

C 언어 EXPRESS(개정3판)



CH07 배열



이번 장에서 학습할 내용



- 반복의 개념 이해
- 배열의 개념
- 배열의 선언과 초기화
- 일차원 배열

배열을
사용하면 한
번에 여러 개의
값을 저장할 수
있는 공간을
할당받을 수
있다.





배열

- 배열을 사용하면 한 번에 여러 개의 변수를 생성할 수 있다.
- `int s[10];`



변수

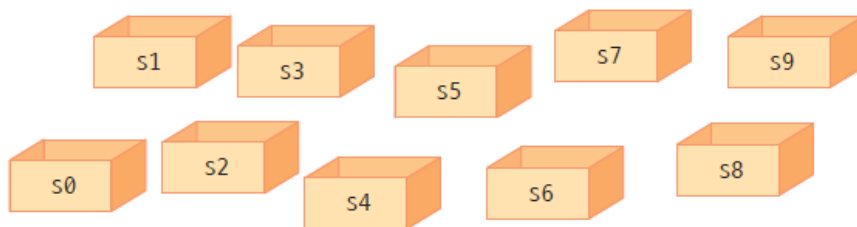


아파트



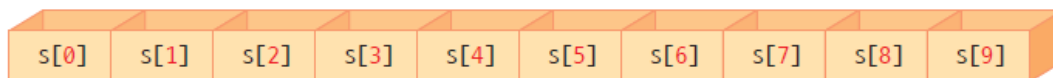
배열의 필요성

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



별도의 이름을 가지니
조작하기가 어렵군!

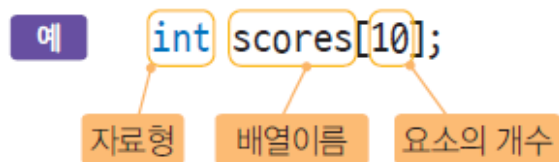
```
// 배열 사용  
int s[10];
```





배열 선언

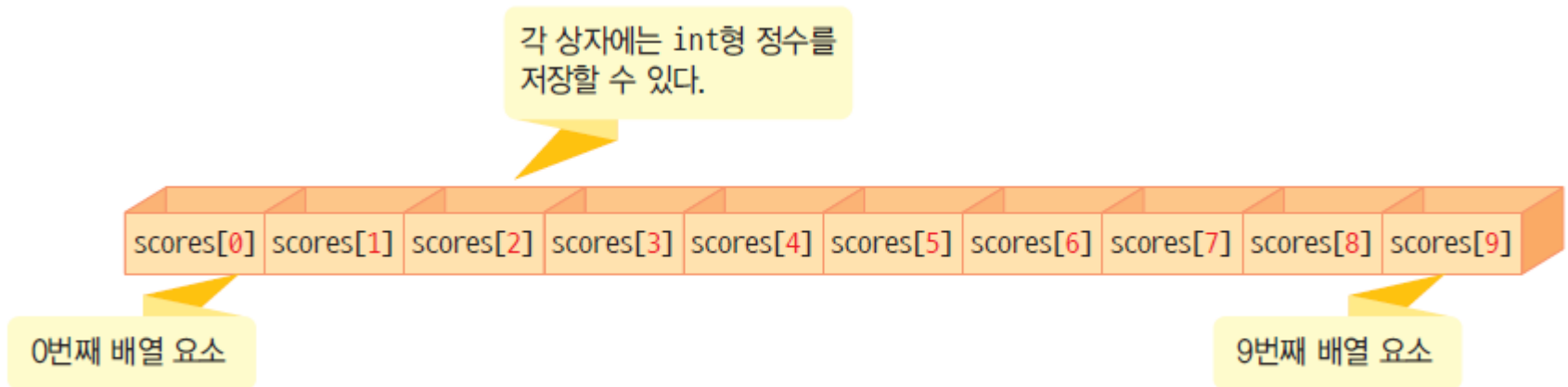
Syntax: 배열 선언





배열 원소와 인덱스

- 인덱스(index): 배열 원소의 번호





배열 선언의 예

```
int score[60];
```

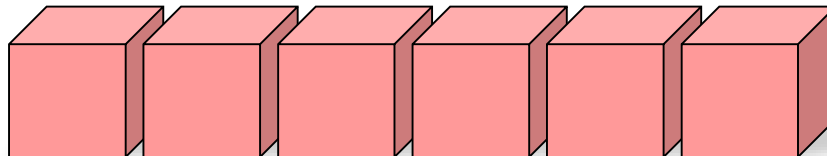
// 60개의 int형 값을 가지는 배열 score

```
float cost[12];
```

