

C언어로 배우는 프로그래밍 기초

Perfect C 3판



제 8 장 배열

- 01 배열 선언과 초기화
- 02 2차원과 3차원 배열



학습목표

- ▶ 배열의 개요와 배열 선언 구문에 대하여 이해하고 설명할 수 있다.
 - 여러 자료의 처리에 필요한 자료구조
 - 자료형, 배열이름, 배열크기를 이용한 배열 선언
 - 생성된 배열에서 원하는 원소를 참조
 - 배열 선언 시 동시에 초기값 지정 방법
 - 배열 선언 초기값 설정에서 배열크기 관계
 - 배열에서의 기본값과 쓰레기 값
- ▶ 다차원 배열에 대하여 다음을 이해하고 설명할 수 있다.
 - 2차원 배열의 개념과 선언 방법
 - 2차원 배열의 배열 선언 초기값 설정
 - 3차원 배열의 개념과 배열 선언과 초기값 설정

배열의 필요성과 정의

- 배열(array)

- 여러 변수들이 같은 배열이름으로 일정한 크기의 연속된 메모리에 저장되는 구조
 - 배열을 이용하면 변수를 일일이 선언하는 번거로움이 없어지고
 - 배열을 구성하는 각각의 변수를 반복 구문으로 쉽게 참조 가능

- 배열 정의와 선언 구문

- 자료유형의 저장공간인 원소를 동일한 크기로 지정된 배열크기만큼 확보한 연속된 저장공간
- 배열의 중요 요소
 - 배열이름, 원소 자료유형, 배열크기

배열 선언

원소자료형 배열이름[배열크기];

배열크기는 리터럴 상수, 매크로 상수 또는 이들의 연산식이 허용되나 변수는 사용할 수 없다.

```
#define SIZE 5  
  
int score[10];  
double point[20];  
char ch[80];  
float grade[SIZE];  
int score[SIZE+1];  
int degree[SIZE*2];
```

매크로 상수는 결국 리터럴 상수로 바뀌어 컴파일되므로 문제 없이 선언이 가능하다.

배열원소 접근

• 첨자(index)를 이용

- 배열이름 뒤에 대괄호 사이
 - 첫 번째 배열원소를 참조하는 첨자 값은 0, 다음 두 번째 원소는 1
 - 유효한 첨자의 범위: 0부터 (배열크기-1)까지
- 배열 선언 시 대괄호 안의 수는 배열크기
 - 선언 이후 대괄호 안의 수는 원소를 참조하는 번호 첨자

```
int score[5];  
  
//배열 원소에 값 저장  
score[0] = 78;  
score[1] = 97;  
score[2] = 85;  
//배열 4번째 원소에 값 저장하지 않아 쓰레기값 저장  
score[4] = 91;  
score[5] = 50; //문법오류는 발생하지 않으나 실행오류 발생
```

❌ C4789 버퍼 'score'(크기: 20바이트)이(가) 오버런됩니다. 4바이트가 오프셋 20부터 쓰입니다.

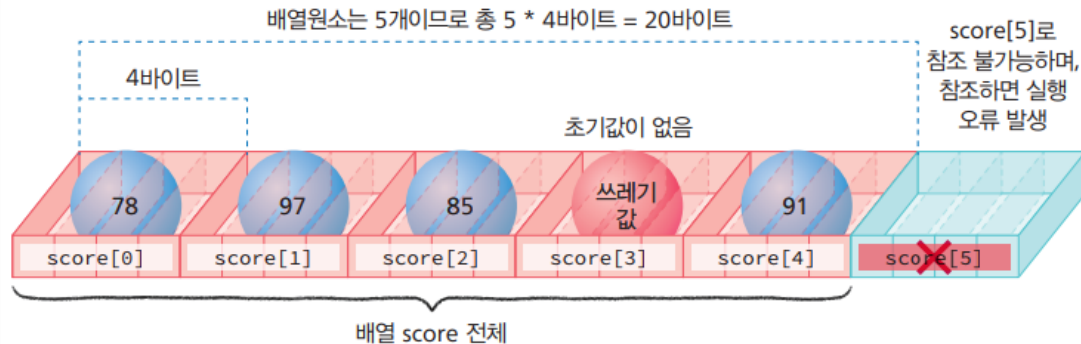


그림 8-3 배열 선언과 원소 참조

배열원소 일괄 출력

- 배열 선언 후 배열원소에 값을 저장하고 순차적으로 출력

실습예제 8-1	Prj01	01deccarray.c	배열 선언 후 배열원소에 값을 저장하고 순차적으로 출력	난이도: ★	
<pre>01 #include <stdio.h> 02 03 #define SIZE 5 04 05 int main(void) 06 { 07 //배열 선언 08 int score[SIZE]; //int score[5]; 09 10 //배열 원소에 값 저장 11 score[0] = 78; //첨자를 사용해 배열원소에 저장 12 score[1] = 97; 13 score[2] = 85; 14 //배열 4번째 원소에 값을 대입하지 않아 쓰레기 값 저장 15 score[4] = 91; 16 //score[5] = 50; //문법오류 발생 17 18 //배열원소 출력 19 for (int i = 0; i < SIZE; i++) 20 printf("%d ", score[i]); 21 printf("\n"); 22 23 return 0; 24 }</pre>					
결과	78	97	85	-858993460	91

