C 프로그래밍

(001/002)

제9강

신 한 대 학 교
소프트웨어융합학과
교수 송 진 희

신한대학교

7강 – 정리 요약

○지역 변수 / 전역 변수 선언

- 전역 변수 : main() 함수 바깥에서 선언
- 지역 변수 : 블록({ }) 안에서 선언

○변수의 생존 기간

- 정적 할당(static allocation):
  - 프로그램 실행 시간 동안 계속 유지(사용 예 : 웹 페이지 접속자 수)
- 자동 할당(automatic allocation):
  - 블록에 들어갈 때 생성
  - 블록에서 나올 때 소멸

7강 – 정리 요약

○변수의 저장 유형

➢auto(자동 변수) : 함수나 블록 내에서 선언된 지역 변수
- 변수를 선언한 위치에서 생성 / 선언된 블록을 벗어나면 자동 소멸
- auto 예약어 생략 가능

➢static(정적 변수) : 키워드 static 사용해서 변수 선언
- 프로그램이 실행될 때 생성 / 프로그램이 종료되면 소멸

➢register(레지스터 변수) : 데이터를 레지스터에 저장하는 변수
- 키워드 register 사용해서 변수 선언 / 지역 변수에만 사용 가능
- CPU를 사용하므로 처리 속도가 매우 빠름(루프 제어 변수 사용->성능 향상)
- 주소 연산자(&) 사용 불가

➢extern(외부 참조 변수) : 블록에서 전역 변수에 접근

➢volatile(휘발 변수) : 하드웨어가 수시로 변수 값을 변경하는 경우에 사용
- volatile 로 지정된 문장을 실행할 때마다 변수의 새로운 값을 메모리에서 읽어온다.

순환(recursion)이란?

○순환(recursion) : 함수가 자기 자신을 호출하는 것

$$n! = \begin{cases} 1 & n=0 \\ n*(n-1)! & n \geq 1 \end{cases}$$

```
int factorial(int n)
{
   if( n <= 1 )
         return(1);
   else
         return (n * factorial(n-1) );
}
```

0! = 1
1! = 1 * (0!)
2! = 2 * (1!)
3! = 3 * (2!)
:

팩토리얼 구하기

○팩토리얼의 호출 순서

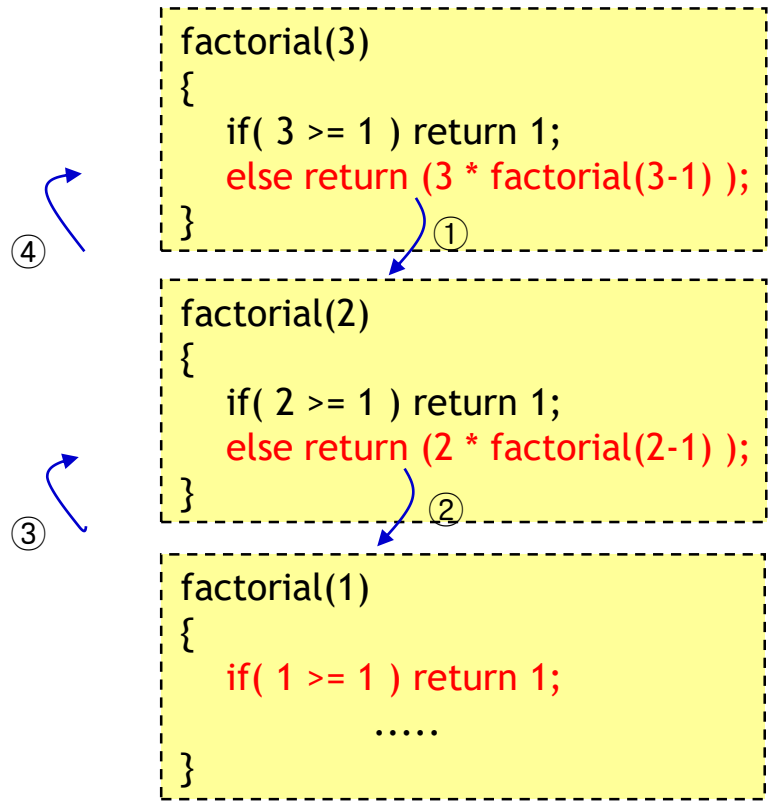
factorial(3) = 3 * factorial(2)

= 3 * 2 * factorial(1)

= 3 * 2 * 1

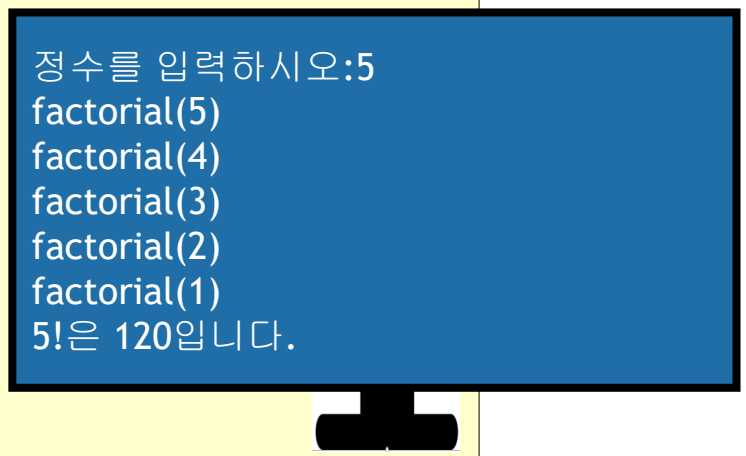
= 3 * 2

= 6



팩토리얼 계산

```c
// 재귀적인 팩토리얼 함수 계산
#include <stdio.h>

long factorial(int n)
{
    printf("factorial(%d) \n", n);

    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}

int main(void)
{
    int x = 0;
    long f;

    printf("정수를 입력하시오:");
    scanf(" % d", &n);
    printf(" %d!은 %d입니다. \n", n, factorial(n));

    return 0;
}
```



C 프로그래밍

제 9 강

- 배열(Array)의 개념
- 1차원 배열 / 다차원 배열
- 배열 선언과 초기화

학습 목표

○배열의 사용 목적을 설명할 수 있다.

○배열명, 배열 요소, 배열 인덱스를 설명할 수 있다.

○1차원 배열을 선언하고 초기화할 수 있다.

○2차원 및 3차원 배열을 선언하고 초기화할 수 있다.

○배열에 데이터를 저장하거나 저장된 배열의 데이터를 검색할 수 있다.

