6강

배열

동양미래대학교 강환수 교수

본 강의 사용 및 참조 자료

- Perfect C, 3판, 강환수 외 2인 공저, 인피니티북스, 2021

8장	배열

목차

01

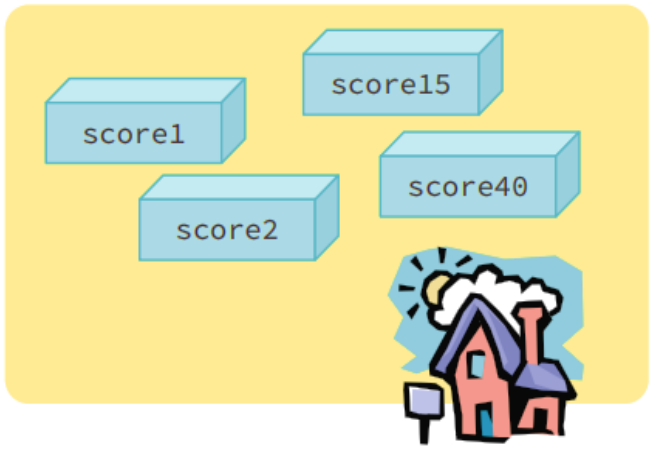
배열 개요와 선언

배열이란

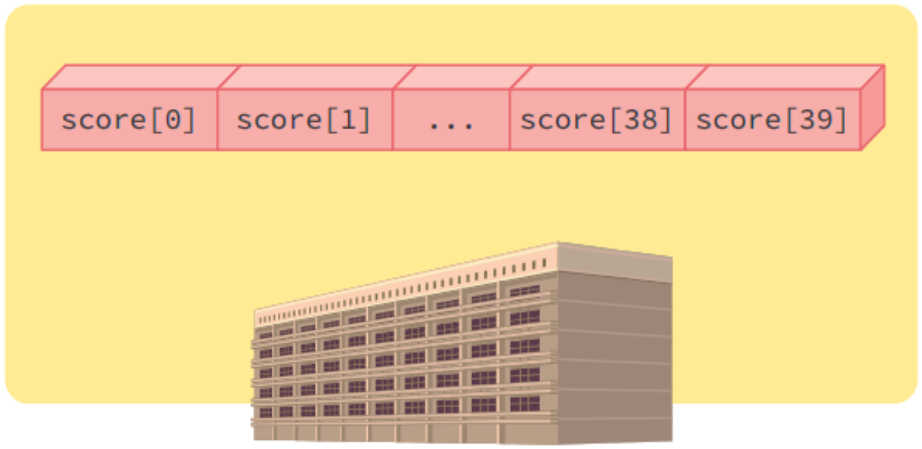
배열(array)

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

일반 변수의 활용



여러 값을 저장하기 위한 배열의 활용



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 342쪽

배열 정의

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

배열 선언 구문

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.

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 343쪽

배열 선언 문법 오류

변수와 const 상수는 배열크기로 사용 불가능

변수와 배열 선언 문장	설명 및 오류 원인
`int n = 5;` `const int size = 6;`	변수 n과 const 상수 size 선언
`int score[n];`	변수 n은 배열크기로 사용 불가

`double point[-3];`
`char ch[0];`
`float grade[3.2];`
	음수는 배열크기로 사용 불가
	0은 배열크기로 사용 불가
	실수는 배열크기로 사용 불가

`int score[n + 2];` `int degree[n * 2];`	변수 n의 연산식은 배열크기로 사용 불가
`int cpoint[size];` `double width[size + 4];`	상수 변수 size의 연산식은 배열크기로 사용 불가

