C 언어 스터디 5주차

복습

```
16  int GetMulti(int x)
17  {
18      int y;
19      printf("두 번째 숫자를 입력하시오 : ");
20      scanf("%d", &y);
21
22      printf("두 수의 곱 : ");
23      return x * y;
24  }
```

```
19  void hanoi(char from, char to, char temp, int n)
20  {
21      if (n == 1)
22          move(from, to, n);
23      else
24      {
25          hanoi(from, temp, to, n - 1);
26          move(from, to, n);
27          hanoi(temp, to, from, n - 1);
28      }
29  }
```

배열의 필요성

예를 들어 학생이 1000명 있고 이들의 성적의 평균을 계산하고자 한다.
그렇다면 학생들의 성적을 저장할 공간(변수)를 선언하여야 한다.

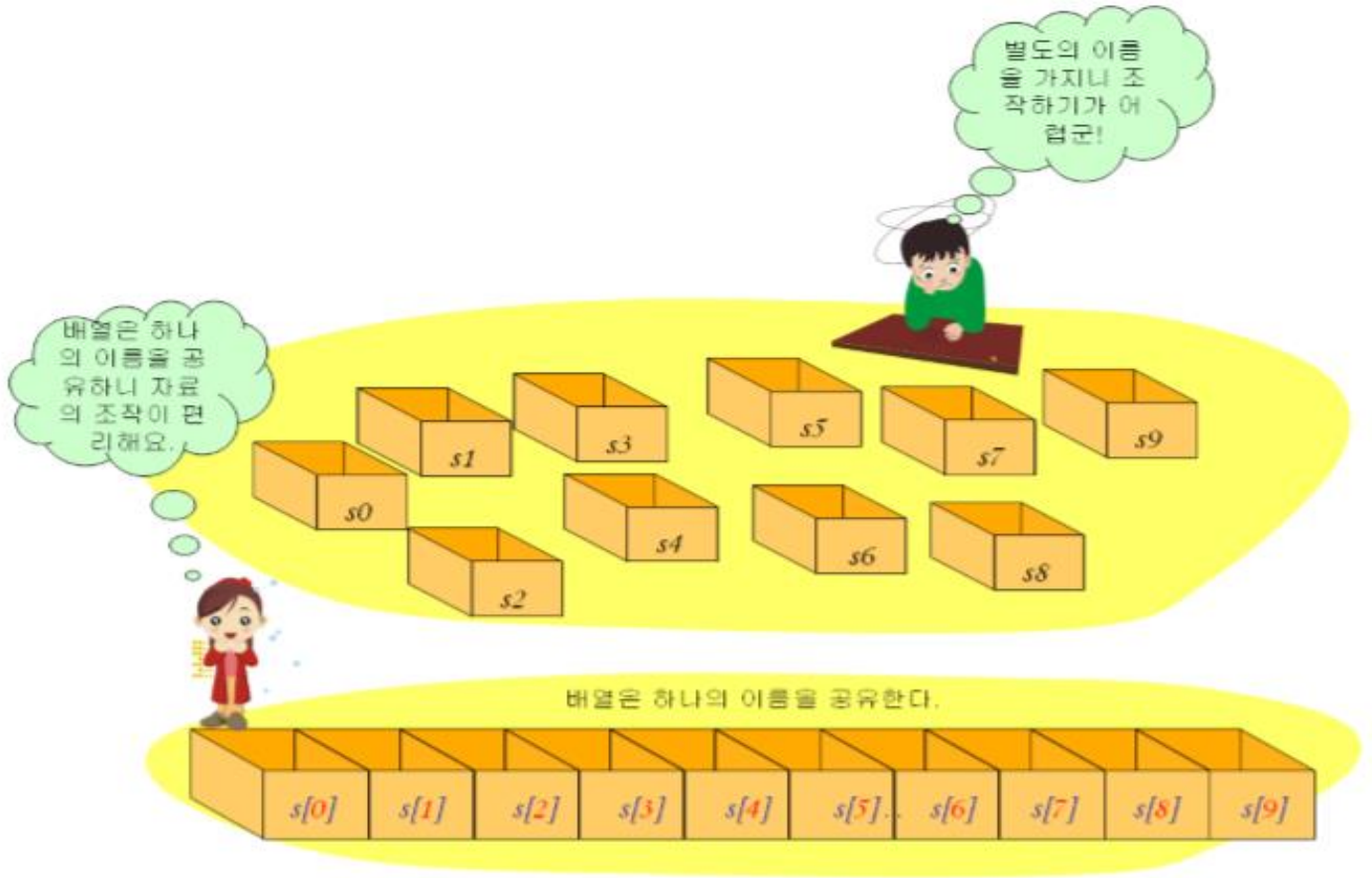
```
int s0, s1, s2, ... , s1000;
```

이를 일일이 선언하고 관리하는 것은 힘들 것이다.

대량의 데이터를 손쉽게 처리할 방법이 필요하고 이것이 바로 **배열**이다.

배열

배열(array)는 동일한 타입의 데이터가 여러 개 저장되어 있는 데이터 저장 장소이다. 각각의 데이터는 배열의 이름에 숫자를 붙여 접근한다.