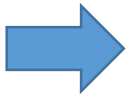
배열(Array)

○배열의 사용 목적

- ➢ **동일한 자료형과 데이터 크기를** 가지는 데이터를 한 개의 배열 이름으로 첨자(index)로 구분하여 저장해서 사용
- ➢배열의 데이터들은 연속적인 공간에 저장 (인덱스는 0부터 시작)
- ➢int s[10]; // 1개의 인덱스를 사용 ➔ 1차원 배열

배열명 ↑ s[10]; 배열 데이터 개수 ↑

```
// 일반 변수 사용
int s0;
int s1;
...
int s9;
```



```
// 배열 사용
int s[10];
```

배열 요소(element) ↑

배열 선언

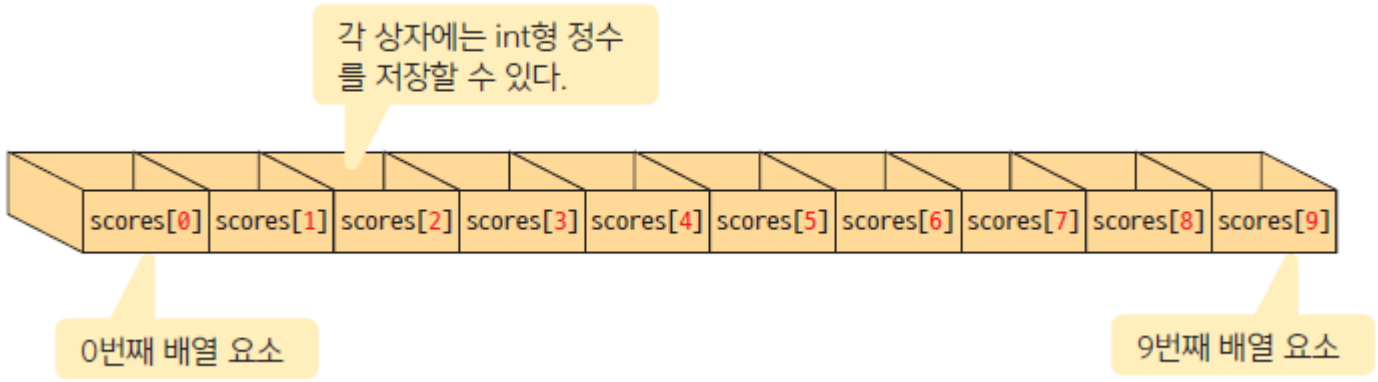
Syntax 배열 선언

자료형 배열 이름 요소의 개수

예

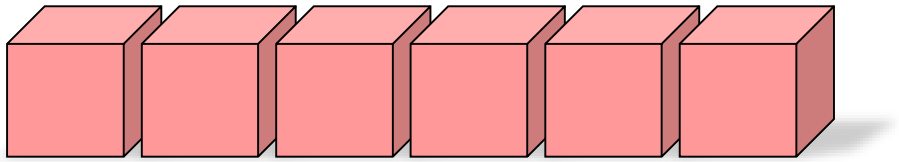
```
int scores[10];
```

○ *인덱스(index):* 배열 원소의 번호



배열 선언의 예

```
// 60개의 int형 값을 가지는 배열 score
int  score[60];

float  cost[12];    // 12개의 float형 값을 가지는 배열 cost

char  name[50];
                    // 50개의 char형 값을 가지는 배열 name
```



배열 크기 지정

○배열의 크기는 상수로 지정

○ 유의 사항

경고

배열의 크기를 나타낼 때는 항상 상수를 사용하여야 한다. 변수를 배열의 크기로 사용하면 컴파일 오류가 된다. 또한 배열의 크기를 음수나 0, 실수로 하면 모두 컴파일 오류이다.

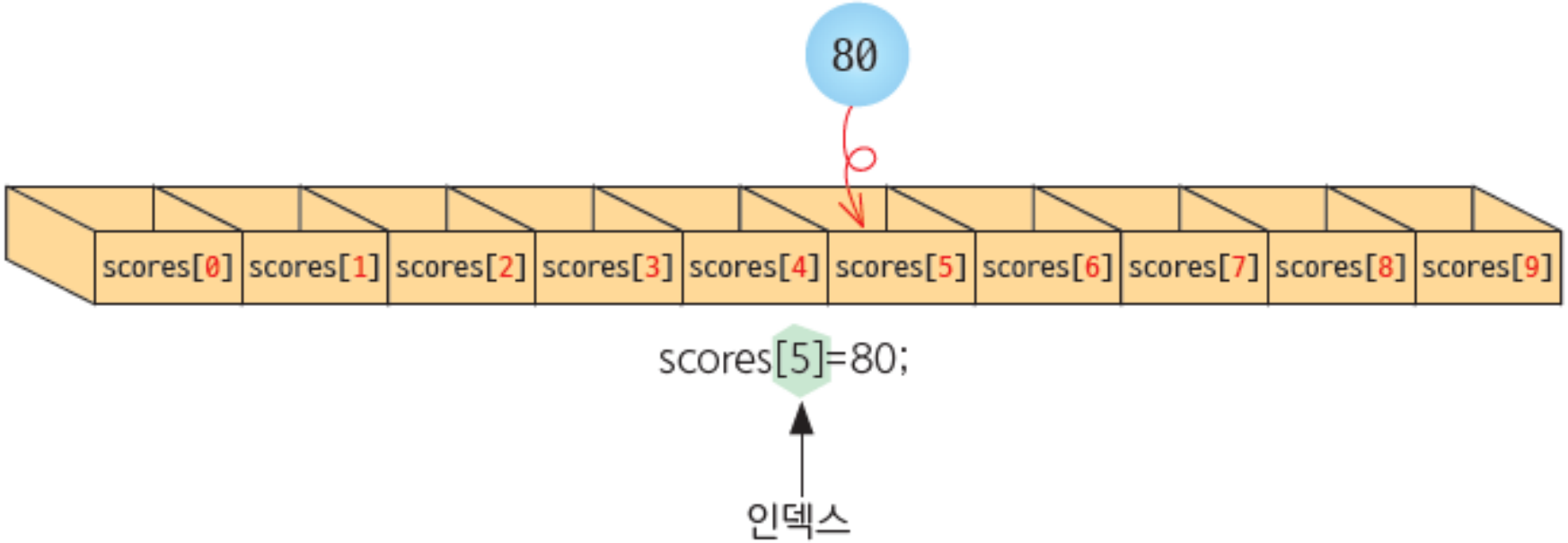
```
int scores[];       // 오류! 배열의 크기를 지정하여야 함
int scores[size];   // 배열의 크기를 변수로 할 수 없음!
int scores[-2];     // 배열의 크기가 음수이면 안 됨
int scores[6.7];    // 배열의 크기가 실수이면 안 됨
```

○배열 크기를 '#define 상수' 로 지정하기

```
#define SIZE 10
int scores[SIZE];
```

#define을 이용한 기호 상수로 배열의 크기를 지정하게 되면 배열의 크기를 변경하기가 쉬워진다. 즉 프로그램의 다른 부분을 수정하지 않고 단지 기호 상수의 정의만 바꾸면 된다.