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 343쪽

배열원소 접근 1/2

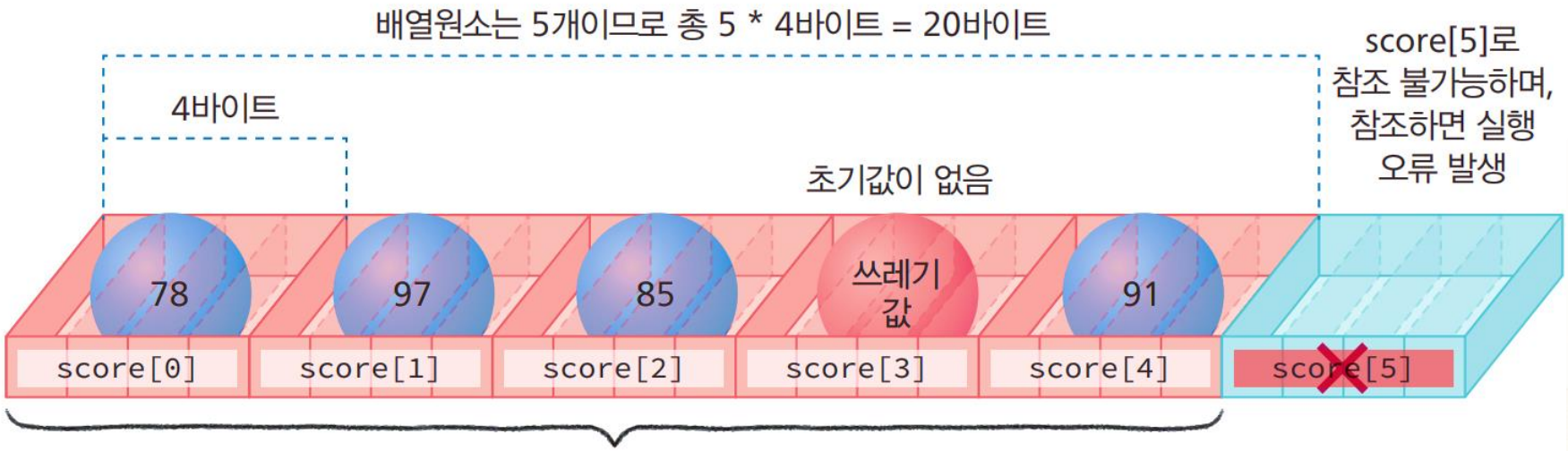
- 첨자(index)를 이용
  - 배열이름 뒤에 대괄호 사이
    - 첫 번째 배열원소를 참조하는 첨자 값은 0
      - 다음 두 번째 원소는 1
    - 유효한 첨자의 범위
      - 0 ~ (배열크기-1)
  - 배열 선언 시 대괄호 안의 수는 배열크기
  - 선언 이후 대괄호 안의 수는 원소를 참조하는 첨자 번호

배열원소 접근 2/2

```
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;
```

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



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 344쪽

실습예제 1/2

Prj01 01decarray.c 배열 선언 후 배열원소에 값을 저장하고 순차적으로 출력 난이도: ★

```
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
```

SIZE는 리터럴 상수 또는 매크로 상수로 양의 정수여야 함

첨자가 0에서 4를 벗어나면 문법오류가 발생

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 345쪽

실습예제 2/2

```
19      for (int i = 0; i < SIZE; i++)
20          printf("%d  ", score[i]);
21      printf("\n");
22
23      return 0;
24  }
```

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

초기값을 저장하지 않아 쓰레기 값이 출력됨

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 345쪽

배열 초기화 1/3

배열선언 초기화

- 배열을 선언하면서 동시에
  원소 값을 손쉽게 저장하는 초기화(initialization) 방법
- 중괄호 사이에 여러 원소 값을 쉼표로 구분하여 기술하는 방법
  - 중괄호 사이에는 명시된 배열크기를 넘지 않게 원소 값 나열 가능

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

배열크기는 생략 가능하며, 생략 시 원소값의 수가 배열크기가 된다.

