REPORT

보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.
2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.
3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.
4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

과　　목 : 프로그래밍 기초와 실습
과 제 명 : 실습7: Arrays &Pointers
담당교수 : 민 형 복 교수
학　　과 : 자연과학계열
학　　년 : 1학년
학　　번 : 2014310407
이　　름 : 이준혁
제 출 일 : 2014년 11월 23일

1. Introduction

이번 실습에서는 array에 대해 공부한다. 그러나 array는 사실 pointer와 몇 가지 점을 제외하면 거의 같으므로, 정확히는 array와 pointer에 대해 공부한다.

array는 같은 type의 변수들을 한 번에 선언하기 위한 data type으로, 이를 이용하면 굉장히 편하게 변수를 선언할 수 있다. 그러나, array는 단순히 변수들을 한 번에 선언하게 하는 기능도 있지만, 그 자체가 pointer라고 할 수도 있다. 바로 array의 이름이 pointer 상수가 되기 때문이다. 물론 변수로 선언된 pointer와 array는 다르지만, pointer를 array의 이름으로 선언하면, 두 개의 기능은 같다고 볼 수 있다. 즉, pointer를 array, array를 pointer로 쓸 수 있는 것이다.

또, 어떤 pointer에 동적 메모리 할당이라는 것을 이용해서, HEAP에 있는 메모리를 넘겨 줄 수도 있다. 이렇게 하면, pointer가 할당 받은 메모리 주소를 가지고 있으므로, pointer를 이용해 data를 담을 수 있게 할 수 있다. 즉, 이것을 array 처럼 쓸 수 있다는 것이다.

이런 array를 이용하면 프로그램을 굉장히 간편하게 구현할 수 있으므로, 이번 실습 역시 저번 실습들만큼 아주 중요하다고 할 수 있다. 또, 동적 메모리 할당을 통해 HEAP에 있는 메모리를 사용할 수 있게 되어, 프로그램이 돌아 갈 때 메모리를 받고, 끝나면 메모리를 돌려주는 식으로 사용할 수 있게 된다. 즉,

프로그램을 아주 효율적으로 사용할 수 있다. 그렇기 때문에 이번 실습에서 동적 메모리 할당을 어떻게 하는 지에 대해서도 확실하게 알아야 한다.

2. Problem Statement

(A)Describe what the problem is

array의 크기를 나타내는 양의 정수 한 개를 입력 받아서, 그것의 크기만큼의 정수형 array를 동적 메모리 할당해야 한다. 이전에 했던 실습과 마찬가지로 이것 역시 양의 정수가 아닌 값을 입력 시 오류 메시지를 줘야 한다.

그 다음에, -3,000 ~ 4,000 사이의 임의의 값들을 위 array에 모두 각각 할당한다. 당연히 위에서 입력 받은, array의 크기 개수만큼 할당해야 한다. 이 때, 별도의 함수를 이용해 할당해야 한다. 그리고 이 array의 값들을 main안에서 1줄에 5개씩 출력해야 한다. 이 값들의 최댓값, 최솟값, 평균, 표준편차를 계산해서 출력할 수 있는 함수를 작성해야 한다. (이 출력은 결과값을 내놓는다는 의미)

최댓값, 최솟값, 평균, 표준편차를 계산하는 함수를 두 개 작성하는데, 하나는 array 표현 식 만으로, 또 다른 하나는 pointer 표현 식만으로 작성한다. 두 개의 결과를 각 출력 해야 한다. 단, 여기서 최댓값, 최솟값은 정수형 변수이나, 평균과 표준편차는 실수형 변수이므로, 이들은 소수점 둘째 자리까지 출력해야 한다.

pointer를 사용하는 함수를 작성할 때, 평균을 계산할 때는 *(p+i) 형태의 pointer 연산을 사용하고, 표준편차를 계산할 때에는*p++, *p-- 등 increment, decrement 만을 이용해야 한다.

즉, array에 데이터 채우기, array를 사용한 최댓값, 최솟값, 평균, 표준편차계산, pointer를 사용한 최대값, 최소값, 평균, 표준편차계산을 독립된 함수를 작성해서 계산해야 한다.

(B) Describe how you solve the problem.

