



Object-Oriented Programming Report

Assignment 1-1

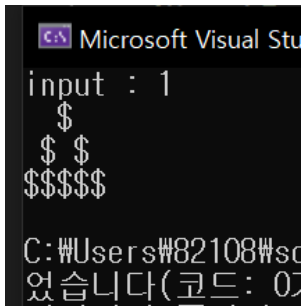
Professor	Donggyu Sim
Department	Computer engineering
Student ID	2022202061
Name	Seoeun Yang
Class (Design / Laboratory)	1 / A (미수강시 0로 표기)
Submission Date	2023. 3. 17

Program 1

□ 문제 설명

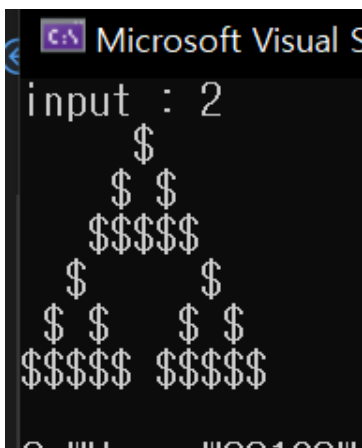
입력 받은 값에 따라서 그에 맞는 삼각형을 출력하는 문제이다. unsigned char 로 입력을 받기 때문에 아스키코드 표에 맞게 48 을 빼 프로그램을 진행시켜야 한다. 1 은 입력 받는 값 중 가장 작은 수로 제일 작은 삼각형(A 라고 하자)을 출력한다. 이 삼각형을 기본형으로 입력값(k)이 1 씩 증가할 때마다 A 의 개수가 3k 씩 증가한다. 입력값이 증가할 때마다 출력되는 전체 삼각형의 길이가 2 배로 늘어나고, 삼각형의 개수가 전 출력값의 3 배가 상, 하 좌, 하우 로 출력되는 것을 확인하였다. 이러한 규칙성을 따라 함수를 재귀적으로 호출해서 가장 작은 삼각형(A)을 위에서부터 순서대로 그리는 프로그램을 작성했다. 최대 8 까지밖에 입력을 받지 못하기 때문에 8 일 때 그려지는 삼각형의 크기를 계산해 배열(Sie_tri)을 전역변수로 설정하여 삼각형을 그렸다. (행: 3×2^7 , 열: $3 \times 2^{7 \times 2 - 1}$)

□ 결과 화면



```
Microsoft Visual Studio
input : 1
$
$$$
$$$$$
C:\Users\82108\src>
```

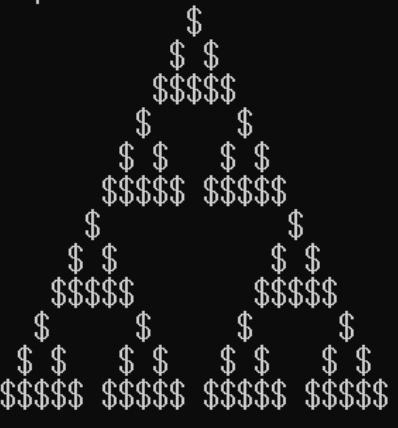
입력값이 1 일 때, 가장 작은 삼각형(A) 출력



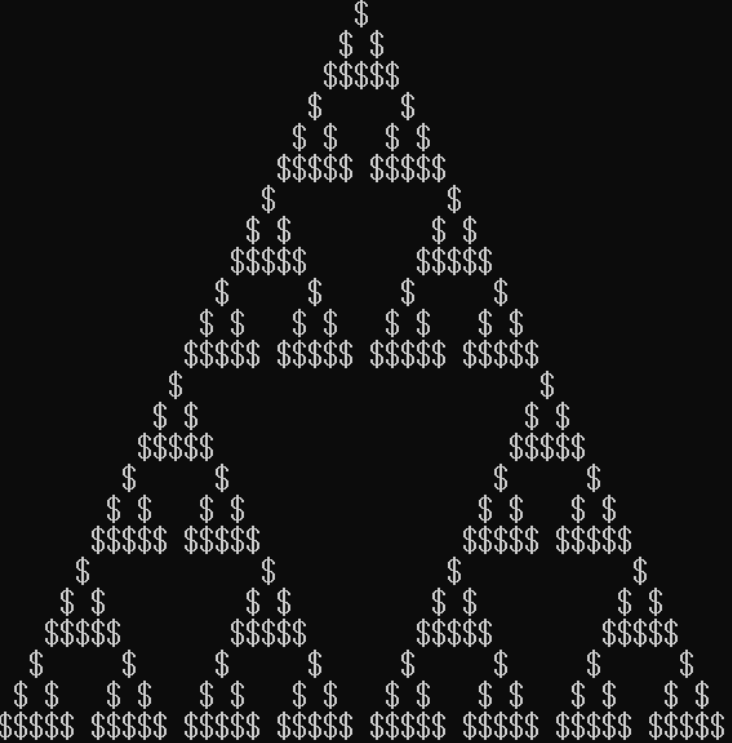
```
Microsoft Visual Studio
input : 2
$
$ $
$ $ $ $
$ $ $ $ $ $
$ $ $ $ $ $
$ $ $ $ $ $
C:\Users\82108\src>
```