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 346쪽

배열 초기화 2/3

배열크기는 생략 가능

- 자동으로 중괄호 사이에 기술된 원소 수가 배열크기
- 원소 값을 나열하기 위해 콤마(,)를 사용하고 전체를 중괄호 {…}로 묶음

배열 선언 초기화

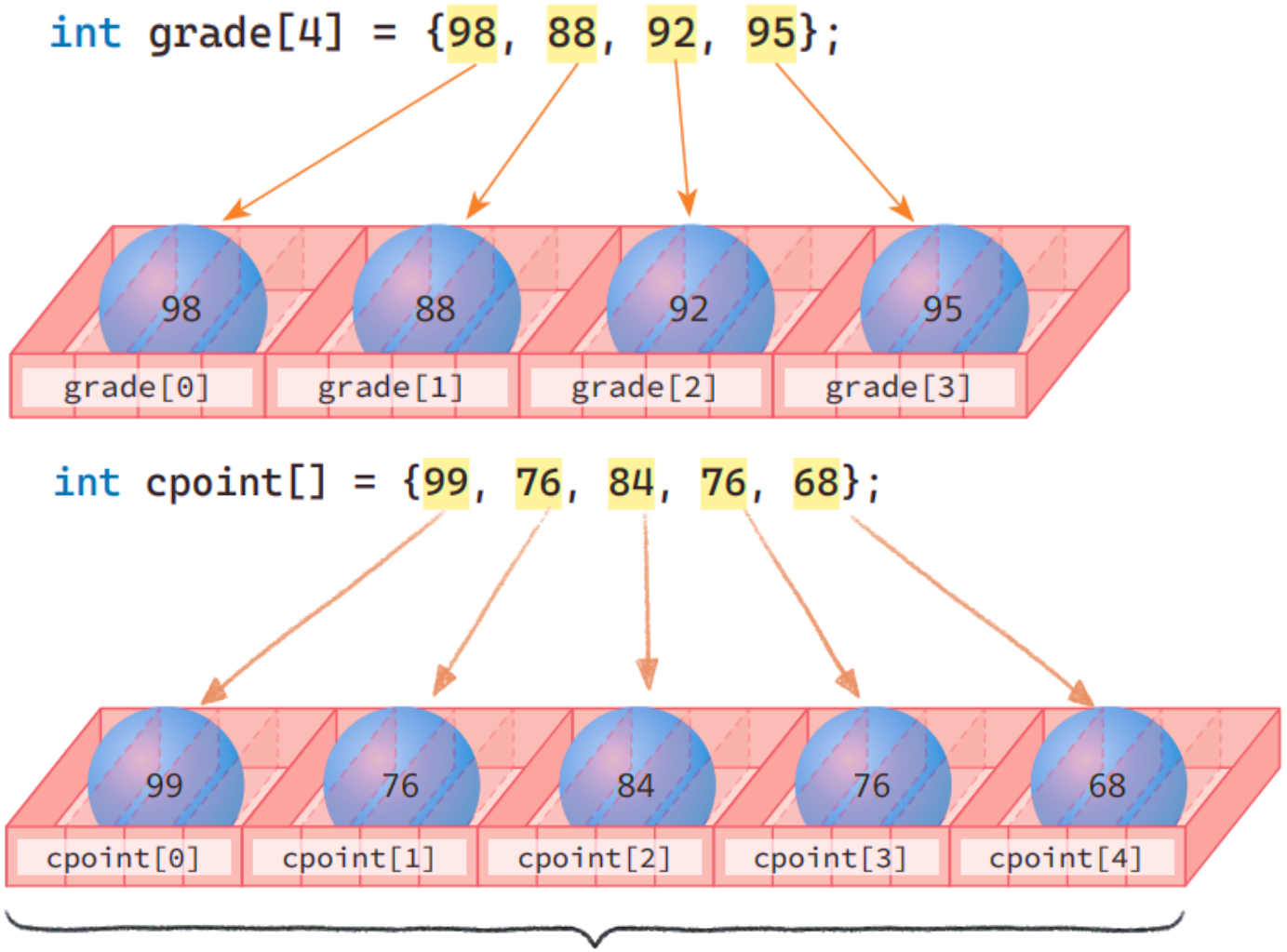
```
원소자료형  배열이름[배열크기] = {원소값1, 원소값2, 원소값3, 원소값4, 원소값5, ... } ;
```

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};
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 346쪽

배열 초기화 3/3



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

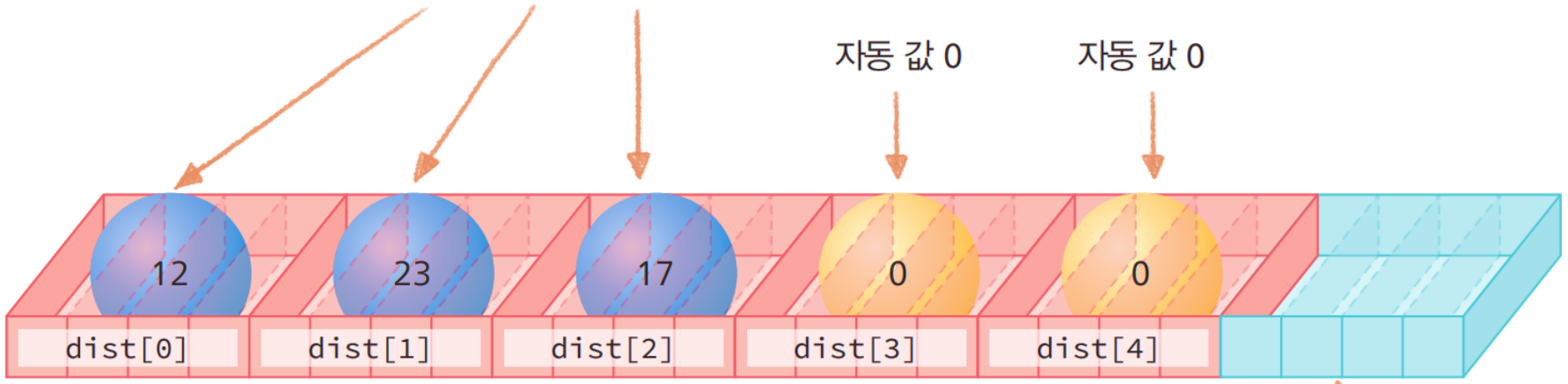
[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 346쪽

배열 초기화 기본 값 1/2

- 지정한 배열크기가 초기값 원소 수보다 크면
  - 지정하지 않은 원소의 초기값은 자동으로 모두 기본값이 저장
    - 기본값이란 자료형에 맞는 0
    - 정수형은 0, 실수형은 0.0
    - 문자 형은 '₩0'인 널문자(문자 코드 번호가 0인 문자)

배열 초기화 기본 값 2/2

```
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};
```

error C2078: 이니셜라이저가 너무 많습니다.