가장 먼저, 프로그램이 실행되는 데 필요한 array하나가 있어야 하겠다고 생각했다. 민형복 교수님의 program template p7.c을 참고해서, 우선 array의 역할을 할 pointer를 선언하기로 생각했다. pointer에 data를 담을 공간이 있으면 사실상 array와 같기 때문이다. 그래서 pointer에 data를 담을 공간을 만들어 주기 위해 문제에도 제시되어 있듯이 malloc함수를 이용하기로 했다.

그런데 malloc의 크기는 array의 개수 * array 하나의 크기인데, array하나의 크기는 sizeof(int)로 해결되지만, array의 개수는 사용자가 직접 입력하는 것이어서, 사용자에게 array의 크기를 먼저 입력 받기로 했다. 또 늘 하던 것과 같이 while문을 이용해 입력 받은 값이 0이하의 정수이면 재입력을 요구하기로 했다.

이제 array가 만들어 졌으니, 값을 할당해 주기 위한 함수를 call하면 좋을 것이라 생각했다. 그 함수에서 array에 -3000 ~ 4000까지의 값을 할당하는 것은 크게 어려워 보이지 않았다. 그런데 생각해 보니 rand함수는 초기값을 변경시켜주지 않으면, 항상 같은 값을 내놓으므로, 수업시간에 배운 것과 같이 <time.h>를 include 시켜 srand함수에 time(NULL)을 넣는 방법을 생각 해 보았다. 딱히 문제될 것도 없고, 무작위성을 높일 수 있을 것 같아서, 그렇게 하기로 했다.

이렇게 되면 array에 어떤 값들이 들어가고, 이것을 출력해야 하는데, 민형복 교수님의 practice7.pdf에도 나와있는 사항이지만, array를 출력하는 함수를 따로 만들면 좋을 것 같아 하나 만들기로 했다. 5번째 마다 ₩n을 출력하게 하게 하고 말이다.

이제 최댓값, 최솟값, 평균, 표준편차를 구하는 함수를 pointer와 array를 이용해서 만들어야 했다. 우선 pointer와 array를 어떻게 이용할 지는 나중에 생각하기로 하고, 최댓값, 최솟값, 평균, 표준편차를 구하는 논리를 먼저 전개해 나갔다.

최댓값과 최솟값을 구하는 것은 저번 시간에서도 했듯이, 최댓값이란 변수를 하나 만들어서, if문을 이용해 최댓값 변수를 한 값으로 초기화 시키고, 다른 변수들이 그 변수보다 큰지를 확인해서, 크다면 그 값을 다시 최댓값 변수에 넣는 방식으로 진행했다. 이 과정에서 여러 번 반복해야 하므로, for문을

이용하기로 했다. 최솟값도 마찬가지 방법으로 말이다.

이 작업을 array의 크기만큼 해야 했으므로, array의 크기를 입력 받기로 했다. 또, 이 변수들이 함수 안에서 정의되면, return을 통해 출력한다 해도 하나밖에 할 수 없었다. 따라서 지난 시간에 했던 것처럼 최댓값, 최솟값을 의미하는 변수를 main에 선언하고, 그 변수들의 주소를 받아서 dereference operator *을 이용해 main에 있는 변수의 값들을 직접 바꾸기로 했다.

평균, 표준편차 역시 마찬가지라고 생각했다. 단, main에 정의된 평균과 표준편차는 문제에서 요구한 것처럼 실수형 변수로 선언하기로 했다. 이 변수들의 주소를 입력 받기로 했다.

다만 평균과 표준편차를 어떻게 계산할 지가 문제였다. 일단 평균을 의미하는 변수를 0으로 초기화 시킨 뒤, for문을 이용해서 array의 값들을 다 더한 후에 array의 크기로 나누면 된다고 생각했다.

표준편차 역시 약간 까다로웠지만, 표준편차의 정의를 잘 생각해 보니 쉬웠다. 위에서 했던 것처럼 표준편차의 변수를 0으로 초기화한 후에, array의 값들에서 위에서 구한 평균을 뺀 후 그것을 제곱한 값을 모두 표준편차를 의미하는 변수에 더하기로 했다. 그 후 array의 크기로 나누고, 제곱근을 씌우기로 했다.

이렇게 되면 main에 있는 최댓값, 최솟값, 평균, 표준편차를 의미하는 변수에

각각 알맞은 값이 들어가게 될 것이므로, 함수를 그냥 여기서 끝내기로 했다. 리고 이 함수의 function call이 끝나는 대로 main에서 이 변수들을 출력하기로 했다.