SIZE는 리터럴 상수 또는 매크로 상수로 양의 정수여야 함

첨자가 0에서 4를 벗어나면 문법오류가 발생

초기값을 저장하지 않아 쓰레기 값이 출력됨

배열 초기화

배열 선언 초기화

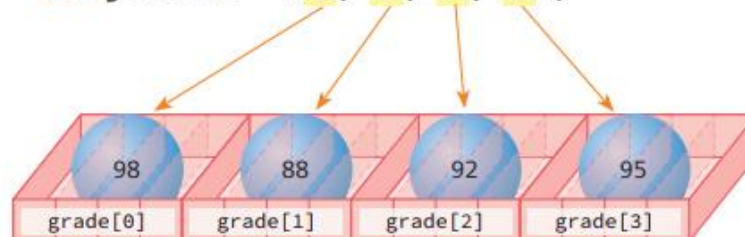
- 배열을 선언하면서 동시에 원소 값을 손쉽게 저장하는 (initialization) 방법
- 중괄호 사이에 여러 원소 값을 쉼표로 구분하여 기술하는 방법
 - 중괄호 사이에는 명시된 배열크기를 넘지 않게 원소 값 나열 가능
- 배열크기는 생략 가능
 - 자동으로 중괄호 사이에 기술된 원소 수가 배열크기
- 원소 값을 나열하기 위해 콤마(,)를 사용하고 전체를 중괄호 {...}로 묶음

원소자료형 배열이름[배열크기] = {원소값1, 원소값2, 원소값3, 원소값4, 원소값5, ... } ;

배열크기는 생략 가능하며, 생략 시 원소값의 수가 배열크기가 된다.

```
int grade[4] = {98, 88, 92, 95};  
double output[] = {78.4, 90.2, 32.3, 44.6, 59.7, 98.9};  
int cpoint[] = {99, 76, 84, 76, 68};
```

```
int grade[4] = {98, 88, 92, 95};
```



배열 기본 값

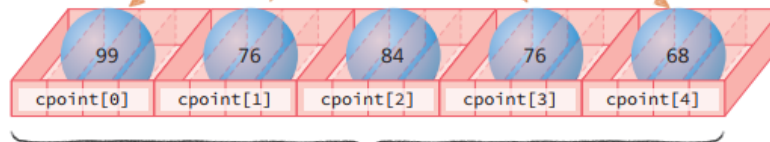
- 배열크기가 초기값 원소 수보다 크면

- 지정하지 않은 원소의 초기값은 자동으로 모두 기본값이 저장

- 기본값이란 자료형에 맞는 0

- 정수형은 0, 실수형은 0.0 그리고 문자 형은 'w0'인 널문자(문자 코드 변화가 0인 문자)

```
int cpoint[] = {99, 76, 84, 76, 68};
```



5: 배열크기를 지정하지 않으면 자동으로 초기값 지정 원소 수가 배열크기가 된다.

```
int dist[5] = {12, 23, 17};
```



배열크기를 지정한 후 초기값 지정 원소 수가 배열크기보다 많으면 다음의 문법오류가 발생한다.

```
int dist[5] = {12, 23, 17, 55, 57, 71};
```

```
int dist[5] = {0};
```

error C2078: 이니셜라이저가 너무 많습니다.

지정한 배열크기보다 초기값 수가 적으면 모두 0으로 채워지므로 모든 배열 원소가 0으로 채워진다.

배열 선언 초기화를 이용한 합과 평균 출력

실습예제 8-2

Prj02

02initarray.c

배열 선언 초기화를 이용한 합과 평균 출력

난이도: ★

```
01  #include <stdio.h>
02  #define SIZE 6
03  int main(void)
04  {
05      //배열 score의 선언과 초기화
06      double score[] = { 89.3, 79.2, 84.83, 76.8, 92.52, 97.4 };
07      double sum = 0;
08
09      //for 문을 이용하여 합을 구함
10      for (int i = 0; i < SIZE; i++)
11      {
12          sum += score[i];
13          printf("score[%d] = %.2f\n", i, score[i]);
14      }
15      printf("성적의 합은 %.2f이고 평균은 %.2f이다.\n", sum, sum/SIZE);
16
17      return 0;
18  }
```

배열 선언과 초기화는 두 문장을 나누어 할 수 없으므로, 다음은 컴파일 오류 발생

`double score[6];`
`score = { 89.3, 79.2, 84.83, 76.8, 92.52, 97.4 };`

SIZE는 배열크기인 매크로 상수로
양의 정수인 6으로 정의

제어문자 i의 첨자는 0에서 5까지 반복

결과

```
score[0] = 89.30
score[1] = 79.20
score[2] = 84.83
score[3] = 76.80
score[4] = 92.52
score[5] = 97.40
성적의 합은 520.050이고 평균은 86.670이다.
```

