



실습: 극장 예약 시스템

- 배열을 이용하여 간단한 극장 예약 시스템을 작성
- 좌석은 **10개**
- 먼저 좌석 배치표를 보여준다.
- 예약이 끝난 좌석은 **1**로, 예약이 안 된 좌석은 **0**으로 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

- *while(1)*
- 사용자로부터 예약 여부(*y* 또는 *n*)를 입력받는다.
- *if* 입력 == '*y*'
- 현재의 좌석 배치표 *seats[]*를 출력한다.
- 좌석 번호 *i*를 사용자로부터 입력받는다.
- *if* 좌석번호가 올바르면
- *seats[i]=1*
- *else*
- 에러 메시지를 출력한다.
- *else*
- 종료한다.



실습: 극장 좌석 예약

```
#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);
```

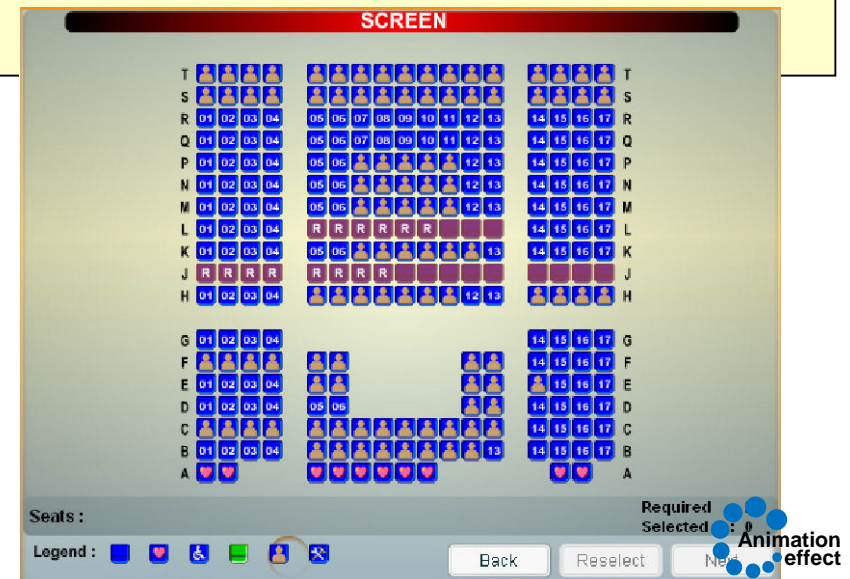




실습: 극장 좌석 예약

```
if(ans1 == 'y')
{
    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);
```

현재 좌석 예약
상태 출력





실습: 극장 좌석 예약

```
if(ans2 <= 0 || ans2 > SIZE) {  
    printf("1부터 10사이의 숫자를 입력하세요\n");  
    continue;  
}  
if(seats[ans2-1] == 0) { // 예약되지 않았으면  
    seats[ans2-1] = 1;  
    printf("예약되었습니다.\n");  
}  
else // 이미 예약되었으면  
    printf("이미 예약된 자리입니다.\n");  
}  
else if(ans1 == 'n')  
    return 0;  
}  
return 0;  
}
```