하지만 위에서도 기술했듯이 이것을 어떻게 array와 pointer로 표시할 지가 약간 문제였다. array의 경우에는 n+1번째 array값을 array의 이름[n]로 표현하면 됐다. pointer도 마찬가지라 생각했다. *(array의 이름 + n)으로 표시하기로 했다. array는 연속적으로 주소를 할당 받으므로, pointer 연산에 의해 (array의 이름 + n) 은 n+1번째 array, 즉 array[n]의 주소를 가질 것이라 생각했기 때문이다.

다만 표준편차를 표현할 때는 increment/decrement만 사용해야 하므로, 약간 당황했지만, *(array의 이름++)와 같이 작성한 다음에 loop를 돌면 역시 n+1번째에 *(array의 이름 + n)가 될 것이기 때문에, 위와 같이 사용하기로 했다.

3. Implementation

3. 1. 헤더 파일 삽입

이번 실습은 include 시킬 헤더파일이 조금 있어서, 우선 그 부분부터 신중하게 작성했다. <stdio.h>말고도 rand, srand, malloc, free등을 이용하기 위한 헤더 파일 <stdlib.h>와, 지수 함수를 나타내는 pow와 제곱근을 나타내는 sqrt라는 함수를 사용하기 위해 필요한 <math.h>가 필요했다. 또, 민형복 교수님의 p7.c에는 없었지만, random number generator의 무작위성을 높이기 위해 이용할

time(NULL)을 가지고 있는 <time.h>역시 필요했다.

헤더파일 작성이 끝난 후에는 main 함수를 생각했다.

3. 2. 메인 함수의 변수 선언

array의 크기를 나타내기 위한 변수인 크기라는 의미를 가진 정수형 변수 size를 선언했고, 프로그램 해결에서 구상한 대로 array처럼 사용할 정수형 pointer 변수 random_array를 선언했다. 이 array에 들어가는 값이 무작위로 선정된 값이기 때문에 그렇게 명명했다. 그 다음에는 최댓값, 최솟값, 평균, 표준편차를 계산하고 출력하기 위한 변수인 min, max, ave, dev를 선언했다. 이 이름들은 각각 최댓값, 최솟값, 평균, 표준편차의 영어 약자로 사용했다. 문제에서도 제시되었지만 min, max는 array안의 수 들이 정수라 정수형으로 선언했지만, 평균과 표준편차는 딱 떨어지는 정수가 아닌 실수가 나올 것이므로 double로 선언했다.

3. 3. 재입력 요구

가장 먼저 할 일은 array의 크기, 즉 size변수의 값을 입력 받는 일이었다. 그리고 while문을 이용해 사용자가 0 이하의 정수값을 입력하면, 오류 메시지 출력 후 다시 입력하도록 했다.

3. 4. 동적 메모리 할당

이제 array의 크기가 정해졌으니, 그 크기만큼 메모리를 할당 받아야 했다. 왜냐하면 random_array라는 pointer는 아직 초기화 된 값이 없어서 사용할 수

없었기 때문이다. 그래서 malloc함수를 사용해 size 와 sizeof(int)의 곱만큼의 메모리를 할당받도록 했다. 이 때, malloc이 return하는 값은 void pointer 타입이므로, type casting을 이용해 정수형 pointer 타입으로 바꿔야 했다. 이렇게 하면 ramdom_array라는 pointer가 마치 array처럼 data를 담을 공간이 생긴 것이다.

3. 5. function call을 이용한 array 할당 및 출력

이제 random_array에 data를 저장할 수 있는 공간이 생겼으므로, 문제에서 제시한 대로 array에 임의의 값을 할당한 후에, 그것을 출력해야 했다. 이를 수행하기 위해 두 개의 함수를 이용했다.

3. 5. 1. assign_array 함수

array(이면서 pointer) 하나와 그 array의 크기를 입력 받아, 모든 array에 -3000 ~ 4000 사이의 임의의 값을 할당하는 함수이다.