입력값이 2일 때, A가 3개 출력되고, 전체 길이가 6으로 입력값이 1일 때보다 2배 증가했다.

```
Microsoft Visual Studio 디버그 콘솔
input : 3
```

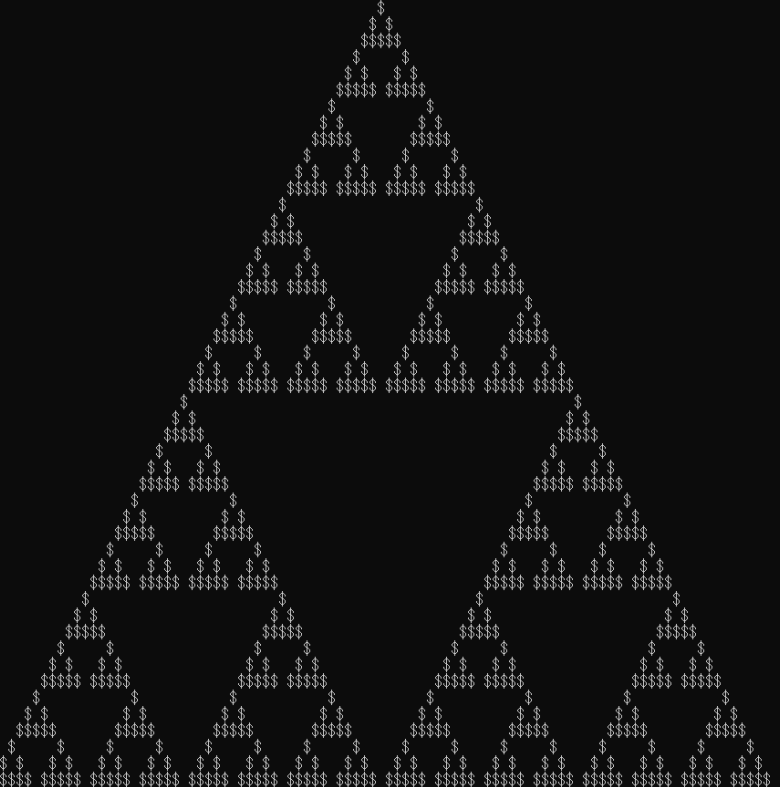


```
Microsoft Visual Studio 디버그 콘솔
input : 4
```

