실습과제 1.2

[과제 설명]

정수 값들로 구성된 2차원 n x n 행렬이 배열 A에 저장되어있을 때, 입력으로 주어진 2차원 행렬 A의 전치행렬 B를 출력한다. 이 때, 전치 행렬은 행과 열의 값이 뒤바뀐 행렬을 뜻한다. 화면에 출력할 때는 원본인 A 행렬과 전치행렬인 B 행렬을 차례로 출력한다.

이 때 조건은 행렬의 크기인 5를 #define으로 정의하며, A, B 행렬은 메인 함수 내에 선언하고 아래의 두 함수를 이용한다.

void Transpose_Mat(int A[][MAT_SIZE]);
전치행렬을 계산해주는 함수
void Print_MAT(int arr[][MAT_SIZE]);
행렬 arr를 출력해주는 함수

배열 인수로 넣는 함수이기 때문에 return 값이 따로 필요없다.

Transpose_Mat 함수로 전치행렬을 구할 때는 중첩반복문을 이용해 순차적으로 B에 행과 열을 바꾼채로 대입하기만 하면 된다. 행렬의 크기와 배열이 main에 주어진채로 실행하기 때문에, scanf를 통한 입력이나 B를 malloc 등으로 동적할당할 필요는 없다.

Print_Mat 함수는 실습할 때 미리 완성된 것이 주어져 있었기 때문에 설명을 생략한다.

[코드]

```
#include <stdio.h>
#define MAT_SIZE 5
//배열의 크기
void Transpose_Mat(int A[][MAT_SIZE], int B[][MAT_SIZE]) {
  //A는 원본 행렬, B는 전치행렬이 될 빈 행렬
  for (int i = 0; i < MAT_SIZE; i++) {
    for (int j = 0; j < MAT_SIZE; j++) {
      B[i][j] = A[j][i]; //행과 열을 바꿔서 대입
    }
  }
}
void Print_MAT(int B[][MAT_SIZE]) {
  for (int i = 0; i < MAT_SIZE; i++) {
    for (int j = 0; j < MAT_SIZE; j++) {
      printf("%d", B[i][j]);
```

```
    }
    puts(""); //행이 바뀔 때마다 줄바꿈
  } //반복문을 통해 행렬을 한 줄씩 출력한다.
}
int main(void) {
  int A[MAT_SIZE][MAT_SIZE] = { {3,2,6,4,5},
    {8,3,5,9,1},
    {0,3,2,7,9},
    {2,1,5,2,4},
    {5,0,8,2,3} }; //원본 행렬

  int B[MAT_SIZE][MAT_SIZE];

  Print_MAT(A); //원본 행렬 출력
  puts(""); //줄바꿈
  Transpose_Mat(A, B); //전치 행렬을 만들어주는 함수 실행
  Print_MAT(B); //전치 행렬 출력
}
```

[과제에 대한 고찰]

전치행렬은 비교적 간단한 알고리즘으로 구현이 가능하다. 과제 1.1에서 알고리즘에 대해서는 대부분 다루기도 했기 때문에 이 과제의 고찰란에는 문법에 관련되어 있는 것을 위주로 적는다.

과제의 함수 조건에 arr[][상수]라고 되어있는데, 이것을 둘 다 비우고 arr[][]라고 적어서 동작시키면 제대로 되지 않는 것을 찾을 수 있었다. 그래서 2차원 배열을 함수의 인수로 넘기는 것이 설명된 포스팅 중 이것에 대한 설명이 있는 것을 찾아보았다. 공인 레퍼런스보다는 비공인 레퍼런스가 더 많아서 아쉽지만 후자 위주로 찾아보았다.

int (*arr)[10] 같은 포인터 표기법으로 받으면 배열의 각 항목을 사용할 때 연산자 우선순위를 맞추기 위해 괄호가 많이 사용되고, 일반적인 포인터보다 이해하기 어려운 표현이다. 마침 C언어에서는 매개 변수에 한해서 int arr[][10] 같은 표기를 추가로 제공하고 있다.
2차원 배열을 매개변수를 선언할 때는 int (*arr)[10] 같은 표현을 int arr[][10]과 같이 적어도 오류는 나지 않지만 원래는 int (*arr)[10]이 더 정확한 표현이라는 것을 알게 되었다.
단, 매개변수에 arr[][] 같은 표현은 불가능하다. 포인터가 주소를 이용해 연산을 하려면 대상의 크기가 정해져야 하는데, 두 쪽을 다 비우게 되면, int (*arr)[] 같은 표현이 된다. 이렇게 되면 포인터가 자신이 가리키는 대상의 크기를 알 수 없게 돼서 오류가 난다. 매개 변수의 앞쪽(행에 해당되는 곳) []에 숫자를 비우는 것은 배열식으로 표기하기 위한 것이라서 가능했던 것이다.

출처: https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=tipsware&logNo=221329432324

지난 학기에 선형대수학 시간에 전치행렬에 대해서 배웠는데 이것을 실제로 코드로 구현할 일은 잘 없었던 것 같다. 이번에 코드로 구현할 때, 정방행렬(행과 열의 크기가 같은 행렬)로만 했지만, 이후에 행과 열의 크기를 입력받거나 입력받은 요소의 개수에 상관 없이 동적 할당을 통해 전치행렬을 만들어주는 함수를 만드는 식으로 응용을 하는 것도 좋을 것 같다고 생각한다.

첫 과제라 그런지 그렇게 어렵지는 않았고, 알고리즘을 생각할 때 당연히 중첩 반복문으로만 생각이 되었다. 하지만 중첩 반복문은 반복 횟수가 작으면 차이가 많이 안 나지만, 반복 횟수가 많아지면 시간 복잡도 부분에서 좋은 편은 아니고, 수업 시간에 추가 설명으로 다른 방법도 알게 되어서 그거에 대해서 자세히 알게 된다면 따로 구현해보고 어느 알고리즘으로 코드를 만드는 것이 더 효율이 좋은지도 비교해보고 싶다.