

## 포인터와 배열



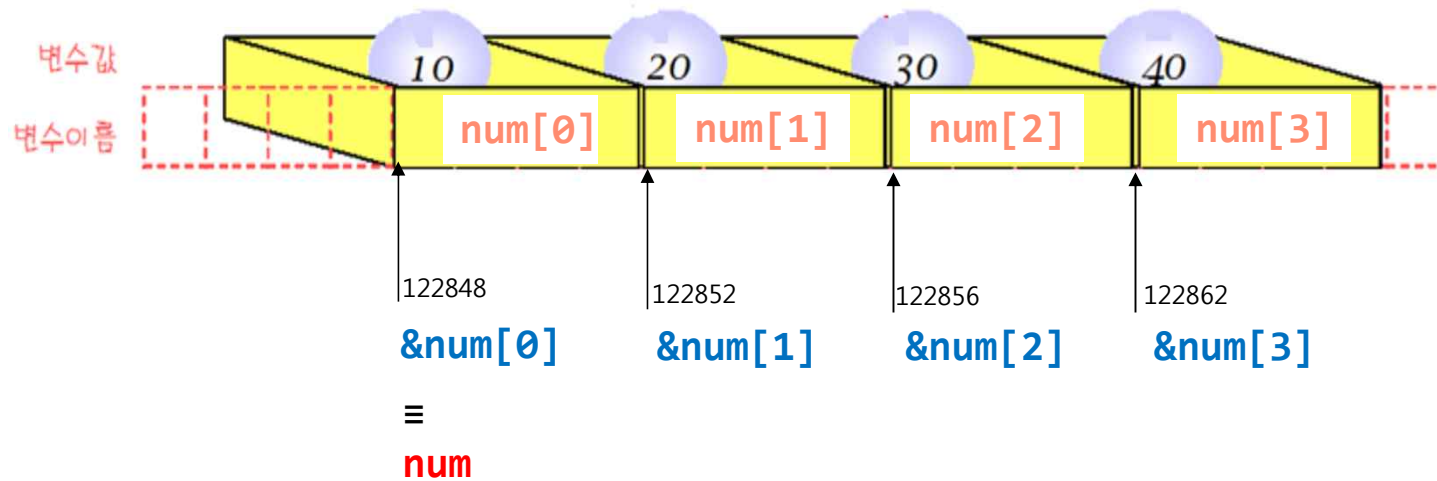
포인터와 배열

포인터 계산과 배열

이중 포인터

## 포인터와 배열명

```
int num[] = { 10, 20, 30, 40 };
```



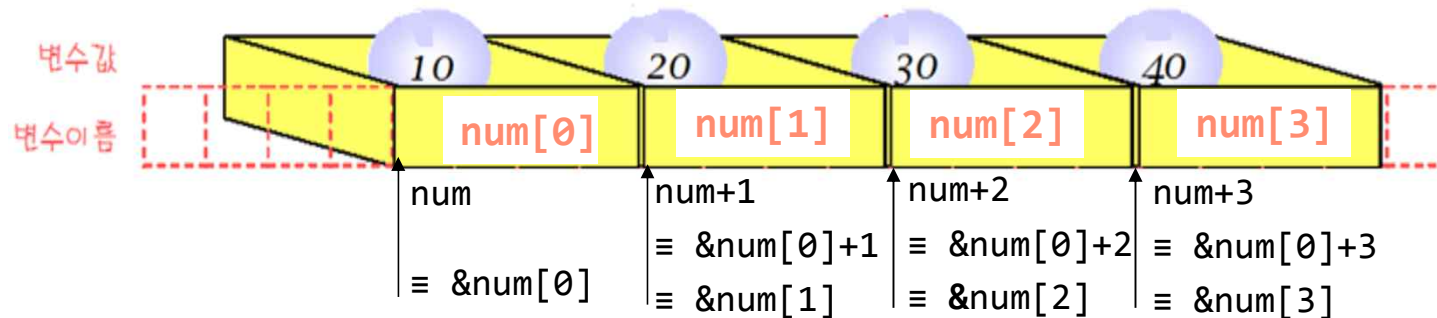
- 배열명은 배열의 첫 번째 원소의 주소

예) num ≡ &num[0]

## 포인터 계산(Pointer Arithmetic)과 배열

int num[] = { 10, 20, 30, 40 } 에서

- 배열명은 배열의 첫 번째 원소의 주소이다
  - num  $\equiv$  &num[0]
- num+1은 num[1]의 주소이다
  - num+1  $\equiv$  &num[0]+1  $\equiv$  &num[1]
- num+3는 num[3]의 주소이다
  - num+3  $\equiv$  &num[0]+3  $\equiv$  &num[3]