// 12개의 float형 값을 가지는 배열 cost

```
char name[50];
```

// 50개의 char형 값을 가지는 배열 name





주의

경고

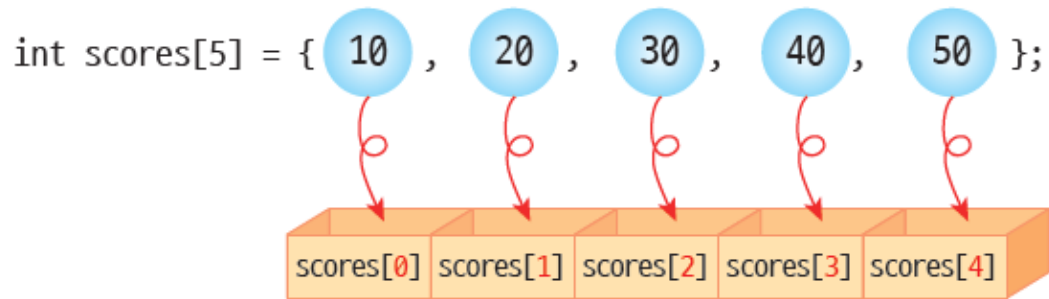
배열의 크기를 나타낼 때는 **항상 상수를 사용하여야 한다.** 변수를 배열의 크기로 사용하면 컴파일 오류가 된다. 또한 배열의 크기를 음수나 0, 실수로 하면 모두 컴파일 오류이다.

```
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
```





배열의 초기화

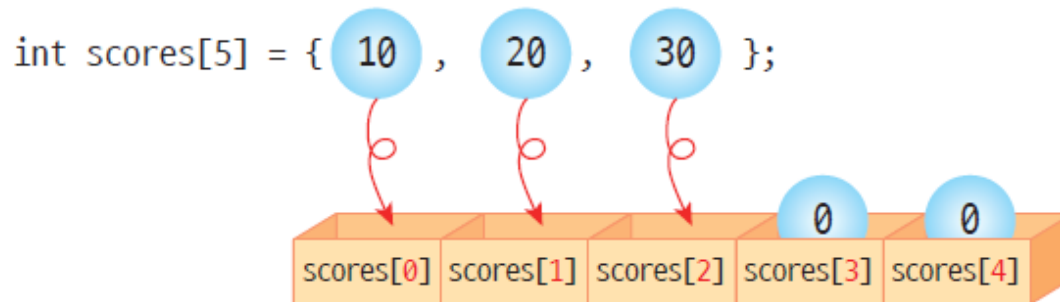


원소들의 초기값을 콤마로 분리하여 중괄호 안에 나열합니다.