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

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

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 347쪽

배열 선언 초기화 주의점 1/2

- 초기화에서도 변수와 const 상수는 배열크기로 사용 불가
- 반드시 배열 선언 시에만 이용이 가능
  - 배열 선언 이후에는 사용할 수 없음

배열 선언 초기화 주의점 2/2

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

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 348쪽

실습예제 1/2

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
```

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 };
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 348쪽

실습예제 2/2

```
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  }
```

SIZE는 배열크기인 매크로 상수로 양의 정수인 6으로 정의

제어문자 i의 첨자는 0에서 5까지 반복

```
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.
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 349쪽

C99: 배열의 첨자 초기화

배열 첨자 초기화(designated initializers)의 다양한 지원

- 배열의 크기가 지정된 배열 a
  - 지정한 첨자에 대해서는 초기값이, 그 외의 원소는 0으로 저장됨
- 배열의 크기가 지정되지 않은 배열 b
  - 지정한 첨자에 대해서는 초기값이, 그 외의 원소는 0으로 저장됨
  - 가장 큰 첨자가 마지막 원소가 되어 배열의 크기가 결정됨
- 일반적인 배열 초기화 방법인 배열 c
  - 순서대로 초기값이 저장되며 지정한 첨자에 대해서는 초기값이 저장됨
  - 그 외의 원소는 0으로 저장됨

```
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
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 349쪽

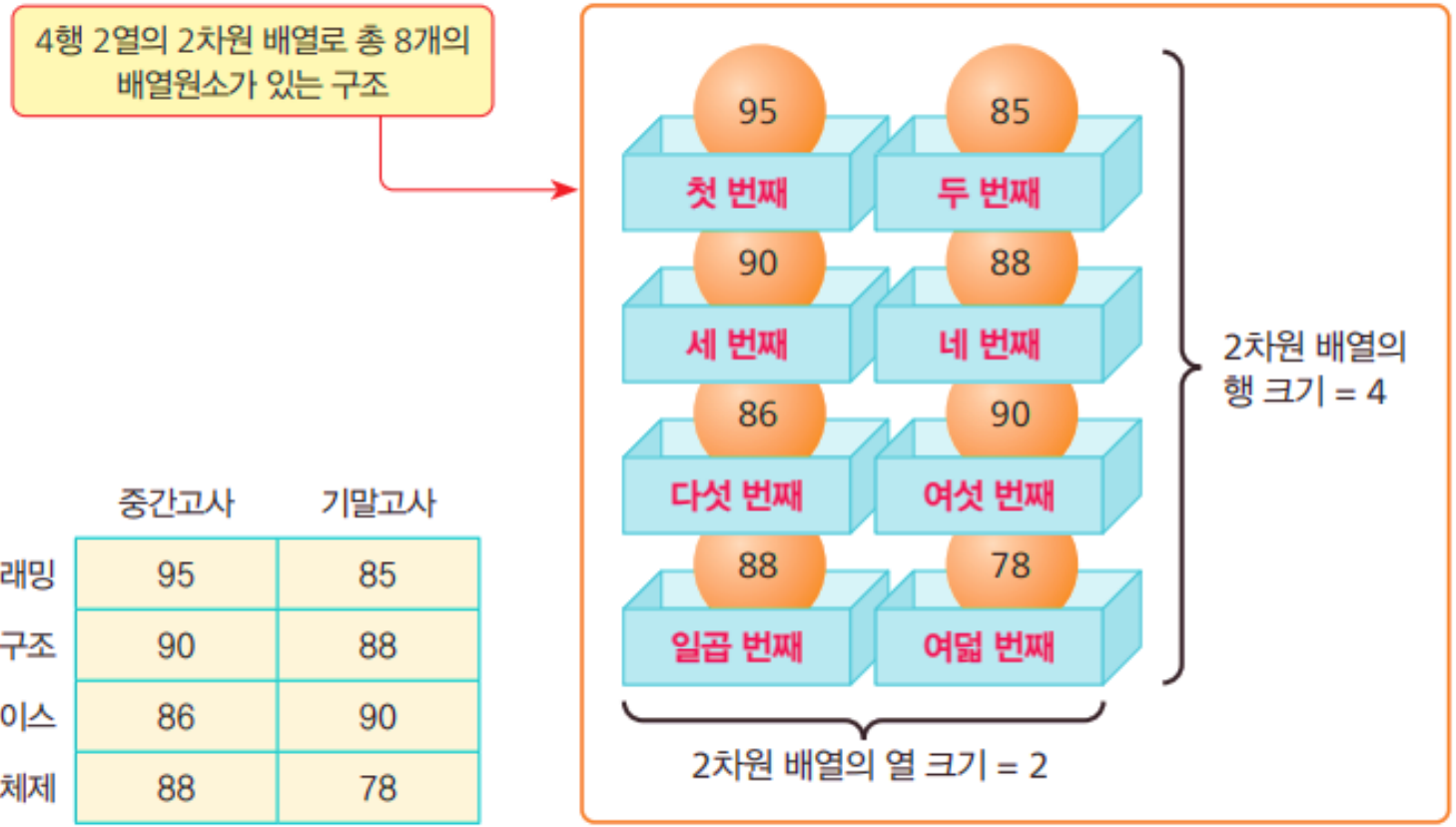
02

다차원 배열

2차원 배열 개요

2차원 배열은 테이블 형태의 구조

- 행(row)과 열(column)의 구조로 표현



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

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 351쪽

2차원 배열 선언 1/2

- 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

2차원 배열 선언 2/2

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];
```