예약 성공



도전문제

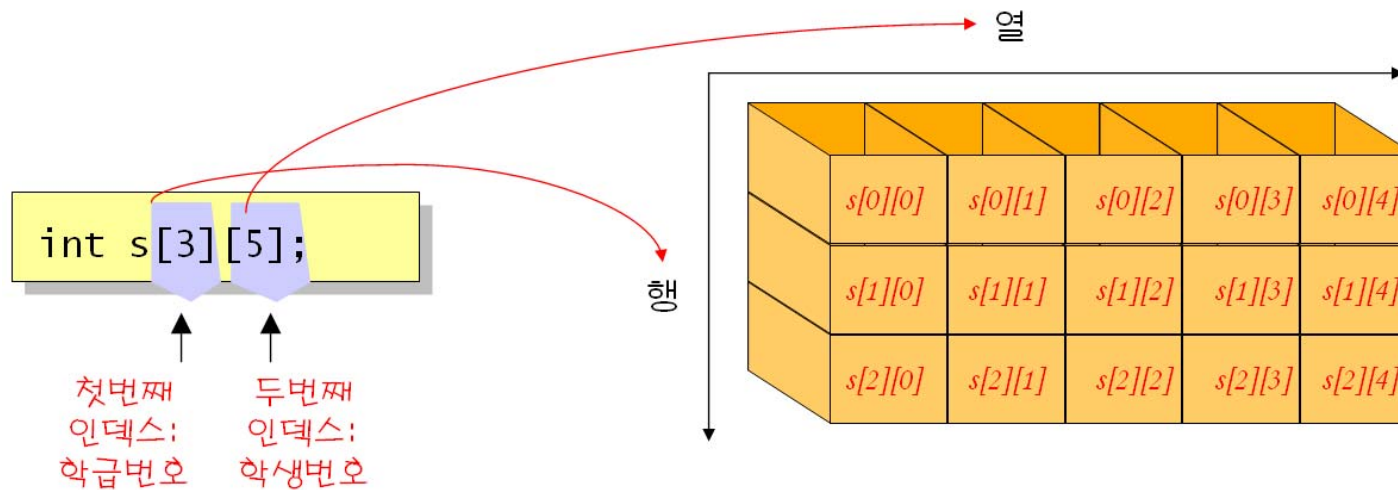
- 위의 프로그램에서는 한명만 예약할 수 있다. 하지만 극장에 혼자서 가는 경우는 드물다. 따라서 한번에 **2명**을 예약할 수 있도록 위의 프로그램을 변경하여 보자.





2차원 배열

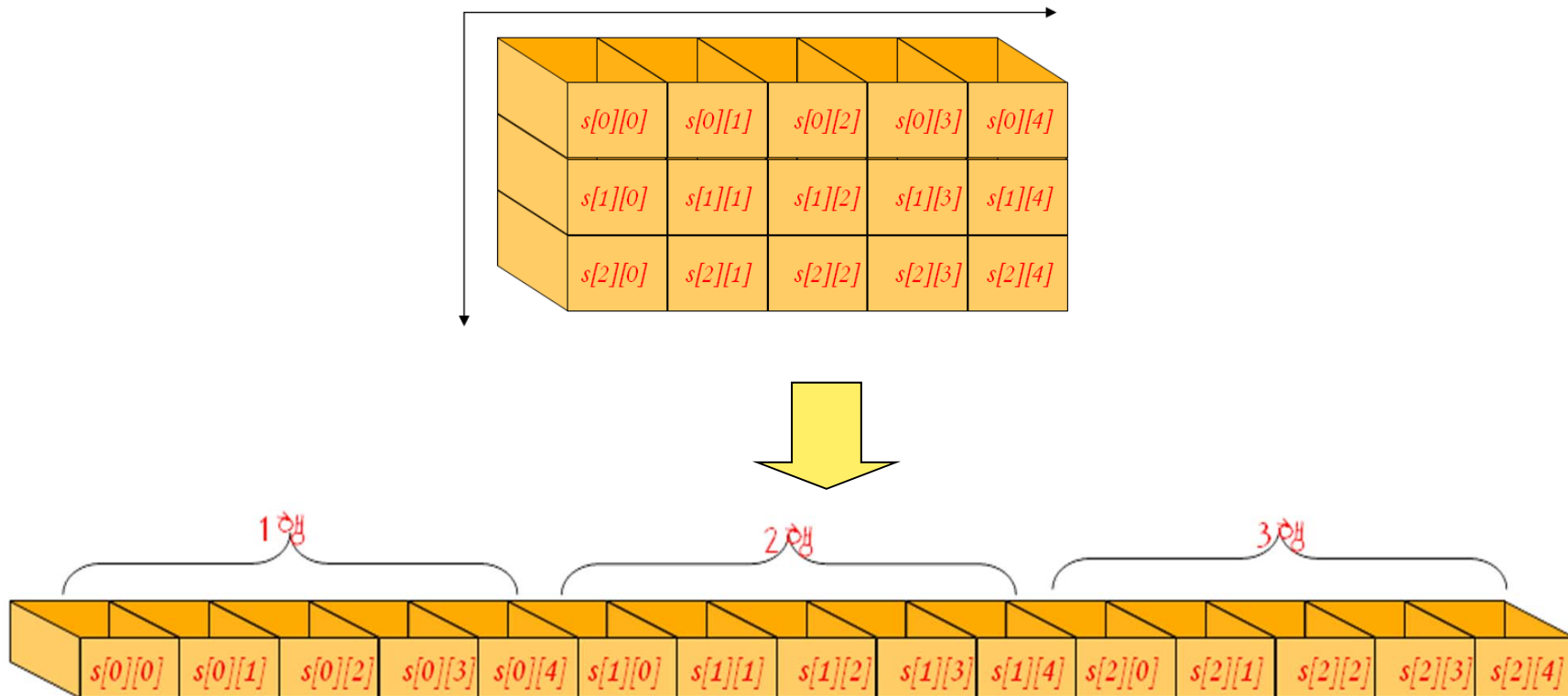
```
int s[10];    // 1차원 배열  
int s[3][10]; // 2차원 배열  
int s[5][3][10]; // 3차원 배열
```





