

Report 3

제어문과 배열(2)

학번	2018212236
학부	전자정보통신공학
이름	김동주

제출일자	2018-09-30
담당교수	반상우

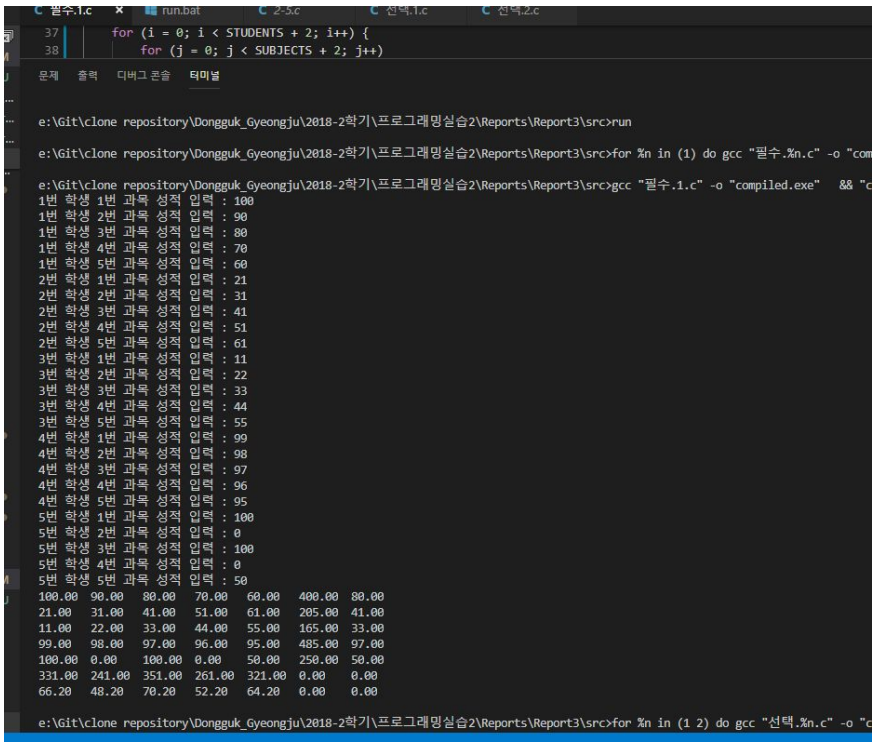
작업환경	Visual Studio Code
컴파일러	MinGW GCC

필수 문제

1. 교과목 성적 산출 프로그램

내용	5명의 학생의 5개 교과목 성적 산출 프로그램 구현. 학생별 성적 총합 및 평균, 과목별 성적 총합 및 평균을 구하여 2차원 형태로 성적 결과를 출력. 단. 성적은 표준입력장치로부터 입력 받는 방식으로 구현.
코드	<pre>#include <stdio.h> #define STUDENTS 5 #define SUBJECTS 5 int main() { int i, j; float score[STUDENTS + 2][SUBJECTS + 2] = { 0 }; for (i = 0; i < STUDENTS; i++) { for (j = 0; j < SUBJECTS; j++) { printf("%d번 학생 %d번 과목 성적 입력 : ", i + 1, j + 1); scanf("%f", &score[i][j]); score[i][SUBJECTS] += score[i][j]; // 학생 점수 합계 } score[i][SUBJECTS + 1] = score[i][SUBJECTS] / SUBJECTS; // 학생 점수 평균 } for (j = 0; j < SUBJECTS; j++) { for (i = 0; i < STUDENTS; i++) { score[STUDENTS][j] += score[i][j]; // 과목 합계 } } }</pre>

	<pre> } score[STUDENTS + 1][j] = score[STUDENTS][j] / STUDENTS; // 과목 평균 } // 출력 for (i = 0; i < STUDENTS + 2; i++) { for (j = 0; j < SUBJECTS + 2; j++) printf("%3.2f\t", score[i][j]); puts(""); } return 0; } </pre>
--	--