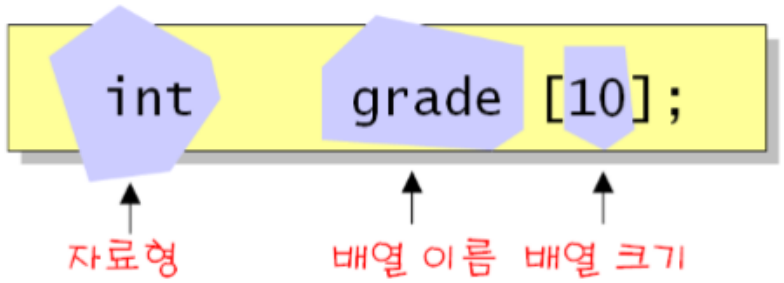


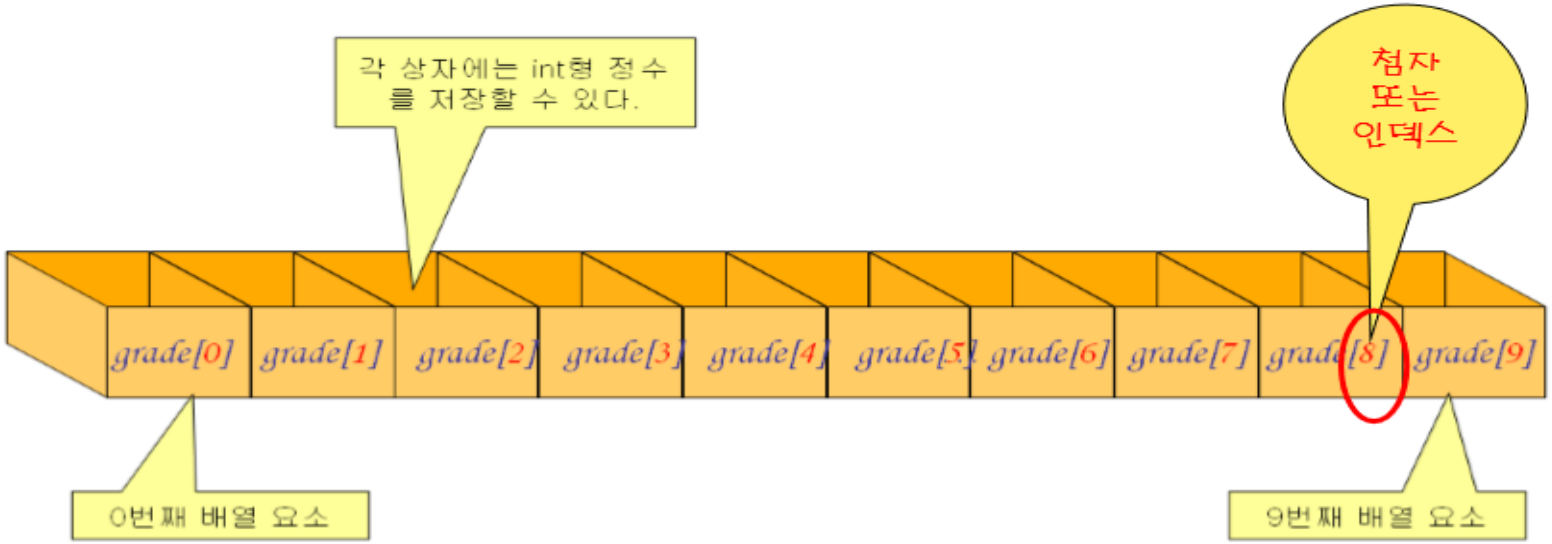
배열은 근본적으로 데이터들에게 하나하나 이름을 붙이지 않고 전체 집단에 하나의 이름을 부여한 후 각각의 데이터를 숫자로 접근하는 방식이다.

배열의 선언

- 배열을 선언하려면 배열 원소의 개수와 원소의 자료형을 선언해야 한다.
- 지정된 타입과 크기를 가지는 배열을 생성한다.

```
Ex) int grade[10];  //10개의 int 타입의 정수를 저장하는 배열 grade
    float values[36]; //36개의 float 타입의 정수를 저장하는 배열 values
    char name[10]; //10개의 문자를 저장하는 배열 name
    int index, days[7]; //일반 변수 index와 배열 days를 동시에 선언
```





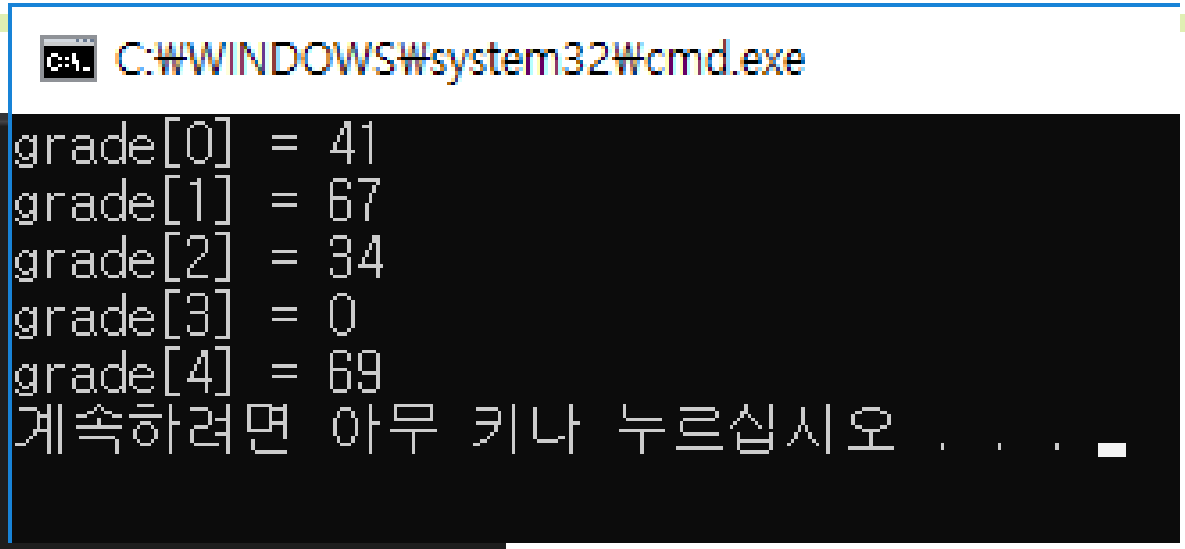
배열의 선언

- 배열을 선언할 때 주의점!!

- int grade[-2]; int grade[6.7]; int grade[size]; 와 같이 배열의 크기에 실수, 음수, 변수는 못 들어감!!

배열의 선언

```c
#include<stdio.h>
#include<stdlib.h>
#define SIZE 5

int main() {
    int i;
    int grade[SIZE];

    for (i = 0; i < SIZE; i++) {
        grade[i] = rand() % 100;
    }
    for (i = 0; i < SIZE; i++)
        printf("grade[%d] = %d\n", i, grade[i]);
    return 0;
}
```

