객체지향프로그래밍

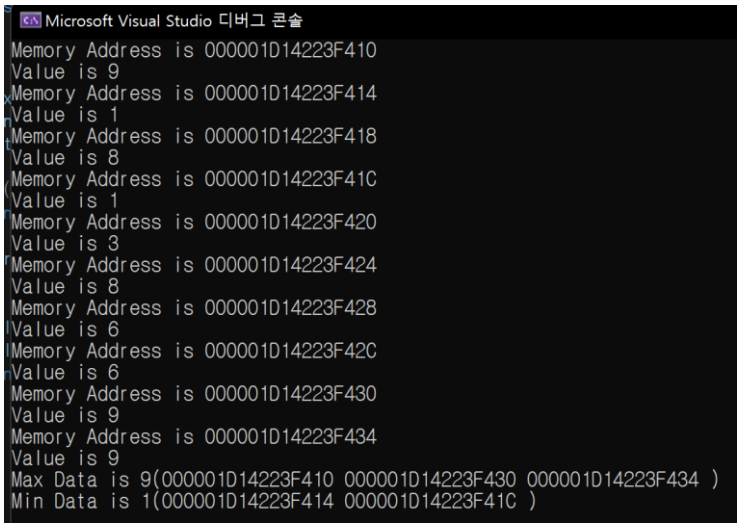
Assignment Report 2-1

2019202050 이강현

1. 10개의 정수(0 ~ 9)를 무작위(rand함수 이용)로 각각 다른 변수에 저장하고 저장된 변수의 값 과 메모리 그리고 최대, 최소값을 저장한 변수의 메모리와 값을 출력하는 프로그램을 작성하시오.

   필요한 개념: rand 함수 사용법을 알아야하고 srand를 이용하여 rand함수에 seed를 제공하는 법을 알아야 한다. 동적할당을 이용한 변수 선언하는 법을 알아야한다등등

   동적할당을 이용하여 난수를 저장할 메모리를 확보한 후 최댓값과 최솟값을 비교연산을 통해 구한 후 해당하는 메모리의 주소값에 접근하여 출력할 것이다.



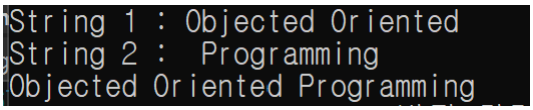
<출력 예시>

   고찰: 난수값을 이용하여 값이 정해지므로 최댓값과 최솟값이 두개 이상 나오는 경우가 있었는데 주소값은 하나만 나오는 문제가 발생하였다. 반복문과 조건문을 적절히 활용하여 이와 같은 문제점을 해결할 수 있었다.

2. 2개의 문자열을 입력 받아 두 문자열을 이어 붙인 새로운 문자열을 출력하는 프로그램을 작성하시오. 출력할 때 두개의 문자열을 이어 출력하지 않고 새롭게 두 개의 문자열을 이어붙인 문자열을 만들어 출력하며 필요한 모든 함수는 직접 구현해 사용한다. (STL,

strcpy, strcat 등 사용불가

필요한 개념: 문자열의 길이가 정해지지 않았기 때문에 정적배열이 아닌 동적배열을 사용하여 길이수가 길 때는 추가 재할당이 가능하게 코드를 작성해야 하므로 calloc 함수와 realloc함수를 이용하여 코드를 작성해야 할 것 같다. 두 문자열을 붙여서 하나로 만드려면 붙인 문자열이 저장될 메모리 길이를 확정하기 위해 두 문자열에서 각각 길이를 반환하게끔 코드를 작성해야 할 듯하다. calloc함수는 메모리를 할당한 후 0으로 값들을 채워주기 때문에 문자열의 길이보다 조금 더 메모리를 할당한 후 0을 만나면 길이를 반환하게끔 하면 적합할 듯 하다. calloc함수와 realloc함수는 null포인터를 반환할 수 있으니 메모리 누수에 조심하자.



<입출력 예시>

고찰: realloc함수와 calloc함수가 null포인터를 반환할 수 있다는 경고문이 떠서 그에 대한 조건문으로 null포인터를 반환하였을 때 프로그램 종료 혹은 메모리 해제구문을 넣어주었다. string함수를 이용한다면 문자열 저장과 길이 반환이 좀 더 간단했겠지만 정적배열을 사용하지 않고 작성하니 필요한 알고리즘을 직접 구현할 수 있다는 점이 더 큰 학습효과를 얻게 했던 것 같다.