실행결과	 <pre> e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실습2\Reports\Report3\src>run e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실습2\Reports\Report3\src>for %n in (1) do gcc "필수.1.c" -o "compiled.exe" && "c 1번 학생 1번 과목 성적 입력 : 100 1번 학생 2번 과목 성적 입력 : 90 1번 학생 3번 과목 성적 입력 : 80 1번 학생 4번 과목 성적 입력 : 70 1번 학생 5번 과목 성적 입력 : 60 2번 학생 1번 과목 성적 입력 : 21 2번 학생 2번 과목 성적 입력 : 31 2번 학생 3번 과목 성적 입력 : 41 2번 학생 4번 과목 성적 입력 : 51 2번 학생 5번 과목 성적 입력 : 61 3번 학생 1번 과목 성적 입력 : 11 3번 학생 2번 과목 성적 입력 : 22 3번 학생 3번 과목 성적 입력 : 33 3번 학생 4번 과목 성적 입력 : 44 3번 학생 5번 과목 성적 입력 : 55 4번 학생 1번 과목 성적 입력 : 90 4번 학생 2번 과목 성적 입력 : 98 4번 학생 3번 과목 성적 입력 : 97 4번 학생 4번 과목 성적 입력 : 96 4번 학생 5번 과목 성적 입력 : 95 5번 학생 1번 과목 성적 입력 : 100 5번 학생 2번 과목 성적 입력 : 0 5번 학생 3번 과목 성적 입력 : 100 5번 학생 4번 과목 성적 입력 : 0 5번 학생 5번 과목 성적 입력 : 50 100.00 90.00 80.00 70.00 60.00 400.00 80.00 21.00 31.00 41.00 51.00 61.00 205.00 41.00 11.00 22.00 33.00 44.00 55.00 165.00 33.00 99.00 98.00 97.00 96.00 95.00 485.00 97.00 100.00 0.00 100.00 0.00 50.00 250.00 50.00 331.00 241.00 351.00 261.00 321.00 0.00 0.00 66.20 48.20 70.20 52.20 64.20 0.00 0.00 e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실습2\Reports\Report3\src>for %n in (1 2) do gcc "선택.1.c" -o "c </pre>
------	---

고찰	<p>수업시간에 비슷한 동작을 하는 코드를 작성하였는데, 그 때 찾았던 일련의 규칙을 떠올렸다.</p> <pre> for (i = 0; i < 5; i++) { for (j = 0; j < 5; j++) { printf("%d번 학생 %d번 과목 성적 입력 : ", </pre>
----	---

```

i + 1, j + 1);
    scanf("%f", &score[i][j]);

    score[i][5] += score[i][j]; // 학생 점수
    합계
}

score[i][6] = score[i][5] / 5; // 학생 점수
    평균
}

```

위 코드에서 i는 학생의 번호를, j는 과목의 번호를 나타내는데, 각 for문 안에서 i, j의 증감을 총 학생수와 전체 과목 수로 제한을 하는 규칙이 있음을 발견했다.

따라서 조금 지저분해 지더라도, #define 전처리를 이용하여 코드를 작성하면 이후에 학생 수, 과목 수를 변경할 일이 생길 때, 수정하기 편하도록 (유지보수의 편의성을 위해) 프로그램을 작성해보았다.

```

#define STUDENTS 5
#define SUBJECTS 5

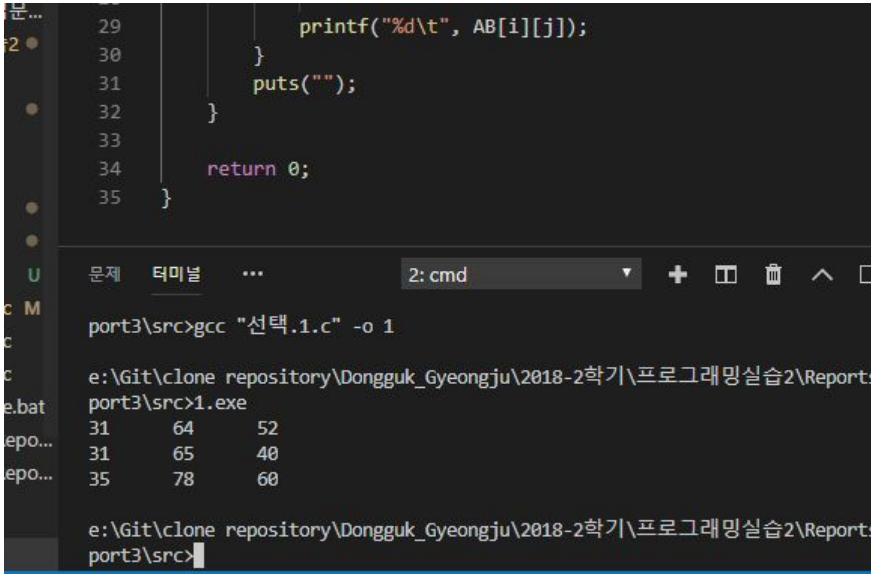
```

나중에 학생 수, 과목 수를 변경해야 하면, 위의 두 값만 바꾸면 되므로 굉장히 편리해 질 것이다.