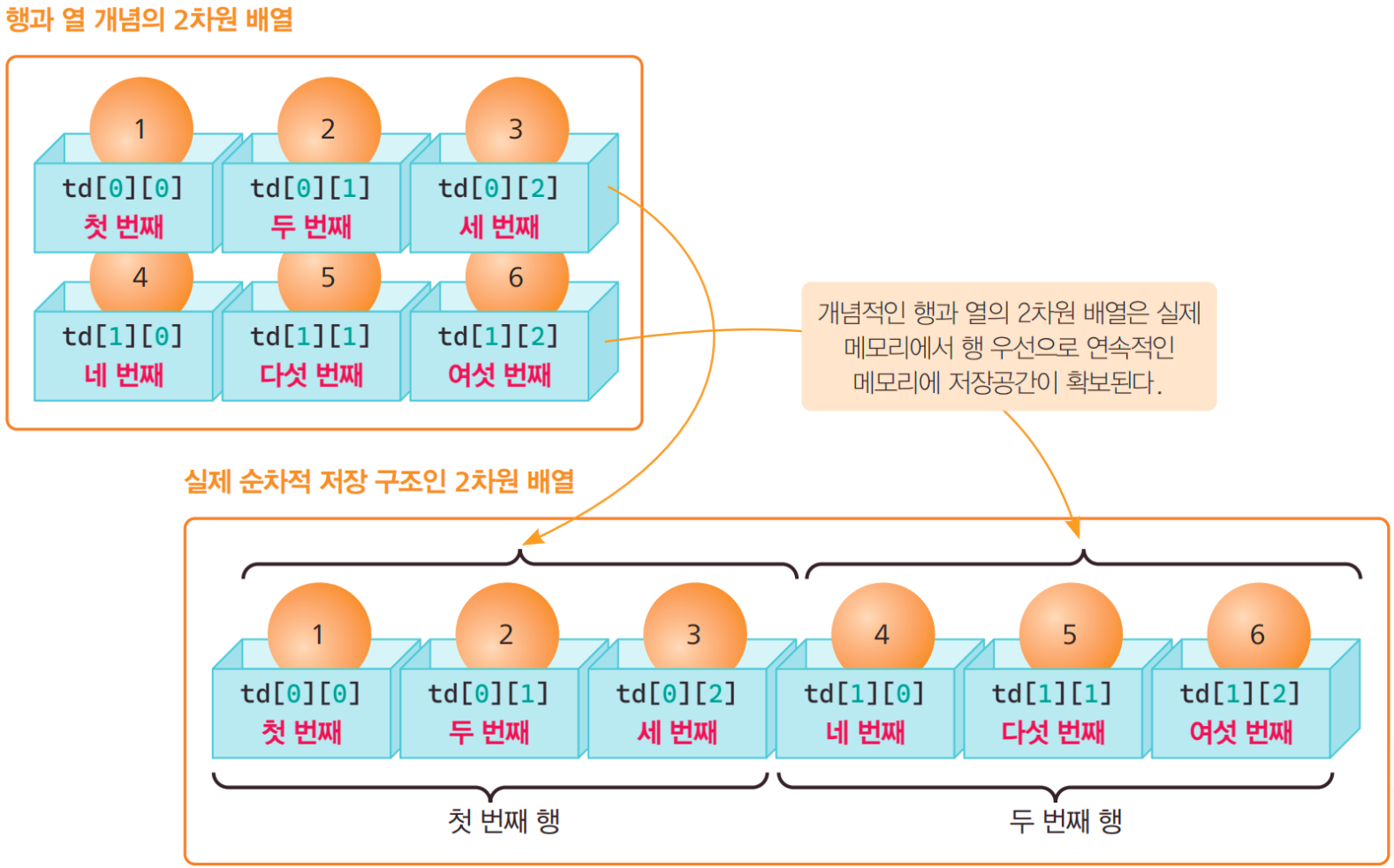
[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 352쪽

행 우선 (row major) 배열 1/2

- 행을 먼저 배치하는 특징
  - 첫 번째 행의 모든 원소가 메모리에 할당된 이후
  - 두 번째 행의 원소가 순차적으로 할당
- 포트란, R 언어
  - 열 우선 배열 지원

행 우선 (row major) 배열 2/2

2행 3열, 2차원 배열 예



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 353쪽

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 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

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 353쪽

실습예제 1/2

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
```

```
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

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 354쪽

실습예제 2/2

```
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  }
```

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

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

```
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
```