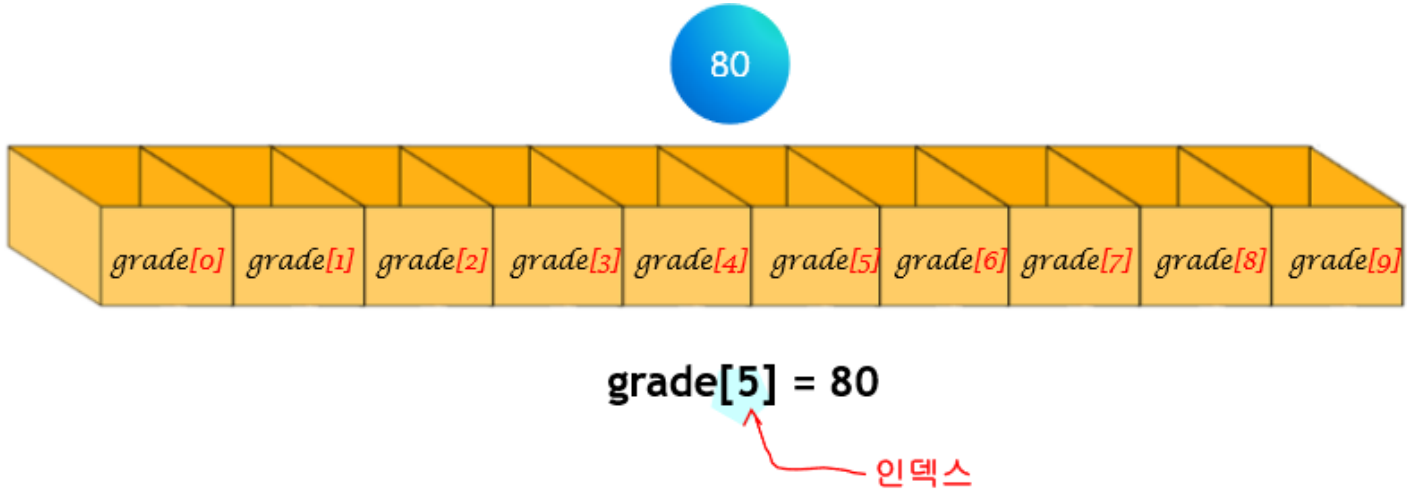


배열 원소 접근

- **인덱스(index)** : 배열 원소의 번호
- 인덱스는 항상 0부터 시작함. 크기 10인 배열이면 0~9까지 있음.

Q. 잠깐! 접근한다는게 무엇인가요?

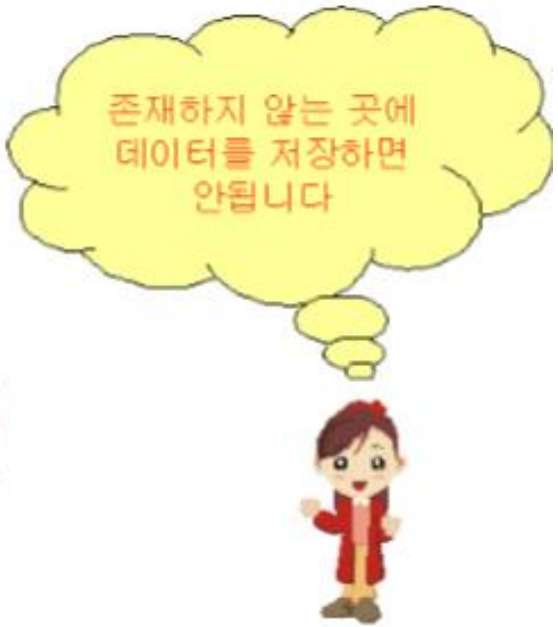
A. grade[4] = 4; 와 같이 그 변수에 있는 값을 변경하거나 참조하는 것을 의미합니다.
+ 선언할 때와 다르게 grade[i]와 같이 변수를 인덱스로 사용하여 접근하는 것은 가능.



배열 원소 접근

- 인덱스는 항상 0부터 시작함. 크기 10인 배열이면 0~9까지 있음.
- 범위를 벗어난 인덱스를 사용하면 프로그램에 오류가 생긴다.
- 컴파일러는 이러한 오류를 잡아주지 않음.

```
int   grade[5];
...
grade[5] = 60;       // 치명적인 오류!
```



배열 초기화

1. 배열을 초기화하는 기본적인 방법

```
int grade[5] = {10, 20, 30, 40, 50};
```

2. 초기화 값의 개수가 배열의 크기보다 작다면?

```
int grade[5] = {10, 20};
```

➔ grade[0] = 10, grade[1] = 20, grade[2] = grade[3] = grade[4] = 0

: 순서대로 앞부터 채워진 후 나머진 0으로 초기화된다.

(모든 원소를 0으로 초기화하고 싶다면 grade[5] = {0}; 이와 같이 해주자.

3. 초기화하지 않고 선언만 한다면?

int grade[5]; ➔ 모든 원소에 쓰레기 값이 들어감.

배열 초기화

★ 배열의 크기를 비워 놓고 초기화만 한다면? ★

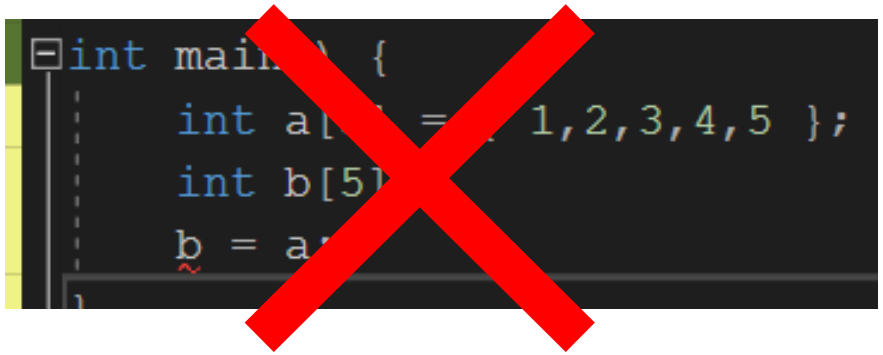
int grade[] = {10, 20, 30}; ➔ 배열의 크기가 자동으로 3으로 잡힌다.

★ 초기화 할 때를 제외하고는 중괄호로 값을 대입할 수 없음!! ★

```
int grade[3];
grade[3] = {10, 20, 30}  //컴파일 오류!!
```

배열의 복사와 비교

배열의 **복사** : 단순히 대입연산자 = 를 사용하면 될 것 같지만 잘못된 방법이다.

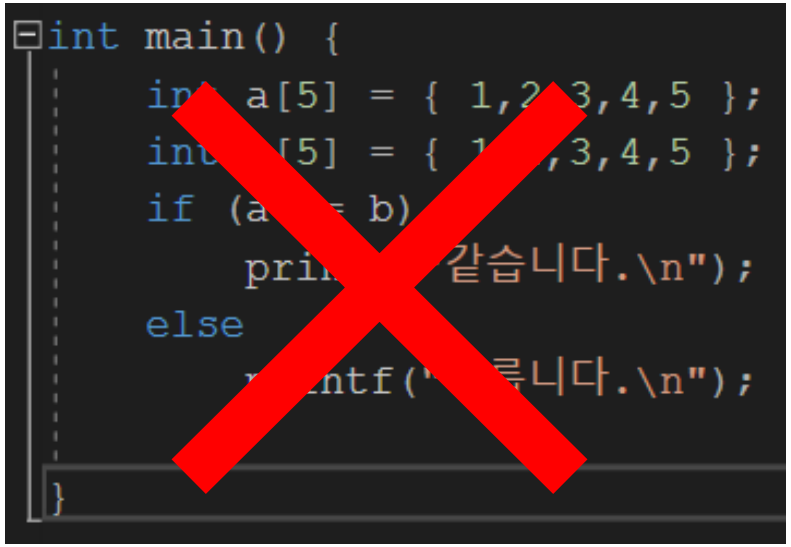



```
int main() {
    int a[5] = { 1,2,3,4,5 };
    int b[5];
    for (int i = 0; i < 5; i++)
        b[i] = a[i];
}
```

하나의 배열을 다른 배열로 복사하기 위해서는 반복문을 통해 원소를 하나하나 복사하여야 한다.