다양한 배열 선언 초기화 구문

• 바른 문장과 잘못된 초기화 문장

```
#define SIZE 3
```

올바른 초기화 문장

```
int grade[4] = {98, 88, 92, 95};  
double output[SIZE] = {8.4, 0.2, 2.3, 44.6};  
int cpoint[] = {99, 76, 84, 76, 68, 93};  
char ch[] = {'a', 'b', 'c'};  
double width[4] = {23.5, 32.1};
```



TIP 배열 선언 초기화에서 주의점

다음은 오류가 발생하는 배열 선언 초기화 문장이다. 초기화에서도 변수와 const 상수는 배열크기로 사용할 수 없다. 마지막 문장은 초보자가 자주 실수하는 문장으로, 중괄호를 사용한 초기화 방법은 반드시 배열 선언 시에만 이용이 가능하며 배열 선언 이후에는 사용할 수 없으니 주의하자.

표 8-2 배열 선언 초기화 시 오류 발생과 원인

변수와 배열 선언 문장	설명 및 오류 원인
<code>int n = 5; const int size = 6;</code>	변수 n과 const 상수 size 선언
<code>int score[n] = {89, 92, 91}; int cpoint[size] = {3, 5, 7};</code>	변수 n은 배열크기로 사용 불가 상수 변수 size는 배열크기로 사용 불가
<code>int grade[3] = {98, 88, 92, 95};</code>	원소 수 4가 배열크기 3보다 큼
<code>int cpoint[] = {99 76 84 76 68 93};</code>	원소값을 구분하는 콤마(,)가 빠짐
<code>char ch[] = {a, b, c};</code>	원소값인 a, b, c가 문자여야 함
<code>double width[4]; width = {23.5, 32.1};</code>	배열 선언 이후에는 중괄호를 사용한 초기화를 사용할 수 없으며, 배열 선언 시 <code>double width[4] = {23.5, 32.1};</code> 로는 가능



NOTE: C99: 배열의 첨자 초기화(designated initializers)

배열의 초기화 방법이 다음과 같이 첨자를 사용해 부분적으로 초기값을 지정할 수 있다. 배열의 크기가 지정된 배열 a에서 지정한 첨자에 대해서는 초기값이, 그 외의 원소는 0으로 지정된다. 배열의 크기가 지정되지 않은 배열 b에서 지정한 첨자에 대해서는 초기값이, 그 외의 원소는 0으로 지정되며, 가장 큰 첨자가 마지막 원소가 되어 배열의 크기가 결정된다. 일반적인 배열 초기화 방법인 배열 c에서 순서대로 초기값이 저장되며 지정한 첨자에 대해서는 초기값이, 그 외의 원소는 0으로 저장된다.

```
int a[8] = { [1] = 10, [3] = 30, [5] = 50 }; // 0 10 0 30 0 50 0 0 저장  
int b[] = { [1] = 10, [3] = 30, [5] = 50 }; // 0 10 0 30 0 50 저장  
int c[] = { 1, 2, [2] = 10, [5] = 50 }; // 1 2 10 0 0 50 저장
```