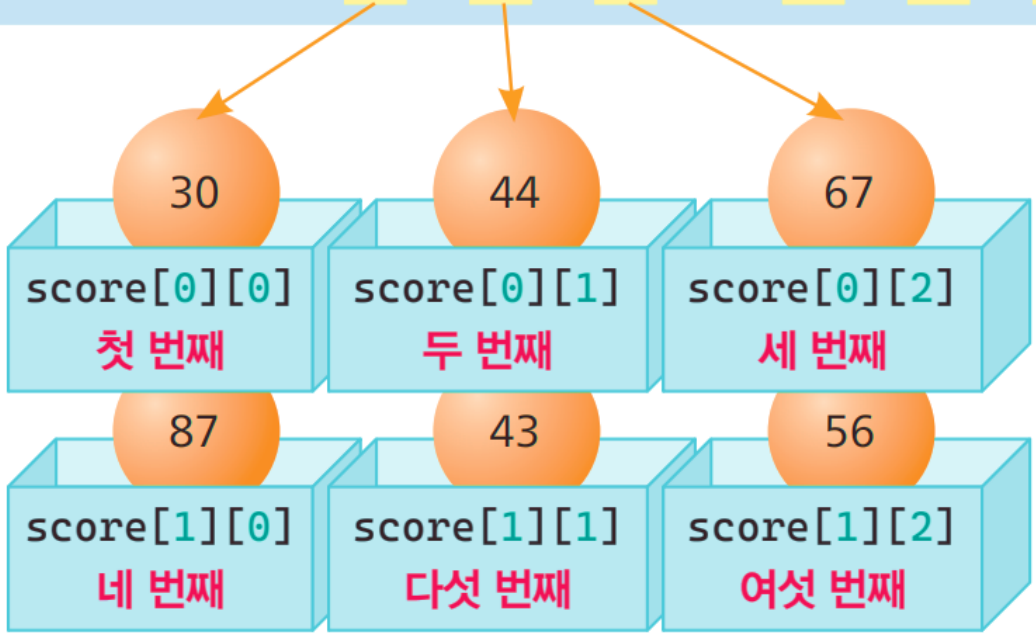
[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 354쪽

2차원 배열 선언 초기화 1/3

- 첫 번째 방법: 중괄호를 중첩되게 이용
  - 2차원 구조를 행과 열로 표현할 수 있는 장점
    - 중괄호 내부에 행에 속하는 값을 다시 중괄호로 묶고, 중괄호와 중괄호 사이에는 쉼표로 분리
    - 행인 중괄호 내부의 초기값들은 쉼표로 분리

2차원 배열 선언 초기화 2/3

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



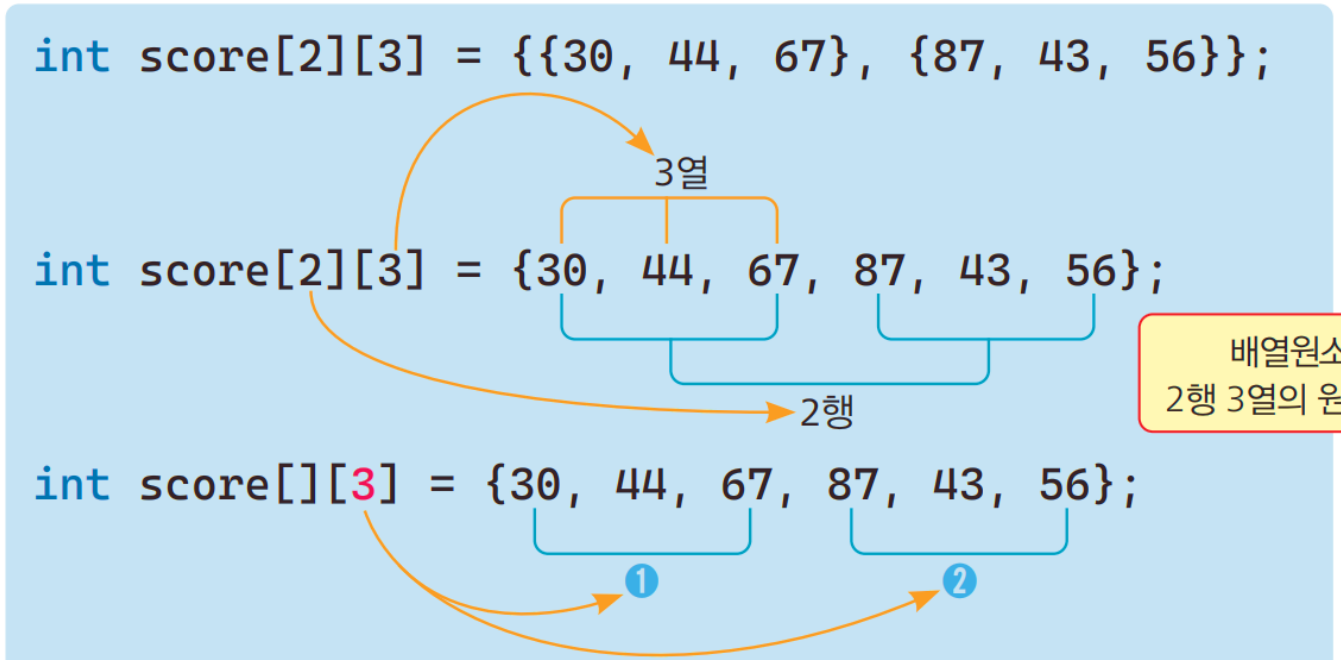
[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 355쪽

2차원 배열 선언 초기화 3/3

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



배열원소를 순차적으로
2행 3열의 원소값으로 인지한다.

