WEEK 06

배열

학습목표

I. 반목문 제어 변경 및 중첩된 반복문 이해
II. 배열과 선언에 대한 기본 개념 이해
III. 2차원 배열 및 3차원 배열과 배열크기 연산 이해

학습목차

Ⅰ. 반목문 제어 변경과 중첩된 반복문

1. break, continue, go to 문
2. 중첩된 반복문

1. break, continue, go to 문

◆ break 문

❖ 반복 내부에서 반복을 종료

실습예제 7-11 | Prj11 | 11break.c | 반복된 정수의 16진수 변환과 break로 종료 | 난이도: ★

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main(void)
05  {
06      int input;
07      while (1)
08      {
09          printf("양의 정수 또는 0[종료] 입력 후 [Enter] >> ");
10          scanf("%d", &input);
11          if (input == 0)
12              break;
13          printf("입력한 정수 %d: 16진수 %#x\n", input, input);
14      }
15      puts("종료");
16
17      return 0;
18  }
```

while (1) 에서 조건식 1이 항상 참이므로 무한반복

결과

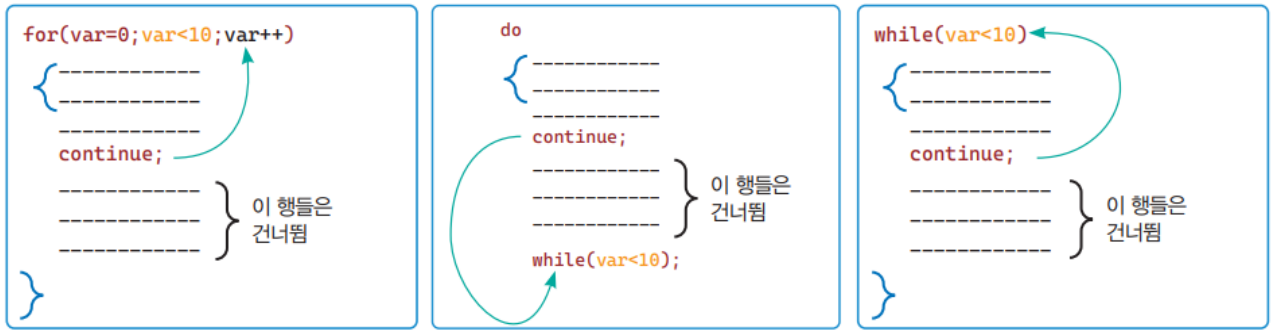
```
양의 정수 또는 0[종료] 입력 후 [Enter] >> 15
입력한 정수 15: 16진수 0xf
양의 정수 또는 0[종료] 입력 후 [Enter] >> 16
입력한 정수 16: 16진수 0x10
양의 정수 또는 0[종료] 입력 후 [Enter] >> 0
종료
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. break, continue, go to 문

◆ 반복의 계속 continue 문

❖ continue는 자신이 속한 가장 근접한 반복에서 다음 반복을 실행



[출처] 강환수 외, Perfect C 3판, 인피니티북스

실습예제 7-12 | Prj12 | 12continue.c | 3으로 나누어지지 않는 정수 출력 | 난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05    const int MAX = 15;
06
07    printf("1에서 %d까지 정수 중에서 3으로 나누어 떨어지지 않는 수\n", MAX);
08    for (int i = 1; i <= MAX; i++)
09    {
10       if (i % 3 == 0) // (!(i % 3))
11          continue;
12       printf("%3d", i);
13    }
14    puts("");
15
16    return 0;
17 }
```

continue를 만나면 실행되지 않고 다음 반복을 위해 i++로 이동

조건식 (i % 3 == 0)은 3으로 나누어 떨어지면 참, 떨어지지 않으면 거짓으로, (!(i % 3))으로도 가능

continue를 만나지 않으면 이 출력문이 실행

결과

```
1에서 15까지 정수 중에서 3으로 나누어 떨어지지 않는 수
  1  2  4  5  7  8 10 11 13 14
```

```
while ( cond1 )
{
   ...
❷ continue;

   for (init; cond2; inc )
   {
      ...
   ❶ continue;

      stmt1;
      stmt2;

   }

}
```

바로 위 ❶ continue를 만나면 실행되지 않는 부분

바로 위 ❷ continue를 만나면 실행되지 않는 부분

1. break, continue, go to 문

◆ go to 문

- ❖ 레이블(label)이 위치한 다음 문장으로 실행 순서를 이동하는 문장
- ❖ 사용하지 않는 것이 바람직

실습예제 7-13 | Prj13 | 13goto.c | goto를 사용해 1에서 10까지의 정수 출력 | 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05     int count = 1;
06
07  loop:
08     printf("%3d", count);
09     if (++count <= 10)
10        goto loop;
11
12     printf("\n종료\n");
13
14     return 0;
15  }
```