선택 문제

1. 행렬의 곱

내용	2개 행렬의 곱을 구하는 프로그램 구현 (445페이지 문제 14번)
코드	<pre>#include <stdio.h> #define M 3 #define N 2 #define P 3 int main(void) { int A[M][N] = { { 3, 5 }, { 4, 2 }, { 5, 7 } }; int B[N][P] = { { 3, 8, 2 }, { 2, 4, 6 } }; int AB[M][P] = { 0 }; int i, j, k; for (i = 0; i < M; i++) { for (j = 0; j < P; j++) { for (k = 0; k < M && k < P; k++) AB[i][j] += A[i][k] * B[k][j]; printf("%d\t", AB[i][j]); } puts(""); } }</pre>

	<pre> return 0; } </pre>
실행결과	
고찰	<p>행렬의 곱 계산방법은 다음을 참고하였다 :</p> <p>A, B를 각각 $m \times n$, $n \times p$ 행렬이라고 하자. A와 B의 곱 AB는 다음과 같은 항을 갖는 $m \times p$ 행렬로 정의된다.</p> $(AB)_{ij} = \sum_{k=1}^n A_{ik}B_{kj}$ <p>링크 : https://ko.wikipedia.org/wiki/행렬_곱셈</p> <p>처음에는 위의 공식을 코드로 잘 표현할 수 있을까 걱정을 했지만, 막상 해보니 생각보다 매우 간단하였다! 등차수열의 시그마 기호가 For문과 기적같이 일치함을 느꼈다. 너무 금방 끝나버려서 뭔가 허무했다...</p> <p>하지만 동시에 교육과정 개편으로 고등학교때 배우지 못하였던 행렬에 대한 호기심이 생기는 것 같다.</p> <p>행렬, 재미있을 것 같다!</p>