배열의 복사와 비교

배열의 **비교** : 이것 또한 ==을 통해 비교하는 것은 옳지 않다.



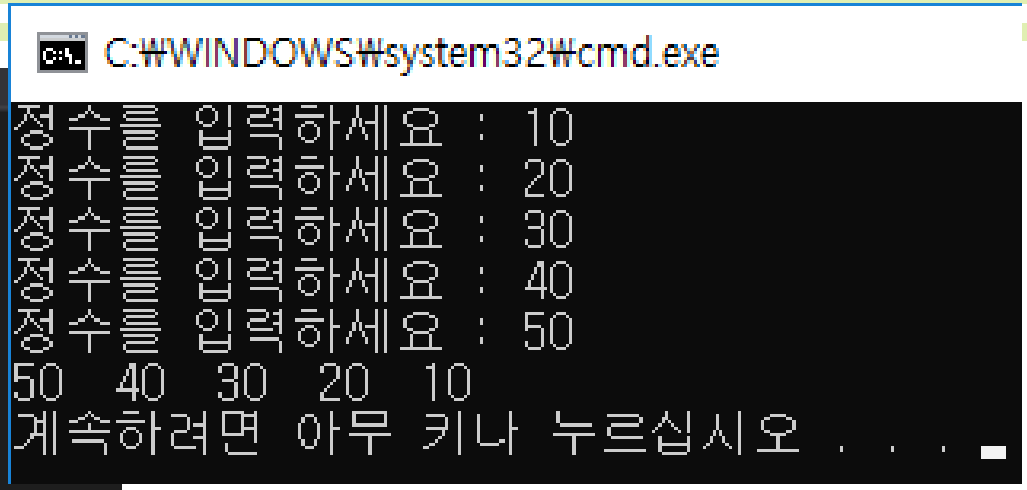
```
int main() {
    int a[5] = { 1,2,3,4,5 };
    int b[5] = { 1,2,3,4,5 };
    if (a == b)
        printf("같습니다.\n");
    else
        printf("다릅니다.\n");
}
```

```
int main(void)
{
    int a[5] = { 1,2,3,4,5 };
    int b[5] = { 1,2,3,4,5 };
    for (int i = 0;i < 5;i++)
    {
        if (a[i] != b[i])
        {
            printf("다릅니다.\n");
            break;
        }
    }
}
```

전자는 배열의 원소는 동일하지만 다르다는 결과를 출력, 후자는 정상종료 될 것임.

배열의 응용

```c
#include<stdio.h>
#define SIZE 5

int main() {
    int data[SIZE];
    int i;
    for (int i = 0; i < SIZE; i++) {
        printf("정수를 입력하세요 : ");
        scanf("%d", &data[i]);
    }
    for (i = SIZE - 1; i >= 0; i--)
        printf("%d  ", data[i]);
    printf("\n");
    return 0;
}
```



배열의 응용

```c
#include<stdio.h>
#include<stdlib.h>
#define SIZE 6

int main() {
    int freq[SIZE] = { 0 };
    int i;
    for (i = 0; i < 6000; i++)
        ++freq[rand() % 6];

    printf("==================\n");
    printf(" 면          빈도\n");
    printf("==================\n");

    for (i = 0; i < SIZE; i++)
        printf("%3d          %3d  \n", i + 1, freq[i]);
    return 0;
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
==================
 면          빈도
==================
  1          1003
  2          1017
  3          983
  4          994
  5          1004
  6          999
계속하려면 아무 키나 누르십시오 . . .
```

배열과 함수

배열 또한 함수의 인수로 사용 가능하다!

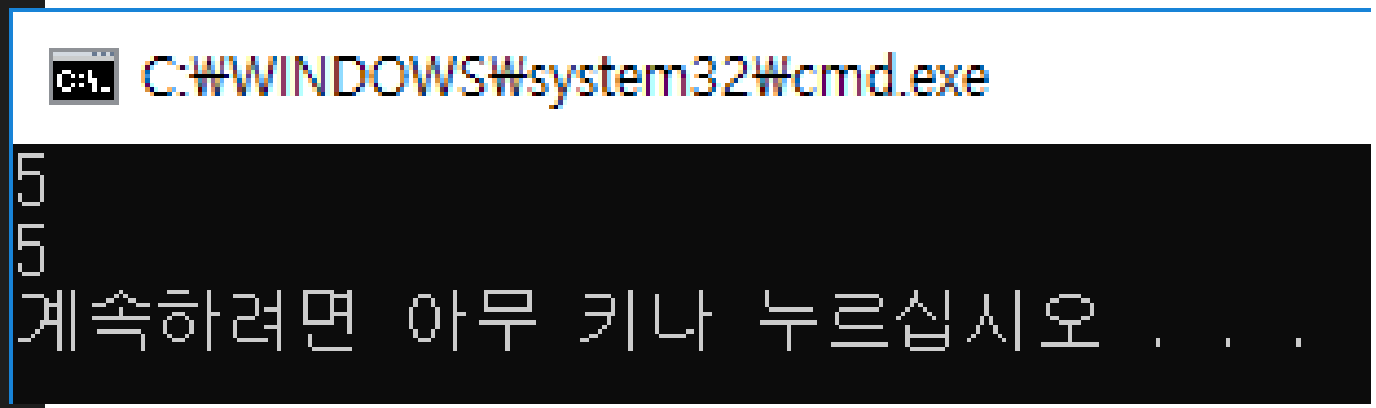
우리는 int나 double 같은 일반 변수는 “값에 의한 호출 " 이라는 것을 알고있다.
(값에 의한 호출 : 원본이 전달되지 않고 복사본이 전달되는 방식)

그러나 배열은 값에 의한 호출이 아닌 원본이 전달된다.

이에 대한 설명은 다음 장인 포인터를 배워야 완전해진다.
일단은 원본이 전달된다고 알아두자.