문장 goto 문은 무조건 label이 있는 곳으로 이동하여 실행

goto 이후에 이동할 레이블을 기술, 변수 count가 11이 되면 if의 조건식이 거짓이 되어 12번 줄로 이동

결과

```
 1  2  3  4  5  6  7  8  9 10
종료
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. break, continue, go to 문

실습예제 7-14 | Prj14 | 14octhex.c | 반복된 정수의 8진수와 16진수 변환과 break로 종료 | 난이도: ★

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int input;
07     do
08     {
09        printf("양의 정수 또는 음수나 0[종료] 입력 후 [Enter] >> ");
10        scanf("%d", &input);
11        if (input <= 0)
12           break;
13        printf("정수 %d: 8진수 %#o  16진수 %#x\n", input, input, input);
14     } while (1);
15
16     return 0;
17  }
```

break는 while 문을 종료

조건식에 1이나 참을 의미하는 값을 넣으면 무한 반복

결과

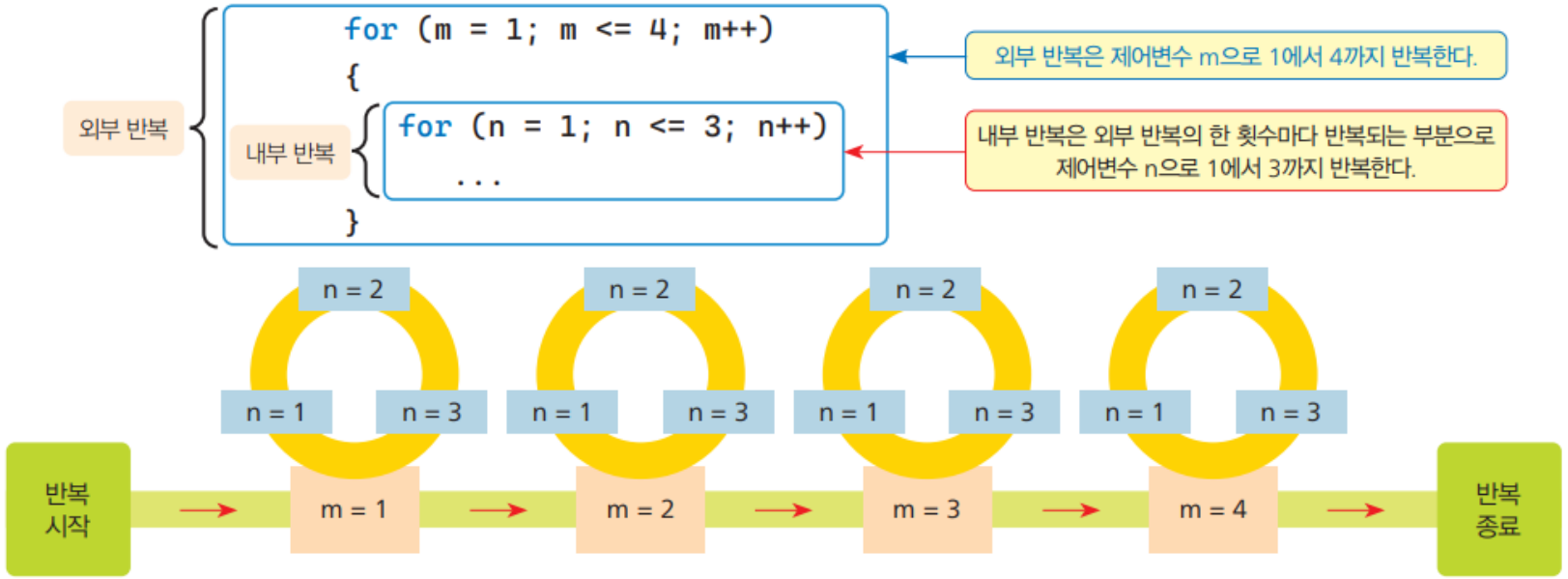
```
양의 정수 또는 음수나 0[종료] 입력 후 [Enter] >> 8
정수 8: 8진수 010  16진수 0x8
양의 정수 또는 음수나 0[종료] 입력 후 [Enter] >> 16
정수 16: 8진수 020  16진수 0x10
양의 정수 또는 음수나 0[종료] 입력 후 [Enter] >> -1
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 중첩된 반복문

❖ 제어 변수는 m, n

- 외부 for 문의 제어 변수는 m이며, 내부 for 문의 제어 변수는 n
- 외부 반복에서 m은 1에서 4까지 반복
- 각각의 m에 대해, 내부 반복에서 n이 1에서 3까지 반복



[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 중첩된 반복문

실습예제 7-15 | Prj15 | 15nestedloop.c | 내부 반복과 외부 반복에서 각각의 변수 값의 변화를 이해 | 난이도: ★

```c
#include <stdio.h>

int main(void)
{
   int m, n;
   for (m = 1; m <= 5; m++)
   {
      printf("m = %-2d\n", m);
      for (n = 1; n <= 7; n++)
         printf("n = %-3d", n);
      puts("");
   }

   return 0;
}
```

외부 반복의 for 문으로 1에서 5까지 반복

내부 반복의 for 문으로 1에서 7까지 반복

결과

```
m = 1
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
m = 2
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
m = 3
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
m = 4
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
m = 5
n = 1  n = 2  n = 3  n = 4  n = 5  n = 6  n = 7
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 중첩된 반복문