문제 해결 방법에서도 구상했듯이 이 함수안에 srand(time(NULL))이라는 statement를 넣어서, rand함수가 어떤 값을 가지지 알 수 없게 해서 무작위성을 높였다.

cnt라는 수를 세기 위한 변수 하나를 선언해서, cnt가 0부터 size-1(size는

입력으로 받은 array의 크기)까지 1씩 커지는 동안 random_array[cnt]에 (rand())%7001 – 3000의 값을 할당했다. (rand())%7001가 0~ 7000까지의 값을 가지므로, (rand())%7001 – 3000는 -3000 ~ 4000까지의 값을 가질 것이라는 사실에서 비롯했다.

array의 값을 변경시키기는 했지만, 딱히 별다른 값을 return할 것은 없어서, 함수의 출력을 void로 뒀다.

3. 5. 2. print_array 함수

위에서 값을 할당 받은 array와 그 array의 크기를 입력 받아, 모든 array의 값을 출력하는 함수이다.

단 문제에도 제시되어 있듯이 한 줄에 5개의 숫자를 출력해야 한다. 즉, 5번째 숫자마다 줄을 바꿔야 한다는 말이다.

역시 이번에도 cnt라는 변수를 선언했고, cnt가 0부터 size-1까지 1씩 커지는 동안 random_array[cnt]를 출력했다. 단, 이것을 출력하기 전에 if문을 이용해 cnt를 5로 나눈 나머지가 0인지 확인하고, 0이면 ₩n을 출력했다. 또, %d가 아닌 %6d를 출력함으로 인해 가독성을 높였다.

이 함수는 출력이 그 목적이므로, 별달리 반환하는 값은 없다.

3. 6. array 표현을 이용하는 함수와 pointer 표현을 이용하는 함수를 통해 array 값들의 최댓값, 최솟값, 평균, 표준편차 계산 후 출력

문제 해결에서 구상한 대로, random_array에 있는 값들의 최댓값, 최솟값, 평균, 표준편차를 계산하는 함수를 만들어야 했다. 그래서 array를 이용한 compute_using_array 함수와 pointer를 이용한 compute_using_pointer 함수를 만들었다.

그리고 main에서 이 함수들을 각각 한번씩 call하고, call이 끝날 때 마다 그 값들을 printf함수를 이용해 출력해 보았다.

3. 6. 1. compute_using_array 함수

문제 해결에서 구상한 것을 직접 구현해 보았다. 먼저 이 함수의 입력은 main함수에서 변경시킬 최댓값, 최솟값, 평균, 표준편차 변수들의 주소, 최댓값, 최솟값, 평균, 표준편차를 계산할 array, 그리고 그 array의 크기였다.

그리고 이 함수 내에서 값을 세기 위한 변수 cnt도 선언했다.

문제 해결 방법을 구상한 것과 같이 최댓값과 최솟값을 구하는 것은 다음처럼 for 문과 if문을 이용했다.

```
*max = random_array[0];
for (cnt = 1; cnt < size; cnt++)
    if (*max < random_array[cnt])
    *max = random_array[cnt];
```

이렇게 되면 array의 값들 중 가장 큰 값이 *max, 즉 main 함수 내의 최댓값을 나타내는 변수에 들어가게 된다. 최솟값 역시 마찬가지로 작성했다.

평균, 표준편차 역시 구상한 대로 for문을 이용해서 구현했다. 우선 평균은 다음과 같이 모든 array값들을 더한 후 그것을 array의 크기로 나누는 방식을 취했다.

```
*ave = 0;
for (cnt = 0; cnt < size; cnt++)
    *ave += random_array[cnt];
*ave /= size;
```

이렇게 되면 평균값이 main의 평균을 나타내는 변수에 들어가게 된다.

표준 편차도 구상한 대로 하니 평균을 작성하는 것과 별 차이는 없었다. for문을 이용해 (array의 값 - 평균)^2를 모두 더한 후 그것을 array의 크기로 나누고,

제곱근을 취했다. 즉, 다음과 같이 작성했다.

```
*dev = 0;
for (cnt = 0; cnt < size; cnt++)
      *dev += pow(random_array[cnt] - *ave, 2.0);
*dev /= size;
*dev = sqrt(*dev);
```

이렇게 하면 역시 표준편차의 정의에 따라, 표준편차 값이 나오게 된다. 이 값이 main에 있는 표준편차를 의미하는 변수에 저장되게 된다.

그리고 이 모든 과정에서, random_array[cnt]와 같이 array를 이용한 표현식을 사용했다.

이 함수는 모든 출력을 pointer를 이용해 main에 있는 변수의 값을 직접 변경시키는 방법으로 하므로, return을 통한 출력은 없다.