배열과 함수

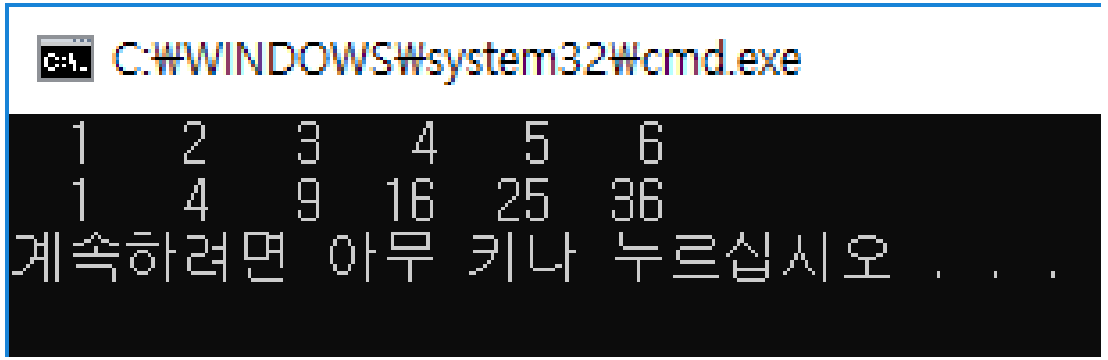
```
1   #include<stdio.h>
2
3   void square(int a) {
4       a = a * a;
5   }
6
7   int main() {
8       int x = 5;
9
10      printf("%d\n", x);
11      square(x);
12      printf("%d\n", x);
13      return 0;
14  }
```



일반 변수는 값에 의한 호출로,
원본은 변하지 않는다.

배열과 함수

```c
#include<stdio.h>
#include<stdlib.h>
#define SIZE 6
void square_array(int a[], int size) {
    for (int i = 0; i < size; i++)
        a[i] = a[i] * a[i];
}
void print_array(int a[], int size) {
    for (int i = 0; i < size; i++)
        printf("%3d ", a[i]);
    printf("\n");
}
int main() {
    int list[SIZE] = { 1,2,3,4,5,6 };

    print_array(list, SIZE);
    square_array(list, SIZE);
    print_array(list, SIZE);
    return 0;
}
```



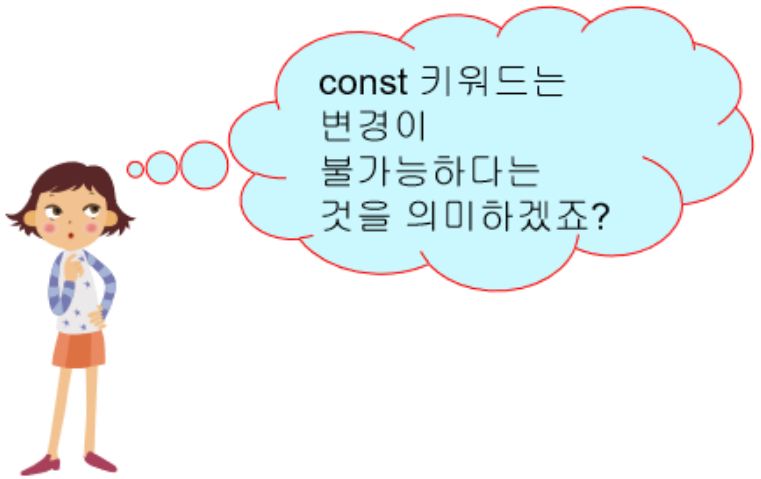
함수를 제곱하고 출력해보았더니 원본이 변경되어 있는 것을 볼 수 있다.

배열과 함수

원본 배열의 변경을 금지하는 방법

```
void print_array(const int a[] int size)
{
        ...
        a[0] = 100;          // 컴파일 오류!
}
```

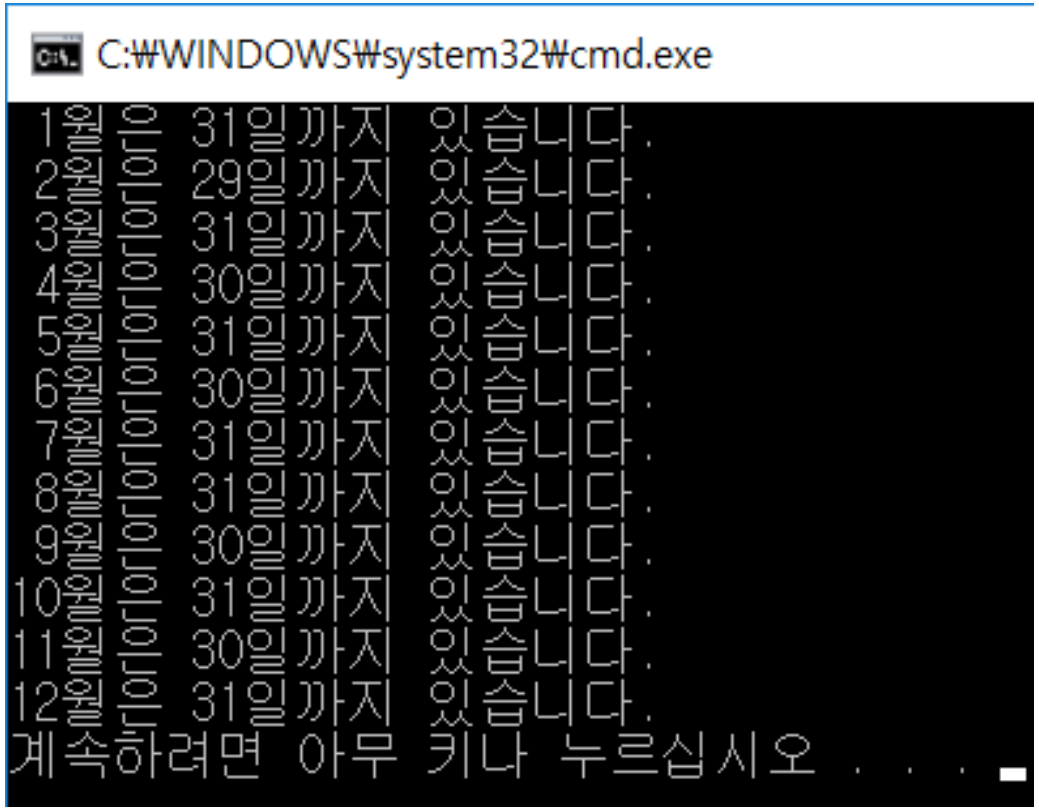
함수 안에서 a[]는 변경할 수 없다.



실습 문제

- 배열 days[]를 다음과 같이 초기화하고 배열 원소의 값을 다음과 같이 출력하는 프로그램을 작성하라.