2차원 배열의 구현

- 2차원 배열은 1차원적으로 구현된다.





2차원 배열의 활용

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

```
int main(void)
```

```
{
```

```
    int s[3][5];    // 2차원 배열 선언  
    int i, j;       // 2개의 인덱스 변수  
    int value = 0;  // 배열 원소에 저장되는 값
```

```
    for(i=0;i<3;i++)  
        for(j=0;j<5;j++)  
            s[i][j] = value++;
```

```
    for(i=0;i<3;i++)  
        for(j=0;j<5;j++)  
            printf("%d\n", s[i][j]);
```

```
    return 0;
```

```
}
```

$s[0][0]$	$s[0][1]$	$s[0][2]$	$s[0][3]$	$s[0][4]$
$s[1][0]$	$s[1][1]$	$s[1][2]$	$s[1][3]$	$s[1][4]$
$s[2][0]$	$s[2][1]$	$s[2][2]$	$s[2][3]$	$s[2][4]$





2차원 배열의 초기화

```
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

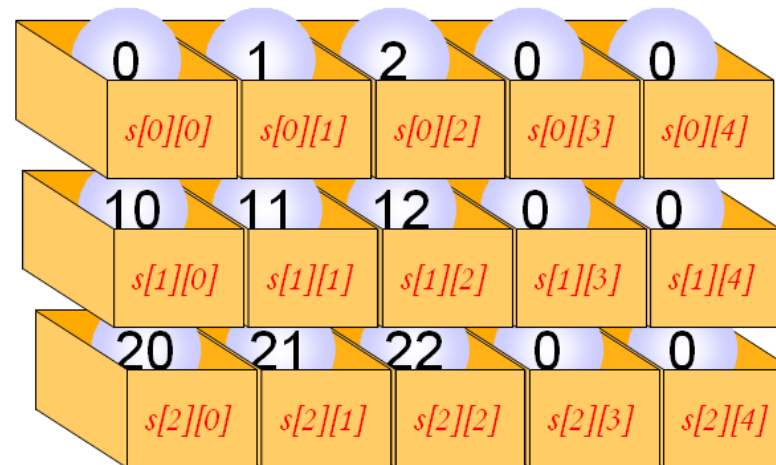
```
int s[ ][5] = {  
    { 0, 1, 2, 3, 4}, // 첫 번째 행의 원소들의 초기값  
    { 10, 11, 12, 13, 14}, // 두 번째 행의 원소들의 초기값  
    { 20, 21, 22, 23, 24}, // 세 번째 행의 원소들의 초기값  
};
```





2차원 배열의 초기화

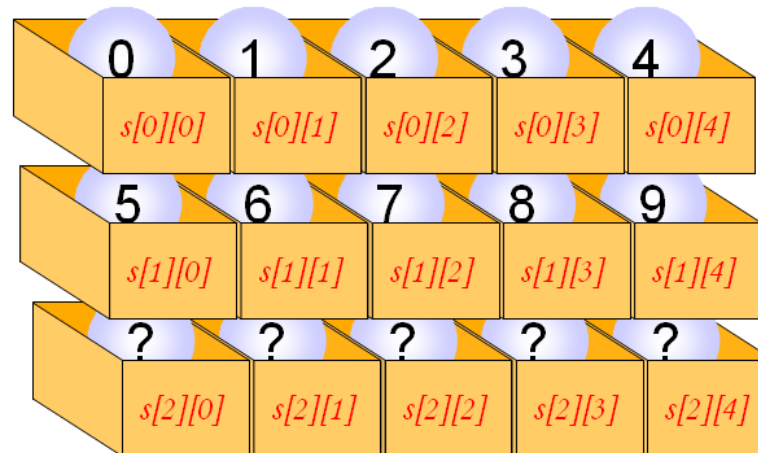
```
int s[ ][5] = {  
    { 0, 1, 2 },      // 첫 번째 행의 원소들의 초기값  
    { 10, 11, 12 },   // 두 번째 행의 원소들의 초기값  
    { 20, 21, 22 }    // 세 번째 행의 원소들의 초기값  
};
```





2차원 배열의 초기화