배열 요소 접근




```
scores[5] = 80;
scores[1] =  scores[0];
scores[i] = 100;              // i는 정수 변수
scores[i+2] = 100;            // 수식이 인덱스가 된다.
scores[index[3]] = 100;       // index[]는 정수 배열
```

배열 기초 예제

- 5개의 정수형 데이터를 저장할 수 있는 배열을 선언한 후,
  10, 20, 30, 40, 50
  의 데이터를 저장하시오.

- 저장된 배열의 데이터를 출력하시오.

```
#include <stdio.h>

int main(void)
{
    int i;
    int scores[5];

    scores[0] = 10;
    scores[1] = 20;
    scores[2] = 30;
    scores[3] = 40;
    scores[4] = 50;

    for(i=0;i < 5; i++)
        printf("scores[%d]=%d\n",  i, scores[i]);
    return 0;
}
```

```
scores[0]=10
scores[1]=20
scores[2]=30
scores[3]=40
scores[4]=50
```

배열과 반복문

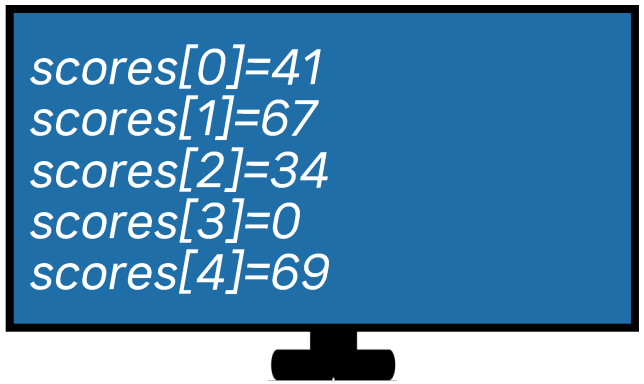
○5개의 정수형 배열에 '0' 으로 초기화

1
```
scores[0] = 0;
scores[1] = 0;
scores[2] = 0;
scores[3] = 0;
scores[4] = 0;
```

2
```
#define SIZE 5
...

for(i = 0;  i < SIZE;  i++)
    scores[i] = 0;
```

배열의 값을 난수로 채우기

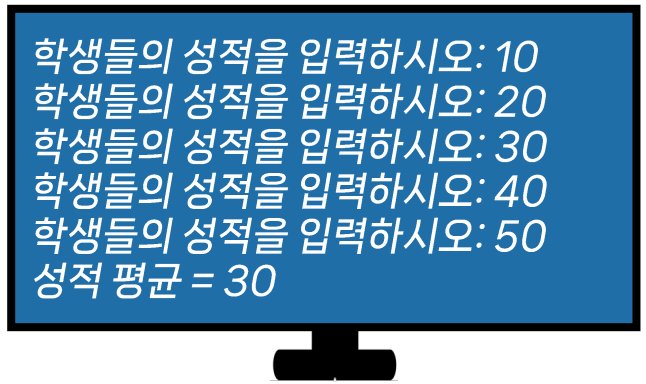
○배열을 정의하고 반복 구조를 사용하여 배열 요소의 값들을 난수로 초기화하고 출력




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

int main(void)
{
    int  i;
    int  scores[SIZE];

    srand((unsigned)time(NULL));    //난수의 seed 지정

    for(i = 0; i < SIZE; i++)
        scores[i] = rand() % 100;

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

예제 #3: 성적 평균 계산하기

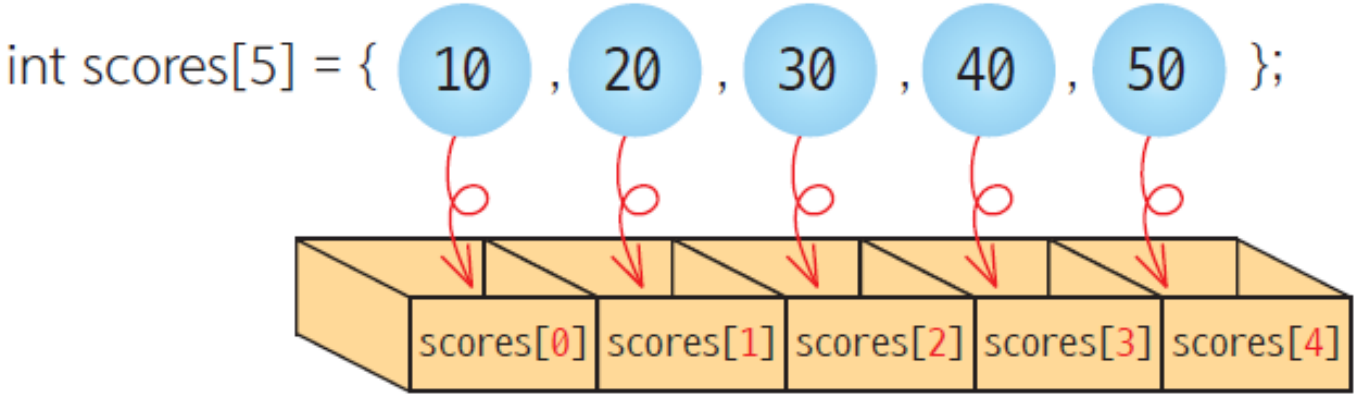
○ 학생 5명의 성적을 배열에 입력한 후, 입력된 학생들의 성적 평균을 구하는 프로그램을 작성해보자.




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

int main(void)
{
	int scores[STUDENTS];
	int sum = 0;
	int i, average;

	for (i = 0; i < STUDENTS; i++)
	{
		printf("학생들의 성적을 입력하시오: ");
		scanf("%d", &scores[i]);
	}

	for (i = 0; i < STUDENTS; i++)
		sum += scores[i];

	average = sum / STUDENTS;
	printf("성적 평균= %d\n", average);