Lab 정수형(int) 배열에 표준입력으로 받은 정수를 저장하여 출력

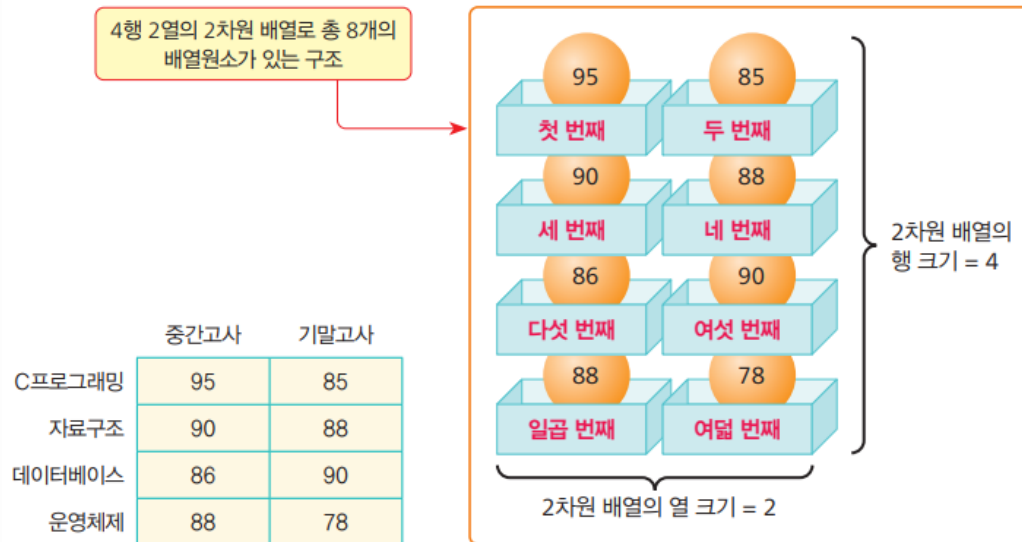
- 자료형 int로 배열 input을 선언

- 표준입력으로 받은 여러 정수를 순서대로 배열 input에 저장하여 출력

Lab 8-1	lab1inputarray.c	난이도: ★
	<pre>01 #define _CRT_SECURE_NO_WARNINGS 02 #include <stdio.h> 03 04 int main(void) 05 { 06 //초기화로 모든 원소에 0을 저장 07 int input[20] = <input type="text"/> ; 08 09 printf("배열에 저장할 정수를 여러 개 입력하십시오."); 10 printf("0을 입력하면 입력을 종료합니다.\n"); 11 int i = 0; 12 do { 13 scanf("%d", <input type="text"/>); 14 } while (input[i++] != 0); 15 16 i = 0; 17 while (input[i] != 0) { 18 printf("%d ", input[i++]); 19 } 20 puts(""); 21 22 return 0; 23 }</pre>	
정답	<pre>07 int input[20] = { 0 }; 13 scanf("%d", &input[i]);</pre>	

2차원 배열 개요

- 2차원 배열은 테이블 형태의 구조
 - 행(row)과 열(column)의 구조로 표현



2차원 배열 선언

원소자료형 배열이름[배열행크기][배열열크기];

배열 선언 시 배열크기는 생략할 수 없으며
배열크기는 리터럴 상수, 매크로 상수
또는 그들의 연산식이 허용된다.

```
#define RSIZE 5
#define CSIZE 2

int score[RSIZE][CSIZE];

double point[2][3];
char ch[5][80];
float grade[7][CSIZE];
```

2차원 배열 구조

- 배열 선언 `int td[2][3];`

- `td[0][0]`으로 첫 번째 원소를 참조
 - 두 번째 원소는 `td[0][1]`,
 - 두 번째 행의 첫 항목인 네 번째 원소는 `td[1][0]`으로 행 첨자가 1 증가
 - 행 첨자는 0에서 (행 크기-1)까지 유효
 - 마찬가지로 열 첨자는 0에서 (열크기 - 1)까지 유효

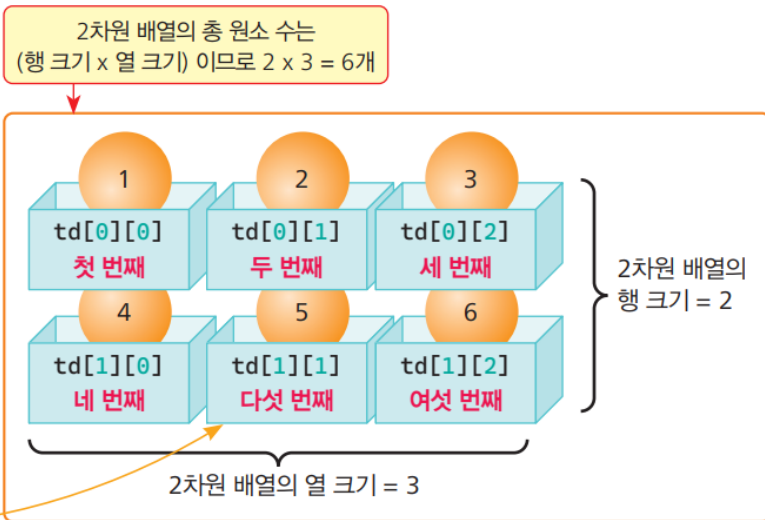
- 행 우선 (row major) 배열

- 행을 먼저 배치하는 특징
 - 첫 번째 행의 모든 원소가 메모리에 할당된 이후
 - 두 번째 행의 원소가 순차적으로 할당

```
#define ROWSIZE 2
#define COLSIZE 3

// 2차원 배열 선언
int td[ROWSIZE][COLSIZE];

// 2차원 배열 원소에 값 저장
td[0][0] = 1; td[0][1] = 2; td[0][2] = 3;
td[1][0] = 4; td[1][1] = 5; td[1][2] = 6;
```



2차원 배열 원소 참조

- 외부 반복 제어변수 i
 - 행을 0에서 (행의 수-1)까지 순차적으로 참조
- 내부 반복 제어변수 j
 - 0에서 (열의 수-1)까지 열을 순차적으로 참조

외부 반복 제어변수 i 는 행을 순차적으로 참조

```
for (i = 0; i < ROWSIZE; i++)  
{  
    for (j = 0; j < COLSIZE; j++)  
        printf("%d ", td[i][j]);  
    puts("");  
}
```

내부 반복 제어변수 j 는 한 행에서 열을 순차적으로 참조

- 배열원소의 저장 값이 행과 열의 관계식이 가능
 - 다음 코드와 같이 중복된 반복문을 사용하여 값을 저장

```
for (i = 0; i < ROWSIZE; i++)  
    for (j = 0; j < COLSIZE; j++)  
        td[i][j] = i*COLSIZE + j + 1;
```