3. 6. 2. compute_using_pointer 함수

이 함수의 논리 자체는 compute_using_array와 같았다. 하지만, array가 사용될 자리에 pointer를 사용해 논리를 전개했다.

우선, 가독성을 높이기 위해 ptr이라는 pointer 변수를 선언해 다음과 같이 초기화 했다.

int *ptr = random_array

이렇게 되면 ptr에는 random_array의 값, 즉 random_array[0]의 주소에 해당하는 값이 들어가 있게 된다.

pointer 연산과 array의 주소의 연속성에 의해 ptr + cnt에는 random_array[cnt]의 주소가 들어가 있게 된다. 즉, *(ptr + cnt)는 random_array[cnt]와 같다고 볼 수 있으므로, compute_using_array라는 함수에서 array 표현을 위와 같이 pointer 표현으로 바꿔 쓰기만 하면 이 함수는 완성된다. 단, 이것은 평균을 구하는 과정까지만이다.

표준편차를 표현할 때는 increment/decrement만 사용해야 하므로, 이대로 바꿔 쓰면 안 된다. 하지만 구상했듯이 loop가 돌면 increment를 이용한 것과 위와 같이 pointer 연산을 이용하는 것과 같다. 즉, 다음 두 개는 같다고 볼 수 있다.

```
for (cnt = 0; cnt < size; cnt++)
    *dev += pow(*(ptr+cnt) - *ave, 2.0);

for (cnt = 0; cnt < size; cnt++)
    *dev += pow(*ptr++ - *ave, 2.0);
```

위 식에서는 한 번 loop를 돌 때 마다 (ptr+cnt)로 쓰고, cnt를 1씩 증가시켰다면, 아래 식에서는 ptr++를 함으로 인해 ptr의 값 자체를 1씩 증가시켰다.

3. 7. 동적 메모리 반환

이제 모든 프로그램이 끝났으니, 다시 메모리를 반환해야 한다. 따라서, free(random_array)를 이용해 random_array에 할당된 메모리의 주소를 반환했다.