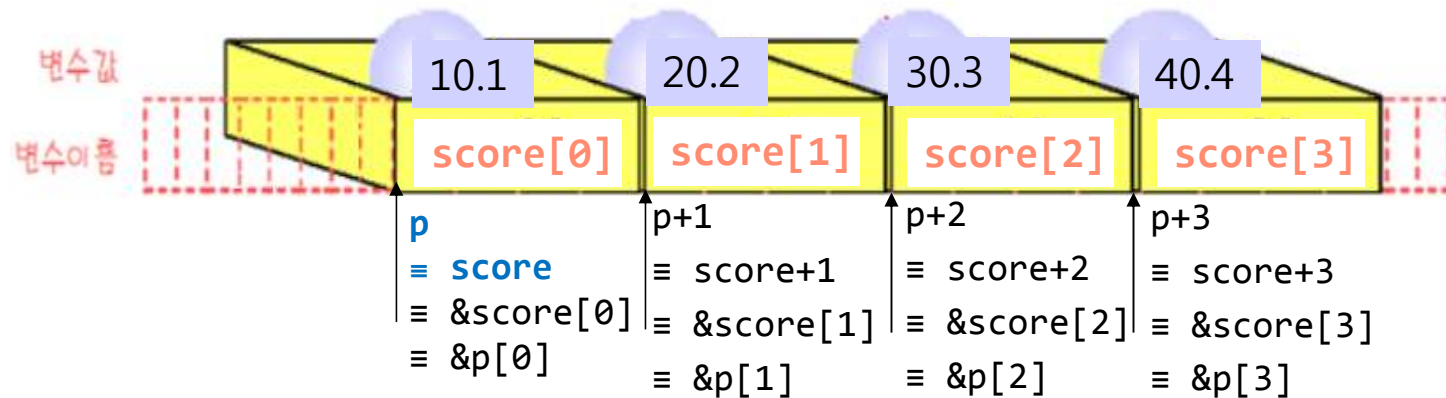
- num+i이 가지고 있는 값은? (단, i는 배열 원소를 가리키는 정수)
  - num+i  $\equiv$  &num[i]

```
double score[] = { 10.1, 20.2, 30.3, 40.4 };
```

```
double *p;
```

```
p = score; 에서
```

- $p+1 \equiv \text{score}+1 \equiv \&\text{score}[1] \equiv \&p[1]$
- $p+3 \equiv \text{score}+3 \equiv \&\text{score}[3] \equiv \&p[3]$



- $p+i$ 와 동일한 표현은? (단,  $i$ 는 배열 원소를 가리키는 정수)
  - $p+i \equiv \&\text{score}[i] \equiv \&p[i]$

## [연습] 아래 코드의 결과를 예측하고, 실행 후 결과와 비교하십시오

```
#include <stdio.h>

int main()
{
    char s[10] = "hello";
    s[0] = 'H';
    s = "i !";
    s[3] = 'W';

    printf("%s\n", s);
    return 0;
}
```

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

```
int main()
```

```
{
```

```
    char s[10] = "hello";
```

```
    s[0] = 'H';
```

```
    s = "i !";
```

```
    s[3] = 'W0';
```

```
    printf("%s\n", s);
```

```
    return 0;
```

```
}
```

-컴파일 에러,

배열명은 배열의 첫 번째 원소의 주소 값인 상수로  
대입 연산자의 왼쪽에 올 수 없음

[연습] 아래 코드의 결과를 메모리 그림으로 그려 예측하고, 실행 후 결과와 비교하시오

```
#include <stdio.h>
int main(void){
    char str[] = "Good Morning@";
    char *p_str;

    p_str = &str[5];
    *p_str = 'E';
    *(p_str + 1) = 'v';

    p_str[2] = 'e';
    str[12] = '#';
    *(str + 12) = '!';


    printf("%s\n", str);
    printf("%s\n", p_str);
    return 0;
}
```

Good Evening!  
Evening!