< 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 >



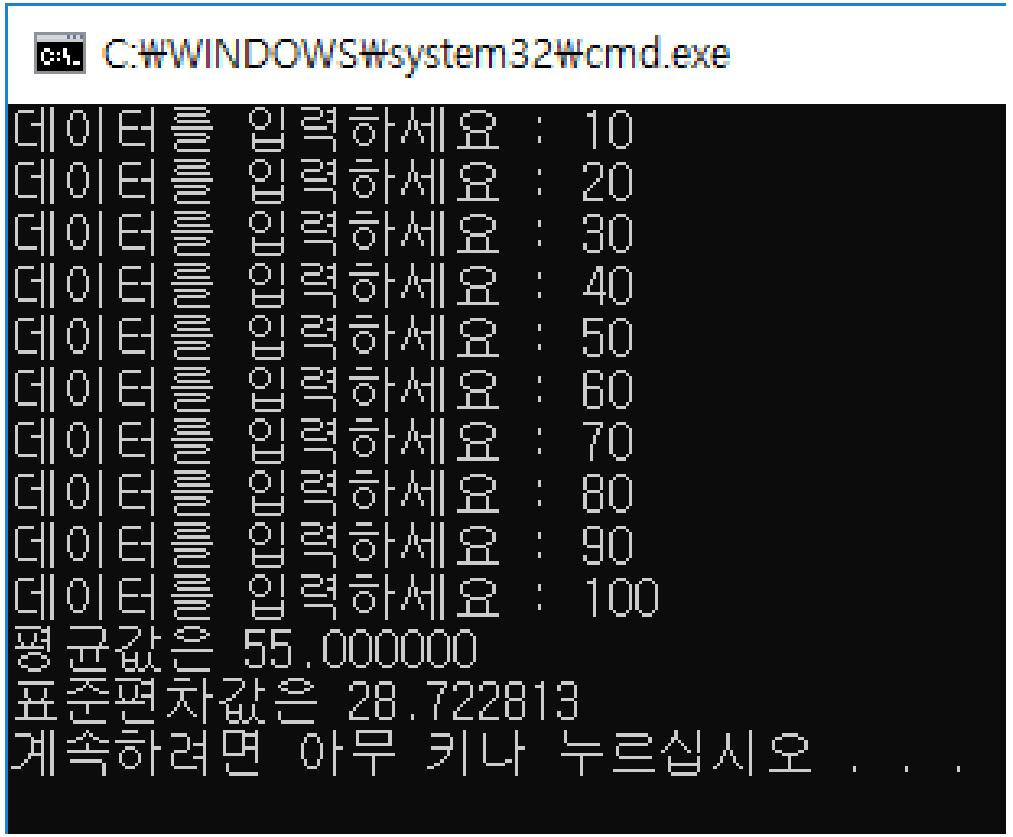
실습 문제

```c
#include<stdio.h>

int main(void)
{
    int i;
    int days[12] = { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

    for (i = 0;i < 12;i++)
    {
        printf("%2d월은 %d일까지 있습니다.\n", i + 1, days[i]);
    }

    return 0;
}
```

실습 문제

- 사용자가 입력하는 10개의 실수 자료를 받아 평균과 표준편차를 계산하는 프로그램을 작성하라.

hint) 1. 평균값은 $m = \frac{1}{n}\sum_{i=1}^{n} x_i$ 분산은 $\mathrm{v} = \frac{1}{n}\sum_{i=1}^{n}(x_i - m)^2$ 으로 구한다.



실습 문제

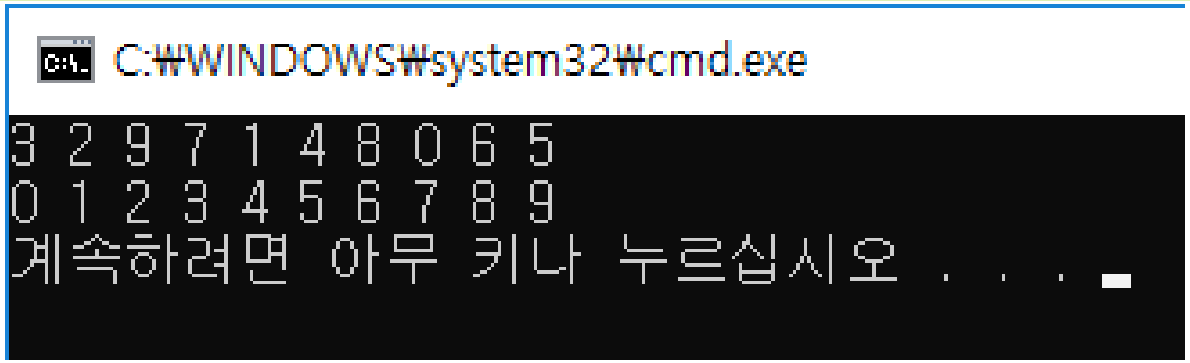
```c
#include<stdio.h>
#include<math.h>

int main(void)
{
    int i;
    double data[10];
    double mean = 0, vari = 0;

    for (i = 0;i < 10;i++)
    {
        printf("데이터를 입력하세요 : ");
        scanf("%lf", &data[i]);
    }

    for (i = 0;i < 10;i++)
        mean += data[i];

    mean /= 10;

    for (i = 0;i < 10;i++)
        vari += (data[i] - mean)*(data[i] - mean);

    vari /= 10;

    printf("평균값은 %lf\n", mean);
    printf("표준편차값은 %lf\n", sqrt(vari));

    return 0;
}
```

선택 정렬

- **선택 정렬(selection sort)은 가장 이해하기 쉬운 정렬 방법이다.**
- **정렬이 안된 원소 중 최소 값을 맨 앞의 원소와 교환.**

선택 정렬

```c
#include<stdio.h>
#define SIZE 10

void print(int a[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", a[i]);
    printf("\n");
}
int main() {
    int list[SIZE] = { 3,2,9,7,1,4,8,0,6,5 };
    int i, j, temp, least;

    print(list, 10);

    for (i = 0; i < SIZE - 1; i++) {
        least = i; //매 반복마다 i번째 원소부터 시작
        for (int j = i + 1; j < SIZE; j++)
            if (list[j] < list[least])
                least = j;
        temp = list[i];
        list[i] = list[least];
        list[least] = temp;
    }
    print(list, 10);

    return 0;
}
```

