

자료구조 보고서

Homework#4

학과 : 소프트웨어학과

학번 : 2018038067

이름 : 정재민

git URL: <https://github.com/jmjung1997/Homework4.git>

(a) 내용

동적 배열을 통해 다차원 배열을 만들고 행렬의 덧셈, 뺄셈, 전치, 곱셈을 계산하는 프로그램을 만들었습니다. 먼저 다차원 배열을 더블 포인터를 이용하여 만들었습니다.

처음에 `int** matrix = malloc(sizeof(int*) * row);`을 통해 더블 포인터 변수에 주소의 크기를 행의 개수만큼 만듭니다. 그 후

```
for (int i = 0; i < row; i++)  
    matrix[i] = malloc(sizeof(int) * col);
```

을 통해 각 행에 해당하는 정수형 사이즈 메모리를 열의 개수만큼 할당해줍니다. 프로그램을 종료하기 직전에는 할당된 메모리를 행부터 해제하고 열을 해제 해줍니다.

`srand(time)`을 통해 런타임 동안 항상 변하는 난수를 만들고 `rand()`함수를 통해 항상 랜덤으로 행렬의 데이터를 생성할 수 있습니다.

메뉴 실행은 `switch` 구문을 통해 각 함수들을 호출하여 실행하였습니다. 먼저 `z`를 누르면 행렬 초기화 함수가 실행되고 `fill_data` 함수를 호출하여 각 행렬에 데이터를 넣습니다.

`p`를 누르면 이중 반복문을 통하여 행렬의 각 데이터를 출력한다. `a`를 누르면 `addition_matrix(matrix_a, matrix_b, row, col)`를 호출하여 `matrix_a`와 `matrix_b`를 더합니다. `s`를 누르면 `subtraction_matrix(matrix_a, matrix_b, row, col);`를 호출하여 `matrix_a`와 `matrix_b`를 뺍니다. `t`를 누르면 `matrix_a`행렬을 전치하게 되는데 이중 반복문을 각 행과 열을 전치 시켜줍니다. `m`를 누르면 `matrix_a`행렬과 전치된 `matrix_a`행렬을 서로 곱합니다. 마지막으로 `q`를 누르면 동적 할당 된 모든 메모리를 해제 시켜주고 반복문을 종료합니다.

(b) 코드

Homework4.c
<pre>#include <stdio.h> #include <stdlib.h> #include <time.h> /* Method Declaration */ int** create_matrix(int row, int col); void print_matrix(int** matrix, int row, int col); int free_matrix(int** matrix, int row, int col); int fill_data(int** matrix, int row, int col); int addition_matrix(int** matrix_a, int** matrix_b, int row, int col); int subtraction_matrix(int** matrix_a, int** matrix_b, int row, int col); int transpose_matrix(int** matrix, int** matrix_t, int row, int col); int multiply_matrix(int** matrix_a, int** matrix_t, int row, int col); int main() {</pre>

```

char command;
printf("[----- [정재민] [2018038067] -----]\n");

int row, col;
srand(time(NULL));

printf("Input row and col : ");
scanf("%d %d", &row, &col); //행과 열을 입력 받는다.

while (row <= 0 || col <= 0)//행과 열을 0이하 입력 받았을 때 다시 입력 받는다.
{
    printf("다시 입력하세요 : ");
    scanf("%d %d", &row, &col);
}

int** matrix_a = create_matrix(row, col);/*동적할당을 통해 배열공간 생성*/
int** matrix_b = create_matrix(row, col);/*동적할당을 통해 배열공간 생성*/
int** matrix_a_t = create_matrix(col, row);/*동적할당을 통해 배열공간 생성*/

printf("Matrix Created.\n");

if (matrix_a == NULL || matrix_b == NULL) { return -1; }

do {
    /*메뉴 출력*/

printf("-----\n");
printf("Matrix Manipulation\n");

printf("-----\n");

printf(" Initialize Matrix    = z          Print Matrix    = p \n");
printf(" Add Matrix            = a          Subtract Matrix    = s \n");
printf(" Transpose matrix_a     = t          Multiply Matrix    = m \n");
printf(" Quit                  = q \n");