	return 0;
}
```

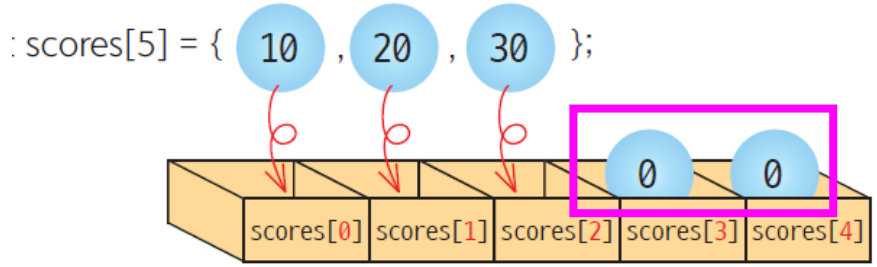
배열의 초기화

○배열을 초기화하려면 초기값들을 콤마로 분리하여 중괄호 { }로 감싼 후에, 배열을 선언할 때 대입



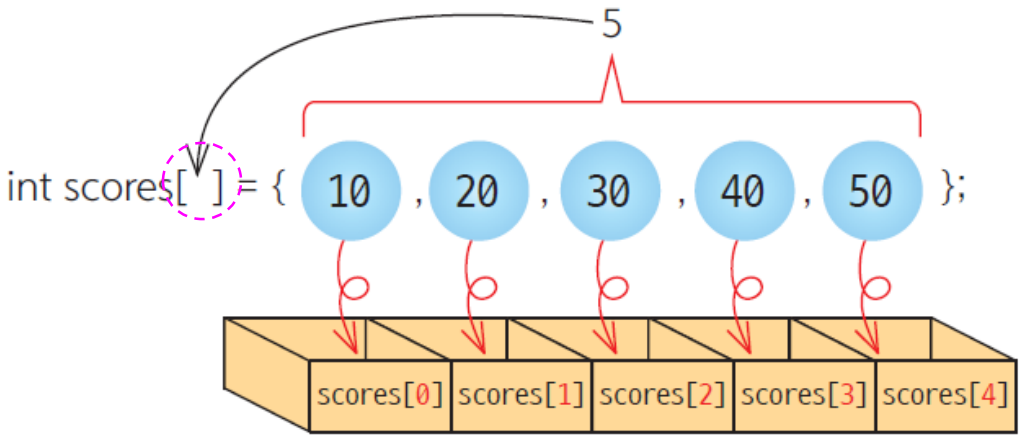
○초기값의 개수가 요소들의 개수보다 적은 경우

➢ 앞에 있는 요소들만 초기화되고, 나머지 배열 요소들은 모두 0으로 초기화



배열의 초기화

○초기화만 하고 배열의 크기를 비워두면 컴파일러가 자동으로 초기값들의 개수만큼의 배열 크기를 잡는다.



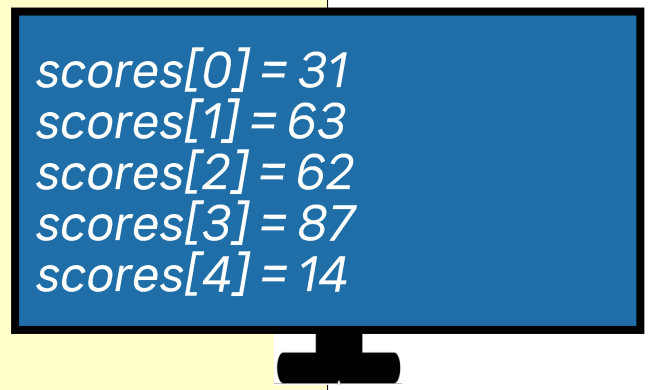
참고사항

배열에서 초기화할 때를 제외하고는 중괄호로 값을 묶어서 대입할 수 없다. 즉 다음과 같이 사용하면 오류가 된다. 배열에 값을 저장하려면 반드시 각 배열 요소에 값을 대입하여야 한다.

```
#define SIZE 3
int main(void)
{
        int scores[SIZE];
        scores = { 6, 7, 8 };   // 컴파일 오류!!
}
```

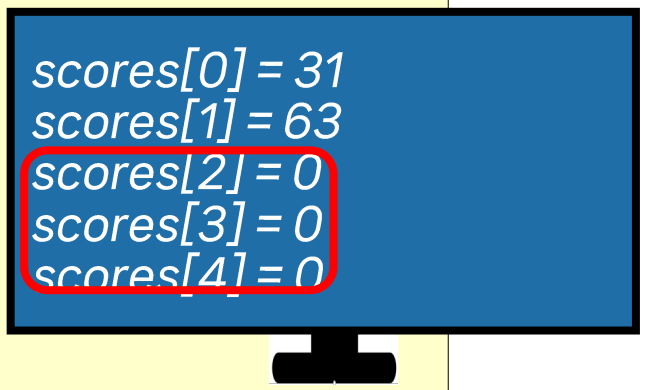
배열 초기화 예제(1)

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

int main(void)
{
    int i;
    int scores[SIZE] = { 31, 63, 62, 87, 14 };


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

    return 0;
}
```



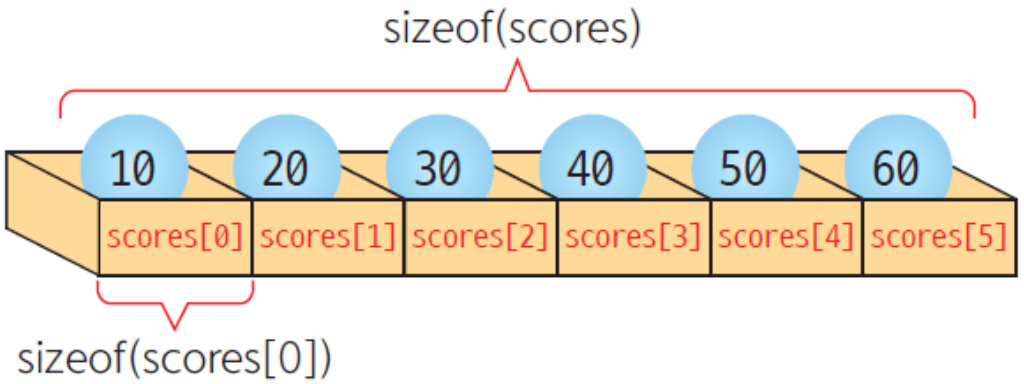
배열 초기화 예제(1)

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

int main(void)
{    //배열 크기보다 초기화  값이 부족한 경우
     int i;
     int scores[SIZE] = { 31, 63 };


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

     return 0;
}
```



배열 원소의 개수 계산

○Sizeof() 함수 : 배열의 크기 또는 배열 원소의 개수를 알려줌




```
int scores[] = { 1, 2, 3, 4, 5, 6 };
int i, size;

size = sizeof(scores) / sizeof(scores[0]);          // 6 / 1 = 6

for(i = 0; i < size ; i++)
      printf("%d ", scores[i]);
```

배열 원소 개수 자동 계산

배열의 복사


```
int a[SIZE] ={1, 2, 3, 4, 5};
int b[SIZE];
a = b;                  // 컴파일 오류!
```

//잘못된 방법

```
int a[SIZE] ={1, 2, 3, 4, 5};
int b[SIZE];
int i;

for(i = 0; i < SIZE; i++)
        b[i] = a[i];
```

원소를 일일이 복사해야 한다

// 올바른 방법

배열의 비교

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

int main(void)
{
        int i;
        int a[SIZE] = { 1, 2, 3, 4, 5 };
        int b[SIZE] = { 1, 2, 3, 4, 5 };

        if( a == b )                    // ① 올바르지 않은 배열 비교
                printf("잘못된 결과입니다.\n");
        else
                printf("잘못된 결과입니다.\n");
```