// 2차원 배열원소에 값 저장

```
td[0][0] = 1; td[0][1] = 2; td[0][2] = 3;  
td[1][0] = 4; td[1][1] = 5; td[1][2] = 6;
```

2차원 배열 선언과 원소 하나하나에 직접 초기값 저장 후 출력

실습예제 8-3

Prj03

03tdarray.c

2차원 배열 선언과 원소 하나하나에 직접 초기값 저장 후 출력

난이도: ★

```
01 #include <stdio.h>
02
03 #define ROWSIZE 2
04 #define COLSIZE 3
05
06 int main(void)
07 {
08     // 2차원 배열 선언
09     int td[ROWSIZE][COLSIZE];
10
11     // 2차원 배열원소에 값 저장
12     td[0][0] = 1; td[0][1] = 2; td[0][2] = 3;
13     td[1][0] = 4; td[1][1] = 5; td[1][2] = 6;
14
15     printf("반복문 for를 이용하여 출력\n");
16     for (int i = 0; i < ROWSIZE; i++)
17     {
18         for (int j = 0; j < COLSIZE; j++)
19             printf("td[%d][%d] == %d ", i, j, td[i][j]);
20         printf("\n"); //행마다 한 줄 출력 후 다음 줄로 이동
21     }
22
23     return 0;
24 }
```

for (i = 0; i < ROWSIZE; i++)
for (j = 0; j < COLSIZE; j++)
td[i][j] = i*COLSIZE + j + 1;

위 반복문으로 대체 가능함.

ROWSIZE는 2차원 배열 행 크기인 매크로 상수로 양의 정수인 2로 정의

COLSIZE는 2차원 배열 열 크기인 매크로 상수로 양의 정수인 3으로 정의

결과

반복문 for를 이용하여 출력

```
td[0][0] == 1 td[0][1] == 2 td[0][2] == 3
td[1][0] == 4 td[1][1] == 5 td[1][2] == 6
```

2차원 배열 선언 초기화

- 첫 번째 방법은 중괄호를 중첩되게 이용

- 중괄호 내부에 행에 속하는 값을 다시 중괄호로 묶고, 중괄호와 중괄호 사이에는 쉼표로 분리
- 행인 중괄호 내부의 초기값들은 쉼표로 분리
- 2차원 구조를 행과 열로 표현 할 수 있는 장점

- 다른 방법

- 1차원 배열과 같이 하나의 중괄호로 모든 초기 값을 쉼표로 분리하는 방법

```
int score[2][3] = {{30, 44, 67}, {87, 43, 56}};
```



```
int score[2][3] = {{30, 44, 67}, {87, 43, 56}};
```

```
int score[2][3] = {30, 44, 67, 87, 43, 56};
```

```
int score[][3] = {30, 44, 67, 87, 43, 56};
```

배열원소를 순차적으로
2행 3열의 원소값으로 인지한다.

명시된 열 수인 3을 보고 3개씩 나누어보면 행이 2인 것을 알 수 있다.

2차원 배열 선언에서의 주의점

- 첫 번째 대괄호 내부의 행의 크기는 명시하지 않을 수 있음
 - 그러나 두 번째 대괄호 내부의 열의 크기는 반드시 명시
 - 행 크기는 명시하지 않고 열 크기만 명시한다면 명시된 배열원소 수와 열 크기를 이용하여 행의 크기를 자동으로 산정
- 2차원 배열에서도 초기값이 총 배열원소 수보다 적게 주어진다면
 - 나머지는 모두 기본값인 0, 0.0 또는 'W0'이 저장

```
int a[2][4] = {10, 30, 40, 50, 1, 3, 0, 0};  
int a[2][4] = {10, 30, 40, 50, 1, 3};  
int a[][4] = {10, 30, 40, 50, 1, 3};  
int a[2][4] = { {10, 30, 40, 50}, {1, 3} };  
int a[][4] = { {10, 30, 40, 50}, {1, 3} };
```



TIP 2차원 배열 선언에서의 오류

다음 왼쪽 초기화 문장은 잘못된 문장이며 오른쪽은 이를 수정한 문장이다. 2차원 배열 선언 초기화에서 첫 번째 열 크기는 생략할 수 있어도, 두 번째 행 크기는 절대 생략할 수 없다는 것에 주의하자.

```
int data[2][2] = {1, 2, 3, 4, 5}; //원소 수 초과  
int data[2][2] = {{1, 2} {3, 4}}; //첨표 , 빠짐  
int data[2][] = {1, 2, 3, 4}; //행 크기만 기술  
int data[][] = {1, 2, 3, 4}; //행, 열 크기 모두 없음
```