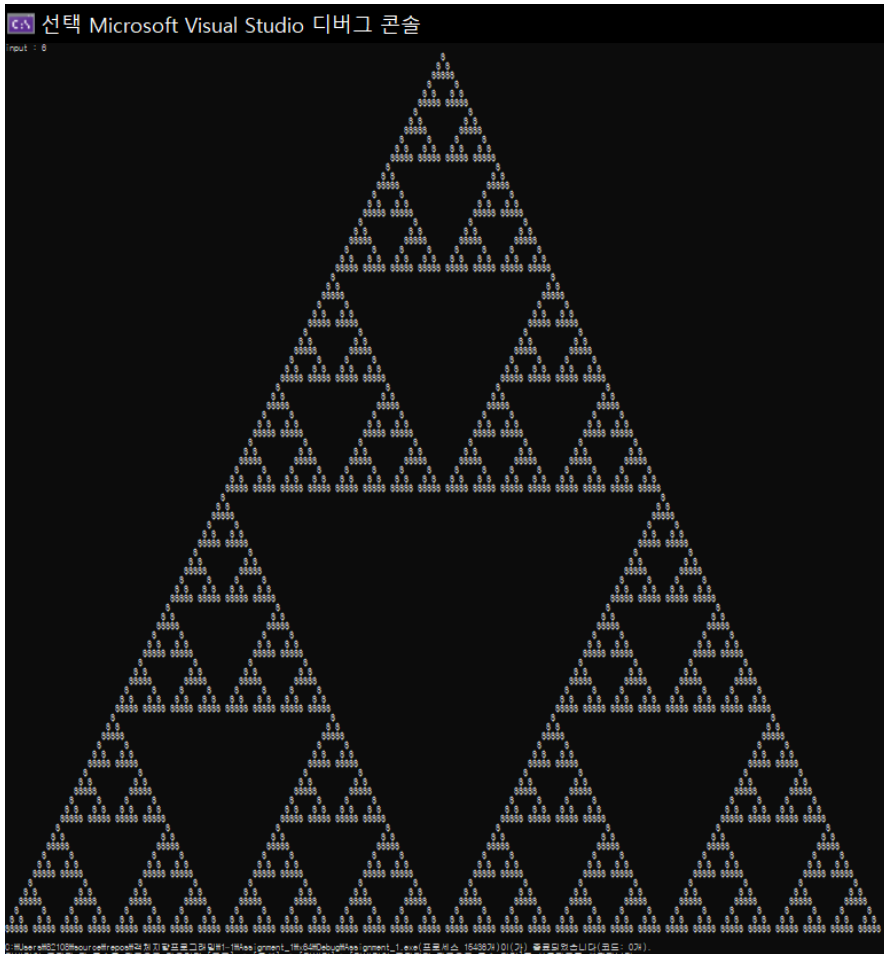


C:\Users\82108\source\repos\객체지향프로그래밍\1-1\A
되었습니다(코드: 0개).

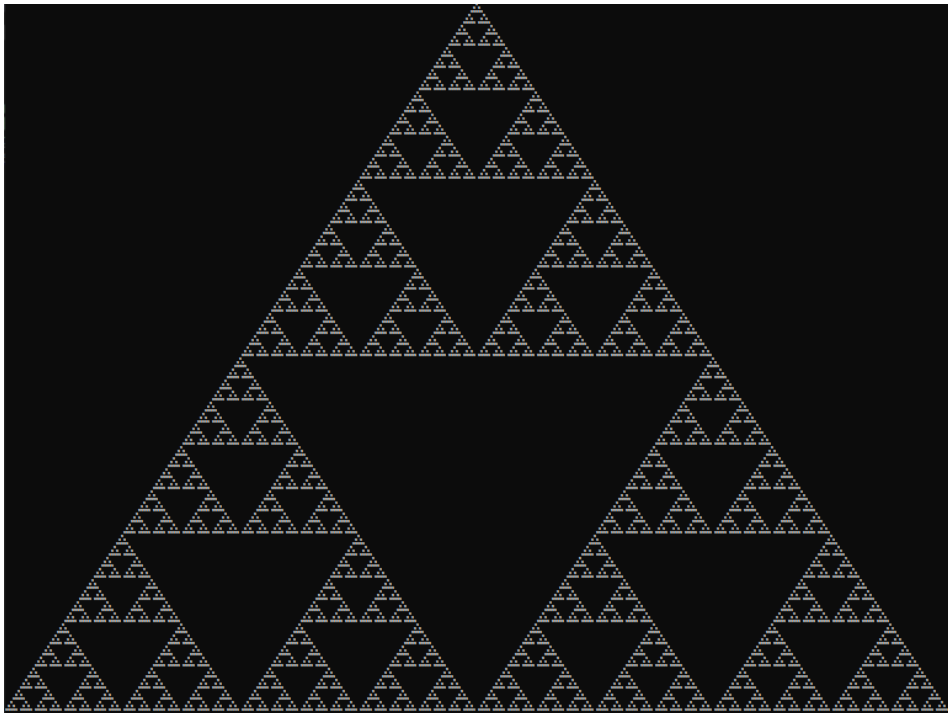
```
Microsoft Visual Studio 디버그 콘솔
input : 5
```