◆ 내부 반복이 외부 반복에 의존

실습예제 7-16 | Prj16 | 16triangle.c | 별을 삼각형 모양으로 출력 | 난이도: ★★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05     const int MAX = 5;
06     int i, j;
07
08     for (i = 1; i <= MAX; i++)
09     {
10        for (j = 1; j <= i; j++)
11           printf("*");
12        puts("");
13     }
14
15     return 0;
16  }
```

내부 반복의 for 문으로 1에서 i(외부 반복의 제어 변수)까지 반복, i는 반복에 따라 1에서 5까지 변화

결과

```
*
**
***
****
*****
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 중첩된 반복문

◆ 3중 중첩 반복

실습예제 7-17 | Prj17 | 17loopsum.c | 1에서 입력된 정수까지의 합 구하기 | 난이도: ★★

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

int main(void)
{
   int input, sum, i, j;
   do
   {
      printf("양의 정수 또는 0(종료)을 입력: ");
      scanf("%d", &input);

      for (i = 1; i <= input; i++)
      {
         for (j = 1, sum = 0; j <= i; j++)
         {
            printf("%d", j);
            j == i ? printf(" = ") : printf(" + ");
            sum += j;
         }
         printf("%d\n", sum);
      }
   } while (input > 0);

   puts("종료합니다.");

   return 0;
}
```

연산식 (j == i)는 j가 마지막이면 참, 반복의 중간이면 거짓임. 중간이면 j 값을 한 줄에 출력한 이후에 + 연산자 출력, 마지막이면 = 를 출력

입력 정수가 양수이면 반복하여 계속 실행하고, 0이나 음수이면 종료

결과

```
양의 정수 또는 0(종료)을 입력: 7
1 = 1
1 + 2 = 3
1 + 2 + 3 = 6
1 + 2 + 3 + 4 = 10
1 + 2 + 3 + 4 + 5 = 15
1 + 2 + 3 + 4 + 5 + 6 = 21
1 + 2 + 3 + 4 + 5 + 6 + 7 = 28
양의 정수 또는 0(종료)을 입력: 3
1 = 1
1 + 2 = 3
1 + 2 + 3 = 6
양의 정수 또는 0(종료)을 입력: 0
종료합니다.
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅰ. 반목문 제어 변경과 중첩된 반복문

1교시 수업을 마치겠습니다.

Ⅱ. 배열 선언과 초기화

1. 배열의 개념
2. 배열의 선언
3. 배열의 사용

1. 배열의 개념

❖ 배열

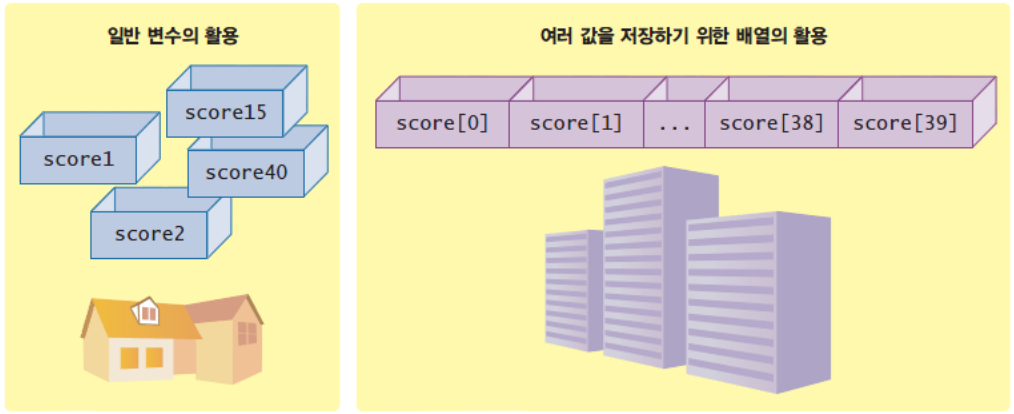
- 여러 변수들이 같은 배열 이름으로 일정한 크기의 연속된 메모리에 저장되는 구조

❖ 필요성

- 배열을 이용하면 변수를 일일이 선언하는 번거로움을 해소할 수 있고, 각 원소를 반복 구문으로 쉽게 참조

❖ 사람이 사는 집을 변수라 생각하면

- 일반 변수는 단독주택에 비유할 수 있고 배열은 아파트



[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 배열의 선언

❖ 구문

- 배열의 크기를 지정하는 부분에는 양수의 정수 상수와 기호 상수 또는 이들의 연산식이 가능
- 그러나 변수로는 배열의 크기를 지정 불가능
- 원소 자료형으로는 모든 자료형이 가능
- 배열 선언 시 초기값 지정이 없다면 반드시 배열 크기는 명시

배열 선언

원소자료형 배열이름(배열크기);

배열 크기는 상수, 기호상수 또는 이들의 연산식이 허용된다 .

```
#define SIZE 5

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

```
Int n = 5;

int score[n];
double point[-3];
char ch[0];
float grade[3.2];
int score[n+2];
int degree[n*2];
```

오류 발생

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 배열의 선언

❖ 배열 원소 접근은 첨자(index) 이용

- 배열 이름 뒤에 대괄호 사이
  - 첫 번째 배열 원소에 접근하는 첨자값은 0, 다음 두 번째 원소는 1
  - 유효한 첨자의 범위는 0부터 (배열 크기-1)까지
- 배열 선언 시 대괄호 안의 수는 배열크기

```
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.