오류
수정

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

그림 8-19 잘못된 2차원 배열 선언 초기화 문장과 수정

2차원 배열 초기화와 원소 출력

실습예제 8-4

Prj04

04inittdary.c

2차원 배열 초기화와 원소 출력

난이도: ★

```
01 #include <stdio.h>
02
03 #define ROWSIZE 2
04 #define COLSIZE 3
05
06 int main(void)
07 {
08     // 2차원 배열 초기화
09     int td[][3] = { { 1 }, { 1, 2, 3 } };
10
11     printf("반목문 for를 이용하여 출력\n");
12     for (int i = 0; i < ROWSIZE; i++)
13     {
14         for (int j = 0; j < COLSIZE; j++)
15             printf("%d ", td[i][j]);
16         printf("\n");
17     }
18
19     return 0;
20 }
```

배열 td를 3열의 2차원 배열로 선언하면서 초기값을 저장, 초기값 { { 1 }, { 1, 2, 3 } }을 통하여 2행임을 알 수 있으며, 1행의 각 원소를 1, 0, 0으로 저장, 2행의 각 원소를 1, 2, 3으로 저장

한 행을 모두 출력한 이후에 다음 줄로 이동하기 위한 출력으로 들여쓰기에 주의

결과

반목문 for를 이용하여 출력

1 0 0

1 2 3

2차원 배열 초기화를 이용한 성적 처리

성적을 2차원 배열에 저장

- 중간고사와 기말고사별로 평균을 출력
- 변수 midsum
 - 중간고사의 합을 저장한 변수
- 변수 finalsum
 - 기말고사의 합을 저장한 변수
- 변수 sum
 - 전체 성적의 합
- 평균 구하기
 - 소수로 출력되도록
 - (double)로 형변환

실습예제 8-5

Prj05

05tdscore.c

2차원 배열 초기화를 이용한 성적 처리

```
01 #include <stdio.h>
02 #define ROWSIZE 4
03 #define COLSIZE 2
04
05 int main(void)
06 {
07     int sum = 0, midsum = 0, finalsum = 0;
```

```
08
09 // 2차원 배열 초기화
10 int score[][COLSIZE] = { 95, 85, 90, 88, 86, 90, 88, 78 };
11
12 printf("      중간      기말\n");
13 printf("      -----\n");
14 for (int i = 0; i < ROWSIZE; i++)
15 {
16     for (int j = 0; j < COLSIZE; j++)
17     {
18         printf("%10d ", score[i][j]);
19         sum += score[i][j];
20         if (j == 0)
21             midsum += score[i][j];
22         else
23             finalsum += score[i][j];
24     }
25     puts("");
26 }
27
28 printf("      -----\n");
29 printf("평균: %6.2f %10.2f\n", (double)midsum /
30                               ROWSIZE, (double)finalsum / ROWSIZE);
31 printf("\n성적의 합은 %d이고 ", sum);
32 printf("평균은 %.2f이다.\n", (double)sum / (ROWSIZE * COLSIZE));
33
34 return 0;
35 }
```

제어문자 i를 첨자로 사용하여 0에서 3까지 반복하며, 이 반복은 내부 반복 for와 puts()의 두 문장으로 구성되므로 15행과 26행의 종결호는 반드시 필요

제어문자 j를 첨자로 사용하여 0에서 1까지 반복하며, 이 반복은 18행에서 23행까지의 여러 문장으로 구성되므로 종결호는 반드시 필요

결과

	중간	기말
	95	85
	90	88
	86	90
	88	78

평균: 89.75 85.25

성적의 합은 700이고 평균은 87.50이다.

3차원 배열 구조와 선언

- 1차원, 2차원, 3차원 배열을 위한 선언과 그 구조
- 배열 선언

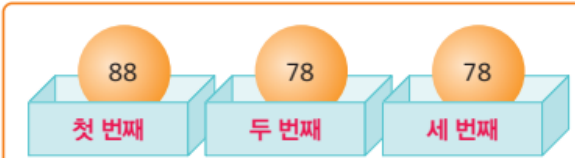
- int threed[2][2][3];
 - 총 $2 \times 2 \times 3 = 12$ 개의 배열원소
 - 대괄호 내부 세 개의 크기는 모두 필요

```
int threed[2][2][3]; //총 2*2*3 = 12개 원소의 3차원 배열
```

```
threed[0][0][0] = 1; // 첫 번째 원소  
threed[0][0][1] = 1; // 두 번째 원소  
threed[0][0][2] = 1; // 세 번째 원소  
threed[0][1][0] = 1; // 네 번째 원소  
... (중간생략)  
threed[1][1][2] = 1; // 열두 번째(마지막) 원소
```

```
int a[3];
```