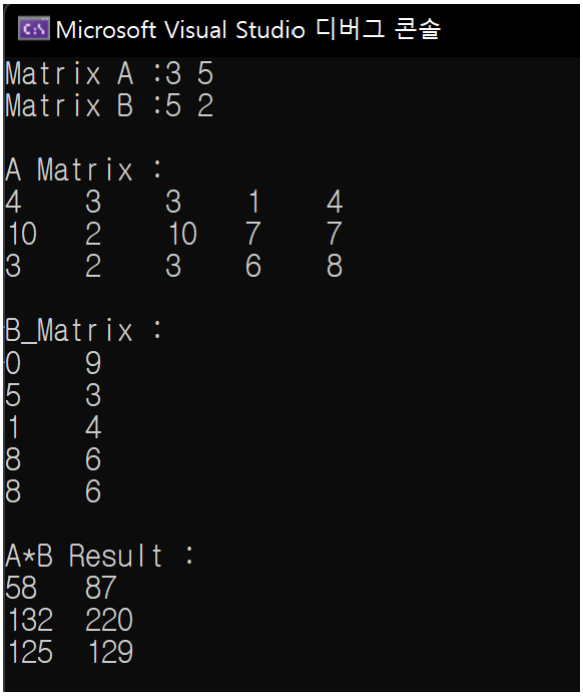
3. 2개의 Matrix의 크기(row, column)를 입력 받아 내부를 10이하의 무작위 자연수로 채운 입력 받은 크기의 Matrix들을 출력하고 만약 두 Matrix가 Matrix Multiplication을 할 수 있다면 두 Matrix의 곱셈 결과를 출력하는 프로그램을 작성하시오. 이 때 필요한 모든 함수는 직접 구현 해서 사용한다.

필요한 개념: 2차원 배열의 크기를 입력값을 이용하여 할당해야하므로 동적할당의 사용이 필요하다. 무작위 자연수를 사용해야 하므로 rand함수의 사용법을 알아야한다.

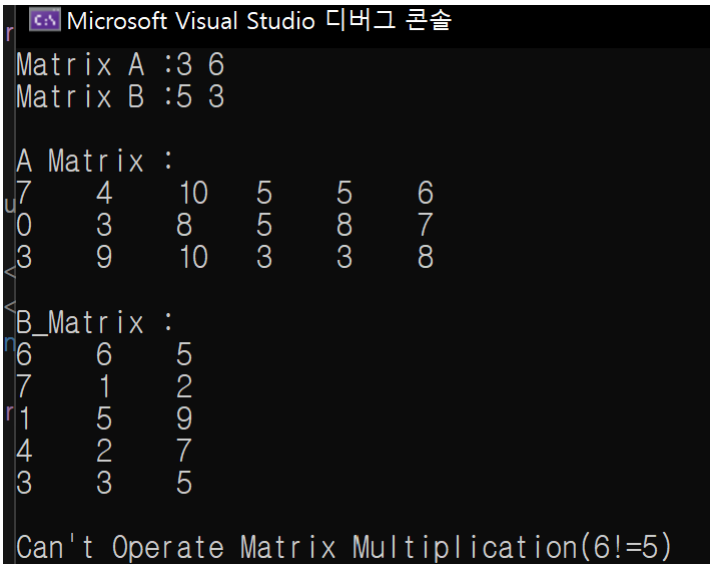
행렬의 곱셈방법과 곱셈이 가능한 조건을 알아야 한다.

먼저 두 배열의 크기를 담을 변수를 선언하고 입력값을 토대로 동적할당을 통해 행렬의 크기를 정한다. 그리고 반복문을 통해 무작위 자연수 값을 배열에 입력하고

반복문을 이용해 출력한다. 행렬의 곱셈이 가능한지 조건문으로 확인한 후 가능하다면 곱셈연산후의 행렬을 출력한다. 불가능하다면 그 이유와 함께 불가능하다는 메시지를 출력한다.



<곱을 연산할 수 있을 때>



<곱 연산이 불가할 때>

고찰: 행렬의 구조를 구현하거나 출력하는데는 큰 어려움이 없었으나 행렬의 곱셈을 구현하는데 시간이 많이 들었다. 반복문에서 사용할 변수를 어떻게 설정해야 할지 또한 곱셈이 불가능한 조건이 무엇인지 생각하느라 시간을 많이 사용하였다.