배열의 초기화

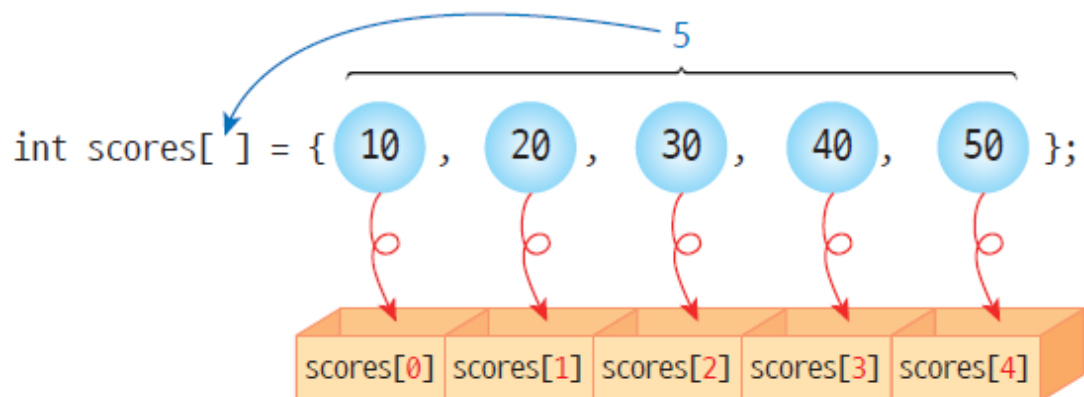


초기값을 일부만 주면 나머지
원소들은 0으로 초기화됩니다.





배열의 초기화



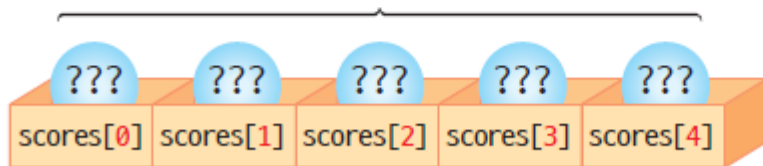
배열의 크기가 주어지지 않은 경우에는 초기값의 개수가 배열의 크기가 됩니다.





초기값을 주지 않았다면?

```
int main(void)
{
    int scores[5];
    ...
}
```



배열을 지역변수로 선언하면
초기화되지 않은 배열은
쓰레기값을 가지게 됩니다.





배열 초기화 비교 예제

Ch07_ex1.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    int empty_arr[5] ;
    int scores[5] = { 3,6 };
    int i = 0;

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

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

    return 0;
}
```

```
empty_arr[0] = -858993460
empty_arr[1] = -858993460
empty_arr[2] = -858993460
empty_arr[3] = -858993460
empty_arr[4] = -858993460
scores[0] = 3
scores[1] = 6
scores[2] = 0
scores[3] = 0
scores[4] = 0
```



경고



경고

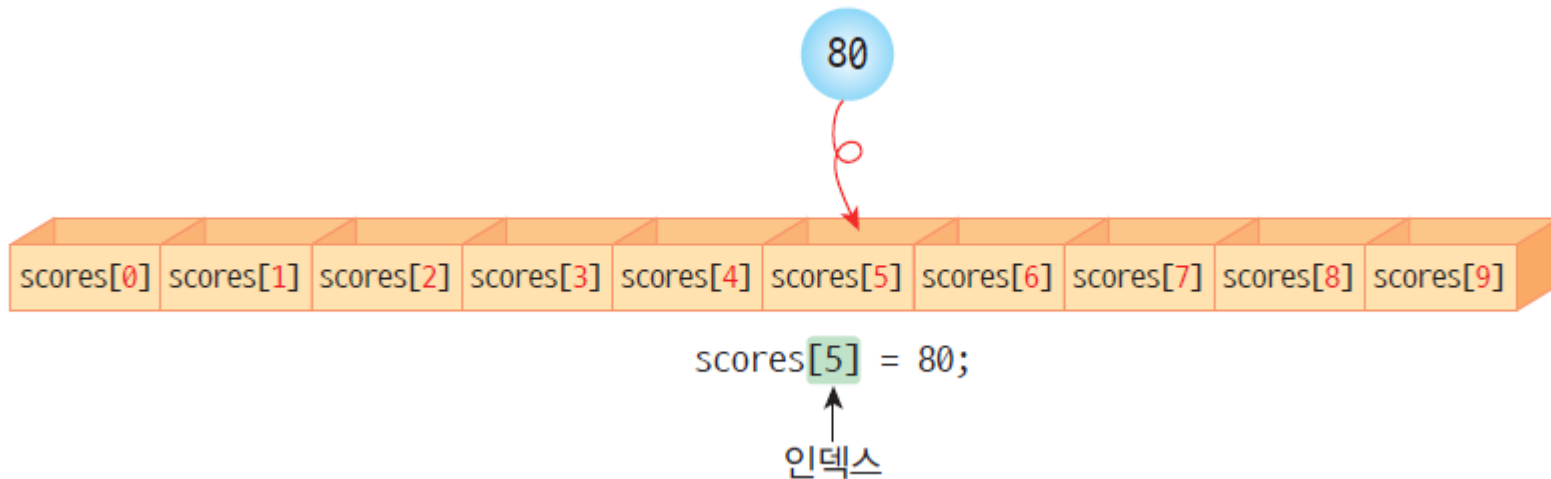
배열의 모든 요소를 10으로 초기화하려고 다음과 같이 작성하면 오류이다.