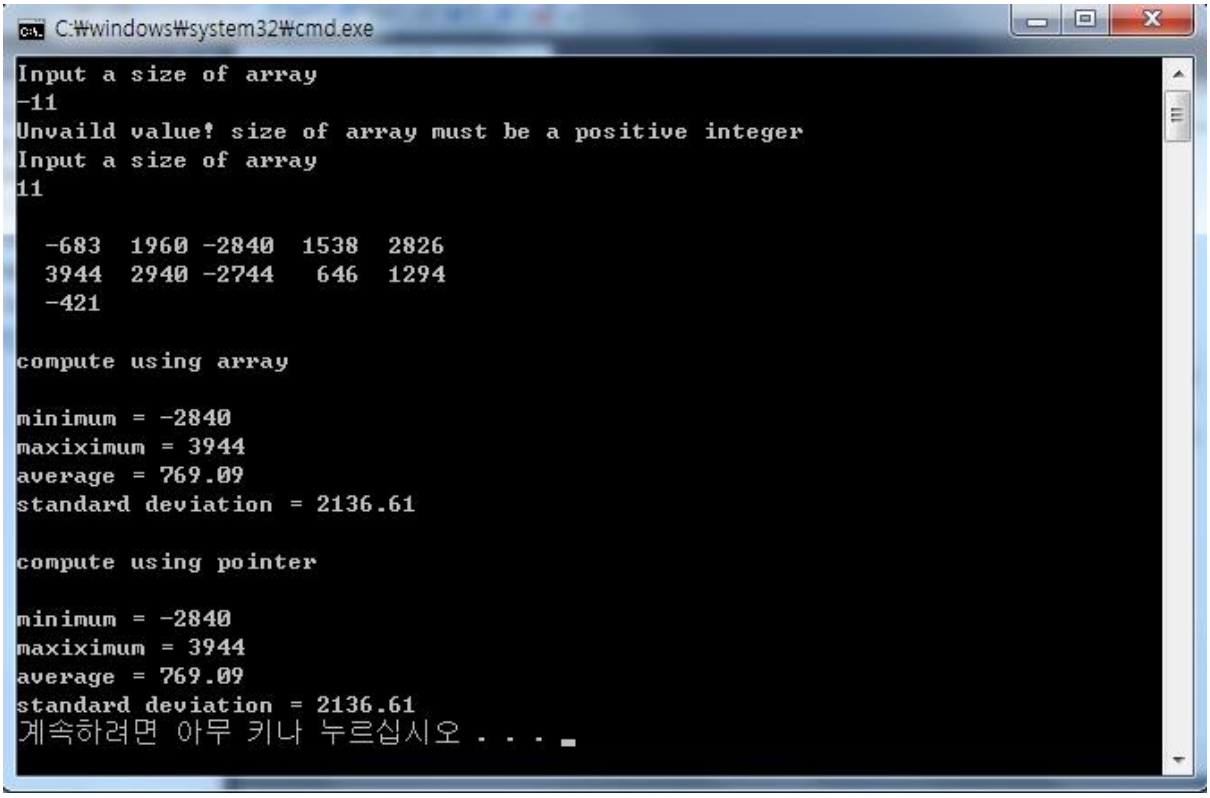
4. Result

가장 먼저, array의 크기를 잘못된 값을 줬을 때 오류 메시지를 주고 재입력을 요구하는 지 확인 하는 것이 첫 번째 목표였다.

그리고 array를 출력할 때 5번째 마다 다음 줄로 바뀌는 지도 확인해야 했다. 게다가 값들이 문제에서 요구하는 -3000 ~ 4000 사이의 값인지 역시 확인해야 했다.

마지막으로 최댓값, 최솟값, 평균, 표준편차가 정확하게 나오는 지, array를 이용한 것과 pointer를 이용한 것이 같은지 확인하려 했다.

다음은 결과 화면이다.



1. -11을 주었을 때 예상대로 오류 메시지를 주고 재입력을 요구했다.

2. 11을 주었을 때 5개/ 5개/1개 순으로 1줄에 5개씩 출력되었다. 역시 예상과 맞아떨어졌다. 게다가 값들 역시 -3000 ~ 4000 사이의 값이었다.

3. 우선 array와 pointer를 이용했을 때 결과값을 똑같았다. 그러나 이 값이 옳은 값인지 역시 확인해야 했다. 둘 다 잘못된 값이 나올 수도 있었기 때문이다.

다른 방법으로 계산해본 결과 위 값들 중 최솟값은 -2840, 최댓값은 3944, 평균은 약 769, 표준편차는 약 2136 이었다. 즉, 이 프로그램은 최댓값, 최솟값,

평균, 표준편차를 완벽하게 구했다.

5. Conclusion & Evaluation

이번 실습에서는 array에 대해 배웠다. 처음엔 array에 대한 내용 자체는 굉장히 쉬워서 실습도 쉽게 할 수 있을 것이라 보았다. 하지만 그것이 pointer와 연관되어 있어서 array를 pointer처럼, pointer를 array처럼 쓸 수 있다는 사실은 살짝 이해가 되지 않았다. 그러나 이번 실습을 통해 직접 pointer를 선언하고 그것을 array처럼 사용하는 프로그램을 만들어 보니 그다지 어렵진 않았다. 쉽게 이해가 갔다.

하지만 아직도 동적 메모리 할당을 하는 부분은 아직 직관적으로 잘 이해가 가지 않는다. 물론 이번 실습에서 동적 메모리 할당을 하는 법, 그리고 그것의 원리는 모두 이해했다. 다만 동적 메모리 할당이라는 것 자체를 처음 배워서 정확하게 감을 잘 못 잡겠다. 하지만 이 부분은 많이 사용하다 보면 익숙해 질 것 같다.

전공이 수학이라, 프로그래밍 기초와 실습을 다 들으면 역행렬과 의사 역행렬을 구하는 프로그램을 직접 한 번 만들어 보고 싶었다. 그런데 행렬을 구현하는 법에 대해서는 무지했는데, 이번에 한 array라는 실습을 통해, 행렬을 어떻게 구현할 지를 확실하게 알게 되었다. 계획한 대로 진행할 수 있게 되어서 만족스럽다.

6. 참고 문헌

[1]민형복, program template, p7.c.

[2] 프로그래밍기초와실습사이트,http://class.icc.skku.ac.kr/~min/C/,보고서작성요령, 2014 년 11 월.

[3] Al Kelly, Ira Pohl, *C by Dissection: The Essentials of C Programming, Fourth Edition,* Pearson, p.303 ~ p.312.

[4] 민형복, practice7.pdf.