## 함수 호출 시 배열을 전달하는 방법

-참조에 의한 호출 방식만 사용 가능

```
main()
{
    int a[10000];
    
    sub(a, 10000);
}
```

```
void sub(int *b, int n)
{
    ...
    //(b+3), b[i]등으로 활용
}
```

- 배열의 원소 수는 상당히 큰 것이 일반적이므로,
  - 배열명(=배열 첫 요소의 주소)을 전달하여 기억 장소 및 원소 값 복사 시간에 대한 낭비를 줄임
  - 호출된 함수에서, 전달받은 배열을 변경할 경우 원래의 배열 또한 수정됨  
(단, 호출된 함수에서는 배열의 인덱스 안에서만 사용하도록 주의하여야 함)

## 함수 호출 시 1차원 배열을 전달하는 방법

```
#include <stdio.h>
#define SIZE 5
void PrintArray(int arr[SIZE]);

int main(void) {
    int x[SIZE] = { 10, 20, 30, 40, 50 };

    printf("x 배열 : ");
    PrintArray(x);
    return 0;
}

void PrintArray(int arr[]) {
    int i;
    for (i = 0; i < SIZE; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

### 호출될 함수의 선언

**반환값형** 함수명(자료형 배열명[원소수\_매크로]);  
Void PrintArray(int arr[SIZE]);

### 함수 호출

**호출될 함수명(배열명);**  
int x[SIZE] = { 10, 20, 30, 40, 50 };  
...  
PrintArray(x);

### 호출될 함수의 정의

**반환값형** 함수명(자료형 배열명[ ]);  
void PrintArray(int arr[ ])

생략된 배열 원소 수는 어떻게 알 수 있을까?  
-매크로 활용

[연습] 아래 PrintArray 함수에서 1차원 배열을 포인터로 받아 처리하도록 프로그램을 다시 작성하시오

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

void PrintArray(int arr[SIZE]);

int main(void) {
    int x[SIZE] = { 10, 20, 30, 40, 50 };

    printf("x 배열 : ");
    PrintArray(x);
    return 0;
}

void PrintArray(int arr[]) {
    int i;

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

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

void PrintArray(int *arr);

int main(void) {
    int x[SIZE] = { 10, 20, 30, 40, 50 };

    printf("x 배열 : ");
    PrintArray(x);
    return 0;
}

void PrintArray(int *arr) {
    int i;

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

### 호출될 함수의 선언

반환값형 함수명(자료형 \*포인터);  
void PrintArray(int \*arr);

### 함수 호출

호출될 함수명(배열명);  
int x[SIZE] = { 10, 20, 30, 40, 50 };  
...  
PrintArray(x);

### 호출될 함수의 정의

반환값형 함수명(자료형 \*포인터);  
Void PrintArray(int \*arr);

## [실습] 다음과 같은 조건을 만족하는 프로그램을 작성하시오

### [조건]

- main 함수에 정의된 배열을 인자로 전달받아  
배열 원소의 평균을 구하는 함수와  
배열 각 요소의 값을 제공하는 함수를 각각 정의

- 그 결과는 모두 main 함수에서 출력

배열 원소의 값	:	1	2	3	4	5	6	7	
배열 원소의 평균	:	4							
배열 원소의 제공	:	1	4	9	16	25	36	49	

### [생각해보기]

- 코딩 후, main 함수에 정의된 원본 배열의 값이 변경되었는가?  
원본 배열의 값이 변경되지 않도록 프로그램을 작성할 수 있는 방법은 무엇인가?

## 함수 호출 시 2차원 배열을 전달하는 방법

```
#include <stdio.h>
#define SIZE_1 4
#define SIZE_2 10

void PrintStrArray(char str[SIZE_1][SIZE_2]);

int main(void) {
    char flower[SIZE_1][SIZE_2]
        = { "rose", "sunflower", "lily", "cosmos" };

    printf("꽃 이름 : ");
    PrintStrArray(flower);
    return 0;
}

void PrintStrArray(char str[SIZE_1][SIZE_2]) {
    int i;

    for (i = 0; i < SIZE_1; i++)
        printf("%s ", str[i]);
    printf("\n");
}
```

### 호출될 함수의 선언

```
반환값형 함수명
(자료형 배열명[매크로 1 ][매크로 2]);
void PrintStrArray( char str[SIZE_1][SIZE_2] );
```

### 함수 호출

```
함수명(배열명);
char flower[SIZE_1][SIZE_2]
...
PrintStrArray(flower);
```

### 호출될 함수의 정의