2. 파스칼 배열

내용	파스칼 배열 출력 프로그램 구현(445페이지 문제 15번)
----	----------------------------------

코드

```
#include <stdio.h>

#define SIZE 10

int main() {
    int pascal[SIZE][SIZE] = { 0 };

    int n, r;
    int f1, f2;

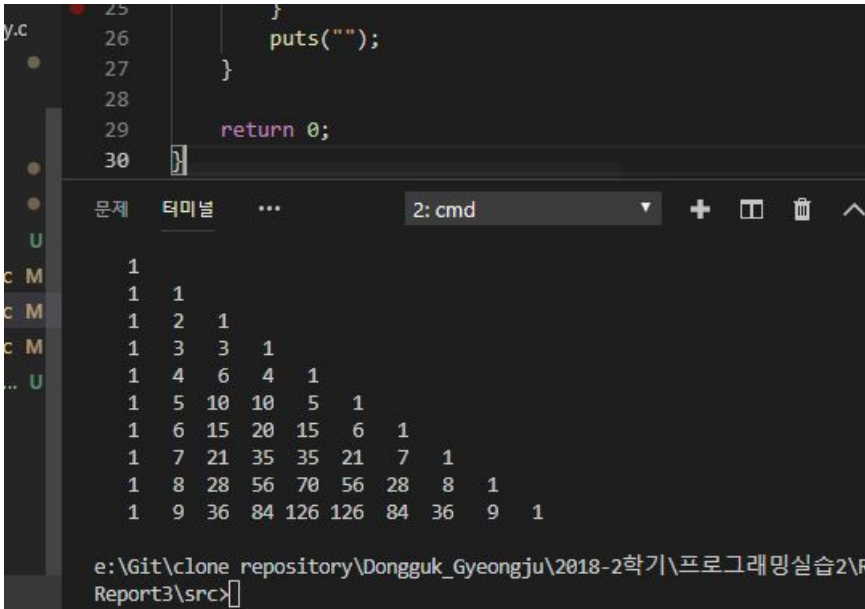
    for (n = 0; n < SIZE; n++) {
        for (r = 0; r <= n; r++) {
            // get Factorial.
            f1 = 1;
            for (int i = n; i > n - r; i--) // n ~
n - r + 1 까지만 곱하기 (분자) -- >f1
                f1 *= i;

            f2 = 1;
            for (int i = r; i > 0; i--) // r ~ 1
까지 곱하기 (분모) --> f2
                f2 *= i;

            pascal[n][r] = f1 / f2; // 조합의 공식 :
 $n! / (r!(n-r)!) = f1 / f2$ 

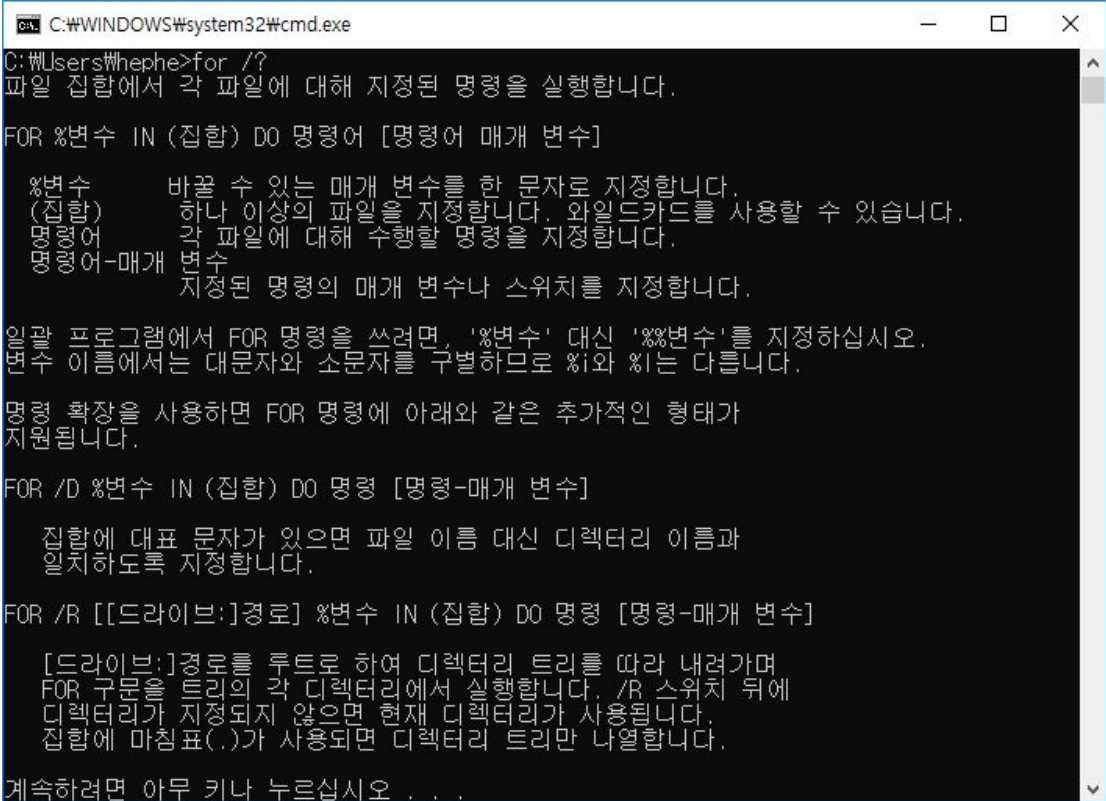
            printf("%4d", pascal[n][r]);
        }
        puts("");
    }

    return 0;
}
```