```
int scores[10] = { 10 };
```

이 때는 첫 번째 요소만 10이 되고 나머지 요소는 전부 0이 된다.



배열 요소 접근



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



배열과 반복문

- 배열의 가장 큰 장점은 반복문을 사용하여 배열의 원소를 간편하게 처리할 수 있다는 점



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

```
#define SIZE 5  
...  
for(i=0 ; i<SIZE ; i++)  
    scores[i] = 0;
```

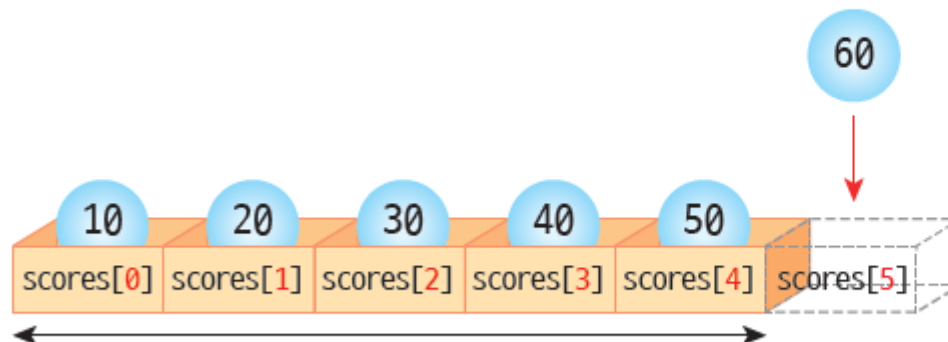




잘못된 인덱스 문제

- 인덱스가 배열의 크기를 벗어나게 되면 프로그램에 치명적인 오류를 발생시킨다.
- C에서는 프로그래머가 인덱스가 범위를 벗어나지 않았는지를 확인하고 책임을 져야 한다.

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



존재하지 않는 곳에 데이터를 저장하면 안됩니다.





```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    int scores[5] = { 31, 63, 62, 87, 14 };
    int i = 0;
    int arr_val = 0;

    arr_val = scores[2]; // 배열값 가져 오기
    printf("arr_val=%d\n", arr_val);

    scores[3] = 100; // 배열값 변경하기

    // 반복문으로 배열값 확인 변경하기
    for(i = 0; i < 5; i++){
        printf("scores[%d] = %d\n", i, scores[i]);
    }

    for (i = 0; i < 5; i++) {
        scores[i] = i*10;
    }
    for (i = 0; i < 5; i++) {
        printf("scores[%d] = %d\n", i, scores[i]);
    }
}
```

```
arr_val=62
scores[0] = 31
scores[1] = 63
scores[2] = 62
scores[3] = 100
scores[4] = 14
scores[0] = 0
scores[1] = 10
scores[2] = 20
scores[3] = 30
scores[4] = 40
```



참고

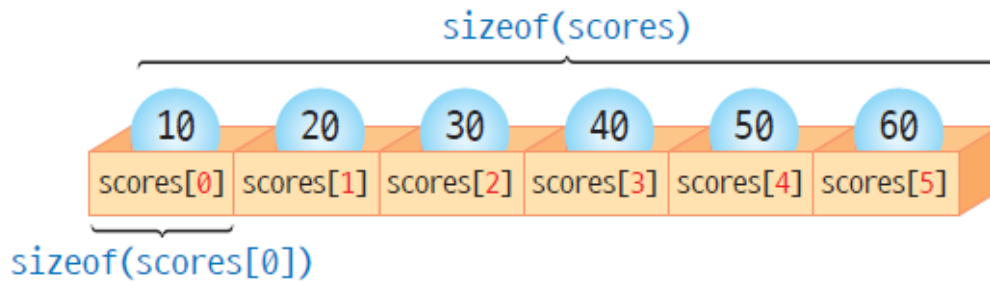
★ 참고사항

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 원소의 개수 계산



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

Ch07_ex3.c