[출처] 강환수 외, Perfect C 3판, 인피니티북스

실습예제 8-1 | Prj01 | 01decarray.c | 배열 선언 후 배열원소에 값을 저장하고 순차적으로 출력 | 난이도: ★

```c
#include <stdio.h>

#define SIZE 5

int main(void)
{
    //배열 선언
    int score[SIZE];//int score[5];

    //배열 원소에 값 저장
    score[0] = 78; //첨자를 사용해 배열원소에 저장
    score[1] = 97;
    score[2] = 85;
    //배열 4번째 원소에 값을 대입하지 않아 쓰레기 값 저장
    score[4] = 91;
    //score[5] = 50; //문법오류 발생

    //배열원소 출력
    for (int i = 0; i < SIZE; i++)
        printf("%d  ", score[i]);
    printf("\n");

    return 0;
}
```

SIZE는 리터럴 상수 또는 매크로 상수로 양의 정수여야 함

첨자가 0에서 4를 벗어나면 문법오류가 발생

첨자가 0에서 SIZE-1까지 순회해야 하므로 이 조건을 이용함

결과

```
78  97  85  -858993460  91
```

초기값을 저장하지 않아 쓰레기 값이 출력됨

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 배열의 선언

◆ 배열 초기화

- ❖ 배열을 선언하면서 동시에 원소 값을 손쉽게 저장하는 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

[출처] 강환수 외, Perfect C 3판, 인피니티북스

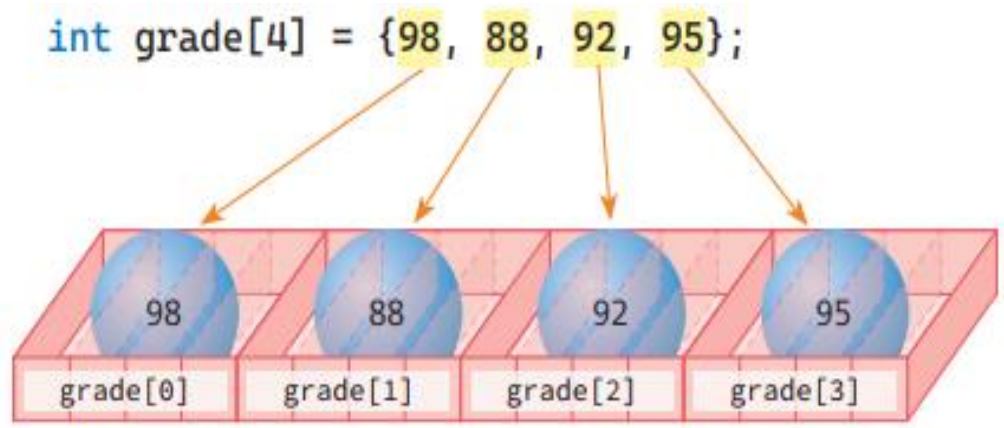
2. 배열의 선언

◆ 배열 초기화

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};
```



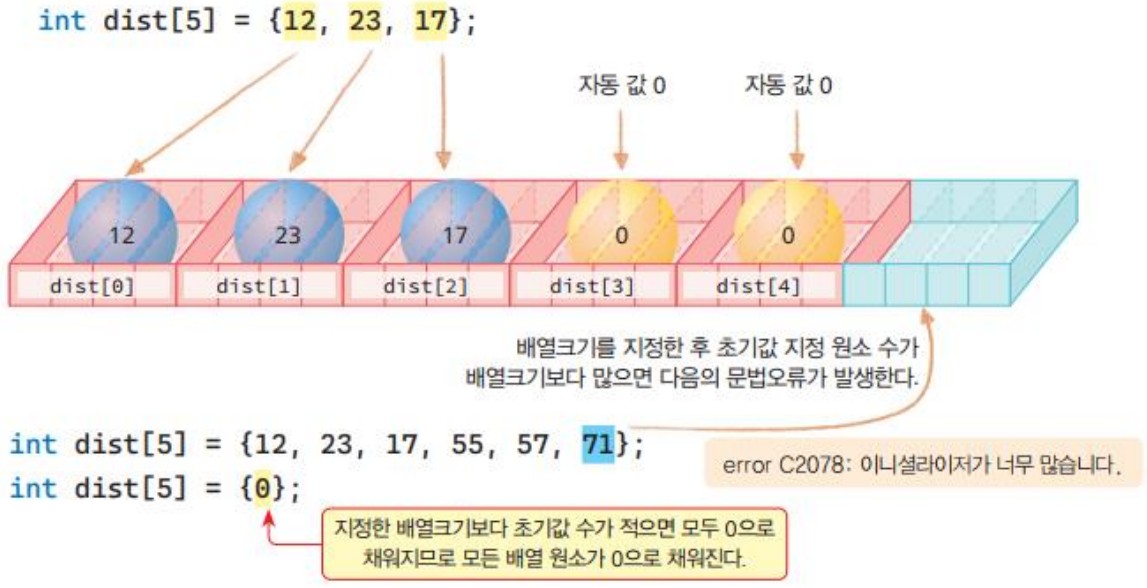
[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 배열의 선언

◆ 배열 초기화

❖ 초기 값이 없는 원소는 기본값으로 저장

- 만일 배열 크기가 초기값 원소의 수보다 크면 지정하지 않은 원소의 초기값은 자동으로 모두 기본값으로 저장
- 여기서 기본값이란 자료형에 맞는 0
  - 즉 정수형은 0, 실수형은 0.0 그리고 문자형은'₩0'인 널문자



[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 배열의 사용

실습예제 8-2 | Prj02 | 02initarray.c | 배열 선언 초기화를 이용한 합과 평균 출력 | 난이도: ★