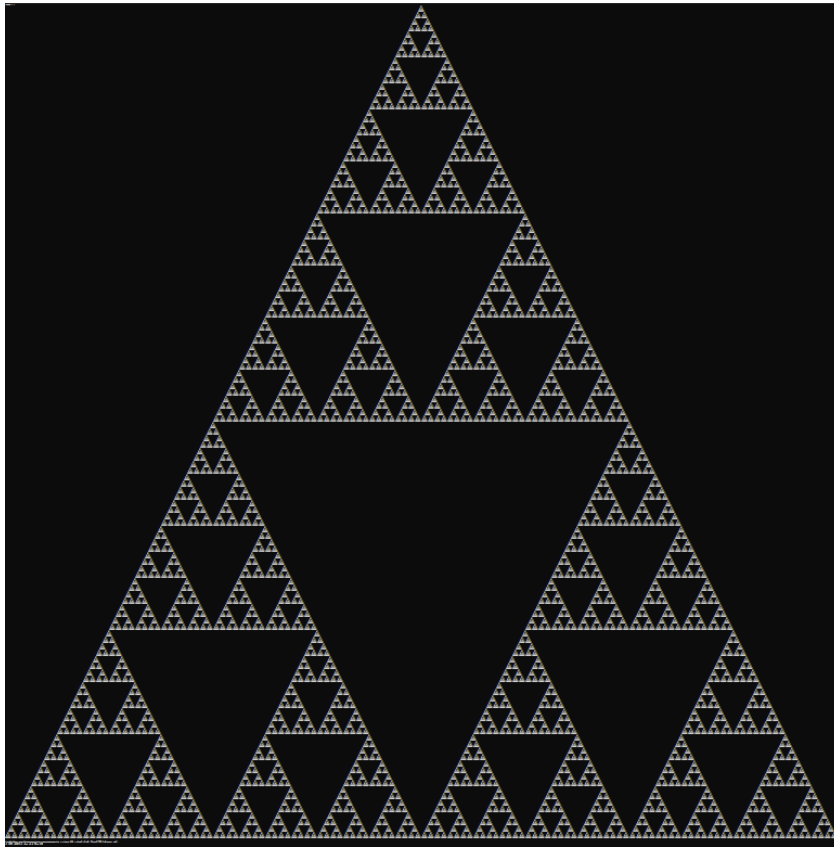
입력값이 5 일 때



입력값이 6 일 때



입력값이 7 일 때



입력값이 8 일 때

□ 고찰

처음엔 입력값에 따라 삼각형 배열(Sie_tri)의 크기를 동적으로 할당해 문제를 풀려고 했었다. 하지만 main 함수에서 다른 함수를 호출하면서 삼각형 배열을 채우는 형태로 코드를 짤기 때문에 배열 자체를 매개변수로 움직여야 했다. 코드를 짤 당시엔 막혀서 최대 크기의 삼각형 배열을 전역변수로 설정함으로 문제를 해결했는데 고찰을 쓰고 있는 현재, 포인터를 사용해 메모리를 매개변수로 넘겼다면 충분히 구현할 수 있을 것으로 판단된다.

Program 2

□ 문제 설명

이차방정식 $ax^2+bx+c=0$ 의 식에서 각 a,b,c 를 입력 받아 근을 찾아주는 문제이다. a 가 0 이면 문제가 성립하지 않으며 서로 다른 실근 2 개, 중근, 허근을 가질 수 있다. 프로그램을 짤 때 유의해야 할 점이 있는데 <그림 1>의 x_1 을 구하는 과정에서 오차가 발생한다. b 가 a,c 에 비해 매우 큰 수 일 때 루트 값이 b 와 근사적이게 되어 결국 $b-b$, 즉 0 에 가까운 값을 가지게 된다. 이러한 계산 과정에서 오차가 발생하기 때문에 x_1 에 한정하여 식을 변형시켜야 한다. 분자를 유리화시켜 루트가 분모에 위치하도록 바꿔준다.