```
int main(void)
```

```
{
```

```
    int score[] = { 1, 2, 3, 4, 5, 6 };
```

```
    int i, size;
```

```
    size = sizeof(score) / sizeof(score[0]);
```

배열 원소 개수 자동 계산

```
    for(i = 0; i < size ; i++)
```

```
        printf("score[%d] = %d\n", i, score[i]);
```

```
    return 0;
```

```
}
```



배열의 복사



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

```
int b[SIZE];
```

```
a = b; // 컴파일 오류!
```

잘못된 방법



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

```
int c[SIZE];
```

```
int i;
```

```
for(i = 0; i < SIZE; i++)
```

```
    c[i] = b[i];
```

원소를 일일이
복사한다

올바른 방법



배열의 비교

Ch07_ex4.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};
```

```
    int c[SIZE];
```

```
    printf("%d - %d\n", a,b);
```

```
    for(i = 0; i < SIZE ; i++) { //배열 값비교
```

```
        if ( a[i] != b[i] ) {
```

```
            printf("다릅니다.\n");
```

```
            break;
```

```
        }
```

```
    }
```

```
    for (i = 0; i < SIZE; i++){
```

```
        c[i] = b[i];
```

```
    }
```

```
    for (i = 0; i < SIZE; i++) {
```

```
        printf("%d ", c[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

원소를 하나씩 비교한다



배열과 함수

Ch07_ex5.c

```
#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;
}
```

배열이 인수인 경우,
배열의 주소가 전달된다.

배열의 원본이
score[]로 전달

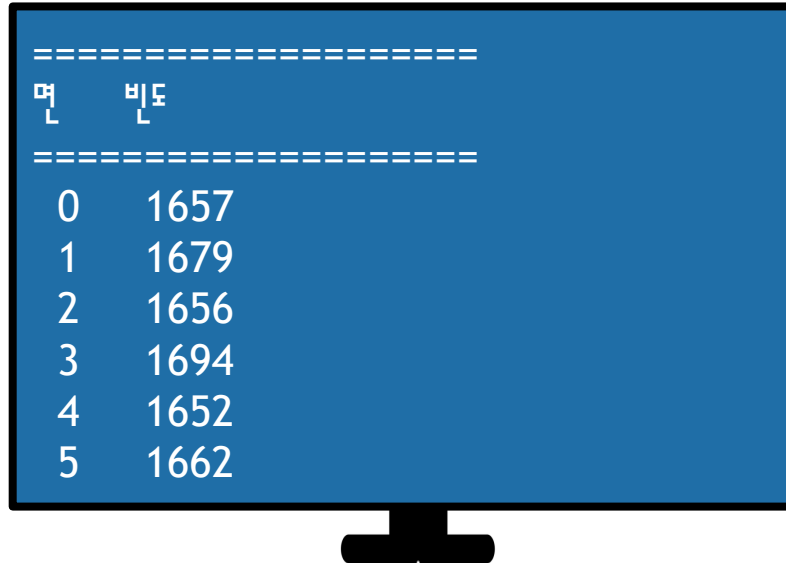
```
int get_average(int scores[], int n) // ②
{
    int i;
    int sum = 0;

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



Lab: 주사위 던지기

- 이 Lab에서는 주사위를 1000번 던져서 각 면이 나오는 횟수를 출력하여 보자.