```

```

printf("-----\n");

printf("Command = ");
scanf(" %c", &command); //switch명령어 입력

switch (command)
{
case 'z': case 'Z': //행렬 초기화 명령
    printf("Matrix Initialized\n");
    fill_data(matrix_a, row, col); //fill_data함수를 이용해서 데이터 넣기
    fill_data(matrix_b, row, col); //fill_data함수를 이용해서 데이터 넣기
    break;
case 'p': case 'P': //matrix_a, matrix_b 행렬 출력
    printf("Print matrix\n\n\n");
    printf("matrix_a\n\n");
    print_matrix(matrix_a, row, col);
    printf("\nmatrix_b\n\n");
    print_matrix(matrix_b, row, col);
    break;
case 'a': case 'A': //matrix_a, matrix_b 행렬 덧셈
    printf("Add two matrices\n");
    addition_matrix(matrix_a, matrix_b, row, col);
    break;
case 's': case 'S': //matrix_a, matrix_b 행렬 뺄셈
    printf("Subtract two matrices \n");
    subtraction_matrix(matrix_a, matrix_b, row, col);
    break;
case 't': case 'T': //matrix_a 행렬 전치
    printf("Transpose matrix_a \n");
    printf("matrix_a\n");
    int **matrix_t=transpose_matrix(matrix_a, matrix_a_t, col, row);
    print_matrix(matrix_t, col, row);
    break;
case 'm': case 'M': //matrix_a와 matrix_a_t의 곱
    printf("Multiply matrix_a with transposed matrix_a \n");
    transpose_matrix(matrix_a, matrix_a_t, col, row);
    multiply_matrix(matrix_a, matrix_a_t, row, col);
    break;
case 'q': case 'Q': //동적할당 된 메모리들을 해제

```

```

        printf("Free all matrices..\n");
        free_matrix(matrix_a_t, col, row);
        free_matrix(matrix_a, row, col);
        free_matrix(matrix_b, row, col);

        break;
    default:
        printf("\n      >>>>>   Concentration!!   <<<<<   \n");
        break;
    }

    } while (command != 'q' && command != 'Q');

    return 1;
}

/* create a 2d array whose size is row x col using malloc() */
int** create_matrix(int row, int col) //행렬 동적할당 함수
{
    int** matrix = malloc(sizeof(int*) * row); //더블 포인터를 이용하여 주소의 크기를 행수만큼 할당
    for (int i = 0; i < row; i++)
        matrix[i] = malloc(sizeof(int) * col); //각 행에 정수형 사이즈만큼 열을 배열한다.

    return matrix;
}

/* print matrix whose size is row x col */
void print_matrix(int** matrix, int row, int col) //행렬 출력함수
{

```

```

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            printf("%2d  ", matrix[i][j]);
        }
        printf("\n");
    }
}

/* free memory allocated by create_matrix() */
int free_matrix(int** matrix, int row, int col) //할당된 메모리 해제함수
{
    for (int i = 0; i < row; i++) //먼저 각 행에 대한 메모리부터 해제한다.
    {
        free(matrix[i]);
    }
    free(matrix); //열에 해당하는 공간 메모리를 해제한다.
}

/* assign random values to the given matrix */
int fill_data(int** matrix, int row, int col) //행렬에 데이터를 넣는다.
{
    for (int i = 0; i < row; i++)
        for (int j = 0; j < col; j++)
            matrix[i][j] = rand() % 20; //rand 함수를 이용하여 행렬에 데이터를 넣는다.
    return matrix;

    if (matrix == NULL) //만약 행렬 전부 값이 0이 될 경우일때 fill_data함수를 다시 호출한다.
    {
        fill_data(matrix, row, col);
    }
}

```

```

    }
    return matrix;

}

/* matrix_sum = matrix_a + matrix_b */
int addition_matrix(int** matrix_a, int** matrix_b, int row, int col)//두 개의 행렬을 더하는 함수
{
    int** matrix_sum = malloc(sizeof(int*) * row); /*행렬을 더해서 저장 할 수 있는 변수를 동적할당을 통해 생성*/

    for (int i = 0; i < row; i++)
        matrix_sum[i] = malloc(sizeof(int) * col);

    for (int i = 0; i < row; i++)
        for (int j = 0; j < col; j++)
        {
            matrix_sum[i][j] = matrix_a[i][j] + matrix_b[i][j];
        }

    print_matrix(matrix_sum, row, col);//더한 행렬 출력하기

    free_matrix(matrix_sum, row, col);//matrix_sum 할당된 메모리 해제 하기
}

/* matrix_sub = matrix_a - matrix_b */
int subtraction_matrix(int** matrix_a, int** matrix_b, int row, int col)//두 개의 행렬을 빼는 함수
{
    int** matrix_sub = malloc(sizeof(int*) * row);/*행렬을 빼서 저장 할 수 있는 변수를 동적할당을 통해 생성*/

    for (int i = 0; i < row; i++)
        matrix_sub[i] = malloc(sizeof(int) * col);

    for (int i = 0; i < row; i++)
        for (int j = 0; j < col; j++)
        {