```c
#include <stdio.h>
#define SIZE 6
int main(void)
{
    //배열 score의 선언과 초기화
    double score[] = { 89.3, 79.2, 84.83, 76.8, 92.52, 97.4 };
    double sum = 0;

    //for 문을 이용하여 합을 구함
    for (int i = 0; i < SIZE; i++)
    {
        sum += score[i];
        printf("score[%d] = %.2f\n", i, score[i]);
    }
    printf("성적의 합은 %.2f이고 평균은 %.2f이다.\n", sum, sum/SIZE);

    return 0;
}
```

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

```c
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 520.05이고 평균은 86.67이다.
```

[출저] 강환수 외, Perfect C 3판, 인피니티북스

3. 배열의 사용

올바른 초기화 문장

```
#define SIZE 3

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

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅱ. 배열 선언과 초기화

2교시 수업을 마치겠습니다.

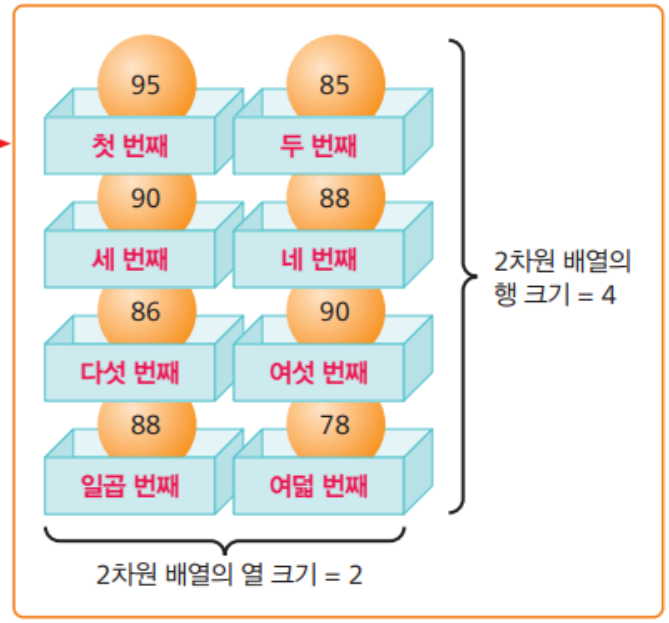
Ⅲ. 2차원 배열 및 3차원 배열과 배열크기 연산

1. 2차원 배열 선언과 사용
2. 2차원 배열 초기화
3. 3차원 배열
4. 배열크기 연산

1. 2차원 배열 선언과 사용

◆ 2차원 배열은 테이블 형태(행렬)의 구조

4행 2열의 2차원 배열로 총 8개의 배열원소가 있는 구조



	중간고사	기말고사
C프로그래밍	95	85
자료구조	90	88
데이터베이스	86	90
운영체제	88	78

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];
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 2차원 배열 선언과 사용