4. 0 ~ 100까지 임의의 자연수로 채워져 있는 Matrix를 int**형 변수에 저장해 출력하고 Matrix를 행 단위로 오름차순 정렬한 후 재 출력, 행의 총 합을 기준으로 오름차순으로 정렬한 후 재 출 력하는 프로그램을 작성하시오. 이 때 행의 총 합을 기준으로 정렬할 때 값을 직접 바꾸지 않고 포인터가 가리키는 주소를 바꿔 정렬한다.

필요한 개념: 행렬의 구조 구현, 오름차순 정렬을 위한 정렬 알고리즘, 행의 총합을 구하기 위해 값에 접근하는 법, 총 합을 기준으로 정렬시 주소값에 접근하는 법

우선 배열의 크기를 10*11로 동적할당 한 후 그 값들에 반복문을 이용하여 접근하면서 행의 마지막 공간을 제외한 모든 곳에 무작위 자연수를 넣어주었다. 그 이후 반복문을 이용하여 original matrix를 출력한 후 행단위로 선택정렬을 이용하여 정렬해주고 출력했다.

행의 총 합을 각 행의 마지막 공간에 넣어주었고 그로 인해 같은 배열안에 총 합이 들어있기 때문에 그 행의 첫번째 배열 주소값을 바꾸어 주면 쉽게 행단위 정렬이 가능했다.

```
Original Matrix
88      98      45      6       40      86      6       32      16      99
69      78      52      83      0       3       64      95      100     67
60      13      81      95      69      24      34      91      31      92
53      5       5       92      23      9       44      64      90      100
63      71      58      18      43      8       36      100     50      28
8       90      96      14      98      24      29      27      18      14
18      42      99      24      91      21      56      24      67      84
81      19      33      80      4       31      61      69      77      27
46      89      81      60      74      21      95      83      23      7
95      70      94      35      54      79      4       80      45      85
Sort by row
6       6       16      32      40      45      86      88      98      99      Sum is 516(0000025FEEF27918)
0       3       52      64      67      69      78      83      95      100     Sum is 611(0000025FEEF27988)
13      24      31      34      60      69      81      91      92      95      Sum is 590(0000025FEEF29688)
5       5       9       23      44      53      64      90      92      100     Sum is 485(0000025FEEF33B58)
8       18      28      36      43      50      58      63      71      100     Sum is 475(0000025FEEF34098)
8       14      14      18      24      27      29      90      96      98      Sum is 418(0000025FEEF33DF8)
18      21      24      24      42      56      67      84      91      99      Sum is 526(0000025FEEF33CA8)
4       19      27      31      33      61      69      77      80      81      Sum is 482(0000025FEEF33D88)
7       21      23      46      60      74      81      83      89      95      Sum is 579(0000025FEEF33E68)
4       35      45      54      70      79      80      85      94      95      Sum is 641(0000025FEEF33ED8)
Sort by Sum
8       14      14      18      24      27      29      90      96      98      Sum is 418(0000025FEEF33DF8)
8       18      28      36      43      50      58      63      71      100     Sum is 475(0000025FEEF34098)
4       19      27      31      33      61      69      77      80      81      Sum is 482(0000025FEEF33D88)
5       5       9       23      44      53      64      90      92      100     Sum is 485(0000025FEEF33B58)
6       6       16      32      40      45      86      88      98      99      Sum is 516(0000025FEEF27918)
18      21      24      24      42      56      67      84      91      99      Sum is 526(0000025FEEF33CA8)
7       21      23      46      60      74      81      83      89      95      Sum is 579(0000025FEEF33E68)
13      24      31      34      60      69      81      91      92      95      Sum is 590(0000025FEEF29688)
0       3       52      64      67      69      78      83      95      100     Sum is 611(0000025FEEF27988)
4       35      45      54      70      79      80      85      94      95      Sum is 641(0000025FEEF33ED8)
```

<출력 예시>

고찰: 처음엔 합을 다른 변수에 저장하여 비교하면서 인덱스를 반환하여 순서를 결정한 후 그것을 토대로 정렬하려했지만 복잡했다. 결국 쉽고 명확한 방법을 찾다가 행 단위 정렬이 목적이므로 행에 저장해두면 되겠다라는 생각을 하여 위와 같은 결과를 얻을 수 있었다. 첫번째 배열의 주소값이 곧 배열의 주소값이라는 기본적인 개념이 더욱 문제를 쉽게 해결할 수 있게 도와준 것 같다.