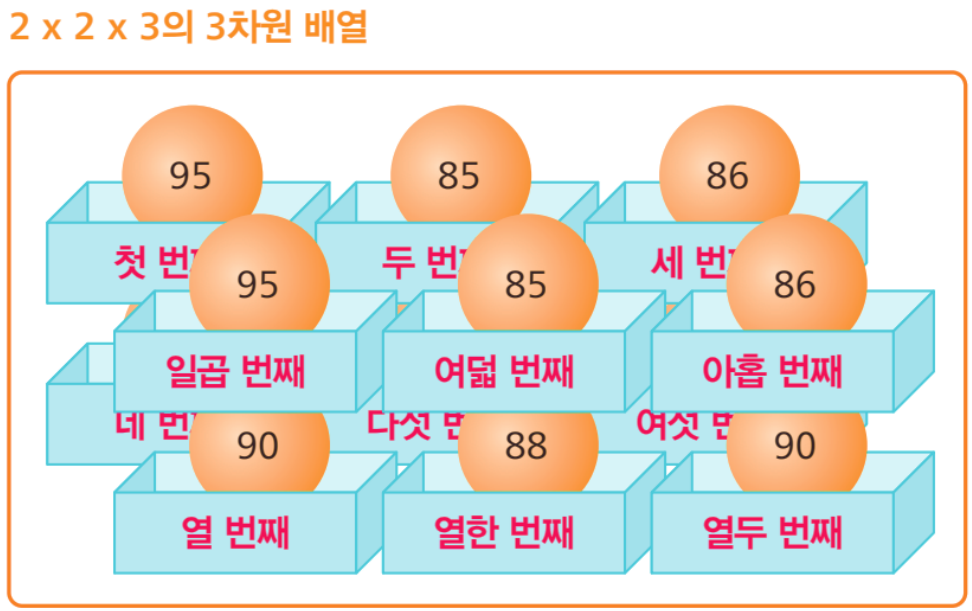
명시된 열 수인 3을 보고 3개씩 나누어보면 행이 2인 것을 알 수 있다.

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 356쪽

3차원 배열 선언

- int threed[2][2][3];
  - 총 2×2×3 = 12개, 배열원소
    - 2행 3열 2차원 배열이 2개로 해석
  - 대괄호 내부 세 개의 크기는 모두 필요

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



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 359쪽

3차원 배열 초기화

- 3차원 배열, 학생의 점수를 초기값으로 저장
  - score [2] [4] [2]