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \text{ and } x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad \text{<그림 1>}$$

□ 결과 화면

```
Microsoft Visual Studio 디버그 콘솔
enter a,b,c : 1 2 1
the roots of 1x^2+2x+1=0 :
x:-1 (double root)
```

중근을 가질 때

```
Microsoft Visual Studio 디버그 콘솔
enter a,b,c : 1 2 3
the roots of 1x^2+2x+3=0 :
The equation has no real number solutions.
```

허근을 가질 때

```
Microsoft Visual Studio 디버그 콘솔
enter a,b,c : 0 3 4
Unexpected factor of a quadratic term
C:\Users\82108\source\repos\객체지향프로그...
```

이차방정식이 성립하지 않을 때

```
Microsoft Visual Studio 디버그 콘솔
enter a,b,c : 1 62.1 1
the roots of 1x^2+62.1x+1=0 :
x_1:-0.0161072
x_2:-62.0839
C:\Users\82108\source\repos\객체지향...
```

근의 공식 유리화를 통해 근의 오차를 줄인 경우 (과제에 제시되어 있는 예시)

```
Microsoft Visual Studio 디버그 콘솔
enter a,b,c : 1 -2 -3
the roots of 1x^2+-2x+-3=0 :
x_1:3
x_2:-1
C:\Users\82108\source\repos\객체지향...
```

서로 다른 두 실근을 가지는 경우

□ 고찰

오차를 줄일 수 있는 방법을 찾는 것이 이 문제의 관건이었던 것 같다. 부동소수점 오류로 인해 오차가 생기는 것으로 판단된다. 따라서 비슷한 수의 뿔셈에서 오차가 발생, 틀린 값이 연산되어 출력되는 것이다. 이를 방지하기 위해선 float 보다는 double 형 변수를 사용하는 것이 좋다. 식을 변형한 후, 같은 수를 대입해 근을 계산했을 때 오류가 정정되는 것을 알 수 있으며 다른 수를 대입해도 알맞은 값이 출력되는 것을 확인할 수 있다.

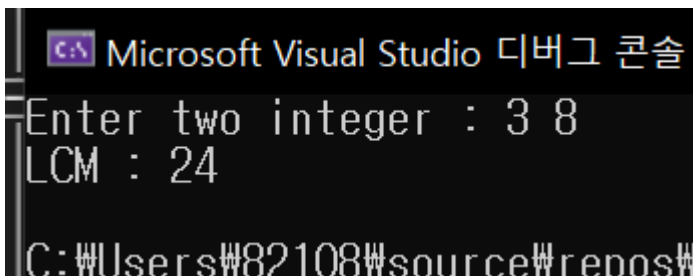
Program 3

□ 문제 설명

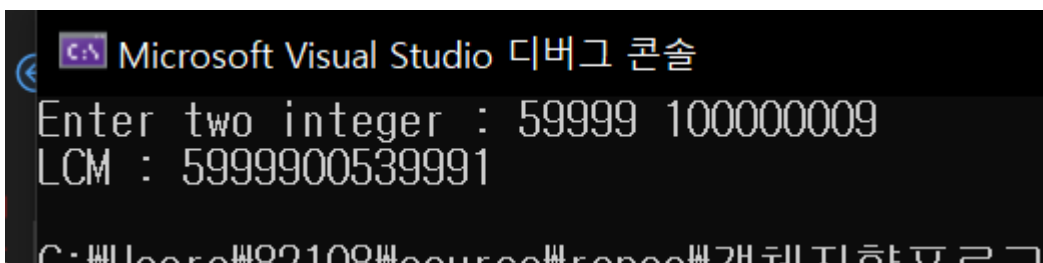
두 수를 입력 받고, 최대공약수를 통해 최소공배수를 구하는 문제이다. 입력 받는 두 수 중 하나가 0 이면 최대공약수가 없으므로 최소공배수 또한 없다. 최대공약수는 유클리드 호제법으로 구하고 c++ 라이브러리에 있는 gcd() 함수를 사용할 수 없다. 최소공배수가 int 형 변수의 범위를 초과할 수 있기 때문에 long long 형 변수를 선언해 최소공배수를 저장해 출력했다. 변수형이 같아야 연산이 가능하기 때문에 입력 받은 int 형 변수도 long long 변수에 각각 저장했다.

유클리드 호제법은 입력 받은 두 수 중, 큰 수를 작은 수로 나누고, 작은 수와 나머지를 나눠주면서 나머지가 0 이 나올 때까지 연산한다. 나머지가 0 이 나올 때의 나누는 수가 두 수의 최대공약수가 된다. 최소공배수의 계산 방법은 입력 받은 두 수를 곱하고 최대공약수로 나눠주는 것이다.