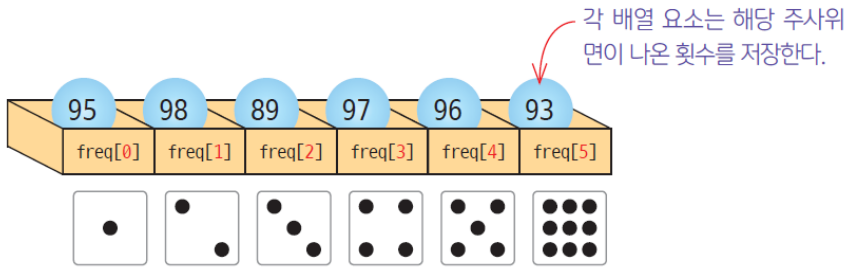

```c
        for(i = 0; i < SIZE ; i++) // ② 올바른 배열 비교
        {
                if ( a[i] != b[i] )
                {
                        printf("a[]와 b[]는 같지 않습니다.\n");
                        return 0;
                }
        }
        printf("a[]와 b[]는 같습니다.\n");
        return 0;
}
```

원소를 하나씩 비교한다

실습#1 : 주사위 던지기

○주사위를 10000번 던져서 6종류의 면들이 나오는 횟수를 출력

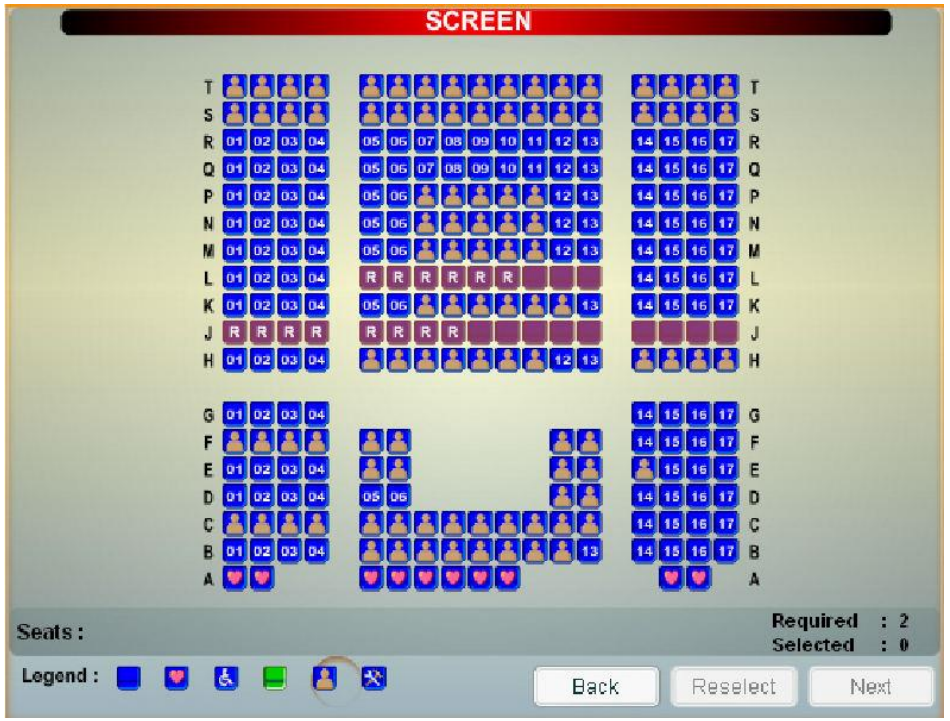
```
====================
숫자     빈도
====================
  0     1657
  1     1679
  2     1656
  3     1694
  4     1652
  5     1662
```




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

#define SIZE 6

int main(void)
{
    int freq[SIZE] = { 0 };// 주사위의 면의 빈도를 0으로 한다.
    int i;

    for (i = 0; i < 10000; i++)// 주사위를 10000번 던진다.
        ++freq[rand() % 6];// 해당면의 빈도를 하나 증가한다.

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

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

    return 0;
}
```

실습#2 : 극장 예약 시스템

- 배열을 이용하여 간단한 극장 예약 시스템을 작성
- 좌석은 10개
- 예약이 끝난 좌석은 1로, 예약이 안 된 좌석은 0으로 나타낸다.




```
좌석을 예약하시겠습니까?(y 또는 n) y
------------------------------
1 2 3 4 5 6 7 8 9 10
------------------------------
0 0 0 0 0 0 0 0 0 0
몇 번째 좌석을 예약하시겠습니까? 1
예약되었습니다.
좌석을 예약하시겠습니까?(y 또는 n) y
------------------------------
1 2 3 4 5 6 7 8 9 10
------------------------------
1 0 0 0 0 0 0 0 0 0
몇 번째 좌석을 예약하시겠습니까? 1
이미 예약된 자리입니다.
좌석을 예약하시겠습니까?(y 또는 n) n
```

알고리즘

1. *while(1)*
2. *사용자로부터 예약 여부(y 또는 n)를 입력받는다.*
3. *if 입력 == 'y'*
4. *현재의 좌석 배치표 seats[]를 출력한다.*
5. *좌석 번호 i를 사용자로부터 입력받는다.*
6. *if 좌석번호가 올바르면*
7. *seats[i]=1*
8. *else*
9. *에러 메시지를 출력한다.*
10. *else*
11. *종료한다.*

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#define SIZE 10

int main(void)
{

   char  ans1;
   int  ans2, i;
   int seats[SIZE] = { 0 };    // '0'은 예약 가능 상태임

   while (1)
   {
       printf("좌석을 예약하시겠습니까?(y 또는n) ");
       scanf(" %c", &ans1);

       if (ans1 == 'n')        // 예약하지 않음
          break;
```

Lab: 극장 좌석 예약

```
    printf("-----------------------------\n");
    printf(" 1 2 3 4 5 6 7 8 9 10\n");
    printf("-----------------------------\n");

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

    printf("몇 번째 좌석을 예약하시겠습니까");
    scanf("%d", &ans2);              // 예약 번호 입력

    if (seats[ans2 - 1] == 0) {     // 예약 가능하면
        seats[ans2 - 1] = 1;         // 예약 완료
        printf("예약되었습니다.\n");
    }
    else// 이미 예약되었으면
        printf("이미 예약된 자리입니다.\n");
  }
  return 0;
}
```

현재 좌석 예약 상태 출력

실습 제출 #3

○위의 프로그램에서는 한 명만 예약할 수 있다. 극장에 친구와 가는 경우, 한번에 2명을 예약할 수 있도록 위의 프로그램을 변경하여 보자. (예약 가능한 좌석 수가 2개 미만인 경우의 처리 필요함)

○ 작성된 소스 프로그램을 사이버캠퍼스에 제출하세요.