<p>실행결과</p>	 <pre> 25 } 26 puts(""); 27 } 28 29 return 0; 30 } </pre> <p>문제 터미널 ... 2: cmd</p> <pre> 1 1 1 1 2 1 1 3 3 1 1 4 6 4 1 1 5 10 10 5 1 1 6 15 20 15 6 1 1 7 21 35 35 21 7 1 1 8 28 56 70 56 28 8 1 1 9 36 84 126 126 84 36 9 1 </pre> <p>e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실습2\Report3\src></p>
<p>고찰</p>	<p>${}_nC_r = \frac{n!}{r!(n-r)!}$ 의 식을</p> <p>$\frac{n!}{r!(n-r)!} = \frac{n(n-1)(n-2)\dots(n-r+1)}{r!}$ 로 변형한 후,</p> <p>분자와 분모의 두 등비수열을 for문을 통하여 구현함으로써 답을 구하였다.</p> <p>언젠가 SNS(페이스북)에서 실력있는 프로그래머가 되려면 수학을 잘해야 한다/아니다 라는 주제로 논쟁이 일어난 것을 보았다. 이에 ‘배열을 잘 다루려면 행렬을 배워야 할 것이다, 그렇지 않더라도 수학은 당연히 잘 해야하지 않겠나.’ 라고 생각했고, 그 외의 이유는 생각해 본 적이 없었다.</p> <p>하지만 이 문제를 풀면서, 특히 분자와 분모로 식을 분리하고, 충분히 생략가능한 연산을 파악, 줄여가며 그 나름의 최적화를 하면서 수학이 필요한 중요한 이유를 하나 더 알게 되었다. 수학에 익숙하지 못하였다면, 위의 문제를 더 복잡하고 더 많은 연산을 통해 답을 구했을 것이고 그만큼 비효율적이었을 것이다.</p> <p>세상의 많은 것들은 수학을 이용하는 것들이 많다. 수학, 과학에 쓰이는 프로그램이 외에도, 디자인에서 모션 그래픽의 타이밍 옵션 (ease, linear, bezier ...), 그라데이션 배치, 등등.. 그렇기에 빠르고, 간단한 효율적인 코드를 작성하기 위해서는 수학이 꼭 필요한 것이다.</p>

P.S.

1. CMD (.bat) 파일



```
C:\WINDOWS\system32\cmd.exe
C:\Users\whephe>for /?
파일 집합에서 각 파일에 대해 지정된 명령을 실행합니다.

FOR %변수 IN (집합) DO 명령어 [명령어 매개 변수]

%변수          바꿀 수 있는 매개 변수를 한 문자로 지정합니다.
(집합)         하나 이상의 파일을 지정합니다. 와일드카드를 사용할 수 있습니다.
명령어         각 파일에 대해 수행할 명령을 지정합니다.
명령어-매개 변수 지정된 명령의 매개 변수나 스위치를 지정합니다.

일괄 프로그램에서 FOR 명령을 쓰려면, '%변수' 대신 '%~변수'를 지정하십시오.
변수 이름에서는 대문자와 소문자를 구별하므로 %I와 %i는 다릅니다.

명령 확장을 사용하면 FOR 명령에 아래와 같은 추가적인 형태가
지원됩니다.

FOR /D %변수 IN (집합) DO 명령 [명령-매개 변수]

    집합에 대표 문자가 있으면 파일 이름 대신 디렉터리 이름과
    일치하도록 지정합니다.

FOR /R [[드라이브:]경로] %변수 IN (집합) DO 명령 [명령-매개 변수]

    [드라이브:]경로를 루트로 하여 디렉터리 트리를 따라 내려가며
    FOR 구문을 트리의 각 디렉터리에서 실행합니다. /R 스위치 뒤에
    디렉터리가 지정되지 않으면 현재 디렉터리가 사용됩니다.
    집합에 마침표(.)가 사용되면 디렉터리 트리만 나열합니다.

계속하려면 아무 키나 누르십시오 . . .
```

평소 레포트를 하기 위해 코드를 작성할 때, [(필수|선택)\.<문제번호>\.c]의 규칙으로 파일에 이름을 붙여왔다.

이 규칙덕에 복습이 필요할 때 열람하는 것은 편하지만, 매번 컴파일을 할 때, 파일 이름이 길고, 한글과 영어, 숫자, 특수기호가 모두 포함되어 입력하는데 시간이 오래 걸렸다.

이 딜레마를 해결해보고자 이번에 DOS Batch에서의 for문 문법을 학습하였고, 이를 활용해보았다.

```
REM "compile.bat"의 내용
for %%n in (1) do gcc "필수.%%n.c" -o "필수.%%n.exe"
for %%n in (1,2) do gcc "선택.%%n.c" -o "선택.%%n.exe"
```