```

```

        matrix_sub[i][j] = matrix_a[i][j] - matrix_b[i][j];
    }
    print_matrix(matrix_sub, row, col); //뺀 행렬 출력하기

    free_matrix(matrix_sub, row, col); //matrix_sub 할당된 메모리 해제 하기
}

/* transpose the matrix to matrix_t */
int transpose_matrix(int** matrix, int** matrix_t, int row, int col) //전치 행렬 함수
{

    for (int j = 0; j < row; j++)
        for (int i = 0; i < col; i++)
            matrix_t[j][i] = matrix[i][j]; //각 열과 각 행의 있는 데이터를 서로 바꾼
다.

    return(matrix_t); //전치된 함수 리턴

}

/* matrix_axt = matrix_a x matrix_t */
int multiply_matrix(int** matrix_a, int** matrix_t, int row, int col) //matrix_a와 전치
된 matrix_t 곱 구하기
{
    int mul = 0;
    int** matrix_multi = malloc(sizeof(int*) * row); //두 행렬을 곱해서 저장 할 수
있는 변수를 동적할당을 통해 생성*/

    for (int i = 0; i < row; i++)
        matrix_multi[i] = malloc(sizeof(int) * row);

    for (int t = 0; t < row; t++) // matrix_a의 행 반복
    {
        for (int i = 0; i < row; i++) //matrix_t의 열 반복
        {
            mul = 0;
            for (int j = 0; j < col; j++) //matrix_a 열과 matrix_b의 행 반복

```



```

        {
            mul += matrix_a[t][j] * matrix_t[j][i]; //두 행렬의 데이터 곱

        }
        matrix_multi[t][i] = mul; //각 행,열에 해당하는 데이터 곱을
matrix_multi에 저장
    }

}

print_matrix(matrix_multi, row, row);
free_matrix(matrix_multi, row, row); //동적 메모리 할당 해제
}

```

(c) 그림

행렬 초기화(z실행시).jpg

```

> Executing task: cmd /C 'C:\Users\ash J\Desktop\새 폴더 (2)\Homework4' <

[----- [절재민] [2018038067] -----]
Input row and col : 3 4
Matrix Created.

-----
                        Matrix Manipulation
-----
Initialize Matrix   = z          Print Matrix         = p
Add Matrix          = a          Subtract Matrix      = s
Transpose matrix_a  = t          Multiply Matrix       = m
Quit                = q

-----
Command = z
Matrix Initialized

```

행렬 출력(p실행시).jpg

```
-----
                        Matrix Manipulation
-----
Initialize Matrix   = z          Print Matrix           = p
Add Matrix          = a          Subtract Matrix        = s
Transpose matrix_a  = t          Multiply Matrix         = m
Quit                = q
-----
Command = p
Print matrix

matrix_a

11 13  8 12
14  6 10 15
14 10 15 13

matrix_b

15  7  1  2
 4  4  4 13
19  9 16 15
-----
```

행렬 덧셈(a실행시).jpg

```
-----
                        Matrix Manipulation
-----
Initialize Matrix   = z          Print Matrix           = p
Add Matrix          = a          Subtract Matrix        = s
Transpose matrix_a  = t          Multiply Matrix         = m
Quit                = q
-----
Command = a
Add two matrices
26 20  9 14
18 10 14 28
33 19 31 28
-----
```

행렬 뺄셈(s실행시).jpg

```
-----  
Matrix Manipulation  
-----  
Initialize Matrix = z      Print Matrix      = p  
Add Matrix        = a      Subtract Matrix   = s  
Transpose matrix_a = t      Multiply Matrix    = m  
Quit              = q  
-----  
Command = s  
Subtract two matrices  
-4  6  7 10  
10  2  6  2  
-5  1 -1 -2  
-----
```

행렬 전치(t실행시).jpg

```
-----  
Matrix Manipulation  
-----  
Initialize Matrix = z      Print Matrix      = p  
Add Matrix        = a      Subtract Matrix   = s  
Transpose matrix_a = t      Multiply Matrix    = m  
Quit              = q  
-----  
Command = t  
Transpose matrix_a  
matrix_a  
11 14 14  
13  6 10  
 8 10 15  
12 15 13  
-----
```

행렬 곱(m실행시).jpg

```
-----  
Matrix Manipulation  
-----  
Initialize Matrix = z      Print Matrix      = p  
Add Matrix        = a      Subtract Matrix   = s  
Transpose matrix_a = t      Multiply Matrix    = m  
Quit              = q  
-----  
Command = m  
Multiply matrix_a with transposed matrix_a  
498 492 560  
492 557 601  
560 601 690  
-----
```

할당된 메모리 해제 및 프로그램 종료(q실행시).jpg

```
-----  
Matrix Manipulation  
-----  
Initialize Matrix = z      Print Matrix      = p  
Add Matrix       = a      Subtract Matrix   = s  
Transpose matrix_a = t      Multiply Matrix    = m  
Quit             = q  
-----  
Command = q  
Free all matrices..  
The terminal process "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -Command cmd /C 'C:\Users\ash J\Desktop\새 폴더 (2)\Homework4'" terminated with exit code: 1.
```