□ 결과 화면



```
C:\> Microsoft Visual Studio 디버그 콘솔
Enter two integer : 3 8
LCM : 24
C:\Users\82108\source\repos\3, 8 입력, 최소공배수: 24
```



```
C:\> Microsoft Visual Studio 디버그 콘솔
Enter two integer : 59999 100000009
LCM : 5999900539991
C:\Users\82108\source\repos\개체지향프로그
```

5999, 100000009 입력, 최소공배수: 5999900539991 (int 형 범위 벗어남)


```
Microsoft Visual Studio 디버그 콘솔
Enter two integer : 4 0
GCD is 0. LCM not exist
C:\Users\82108\source\repos\2
```

입력 받은 두 수 중에 0 이 포함되어 있을 때 (예외처리)

□ 고찰

구현하는 데에 어려움은 없었던 것 같다. 입력 받은 두 정수를 long long 으로 변환해주고 lcm 함수를 구현해 long long 형으로 선언된 결과값에 저장했다. 입력 받은 두 수 중에 0 이 있을 때 최소공배수가 없음을 알려주는 예외처리를 포함했으며 유클리드 호제법을 사용하기 위해 입력 받은 두 수의 크기 비교 또한 수행해줬다.

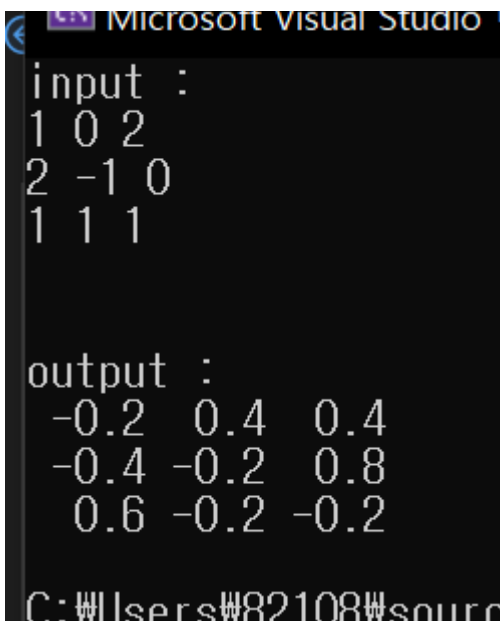
Program 4

□ 문제 설명

정수형으로 3*3 배열(A)을 선언하고 숫자를 입력 받는다. A 행렬의 역행렬을 계산하는 프로그램이다. 총 6 단계로 프로그램을 작성했다.

1. 행렬을 입력 받는다.
2. $\det(A)$ 를 계산한다.
3. cofactor matrix 를 구한다.
4. cofactor matrix 를 transpose 한 행렬을 구한다.
새로운 행렬을 선언해 cofactor matrix 를 복사하고, temp 변수를 설정해 적절하게 숫자를 재배치한다.
5. 역행렬을 계산한다.
소수가 나올 수 있으므로 float 형으로 역행렬을 선언하고, 숫자들 또한 float 형으로 강제적으로 변환해 계산해 저장한다.
6. 출력한다.

□ 결과 화면



```
Microsoft Visual Studio
input :
1 0 2
2 -1 0
1 1 1

output :
-0.2 0.4 0.4
-0.4 -0.2 0.8
0.6 -0.2 -0.2

C:\Users\82108\source
```

문제에 제시되어 있는 행렬을 입력하여 올바른 역행렬이 출력되는 것을 확인함.

```
Microsoft Visual Studio 디버그 콘솔
input :
2 2 1
-1 1 0
0 0 0
The inverse matrix does not exist.
C:\Users\82108\source\repos\객체지향프로그래밍\obj\Debug\객체지향프로그래밍.exe
업니다(코드: 0개).
```

입력 받은 행렬의 det 값이 0 일 때 (역행렬이 존재하지 않음)

□ 고찰

문제 상황에 맞게 구현은 했지만 아쉬움이 남는다. 정말 무식하게 코드를 짰 것 같다. cofactor matrix 를 계산하거나, det 값을 계산할 때 반복문이나 조건문을 사용해 간소화할 수 있었을 것 같다. 본인이 짰 코드 3*3 배열에만 사용할 수 있다는 점이 가장 치명적인 단점이다. 사용자가 배열의 크기를 정할 수 있고 그에 따른 역행렬을 계산할 수 있는 코드를 짰다면 더 실용적이고 완성도 있는 프로그램이 될 수 있을 것이다.