◆ 2차원 배열 선언과 원소 참조

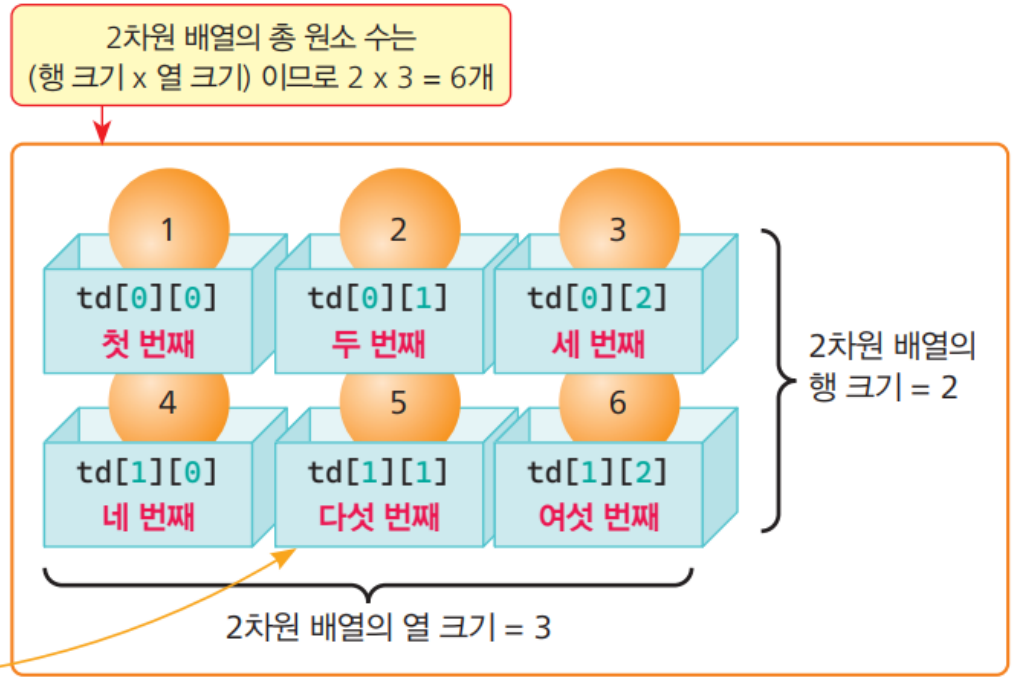
❖ 2개의 첨자 필요

- 행 첨자는 0에서 (행크기-1)까지 유효
- 마찬가지로 열 첨자는 0에서 (열크기-1)까지 유효

```
#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;
```

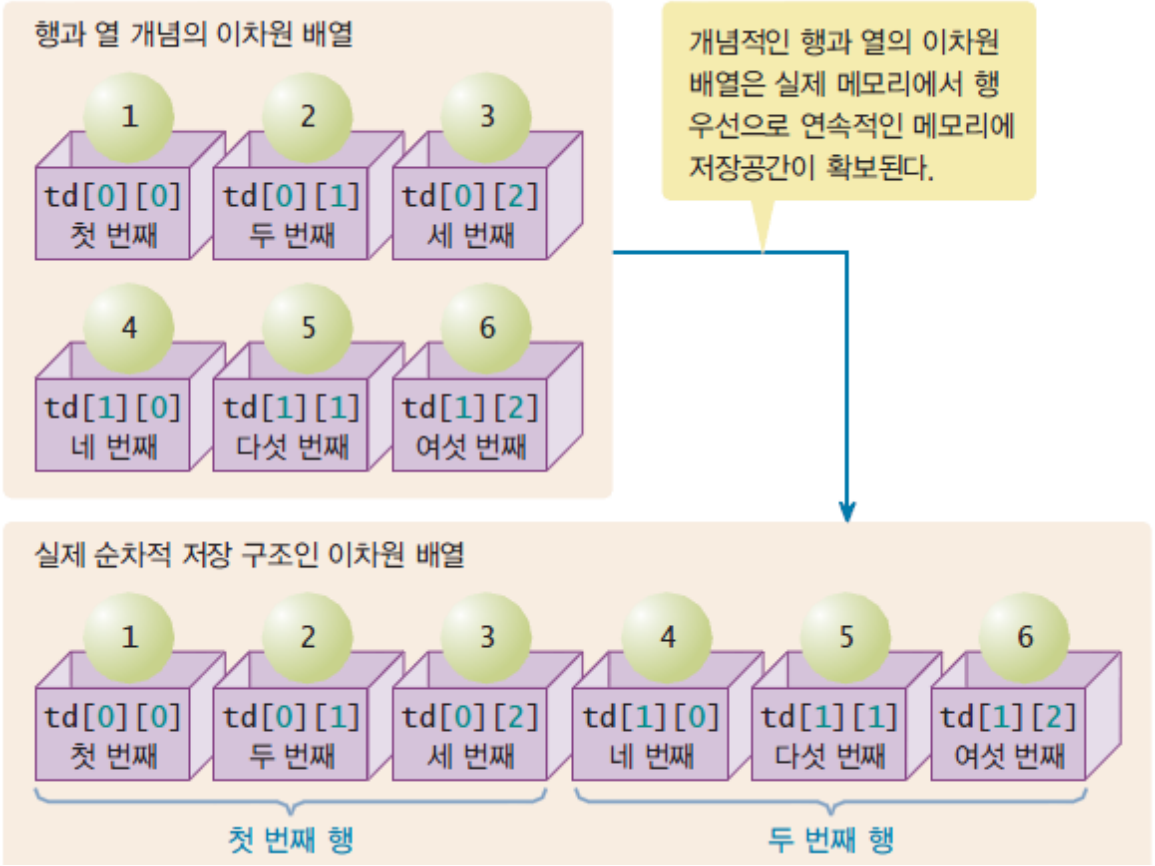


[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 2차원 배열 선언과 사용

◆ C 언어의 행 우선 배열

❖ 첫 번째 행의 모든 원소가 메모리에 할당된 이후에 두 번째 행의 원소가 순차적으로 할당



[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 2차원 배열 선언과 사용

◆ 2차원 배열 원소 참조

외부 반복 제어변수 i는 행을 순차적으로 참조

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

내부 반복 제어변수 j는 한 행에서 열을 순차적으로 참조

◆ 연산식으로 배열원소에 값 대입

```
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;
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 2차원 배열 선언과 사용

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("반목문 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 < ROWZIZE; i++)
   for (j = 0; j < COLZIZE; j++)
      td[i][j] = i*COLZIZE + 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

```
반목문 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
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 2차원 배열 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}};
```

3열

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

2행

배열원소를 순차적으로
2행 3열의 원소값으로 인지한다.

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

❶ ❷

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

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 2차원 배열 초기화

❖ 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 주어지면

- 나머지는 모두 기본값인 0, 0.0 또 는 '₩0'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} };
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 2차원 배열 초기화

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}}; //쉼표 , 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

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 2차원 배열 초기화

실습예제 8-4 | Prj04 | 04inittdary.c | 2차원 배열 초기화와 원소 출력 | 난이도: ★