```
반환값형 함수명
(자료형 배열명[매크로 1 ][매크로 2]);
void PrintStrArray( char str[SIZE_1][SIZE_2] );
```

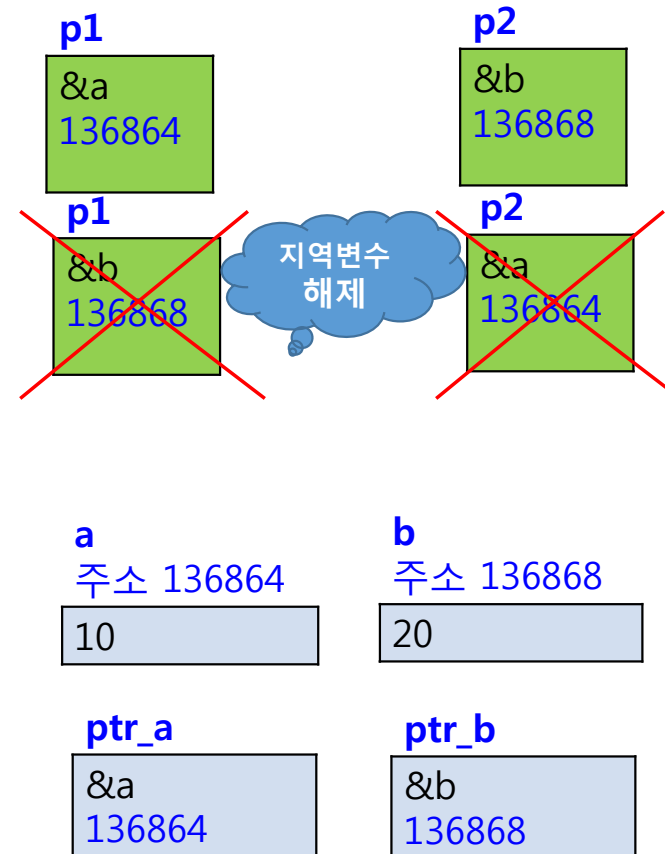
다음 코드의 결과는 무엇이겠습니까? 메모리 그림을 그려서 그 이유를 확인하시오.

```
#include <stdio.h>
void SwapPtr(int *p1, int *p2)
{
    int * temp = p1;
    p1 = p2;
    p2 = temp;
}

void main( )
{
    int a = 10, b = 20;
    int *ptr_a, *ptr_b;
    ptr_a = &a, ptr_b = &b;

    printf("*ptr_a, *ptr_b: %d %d  a, b :%d %d\\n",
           *ptr_a, *ptr_b, a, b);
    SwapPtr(ptr_a, ptr_b);
    printf("*ptr_a, *ptr_b: %d %d  a, b :%d %d\\n",
           *ptr_a, *ptr_b, a, b);
}
```

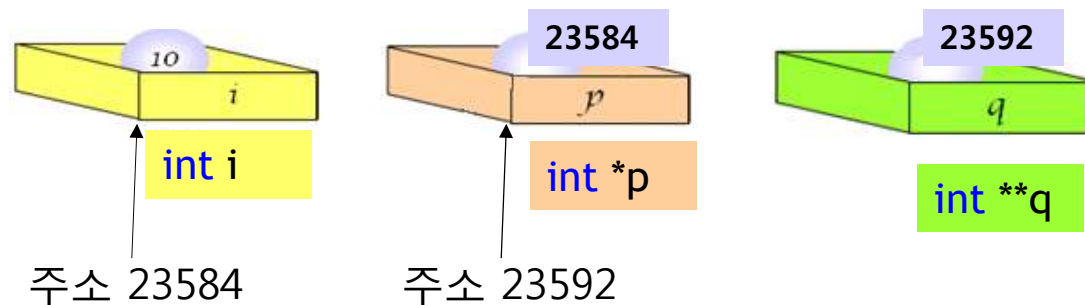
```
10 20 10 20
10 20 10 20
```



## 이중 포인터(Double Pointer)

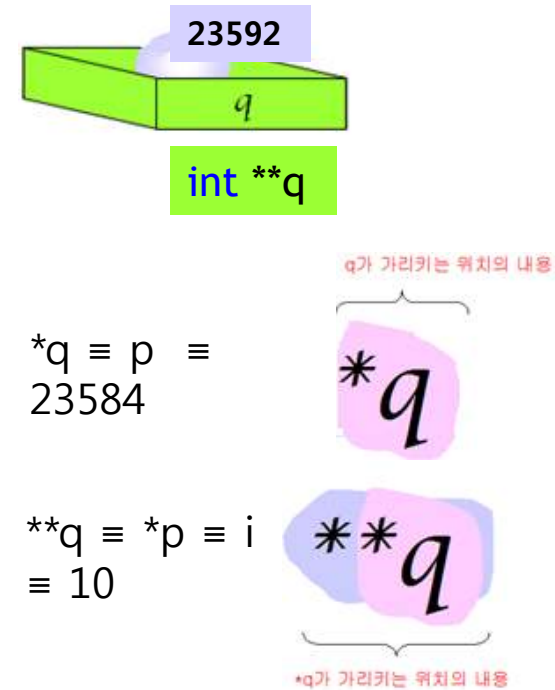
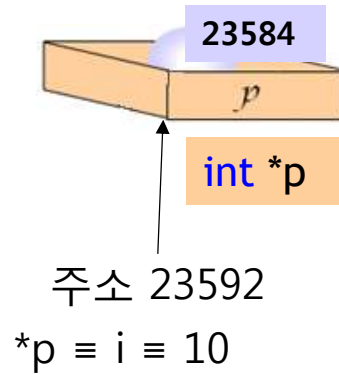
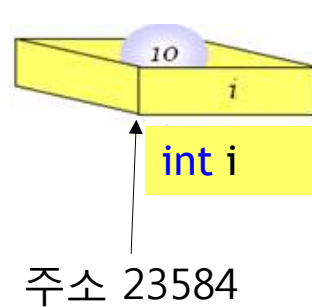
- SwapPtr함수에서 포인터 ptr\_a와 ptr\_b의 값을 변경하려면,
  - 함수 호출 시 **포인터 변수의 주소**를 보내야 한다
- 호출된 함수에서는
  - 포인터 변수의 주소**를 받을 수 있는 **이중 포인터 자료형**을 사용한다