면	빈도
0	1657
1	1679
2	1656
3	1694
4	1652
5	1662





```
#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("면    빈도\n");
```

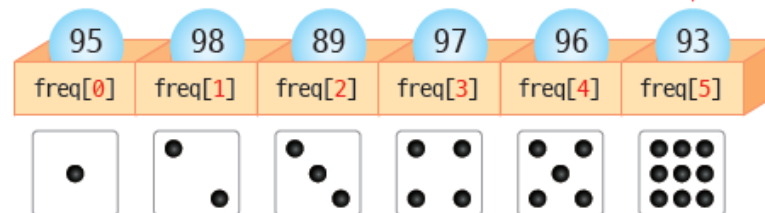
```
    printf("=====\n");
```

```
    for (i = 0; i < SIZE; i++)
```

```
        printf("%3d    %3d \n", i, freq[i]);
```

```
    return 0;
```

```
}
```

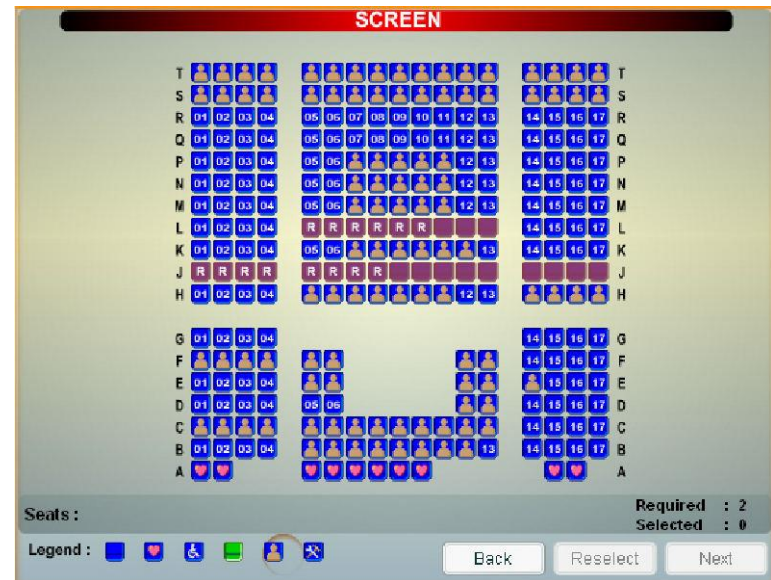


각 배열 요소는 해당 주사위 면이 나온 횟수를 저장한다.



Lab : 극장 예약 시스템

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





실행 결과

```
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
몇번째 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
몇번째 좌석을 예약하시겠습니까? 1
이미 예약된 자리입니다. 다른 좌석을 선택하세요
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.
```



Lab: 극장 좌석 예약

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#define SIZE 10

int main(void)
{

    char ans1;
    int ans2, i;
    int seats[SIZE] = { 0 };