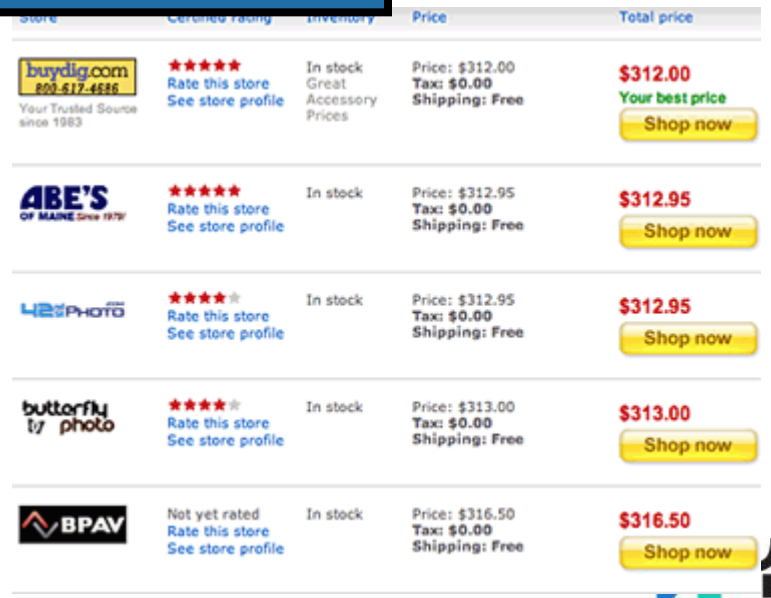
실습 : 최소값 찾기

○우리는 인터넷에서 상품을 살 때, 가격 비교 사이트를 통하여 가장 싼 곳을 검색한다.

○➔ 배열에 들어 있는 정수 중에서 최소값을 찾는 문제와 동일

실행 결과

```
----------------------------------------
1  2  3  4  5  6  7  8  9  10
----------------------------------------
28 81 60 83 67 10 66 97 37 94

최소값은 10입니다.
```



알고리즘

1. *배열 prices[]의 원소를 난수로 초기화한다.*
2. *일단 첫 번째 원소를 최소값 minium이라고 가정한다.*
3. *for(i=1; i< 배열의 크기; i++)*
4. *if (prices[i] < minimum)*
5. *minimum = prices[i]*
6. *반복이 종료되면 minimum에 최소값이 저장된다.*

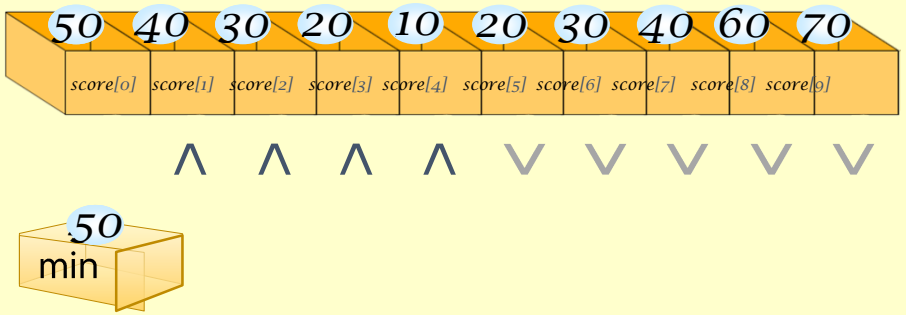
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 10

int main(void)
{
        int prices[SIZE] = { 0 };
        int i, minimum;
        printf("-------------------------------------\n");
        printf("1 2 3 4 5 6 7 8 9 10\n");
        printf("-------------------------------------\n");
        srand( (unsigned)time( NULL ) );
        for(i = 0; i < SIZE; i++){
                prices[i] = (rand()%100)+1;
                printf("%-3d ", prices[i]);
        }
        printf("\n\n");
```

물건의 가격 출력

실습 : 최소값 찾기

```
        // 첫 번째 배열 원소를 최소값으로 가정
        minimum = prices[0];
        for(i = 1; i < SIZE; i++)
        {
                if( prices[i] < minimum )
                        minimum = prices[i];
        }
        printf("최소값은 %d입니다.\n", minimum);
        return 0;
}
```

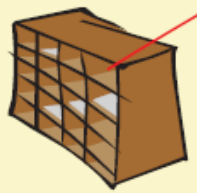
현재의 최소값보다 배열 원소값이 작으면, 배열 원소를 최소값으로 교체한다.



배열과 함수

❍배열을 함수의 매개변수로 전달할 경우, 사본이 아닌 원본이 전달된다.

```
int main(void)
{
  ...

  get_average(  , STUDENTS);

  ...
}
```



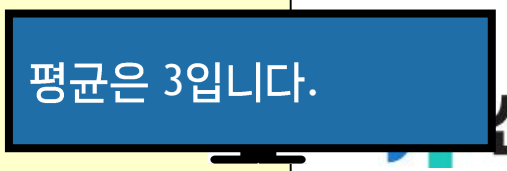
```
int get_average(  , int size)
{
  ...

  sum += scores[i];

  ...
}
```

배열과 함수

```
#include <stdio.h>
#define STUDENTS 5
int get_average(int scores[], int n);    // ①

int main(void)
{
    int scores[STUDENTS] = { 1, 2, 3, 4, 5 };
    int avg;

    avg = get_average(scores, STUDENTS);
    printf("평균은 %d입니다.\n", avg);
    return 0;
}

int get_average(int scores[], int n)    // ②
{
    int i;
    int sum = 0;

    for(i = 0; i < n; i++)
        sum += scores[i];
    return sum / n;
}
```

배열이 인수인 경우,
배열의 주소가 전달된다.

배열의 원본이 score[]로 전달

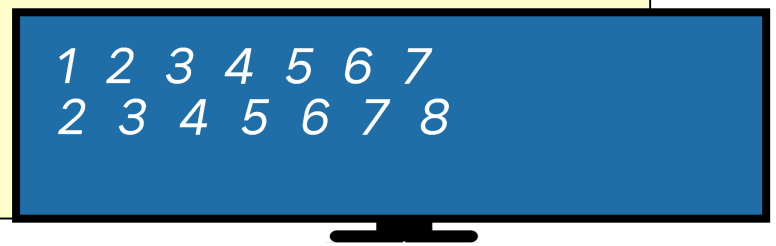


배열이 함수의 인수인 경우 (#1)

```
#include <stdio.h>
#define SIZE 7

void modify_array(int a[], int size);
void print_array(int a[], int size);

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

        print_array(list, SIZE);          // 배열은 주소가 전달되어 출력
        modify_array(list, SIZE);         // 배열은 주소가 전달되어 각 배열 요소 값이 1씩 증가
        print_array(list, SIZE);          // 배열은 주소가 전달되어 출력

        return 0;
}
```

배열이 함수의 인수인 경우(#2)