배열크기 3의 1차원 배열



```
int b[2][3];
```

2행 3열의 2차원 배열



```
int c[2][2][3]
```

2 x 2 x 3의 3차원 배열



3차원 배열 초기화를 이용한 성적 점수 출력

- 3차원 배열 `score[2][4][2]`

- 순서대로 첫 번째 상수 2

- 강좌 수

- 두 번째 상수 4

- 각 반의 학생 수

- 마지막 세 번째 상수 2

- 중간고사와 기말고사인 시험 횟수

실습예제 8-6

Prj06

06thdary.c

3차원 배열 초기화를 이용한 성적 점수 출력

난이도: ★

```
01 #include <stdio.h>
02 #define ROWSIZE 4
03 #define COLSIZE 2
04
05 int main(void)
06 {
07     // 3차원 배열 초기화, 첫 번째 크기는 지정하지 않을 수 있음
08     int score[ROWSIZE][COLSIZE] =
09     {
10         { { 95, 85 },
11           { 85, 83 },
12           { 92, 75 },
13           { 90, 88 } },
14         { { 88, 77 },
15           { 72, 95 },
16           { 88, 92 },
17           { 93, 83 } }
18     };
19
20     for (int i = 0; i < 2; i++)
21     {
22         if (i == 0)
23             printf("[강좌 1]");
24         else
25             printf("[강좌 2]");
26         printf("%11s%7s\n", "중간", "기말");
27
28         for (int j = 0; j < ROWSIZE; j++)
29         {
30             printf("%10s%2d", "학생", j+1);
31             for (int k = 0; k < COLSIZE; k++)
32                 printf("%6d ", score[i][j][k]);
33             printf("\n");
34         }
35         printf("\n");
36     }
37
38     return 0;
39 }
```

3차원 배열 `int score[4][2]`를 선언하면서 초기값 지정. 배열 `score`의 초기화에서 학생 수는 4로 지정하고 2개의 점수를 직접 기술하므로 세 번째 크기는 2가 지정되며, 비어있는 첫 번째 크기는 자동으로 2가 지정됨

제어문자 `k`를 첨자로 사용하여 0에서 1까지 반복하며, 이 반복은 32행의 `printf()` 하나이므로 중괄호는 필요 없으며, 각 학생의 중간고사(`score[i][j][0]`)와 기말고사 성적(`score[i][j][1]`)이 출력

[강좌 1]	중간	기말
학생 1	95	85
학생 2	85	83
학생 3	92	75
학생 4	90	88

[강좌 2]	중간	기말
학생 1	88	77
학생 2	72	95
학생 3	88	92
학생 4	93	83

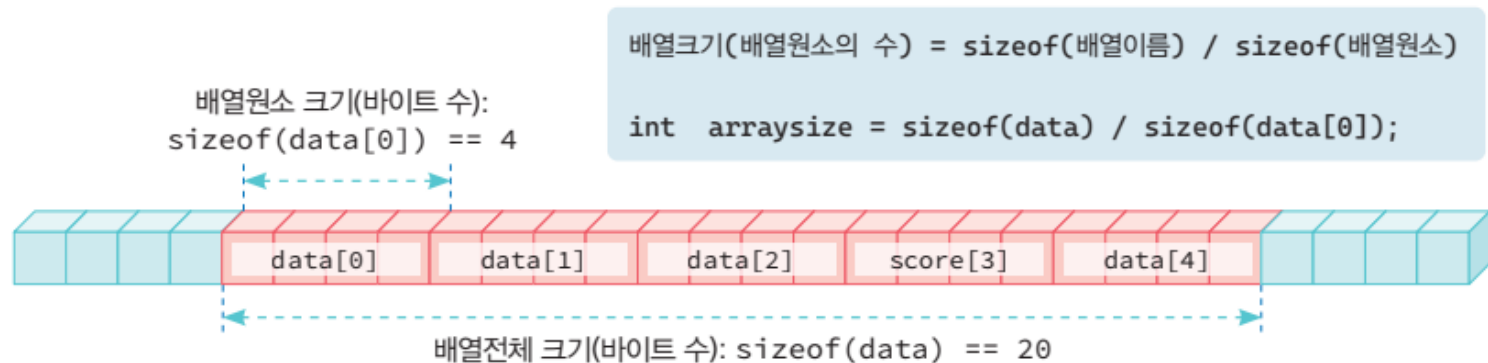
```
int score[2][4][2] = {
    { { 95, 85 },
      { 85, 83 },
      { 92, 75 },
      { 90, 88 } },
    { { 88, 77 },
      { 72, 95 },
      { 88, 92 },
      { 93, 83 } }
};
```

배열크기 연산

• 배열크기 계산방법

- 연산자 sizeof를 이용한 식 (sizeof(배열이름) / sizeof(배열원소))
 - 저장공간의 크기를 바이트 수로 반환하는 연산자 sizeof
- sizeof(배열이름)
 - 배열의 전체 공간의 바이트 수
- sizeof(배열원소)
 - 배열원소 하나의 바이트 수

```
int data[] = {12, 23, 17, 32, 55};
```



2차원 배열 크기 계산방법

- 2차원 배열의 행의 수

- $(\text{sizeof}(x) / \text{sizeof}(x[0]))$
 - **sizeof(배열이름)**
 - 배열의 전체 공간의 바이트 수