```
int i = 10;
int *p = &i;      // p는 i를 가리키는 포인터
int **q = &p;     // q는 포인터 p를 가리키는 이중 포인터
```





```
int i = 10;
int *p = &i;      // p는 i를 가리키는 포인터
int **q = &p;     // q는 포인터 p를 가리키는 이중 포인터
```



## [연습] 아래 코드의 결과를 메모리 그림으로 그려 예측하고, 실행 후 결과와 비교하시오

```
#include <stdio.h>
int main(void) {
    double i = 103.5;
    double j = 500.9;
    double *p;
    double **q;

    p = &i;
    q = &p;
    printf("&i=%d   p=%d   *q=%d \n", &i, p, *q);

    *p = 200.3;
    printf("i=%f   j=%f   *p=%f   **q=%f \n", i, j, *p, **q);
    **q = 300.4;
    printf("i=%f   j=%f   *p=%f   **q=%f \n", i, j, *p, **q);

    p = &j;
    printf("&j=%d   p=%d   *q=%d \n", &j, p, *q);
    *p = 351.7;
    printf("i=%f   j=%f   *p=%f   **q=%f \n", i, j, *p, **q);
    **q = 925.7;
    printf("i=%f   j=%f   *p=%f   **q=%f \n", i, j, *p, **q);

    return 0;
}
```

&i=1832796	p=1832796	*q=1832796	
i=200.300000	j=500.900000	*p=200.300000	**q=200.300000
i=300.400000	j=500.900000	*p=300.400000	**q=300.400000
&j=1832780	p=1832780	*q=1832780	
i=300.400000	j=351.700000	*p=351.700000	**q=351.700000
i=300.400000	j=925.700000	*p=925.700000	**q=925.700000

[연습] 아래 코드를 다음과 같이 출력되도록 이중 포인터를 사용하여 다시 작성하시오

```
#include <stdio.h>
void SwapPtr(int *p1, int *p2)
{
    int * temp = p1;
    p1 = p2;
    p2 = temp;
}

void main( )
{
    int a = 10, b = 20;
    int *ptr_a, *ptr_b;
    ptr_a = &a, ptr_b = &b;

    printf("*ptr_a, *ptr_b: %d %d   a, b :%d %d\n", *ptr_a, *ptr_b, a, b);
    SwapPtr(ptr_a, ptr_b);
    printf("*ptr_a, *ptr_b: %d %d   a, b :%d %d\n", *ptr_a, *ptr_b, a, b);
}
```