```
void modify_array(int a[], int size)
{
        int i;

        for(i = 0; i < size; i++)
                ++a[i];
}

void print_array(int a[], int size)
{
        int i;

        for(i = 0; i < size; i++)
                printf("%3d ", a[i]);
        printf("\n");
}
```



원본 배열의 변경을 금지하는 방법

const 키워드는 변경이 불가능

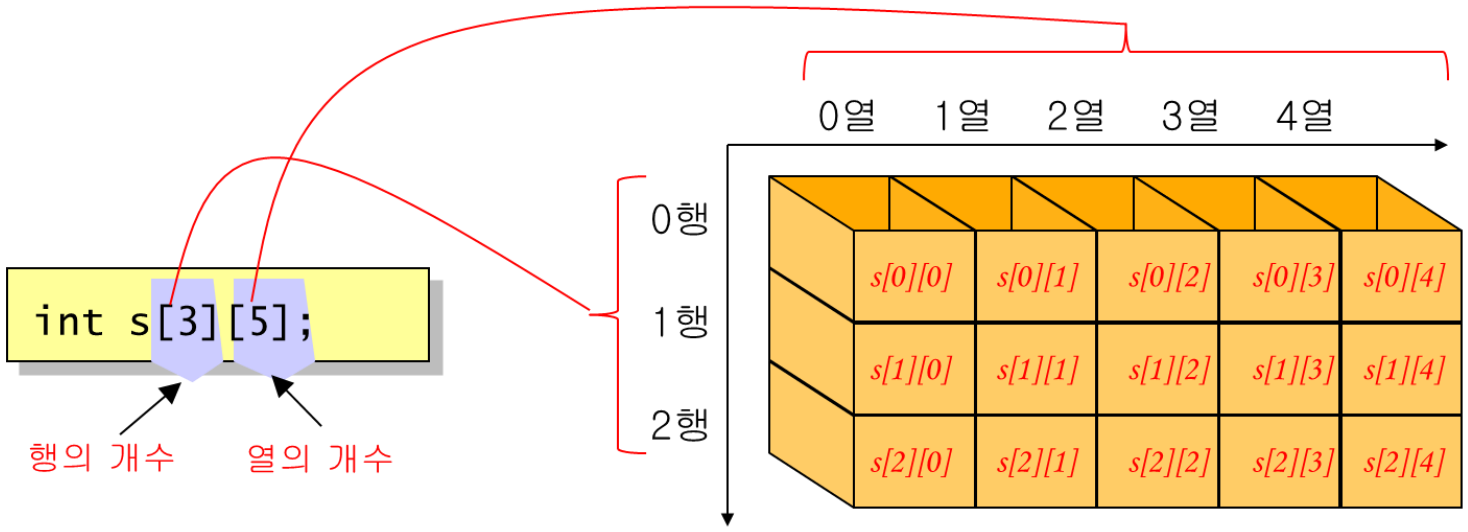
```
void print_array(const int a[], int size)
{
        ...
        a[0] = 100;          // 컴파일 오류!
}
```

함수 안에서 a[]는 변경할 수 없다.

2차원 배열

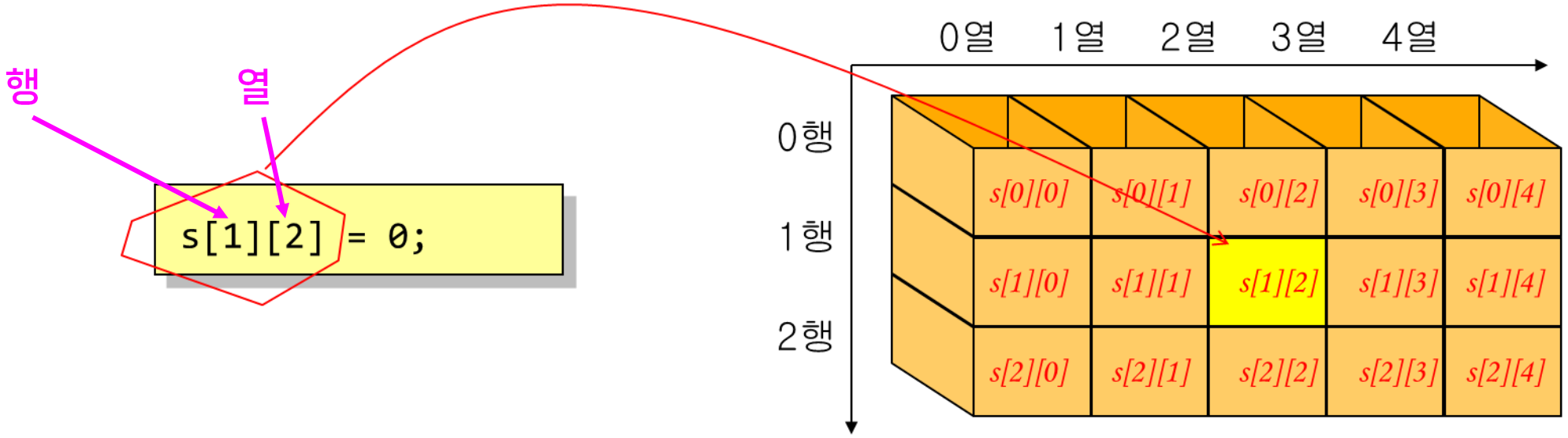
○배열의 데이터를 2개의 인덱스(행, 열)로 표현