```
int s[ ][5] = {  
    0, 1, 2, 3, 4,      // 첫 번째 행의 원소들의 초기값  
    5, 6, 7, 8, 9,      // 두 번째 행의 원소들의 초기값  
};
```





3차원 배열

```
int s [6][3][5];
```

첫번째 두번째 세번째
인덱스: 인덱스: 인덱스:
학년번호 학급번호 학생번호

```
#include <stdio.h>
int main(void)
{
    int s[3][3][3];    // 3차원 배열 선언
    int x, y, z;       // 3개의 인덱스 변수
    int i = 1;         // 배열 원소에 저장되는 값

    for(z=0; z<3; z++)
        for(y=0; y<3; y++)
            for(x=0; x<3; x++)
                s[z][y][x] = i++;

    return 0;
}
```



다차원 배열 인수

```
#include <stdio.h>
#define YEARS    3
#define PRODUCTS 5

int sum(int grade[][PRODUCTS]);

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

    printf("총매출은 %d입니다.\n", total_sale);
    return 0;
}
```




다차원 배열 인수

```
int sum(int grade[][PRODUCTS])  
{  
    int y, p;  
    int total = 0;  
    for(y = 0; y < YEARS; y++)  
        for(p = 0; p < PRODUCTS; p++)  
            total += grade[y][p];  
    return total;  
}
```

첫번째 인덱스의 크기는
적지 않아도 된다.



총매출은 45입니다.



다차원 배열 예제

```
#include <stdio.h>
#define CLASSES 3
#define STUDENTS 5

int main(void)
{
    int s[CLASSES][STUDENTS] = {
        { 0, 1, 2, 3, 4 },    // 첫번째 행의 원소들의 초기값
        { 10, 11, 12, 13, 14 }, // 두번째 행의 원소들의 초기값
        { 20, 21, 22, 23, 24 }, // 세번째 행의 원소들의 초기값
    };
}
```



다차원 배열 예제

```
int clas, student, total, subtotal;
total = 0;
for(clas = 0; clas < CLASSES; clas++)
{
    subtotal = 0;
    for(student = 0; student < STUDENTS; student++)
        subtotal += s[clas][student];
    printf("학급 %d의 평균 성적= %d\n", clas, subtotal / STUDENTS);
    total += subtotal;
}
printf("전체 학생들의 평균 성적= %d\n", total/(CLASSES * STUDENTS));
return 0;
}
```



학급 0의 평균 성적 = 2
학급 1의 평균 성적 = 12
학급 2의 평균 성적 = 22
전체 학생들의 평균 성적 = 12



행렬

- 행렬(matrix)는 자연과학에서 많은 문제를 해결하는데 사용

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 8 & 9 & 1 \\ 7 & 0 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 7 & 0 & 0 \\ 9 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix}$$

Mathematics - ELEMENTARY MATRIX OPERATIONS

OPERATION: MULTIPLY EACH ELEMENT IN 2ND Row by 7:

$A = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$

1) FIND $E = 2 \times 2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix}$
 $I \quad E$

2) PREMULT $\begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 1(0)+0(3) & 1(1)+0(4) & 1(2)+0(5) \\ 0(0)+7(3) & 0(1)+7(4) & 0(2)+7(5) \end{bmatrix}$



다차원 배열을 이용한 행렬의 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];
```