```
*ptr_a, *ptr_b: 10 20   a, b :10 20
*ptr_a, *ptr_b: 20 10   a, b :10 20
계속하려면 아무 키나 누르십시오 . . .
```

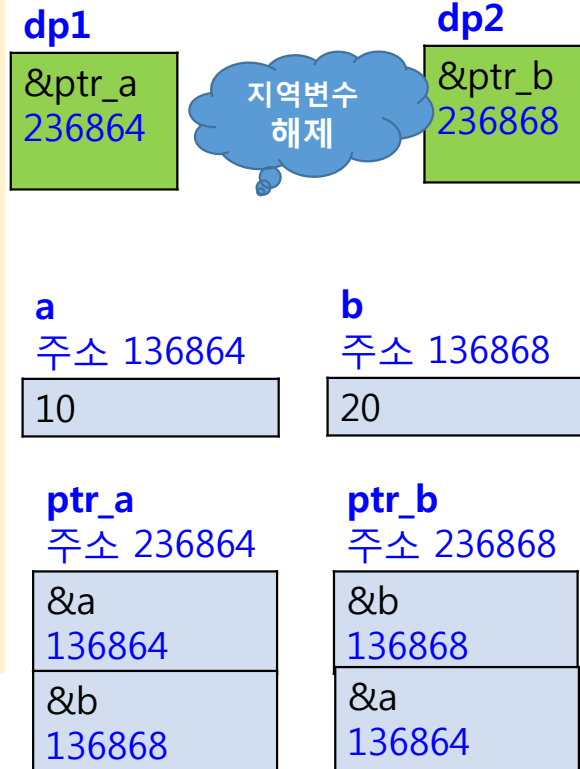
```
#include <stdio.h>
```

```
void SwapPtr(int **dp1, int **dp2){  
    int * temp = *dp1;  
    *dp1 = *dp2;  
    *dp2 = temp;  
}
```

```
void main(){  
    int a = 10, b = 20;  
    int *ptr_a, *ptr_b;  
    ptr_a = &a, ptr_b = &b;
```

```
    printf("*ptr_a, *ptr_b: %d %d   a, b :%d %d\n", *ptr_a, *ptr_b, a, b);  
    SwapPtr(&ptr_a, &ptr_b);  
    printf("*ptr_a, *ptr_b: %d %d   a, b :%d %d\n", *ptr_a, *ptr_b, a, b);  
}
```

```
*ptr_a, *ptr_b: 10 20   a, b :10 20  
*ptr_a, *ptr_b: 20 10   a, b :10 20  
계속하려면 아무 키나 누르십시오 ...
```



## [프로젝트] 다음과 같은 성적처리 함수를 정의하여 코딩하시오

문항	1	2	3	4	5	6	7	8	9	10	점수
1번	O	O	O	O	O	O	O	O	X	O	9
2번	X	O	O	X	X	O	O	O	O	O	7
3번	X	X	O	O	O	O	O	O	O	O	8
4번	O	O	O	O	X	O	O	X	O	O	8
5번	O	O	O	O	O	O	O	X	O	O	9
6번	O	X	X	X	O	X	O	X	O	X	4
7번	O	O	O	O	O	O	O	O	O	O	10

번호	점수	석차
1번	9	2
2번	7	6
3번	8	4
4번	8	4
5번	9	2
6번	4	7
7번	10	1

### [ 조건 ]

- 문항별 정답 및 개별 학생들의 답안이 main함수에 정의되어 있을 때, 개인별 점수 및 석차를 구하는 함수를 작성하시오

- 매크로 : `#define STUDENT 7` // 학생수  
`#define N 10` // 문항수

- 시험의 정답이 저장된 배열:

```
int answer[N] = { 1, 3, 2, 3, 4, 2, 3, 1, 4, 2 };
```

- 학생 STUDENT명의 시험 답안지를 2차원 배열에 초기화하기

```
int paper[STUDENT][N] = { { 1, 3, 2, 3, 4, 2, 3, 1, 3, 2 },
                           { 2, 3, 2, 4, 3, 2, 3, 1, 4, 2 },
                           { 4, 2, 2, 3, 4, 2, 3, 1, 4, 2 },
                           { 1, 3, 2, 3, 3, 2, 3, 4, 4, 2 },
                           { 1, 3, 2, 3, 4, 2, 3, 3, 4, 2 },
                           { 1, 1, 1, 4, 4, 3, 3, 2, 4, 3 },
                           { 1, 3, 2, 3, 4, 2, 3, 1, 4, 2 } };
```

- 이외의 사항은 임의로 결정하되, 한 줄 주석으로 문서화



 T h a n k y o u

## TECHNOLOGY

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Velit ex  
aliquo ipsum, labore sed tempora ratione asperiores des  
cendit hore sed Tempora rati igert one bore sed teni