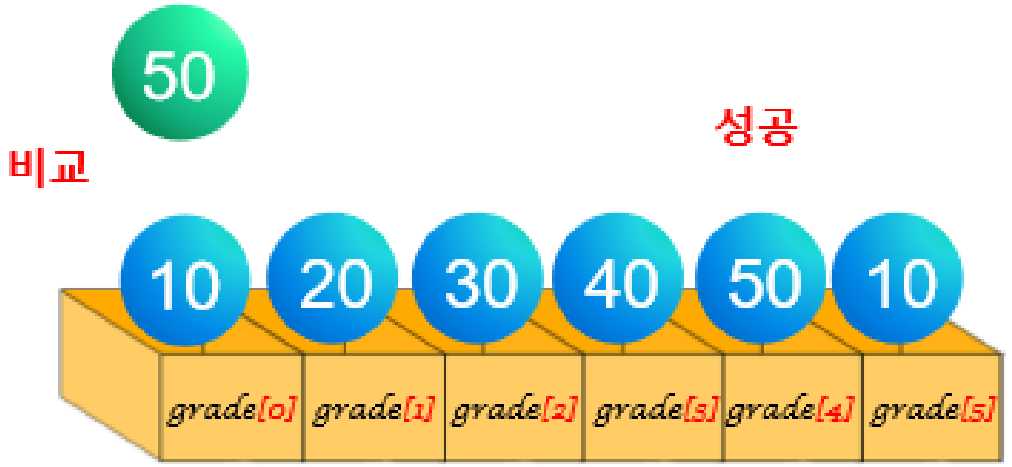


내부 for문. 최소값을 탐색.

Swap. Temp가 있어야 한다는 걸 기억하자.

탐색 – 순차탐색

- **순차탐색(sequential search)**은 가장 간단하고 직접적인 방법이다.
- 배열의 원소를 하나씩 순서대로 비교하는 방법이다.
- 첫번째 원소에서 성공할 수도 있고 마지막까지 갈 수도 있다.



탐색 - 순차탐색

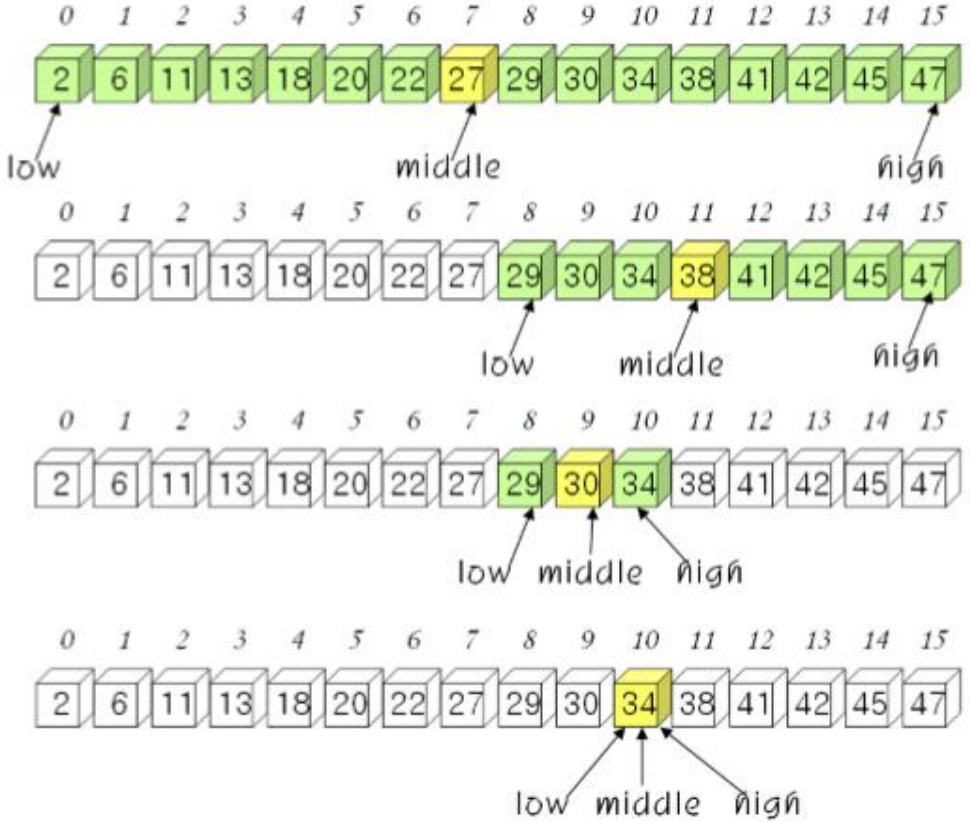
```
C:₩WINDOWS₩system32₩cmd.exe
탐색할 값을 입력하세요 : 5
탐색 성공 인덱스 = 4
탐색종료
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include<stdio.h>
#define SIZE 10

int main() {
    int list[SIZE] = { 1,2,3,4,5,6,7,8,9 };
    int i, key;

    printf("탐색할 값을 입력하세요 : ");
    scanf("%d", &key);

    for (i = 0; i < SIZE; i++) {
        if (list[i] == key)
            printf("탐색 성공 인덱스 = %d\n", i);
    }
    printf("탐색종료\n");

    return 0;
}
```

인덱스는 0부터 시작하므로 1적음!

탐색 – 이진탐색

- 이진 탐색(binary search)은 아주 빠른 탐색 기법이다.
- 대신에 탐색하려는 배열이 정렬이 되어 있어야 한다.
- 배열의 중앙에 있는 값을 탐색키와 비교하고 절반을 버리는 방식을 취한다.



탐색 – 이진탐색

```c
#include<stdio.h>
#define SIZE 16

int binary_search(int list[], int n, int key) {
    int low, middle, high;
    low = 0;
    high = n - 1;
    while (low <= high) {
        printf("[%d %d]\n", low, high);
        middle = (low + high) / 2;
        if (key == list[middle])
            return middle;
        else if (key > list[middle])
            low = middle + 1;
        else
            high = middle - 1;
    }
    return -1;
}
```

```c
int main() {
    int grade[SIZE] = { 2,6,11,13,18,20,22,27,29,31,34,38,41,42,45,47 };
    int key;
    printf("탐색할 값을 입력하세요 : ");
    scanf("%d", &key);
    printf("탐색 결과 : %d\n", binary_search(grade, SIZE, key));

    return 0;
}
```