    while (1)
    {

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

        if (ans1 == 'n')
            break;
```



Lab: 극장 좌석 예약

현재 좌석 예약
상태 출력

```
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("몇번째 좌석을 예약하시겠습니까");  
scanf("%d", &ans2);  
if (seats[ans2 - 1] == 0) { // 예약되지 않았으면  
    seats[ans2 - 1] = 1;  
    printf("예약되었습니다.\n");  
}  
else // 이미 예약되었으면  
    printf("이미 예약된 자리입니다.\n");  
}  
return 0;  
}
```



도전문제

- 위의 프로그램에서는 한 명만 예약할 수 있다. 하지만 극장에 혼자서 가는 경우는 드물다. 따라서 한번에 2명을 예약할 수 있도록 위의 프로그램을 변경하여 보자.





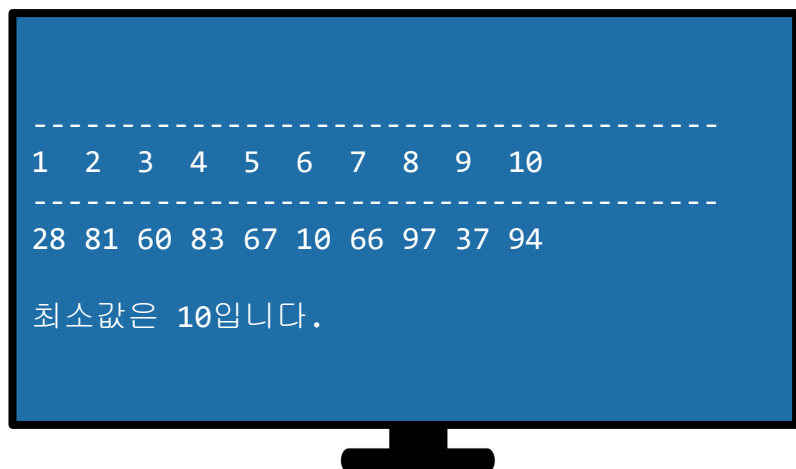
Lab: 최소값 찾기






- 우리는 인터넷에서 상품을 살 때, 가격 비교 사이트를 통하여 가장 싼 곳을 검색한다.
- 일반적으로 배열에 들어 있는 정수 중에서 **최소값**을 찾는 문제와 같다.





실행 결과



Store	Certified rating	Inventory	Price	Total price
 Your Trusted Source since 1983	★★★★★ Rate this store See store profile	In stock Great Accessory Prices	Price: \$312.00 Tax: \$0.00 Shipping: Free	\$312.00 Your best price Shop now
 Since 1979	★★★★★ Rate this store See store profile	In stock	Price: \$312.95 Tax: \$0.00 Shipping: Free	\$312.95 Shop now
	★★★★★ Rate this store See store profile	In stock	Price: \$312.95 Tax: \$0.00 Shipping: Free	\$312.95 Shop now
	★★★★★ Rate this store See store profile	In stock	Price: \$313.00 Tax: \$0.00 Shipping: Free	\$313.00 Shop now
	Not yet rated Rate this store See store profile	In stock	Price: \$316.50 Tax: \$0.00 Shipping: Free	\$316.50 Shop now



알고리즘

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



Lab: 최소값 찾기

```
#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



Lab: 최소값 찾기

```
minimum = prices[0];
```

```
for(i = 1; i < SIZE; i++)  
{
```

```
    if( prices[i] < minimum )  
        minimum = prices[i];
```

```
}
```

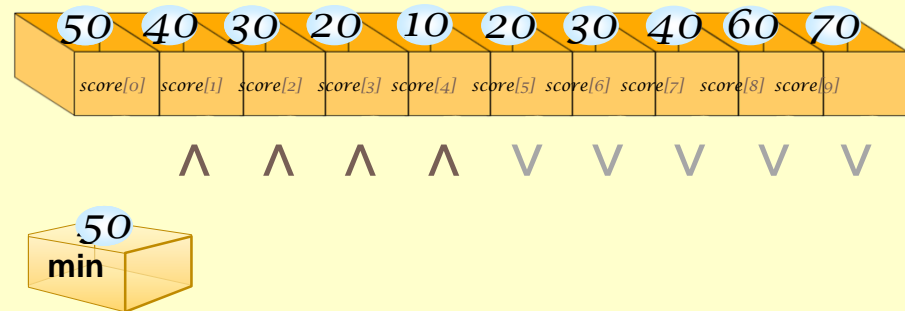
```
printf("최소값은 %d입니다.\n", minimum);
```

```
return 0;
```

```
}
```

첫 번째 배열 원소를 최소
값으로 가정

현재의 최소값보다
배열 원소가
작으면, 배열 원
소를 최소값으로
복사한다.





도전문제

- 위의 프로그램에서는 최소값을 계산하였다. 이번에는 배열의 원소 중에서 최대값을 찾도록 변경하여 보자. 변수 이름도 적절하게 변경하라.