```c
#include <stdio.h>

#define ROWSIZE 2
#define COLSIZE 3

int main(void)
{
   // 2차원 배열 초기화
   int td[][3] = { { 1 }, { 1, 2, 3 } };

   printf("반목문 for를 이용하여 출력\n");
   for (int i = 0; i < ROWSIZE; i++)
   {
      for (int j = 0; j < COLSIZE; j++)
         printf("%d ", td[i][j]);
      printf("\n");
   }

   return 0;
}
```

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
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 2차원 배열 초기화

실습예제 8-5 | Prj05 | 05tdscore.c | 2차원 배열 초기화를 이용한 성적 처리

```c
#include <stdio.h>
#define ROWSIZE 4
#define COLSIZE 2

int main(void)
{
   int sum = 0, midsum = 0, finalsum = 0;

   // 2차원 배열 초기화
   int score[][COLSIZE] = { 95, 85, 90, 88, 86, 90, 88, 78 };

   printf("       중간       기말\n");
   printf("    ------------------\n");
   for (int i = 0; i < ROWSIZE; i++)
   {
      for (int j = 0; j < COLSIZE; j++)
      {
         printf("%10d ", score[i][j]);
         sum += score[i][j];
         if (j == 0)
            midsum += score[i][j];
         else
            finalsum += score[i][j];
      }
      puts("");
   }

   printf("    ------------------\n");
   printf("평균: %6.2f %10.2f\n", (double)midsum /
                                  ROWSIZE, (double)finalsum / ROWSIZE);
   printf("\n성적의 합은 %d이고 ", sum);
   printf("평균은 %.2f이다.\n", (double)sum / (ROWSIZE * COLSIZE));

   return 0;
}
```

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

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

결과

```
       중간       기말
-------------------------
        95         85
        90         88
        86         90
        88         78
-------------------------
평균:   89.75     85.25

성적의 합은 700이고 평균은 87.50이다.
```

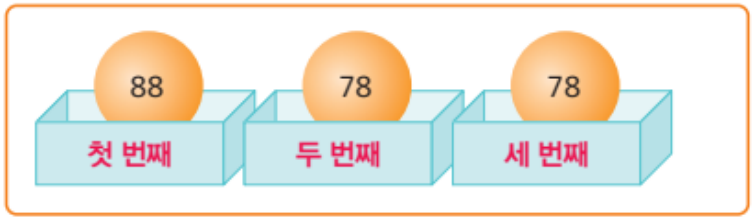
[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 3차원 배열

◆ 배열 선언

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

배열크기 3의 1차원 배열

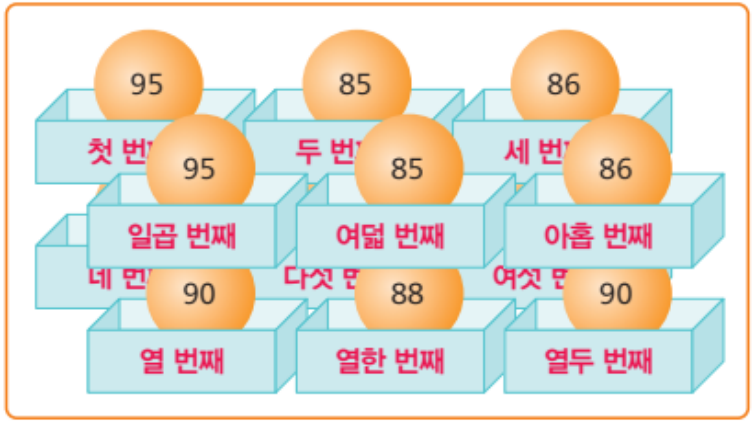



```
int b[2][3];
```

2행 3열의 2차원 배열


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

2 x 2 x 3의 3차원 배열




```
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
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 3차원 배열

❖ 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

[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 } }
};
```

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 score의 초기화에서 학생 수는 4로 지정하고 2개의 점수를 직접 기술하므로 세 번째 크기는 2가 지정되며, 비어있는 첫 번째 크기는 자동으로 2가 지정됨

제어문자 k를 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

[출처] 강환수 외, Perfect C 3판, 인피니티북스

4. 배열크기 연산

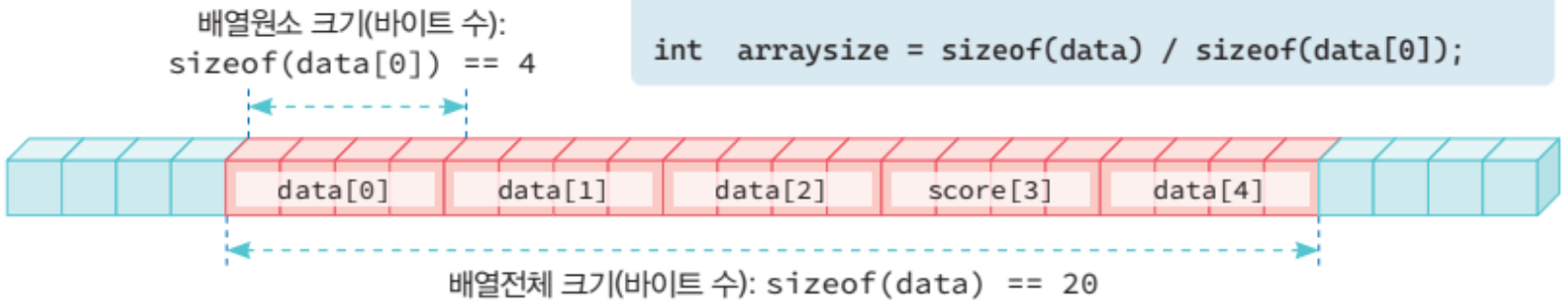
◆ 배열크기 계산 방법

❖ 연산자 sizeof를 이용한 식 (sizeof(배열이름) / sizeof(배열원소))

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