다차원 배열을 이용한 행렬의 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 행렬의 B의 각 원소들을 서로 더하여 행렬 C에 대입한다.





중간 점검

- 다차원 배열 `int a[3][2][10]`에는 몇개의 원소가 존재하는가?
- 다차원 배열 `int a[3][2][10]`의 모든 요소를 0으로 초기화하는 문장을 작성하시오.





실습: tic-tac-toe

- **tic-tac-toe** 게임은 2명의 경기자가 오른쪽과 같은 보드를 이용하여 번갈아가며 **O**와 **X**를 놓는 게임이다.
- 같은 글자가 가로, 세로, 혹은 대각선 상에 놓이면 이기게 된다.





실행 결과



(x, y) 좌표 (종료 -1, -1): 0 0

---	---	---
X		
---	---	---
---	---	---

(x, y) 좌표 (종료 -1, -1): 1 1

---	---	---
X		
---	---	---
	0	
---	---	---

...



알고리즘

- 보드를 초기화한다.
- *while(1)*
- 보드를 화면에 출력한다.
- 사용자로부터 좌표 x, y 를 받는다.
- *if (board[x][y]가 비어 있으면)*
- *if(현재 경기자가 'X'이면)*
- $board[x][y] = 'X'$
- *else*
- $board[x][y] = 'O'$
- *else*
- 오류 메시지를 출력한다



소 스

```
#include <stdio.h>
#include <stdlib.h>
void init_board(char board[][3]);
int get_player_move(int palyer, char board[][3]);
void disp_board(char board[][3]);
int main(void)
{
    char board[3][3];
    int quit=0;
    init_board(board);
    do {
        disp_board(board);
        quit = get_player_move(0, board);
        disp_board(board);
        quit = get_player_move(1, board);
    } while(quit == 0);
    return 0;
}
```



소 스

```
void init_board(char board[][3])
{
    int x, y;
    for(x=0; x<3; x++)
        for(y=0; y<3; y++) board[x][y] = ' ';
}

void disp_board(char board[3][3])
{
    int i;
    for(i=0; i<3; i++){
        printf("--- | --- | ---\n");
        printf(" %c | %c | %c \n",board[i][0], board[i][1], board [i][2]);
    }
    printf("--- | --- | ---\n");
}
```



소 스

```
int get_player_move(int player, char board[3][3])
{
    int x, y, done = 0;

    while(done != 1) {
        printf("(x, y) 좌표(종료-1, -1): ");
        scanf("%d %d", &x, &y);
        if( x == -1 && y == -1 ) return 1;
        if(board[x][y] == ' ') break;
        else printf("잘못된 위치입니다.\n");    }

    }

    if( player == 0 )    board[x][y] = 'X';
    else board[x][y] = 'O';

    return 0;
}
```



실행 결과



(x, y) 좌표 (종료 -1, -1): 0 0

---	---	---
x		
---	---	---

---	---	---

(x, y) 좌표 (종료 -1, -1): 1 1

---	---	---
x		
---	---	---
	0	
---	---	---

---	---	---

...



도전문제

- 보드를 분석하여서 게임이 종료되었는지를 검사하는 함수를 추가하라.
- 컴퓨터가 다음 수를 결정하도록 프로그램을 변경하라. 가장 간단한 알고리즘을 사용한다. 예를 들면 비어 있는 첫 번째 좌표에 놓는다.