[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 } }
};
```

[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 360쪽

03

배열 크기

배열크기 연산

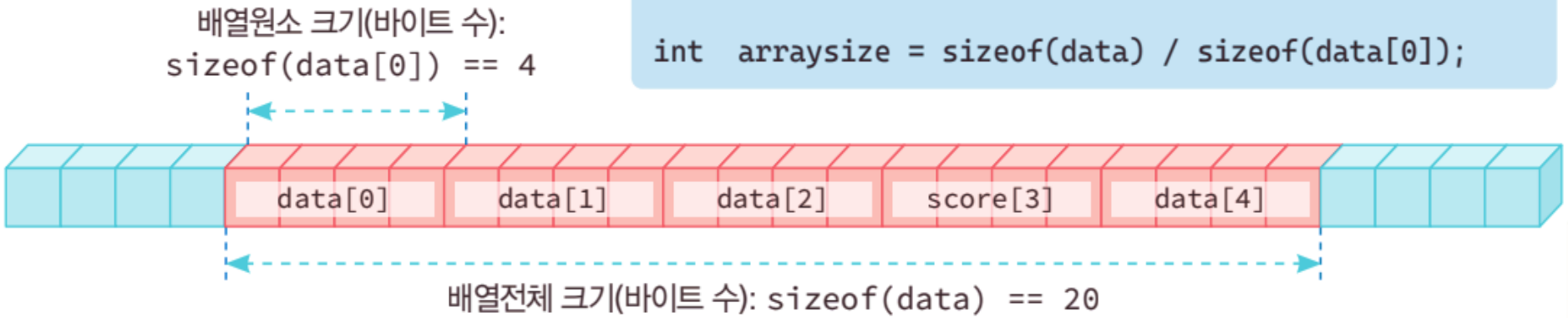
- 연산자 sizeof(변수), sizeof(자료형)
  - 저장공간의 크기를 바이트 수로 반환하는 연산
- 배열크기 계산방법
  - sizeof(배열이름)
    - 배열의 전체 공간의 바이트 수
  - sizeof(배열원소)
    - 배열원소 하나의 바이트 수
  - sizeof(배열이름) / sizeof(배열원소)
    - 배열크기(배열 원수 소) 반환

배열크기 연산

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

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

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



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 362쪽

2차원 배열크기 계산방법

- 2차원 배열의 행의 수
  - sizeof(x) / sizeof(x[0])
    - sizeof(x)
      - 배열의 전체 공간의 바이트 수
    - sizeof(x[0])
      - (첫) 행의 바이트 수
- 2차원 배열, 열의 수
  - sizeof(x[0]) / sizeof(x[0] [0])
    - sizeof(x[0] [0])
      - (첫) 원소의 바이트 수

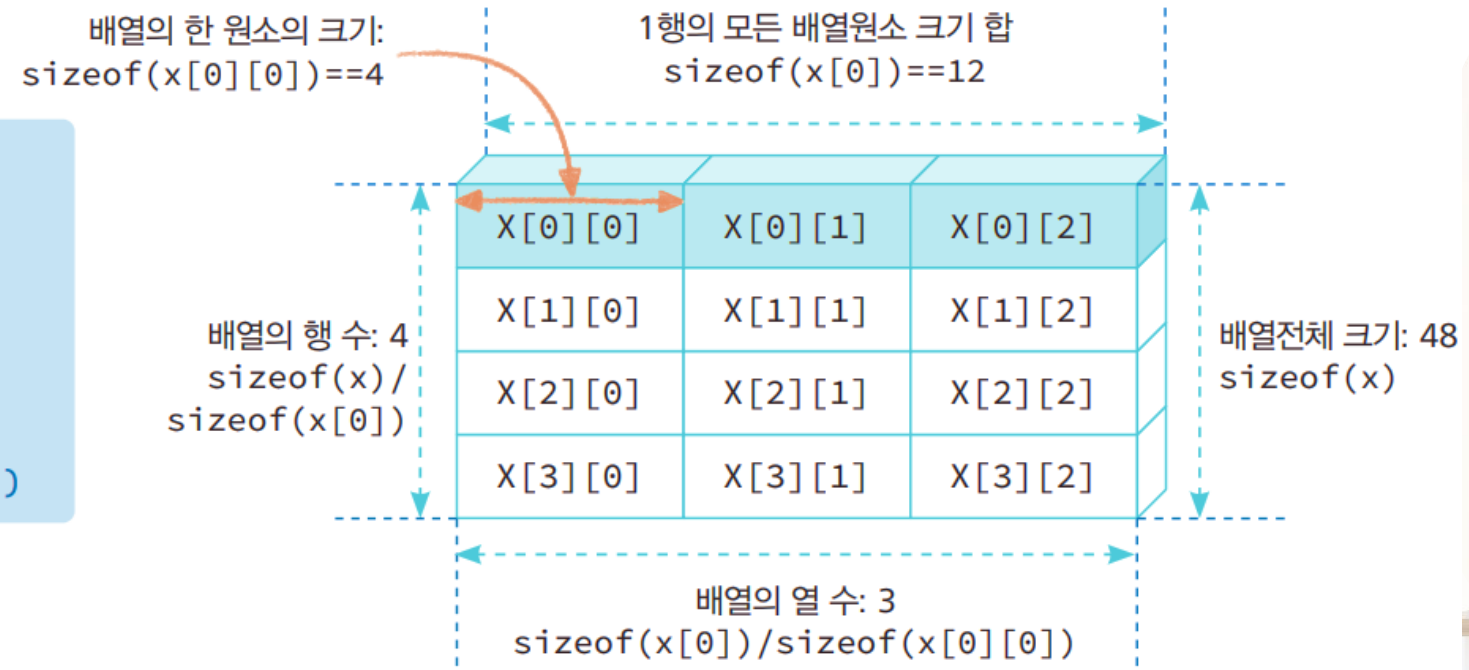
2차원 배열크기 계산방법

4행 3열, 2차원 배열

```
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])
```



[이미지] 강환수 외 2인 공저, Perfect C (3판), 인피니티북스, 2021, 363쪽

정 리 하 기

정리하기

- 배열 필요성을 이해하고 배열을 선언하고 활용한다.
- 배열 선언에서 배열크기는 리터럴 상수와 매크로 상수로 지정한다.
- 선언된 배열에서 배열 원소를 참조한다.
- 배열 선언에서 1차원, 2차원 배열 원소를 초기화한다.
- 2차원과 3차원 등의 다차원 배열을 활용할 수 있다.
- 선언된 배열에서
  배열의 원소 수인 배열 크기를 직접 구해 활용한다.

다음시간안내

7강

함수