Lab: 영상 처리

- 디지털 영상은 픽셀들의 2차원 배열이라 할 수 있다.

```
int image[8][16] = {  
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1 },  
    { 1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1 },  
    { 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1 },  
    { 1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1 },  
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 } };
```







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

```
void display(int image[8][16])
{
    for (int r = 0; r < 8; r++) {
        for (int c = 0; c < 16; c++) {
            if (image[r][c] == 0)
                printf("*");
            else
                printf("_");
        }
        printf("\n");
    }
}

void inverse(int img[8][16])
{
    for (int r = 0; r < 8; r++) {
        for (int c = 0; c < 16; c++) {
            if (img[r][c] == 0)
                img[r][c] = 1;
            else
                img[r][c] = 0;
        }
    }
}
```

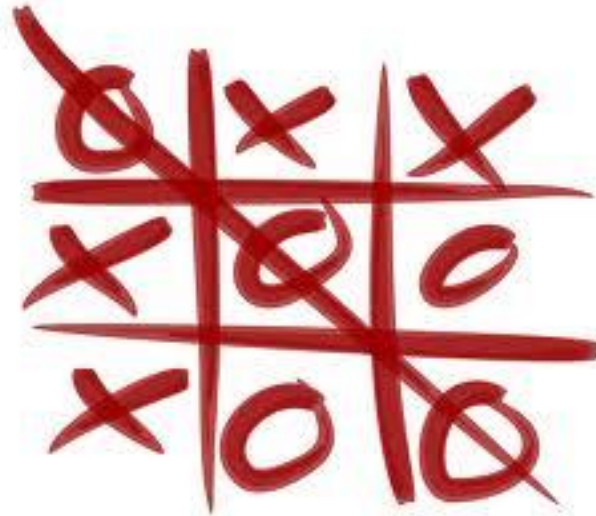



```
int main(void)
{
    int image[8][16] = {
        { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1 },
        { 1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1 },
        { 1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1 },
        { 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1 },
        { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 } };
    printf("변환전 이미지\n");
    display(image);
    inverse(image);
    printf("\n\n변환후 이미지\n");
    display(image);
    return 0;
}
```



Mini Project: tic-tac-toe

- tic-tac-toe 게임은 2명의 경기자가 오른쪽과 같은 보드를 이용하여서 번갈아가며 O와 X를 놓는 게임이다.
- 같은 글자가 가로, 세로, 혹은 대각선 상에 놓이면 이기게 된다.





실행 결과

```
C:\WINDOWS\system32\cmd.exe
(x, y) 좌표: 0 0
X
|
|
|
|
|
(x, y) 좌표: 1 1
X
|
| 0
|
|
(x, y) 좌표:
```



알고리즘

```
1. 보드를 초기화한다.
2. while(1)
3.     보드를 화면에 출력한다.
4.     사용자로부터 좌표 x, y를 받는다.
5.     if (board[x][y]가 비어 있으면)
6.         if( 현재 경기자가 'X'이면 )
7.             board[x][y] = 'X'
8.         else
9.             board[x][y] = 'O'
10.    else
11.        오류 메시지를 출력한다
```



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

int main(void)
{
    char board[3][3];
    int x, y, k, i;

    // 보드를 초기화한다.
    for (x = 0; x < 3; x++)
        for (y = 0; y < 3; y++) board[x][y] = ' ';

    // 사용자로부터 위치를 받아서 보드에 표시한다.
    for (k = 0; k < 9; k++) {
        printf("(x, y) 좌표: ");
        scanf(" %d %d", &x, &y);
        board[x][y] = (k % 2 == 0) ? 'X' : 'O'; // 현재의 순번에 따라 'X', 'O'중 선택
    }
}
```



```
// 보드를 화면에 그린다.  
    for (i = 0; i < 3; i++) {  
        printf("--- | --- | ---\n");  
        printf("%c | %c | %c \n", board[i][0], board[i][1], board[i][2]);  
    }  
    printf("--- | --- | ---\n");  
}  
return 0;  
}
```



도전문제

- (1) 위의 코드를 실행하면 상대방이 놓은 곳에 다시 놓을 수 있다. 이것을 방지하는 코드를 추가하라.
- (2) 보드를 분석하여서 게임이 종료되었는지를 검사하는 함수를 추가하라.
- (3) 컴퓨터가 자동으로 다음 수를 결정하도록 프로그램을 변경하라. 가장 간단한 알고리즘을 사용한다. 예를 들면 비어 있는 첫 번째 좌표에 놓는다.





Q & A