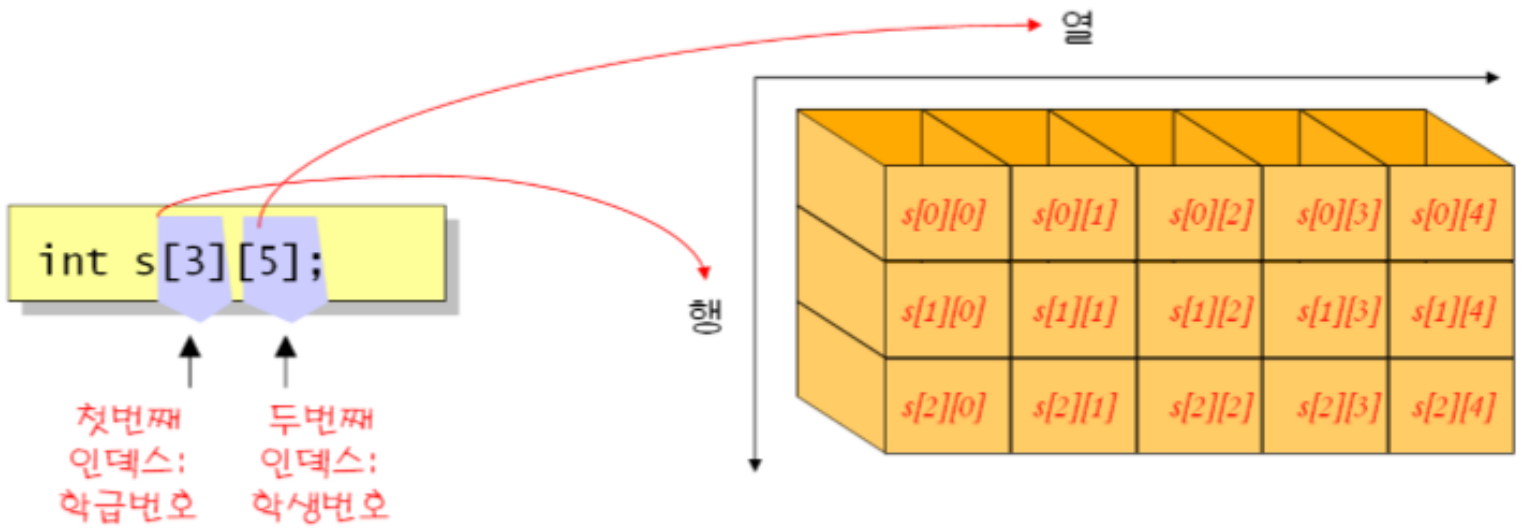
```
C:₩WINDOWS₩system32₩cmd.exe
탐색할 값을 입력하세요 : 11
[0 15]
[0 6]
[0 2]
[2 2]
탐색 결과 : 2
계속하려면 아무 키나 누르십시오 . . .
```

다차원 배열

여태 했던 배열은 1차원 배열이다.

C언어에서는 2차원, 3차원 배열도 가능하며 아래 그림은 2차원 배열을 표현한 것이다.

2차원 배열은 int s[3][5]; , 3차원 배열은 int s[3][5][7];과 같이 선언



실습 문제

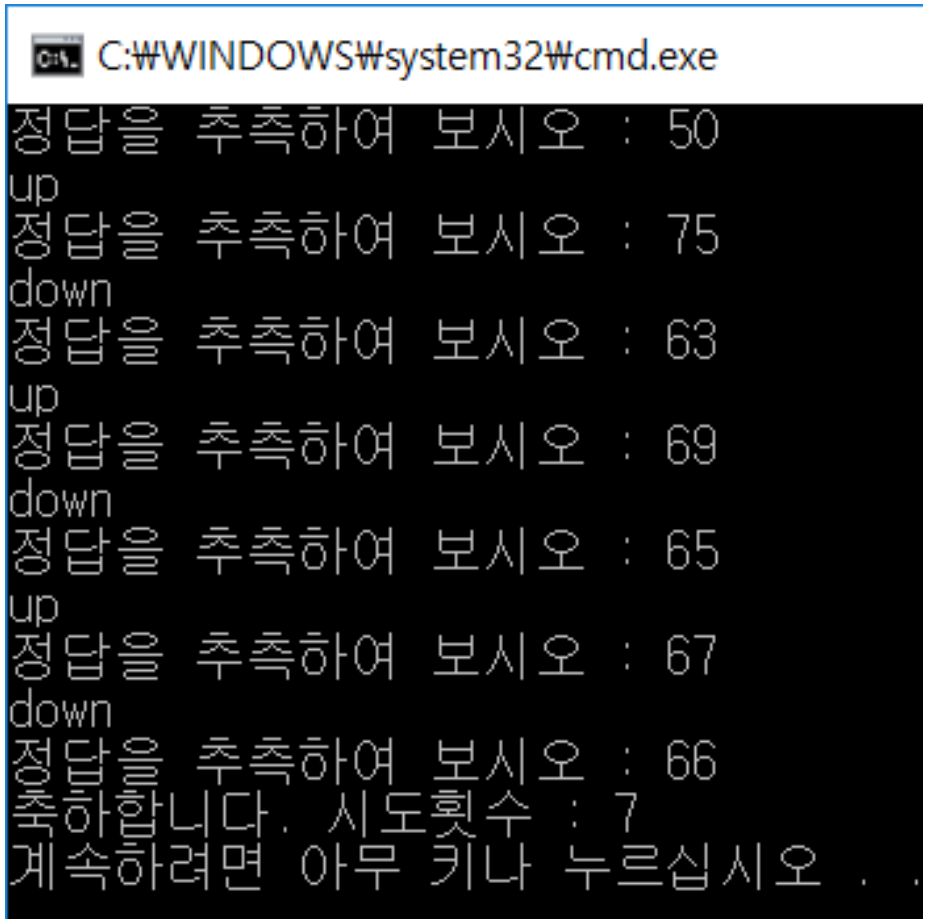
- 간단한 투표 집계 프로그램을 만들어보자. 10명이 후보로 입후보한 경우로 하고 음수를 입력 받으면 종료한다. (입력은 0 ~ 10 사이로 가정)

실습 문제

```c
#include<stdio.h>
#include<math.h>

int main(void)
{
    int candidate[10] = { 0 };

    int i;
    int vote;

    while (1)
    {
        printf("몇 번 후보자를 선택하시겠습니까?(종료 0) : ");
        scanf("%d", &vote);

        if (vote <= 0)
            break;

        candidate[vote - 1]++;
    }

    printf("값\t득표결과\n");

    for (i = 0;i < 10;i++)
    {
        printf("%d\t   %d\n", i + 1, candidate[i]);
    }

    return 0;
}
```

실습 문제

- 업다운 게임을 만들고자 한다. 정답은 랜덤으로 1~100까지 범위로 설정한다.
- 게임이 끝나면 몇 번 시도했는지도 출력하라.



실습 문제

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

int main(void)
{
    int answer, guess;
    int count = 0;

    srand((unsigned)time(NULL));

    answer = rand() % 100 + 1;
    while (1)
    {
        printf("정답을 추측하여 보시오 : ");
        scanf("%d", &guess);
        count++;

        if (guess == answer)
            break;
        else if (guess < answer)
            printf("up\n");
        else
            printf("down\n");
    }


    printf("축하합니다. 시도횟수 : %d\n", count);

    return 0;
}
```

다음주

포인터

감사합니다