++)
                        m[i][j] = arr[i][j];
}

void Matrix::Transpose()
{
        int arr[3][3];

        for (int i = 0; i < 3; i++)
                for (int j = 0; j < 3; j++)
                        arr[i][j] = m[i][j];

        for (int i = 0; i < 3; i++)
                for (int j = 0; j < 3; j++)
                        m[i][j] = arr[j][i];
}

Matrix Matrix::operator + (const Matrix &a)
{
        int arr[3][3];

        for (int i = 0; i < 3; i++)
                for (int j = 0; j < 3; j++)
                        arr[i][j] = m[i][j] + a.m[i][j];

        return Matrix (arr[0][0], arr[0][1], arr[0][2], arr[1][0], arr[1][1], arr[1][2], arr[2][0], arr[2][1], arr[2][2]);
}

Matrix Matrix::operator − (const Matrix &a)
{
        int arr[3][3];

        for (int i = 0; i < 3; i++)
                for (int j = 0; j < 3; j++)
                        arr[i][j] = m[i][j] − a.m[i][j];

        return Matrix (arr[0][0], arr[0][1], arr[0][2], arr[1][0], arr[1][1], arr[1][2], arr[2][0], arr[2][1], arr[2][2]);

}

Matrix Matrix::operator ∗ (const Matrix &a)
{
        int arr[3][3] = {0, 0, 0, 0, 0, 0, 0, 0, 0};

        for (int i = 0; i < 3; i++)
                for (int j = 0; j < 3; j++)
                        for (int k = 0; k < 3; k++)
                                arr[i][k] += m[i][j] ∗ a.m[j][k];

        return Matrix (arr[0][0], arr[0][1], arr[0][2], arr[1][0], arr[1][1], arr[1][2], arr[2][0], arr
```

```
                ↪ [2][1], arr[2][2]);
60
61  }
62
63  void Matrix::operator = (const Matrix &a)
64  {
65          for (int i = 0; i < 3; i++)
66                  for (int j = 0; j < 3; j++)
67                                  m[i][j] = a.m[i][j];
68  }
69
70  void Matrix::ShowMatrix()
71  {
72          for (int i = 0; i < 3; i++)
73          {
74                  for (int j = 0; j < 3; j++)
75                          cout << m[i][j] << " ";
76                  cout << endl;
77          }
78  }
```

```
/* hw3a 와 동일한 부분은 설명을 생략하도록 한다.
 * Matrix 의 크기가 2 * 2 이든 3 * 3 이든, 연산 과정은 동일하다. (배열의 크기 및 반복문 수행 횟수 제외)

*/
```

iii. hw3b.cpp

```
1   #include "matrixb.h"
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7           Matrix matrix1(1, 2, 3, 4, 5, 6, 7, 8, 9);
8           Matrix matrix2(1, 1, 1, 1, 1, 1, 1, 1, 1);
9           Matrix matrix3;
10
11          cout << "--------------------" << endl;
12          cout << "Matrix Transpose" << std::endl;
13          cout << "--------------------" << endl;
14
15          matrix1.Transpose();
16          matrix1.ShowMatrix();
17          matrix1.Transpose();
18
19          cout << "--------------------" << endl;
20          cout << "Matrix Add" << endl;
21          cout << "--------------------" << endl;
22
23          matrix3 = matrix1 + matrix2;
24          matrix3.ShowMatrix();
25
26          cout << "--------------------" << endl;
27          cout << "Matrix Sub" << endl;
28          cout << "--------------------" << endl;
29
30          matrix3 = matrix1 - matrix2;
31          matrix3.ShowMatrix();
32
33          cout << "--------------------" << endl;
34          cout << "Matrix Multi" << endl;
35          cout << "--------------------" << endl;
36
37          matrix3 = matrix1 * matrix2;
38          matrix3.ShowMatrix();
39
40          cout << "--------------------" << endl;
41
42          return 0;
43  }
```

```
/* hw3a 와 동일한 부분은 설명을 생략하도록 한다.
 * Matrix 의 크기가 2 * 2 이든 3 * 3 이든, 연산 과정은 동일하다. (배열의 크기 및 반복문 수행 횟수 제외)

*/
```

iv. results

iv.1 makefile

```
1   hw3b: hw3b.o matrixb.o
2           g++ -o hw3b hw3b.o matrixb.o
3   hw3b.o matrixb.o: matrixb.h
```

iv.2 compile

```
1   [B711222@localhost hw3d]$ ./hw3b
2   --------------------
3   Matrix Transpose
4   --------------------
5   1 4 7
6   2 5 8
7   3 6 9
8   --------------------
9   Matrix Add
10  --------------------
11  2 3 4
12  5 6 7
```

```
8 9 10
----------------
Matrix Sub
----------------
0 1 2
3 4 5
6 7 8
----------------
Matrix Multi
----------------
6 6 6
15 15 15
24 24 24
----------------
```