```
int s[10];          // 1차원 배열 - 열
int s[3][10];       // 2차원 배열 – 행, 열
int s[5][3][10];    // 3차원 배열 – 행, 열, 면
```



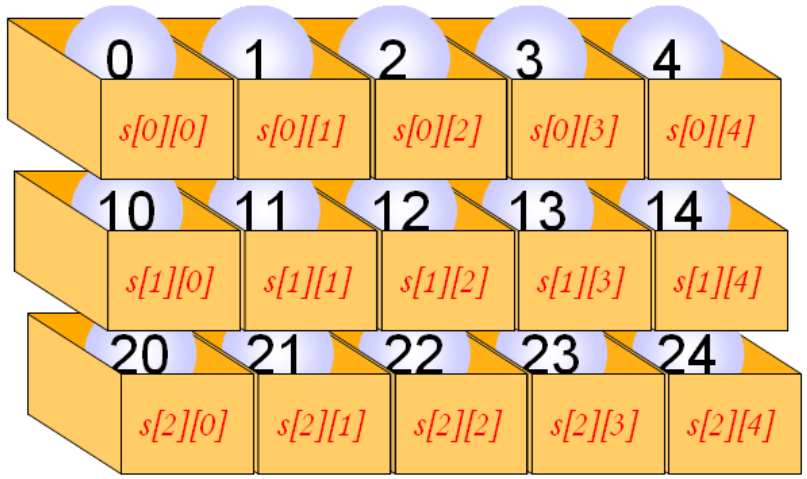
2차원 배열에서 인덱스

○2차원 배열의 데이터 초기화



2차원 배열의 초기화 (1)

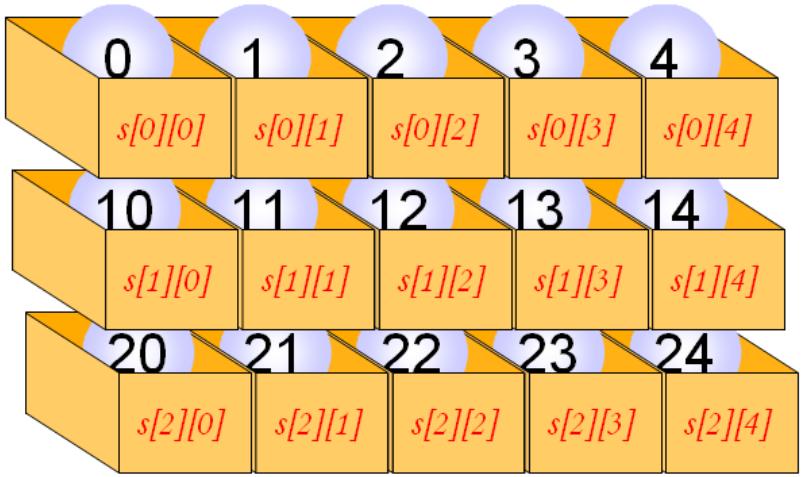
```
int s[3][5] = {
        { 0, 1, 2, 3, 4 },          // 첫 번째 행의 원소들의 초기값
        { 10, 11, 12, 13, 14 },     // 두 번째 행의 원소들의 초기값
        { 20, 21, 22, 23, 24 }      // 세 번째 행의 원소들의 초기값
};
```



2차원 배열의 초기화 (2)

// 행의 개수 미지정

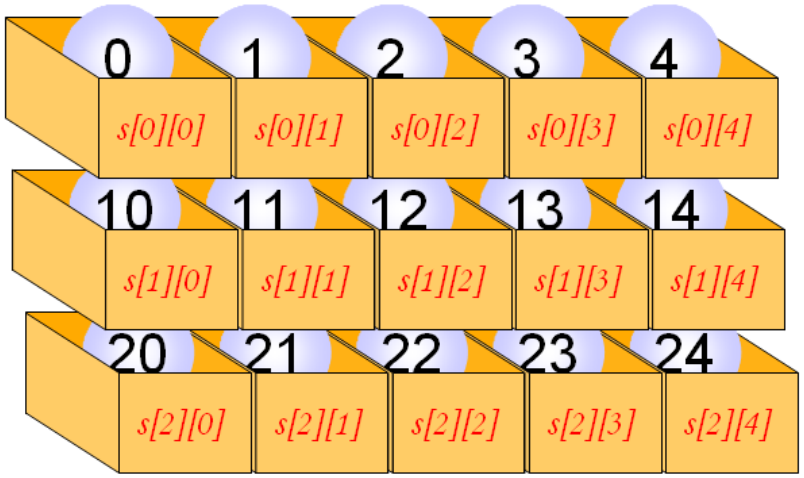
```
int s[ ][5] = {
      { 0, 1, 2, 3, 4 }, // 첫 번째 행의 원소들의 초기값
      { 10, 11, 12, 13, 14 }, // 두 번째 행의 원소들의 초기값
      { 20, 21, 22, 23, 24 }, // 세 번째 행의 원소들의 초기값
};
```



2차원 배열의 초기화 (3)

// 행의 개수 미지정, 배열 요소 값을 연속적으로 표기

```
int s[ ][5] = {
  0, 1, 2, 3, 4, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24
};
```



예제

○학생들의 성적 기록표를 2차원 배열에 저장하고 각 학생의 최종 성적을 계산해보자.

학번	중간고사(30%)	기말고사(40%)	기말과제(20%)	퀴즈점수(10%)	결석횟수(감점)
1	87	98	80	76	3
2	99	89	90	90	0
3	65	68	50	49	0

2차원 배열의 초기화

```
학생 #1의 최종성적= 85.90
학생 #2의 최종성적= 92.30
학생 #3의 최종성적= 61.60
```

```c
#include <stdio.h>
#define ROWS 3
#define COLS 5


int main(void)
{
    int i;

    int a[ROWS][COLS] = {
        { 87, 98, 80, 76, 3 },
        { 99, 89, 90, 90, 0 },
        { 65, 68, 50, 49, 0 }
    };


    for (i = 0;  i < ROWS;  i++) {
        double  final_scores = a[i][0] * 0.3 + a[i][1] * 0.4 +
            a[i][2] * 0.2 + a[i][3] * 0.1 - a[i][4];        // 점수의 각 항목별 비중을 곱함
        printf("학생 #%i의 최종성적=%10.2f \n", i + 1, final_scores);
    }
    return 0;
}
```

2차원 배열을 이용한 행렬 표현

```
#include <stdio.h>
#define ROWS 3
#define COLS 3

int main(void)
{
        int A[ROWS][COLS] = {  { 2,3,0 },
                               { 8,9,1 },
                               { 7,0,5 } };
        int B[ROWS][COLS] = {  { 1,0,0 },
                               { 1,0,0 },
                               { 1,0,0 } };
        int C[ROWS][COLS];
```

2차원 배열을 이용한 행렬 표현

```
        int r, c;


        // 두개의 행렬을 더한다.
        for(r = 0; r < ROWS; r++)
                for(c = 0;  c < COLS;  c++)
                        C[r][c] = A[r][c] + B[r][c];


        // 행렬을 출력한다.
        for(r = 0;  r < ROWS;  r++)
        {
                for(c = 0;  c < COLS; c++)
                        printf("%d ", C[r][c]);
                printf("\n");
        }
        return 0;
}
```

// 중첩 for 루프를 이용하여 행렬 A의 각 원소들과 행렬 B의 각 원소들을 서로 더하여 행렬 C에 대입한다.

2차원 배열을 함수로 전달하기

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

#define  YEARS 3
#define  PRODUCTS 5

int sum(int scores[YEARS][PRODUCTS]);

int main(void)
{
      int sales[YEARS][PRODUCTS] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
      int total_sale;

      total_sale = sum(sales);            //배열명을 전달
      printf("총매출은 %d입니다.\n", total_sale);


      return 0;
}
```

2차원 배열을 함수로 전달하기

```
int sum(int scores[YEARS][PRODUCTS])
{
      int  y, p;
      int  total = 0;

      for (y = 0;  y < YEARS;  y++)
            for (p = 0;  p < PRODUCTS;  p++)
                  total += scores[y][p];

      return total;
}
```

9강 – 정리 요약

○ **배열의 사용 목적**

- **동일한 자료형과 데이터 크기를** 가지는 데이터를 한 개의 배열 이름으로 첨자(index)로 구분하여 저장해서 사용
- 배열의 데이터들은 연속적인 공간에 저장 (인덱스는 0부터 시작)
- 배열 크기 : 상수 또는 #define 상수로 지정

○ **1차원 배열**

- int s[10]; // 1개의 인덱스(열)를 사용 ➔ 1차원 배열

○ **2차원 배열**

- int s[3][10]; // 2개의 인덱스를 사용 ➔ 행=>3 , 열 =>10


신 한 대 학 교
SHINHAN UNIVERSITY