```
배열크기(배열원소의 수) = sizeof(배열이름) / sizeof(배열원소)

int  arraysize = sizeof(data) / sizeof(data[0]);
```



[출처] 강환수 외, Perfect C 3판, 인피니티북스

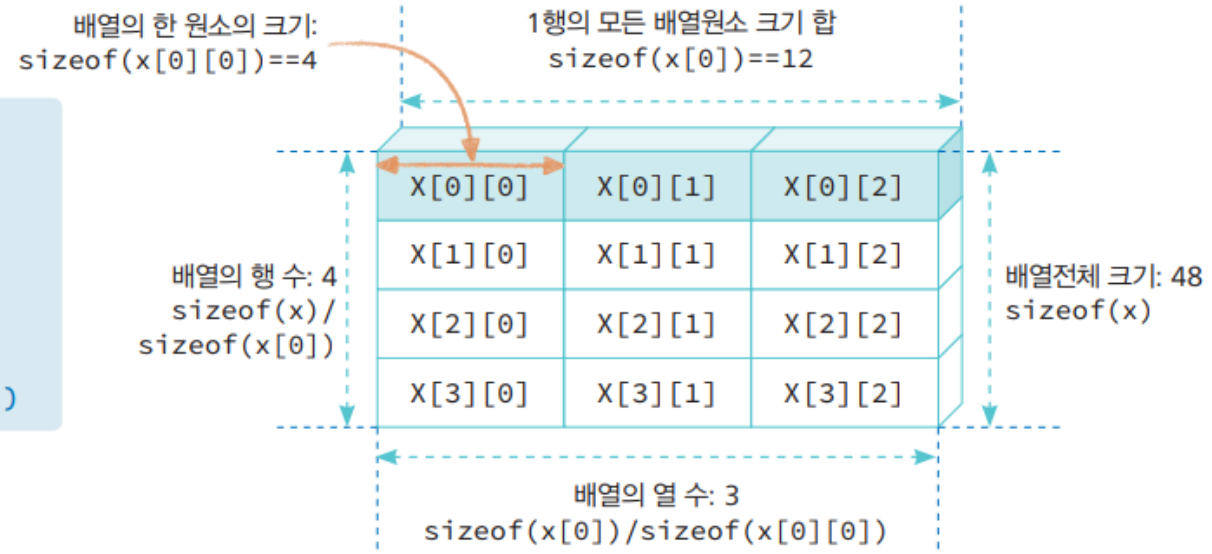
4. 배열크기 연산

◆ 2차원 배열크기 계산 방법

```
배열의 전체 원소 수: 12
sizeof(x) / sizeof(x[0][0])

배열의 행 수: 4
sizeof(x) / sizeof(x[0])

배열의 열 수: 3
sizeof(x[0]) / sizeof(x[0][0])
```



[출처] 강환수 외, Perfect C 3판, 인피니티북스

4. 배열크기 연산

실습예제 8-7 | Prj07 | 07arysize.c | 1차원과 2차원 배열에서 배열 전체 및 원소의 크기 | 난이도: ★★

```c
#include <stdio.h>

int main(void)
{
   int data[] = { 3, 4, 5, 7, 9 };

   printf("%d %d\n", (int) sizeof(data), (int) sizeof(data[0]));
   printf("배열 data 크기 == %d\n", (int) (sizeof(data) / sizeof(data[0])));

   //4 x 3 행렬
   double x[][2] = {
      { 1.2, 2.3},
      { 7.3, 8.9}
   };

   printf("%d %d ", (int) sizeof(x), (int) sizeof(x[0]));
                                                  //전체 크기와 첫 행의 크기
   printf("%d %d\n", (int) sizeof(x[1]), (int) sizeof(x[0][0]));
   int rowsize = sizeof(x) / sizeof(x[0]);
   int colsize = sizeof(x[0]) / sizeof(x[0][0]);
   printf("2차원 배열 x: 행수 = %d  열수 = %d\n", rowsize, colsize);
   printf("2차원 배열 x: 전체 원소 수 = %d\n", (int) (sizeof(x) /
                                                  sizeof(x[0][0])));

   return 0;
}
```

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

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
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅲ. 2차원 배열 및 3차원 배열과 배열크기 연산

3교시 수업을 마치겠습니다.