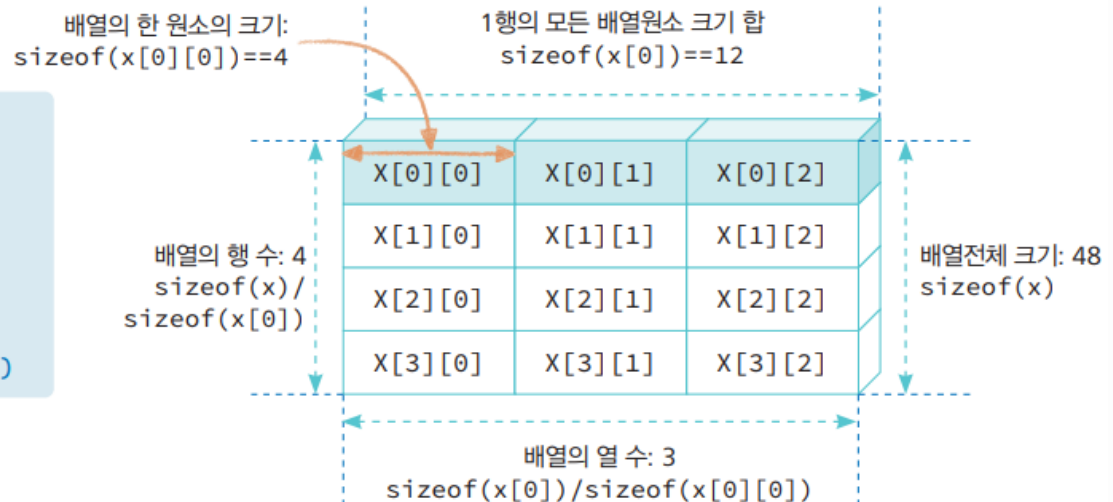
- 2차원 열의 수

- $(\text{sizeof}(x[0]) / \text{sizeof}(x[0][0]))$
 - **sizeof(x[0])**
 - 첫 행의 바이트 수
 - **sizeof(x[0][0])**
 - 첫 번째 원소의 바이트 수

배열의 전체 원소 수: 12
 $\text{sizeof}(x) / \text{sizeof}(x[0][0])$

배열의 행 수: 4
 $\text{sizeof}(x) / \text{sizeof}(x[0])$

배열의 열 수: 3
 $\text{sizeof}(x[0]) / \text{sizeof}(x[0][0])$



1차원과 2차원 배열에서 배열 전체 및 원소의 크기

실습예제 8-7

Prj07

07arysize.c

1차원과 2차원 배열에서 배열 전체 및 원소의 크기

난이도: ★★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int data[] = { 3, 4, 5, 7, 9 };
06
07     printf("%d %d\n", (int) sizeof(data), (int) sizeof(data[0]));
08     printf("배열 data 크기 == %d\n", (int) (sizeof(data) / sizeof(data[0])));
09
10     //4 x 3 행렬
11     double x[][2] = {
12         { 1.2, 2.3 },
13         { 7.3, 8.9 }
14     };
15
16     printf("%d %d ", (int) sizeof(x), (int) sizeof(x[0]));
17
18     printf("%d %d\n", (int) sizeof(x[1]), (int) sizeof(x[0][0]));
19     int rowsize = sizeof(x) / sizeof(x[0]);
20     int colsize = sizeof(x[0]) / sizeof(x[0][0]);
21     printf("2차원 배열 x: 행수 = %d 열수 = %d\n", rowsize, colsize);
22     printf("2차원 배열 x: 전체 원소 수 = %d\n", (int) (sizeof(x) /
23                                                     sizeof(x[0][0])));
24
25     return 0;
26 }
```

연산자 size의 반환값이 size_t 유형이라 (int)로 변환함.
변환하지 않아도 실행은 되나 경고 메시지가 나와 형변환 (int)를 수행함.

1차원 배열의 원소 수인 배열 크기는 연산식
sizeof(data) / sizeof(data[0])으로 계산

//전체 크기와 첫 행의 크기

원소 하나의 크기

2차원 배열의 전체 원소 수

결과

```
20 4
배열 data 크기 == 5
32 16 16 8
2차원 배열 x: 행수 = 2 열수 = 2
2차원 배열 x: 전체 원소 수 = 4
```


LAB 2차원 배열에서 원소 참조 방법과 저장 값을 출력

Lab 8-2

lab2aryprint.c

난이도: ★

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int a[3][4] =
06      {
07          { 1, 2, 7, 3 }, /* initializers for row indexed by 0 */
08          , /* initializers for row indexed by 1 */
09          { 9, 7, 1, 8 } /* initializers for row indexed by 2 */
10      };
11
12      printf("%6s %3s ", "원소", "값");
13      printf("%6s %3s ", "원소", "값");
14      printf("%6s %3s ", "원소", "값");
15      printf("%6s %3s\n", "원소", "값");
16      printf("-----");
17      printf("-----\n");
18
19      for (int i = 0; i < 3; i++)
20      {
21          for (int j = 0; j < 4; j++)
22              printf(" a[%d][%d] %d ", );
23          puts("");
24      }
25
26      return 0;
27  }
```

정답

```
07      { 5, 6, 3, 4 }, /* initializers for row indexed by 1 */
22      printf(" a[%d][%d] %d ", i, j, a[i][j]);
```

감사합니다.