



배열

학습내용

- 선언과 초기화
- 다차원 배열

학습목표

- 배열의 용도를 알고, 배열을 초기화하여 자료처리에 사용할 수 있다.
- 다차원 배열의 개념에 대해 설명할 수 있다.

선언과 초기화

1 배열 이해

1 고급언어와 저급언어의 특징



동일한 이름으로 참조되는 연속된 메모리에
할당된 자료 구조



같은 데이터 타입을 묶어서
하나의 공간처럼 사용할 수 있는 자료 구조



관련 있는 데이터를 묶어서 처리할 필요가 있을 때
유용한 자료 구조

2 효과

01

많은 수의 변수 이름을 생성할 필요가 없음

02

동일한 이름을 사용하므로 반복문으로 구현하기가
용이함

선언과 초기화

1 배열 이해

3 용어

요소(Elements)

배열을 구성하는 각 항목

배열(Array)명

전체 공간에 대한 대표 이름(변수명)

크기

배열요소의 개수

첨자(Index)

각 요소에 부여되는 위치 정보

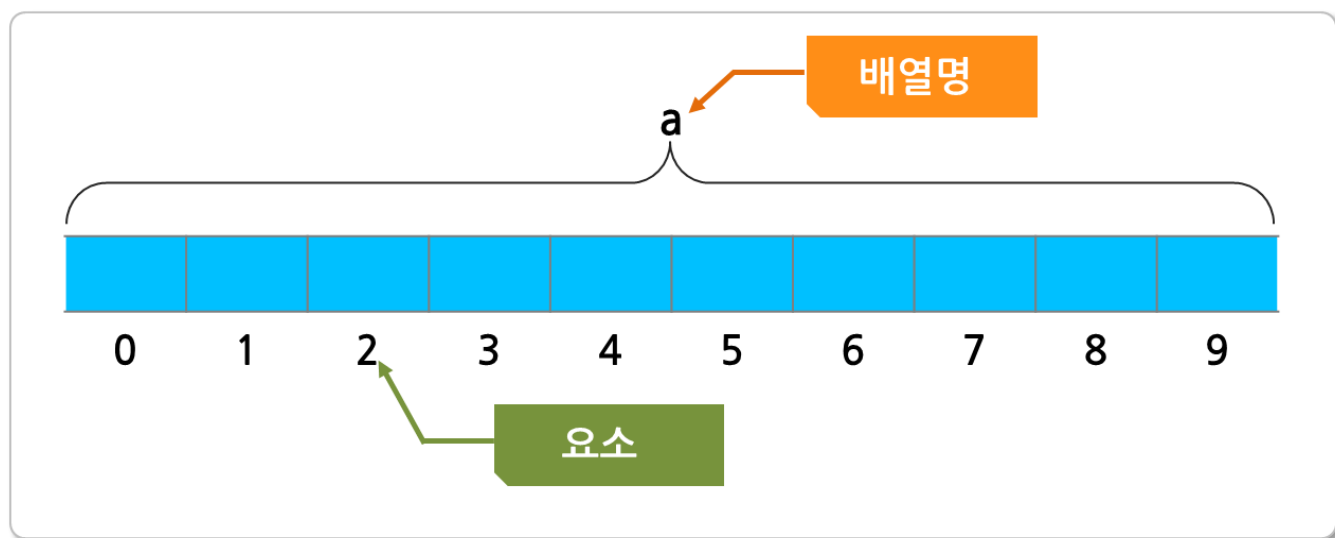


아파트 = 배열구조

선언과 초기화

1 배열 이해

3 용어



2 선언

데이터 타입 배열명[크기] ;

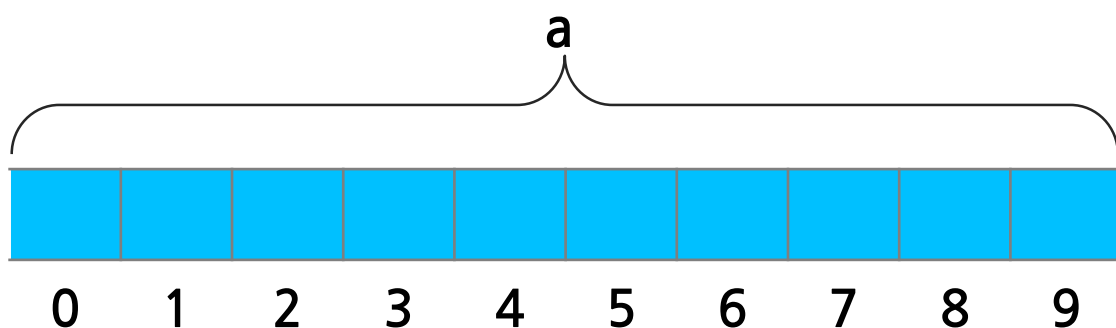
배열 크기

- 양의 정수, 매크로 상수, 관련 연산식으로 지정
- 변수, const 상수 불가

선언과 초기화

2 선언

```
int a[10];
```



```
#define SIZE 5
```

```
int a[100];
double b[20];
int c[SIZE];
char d[SIZE+1];
```

```
int size=5;
```

```
int a[100];
double b[20];
int c[size];
char d[size+1];
float e[3.5];
int f[-3];
short g[0];
int h[ ];
```

선언과 초기화

3 참조

01 각 요소에 대한 참조는 index를 이용

02 0 ~ size-1

03 배열명[index]

04 범위 밖의 요소를 참조하는 경우 실행(Run Time) 오류가 발생

```
int a[5];  
a[0]=11;  
a[1]=22;  
a[2]=33;  
a[3]=44;  
a[4]=55;
```

11

22

33

44

55

선언과 초기화

4 초기화

- 1 형식: 데이터 타입 배열명[크기] = {초기값1, 초기값2, 초기값3, ...};
- 2 각 요소는 순서대로 인덱스 0부터 초기화
- 3 배열 크기보다 초기화 요소수가 적으면 나머지는 0으로 초기화
- 4 초기화하지 않은 지역 배열요소는 쓰레기 값을 가짐
- 5 선언과 초기화를 같이 하는 경우 배열 크기는 생략 가능

```
int a[5] = { 11, 22, 33, 44, 55};
```



```
int a[5] = { 11, 22, 33};
```



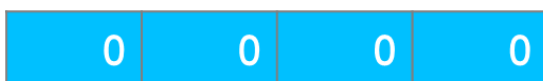
선언과 초기화

4 초기화

```
int a[ ] = { 11, 22, 33 };
```



```
int a[4] = { 0 };
```



학생 50명의 국어 성적을 출력하고 학급평균을 처리하는 프로그램

```
#include <stdio.h>
int main()
{
    int i, kor[50], sum=0;
    for( i=0 ; i<50 ; i++ )
    {
        printf("%d번 학생의 국어점수를 입력하세요 ");
        scanf("%d", &kor[i] );
        sum += kor[i];
    }
}
```

선언과 초기화

4 초기화

학생 50명의 국어 성적을 출력하고 학급평균을 처리하는 프로그램

```
for( i=0 ; i<50 ; i++ )
{
    printf(“%d번 학생의 국어점수 : %d \n“, kor[i] );
}

printf(“학급평균은 %f 이다”, sum/50.0);
return 0;
}
```

다차원 배열

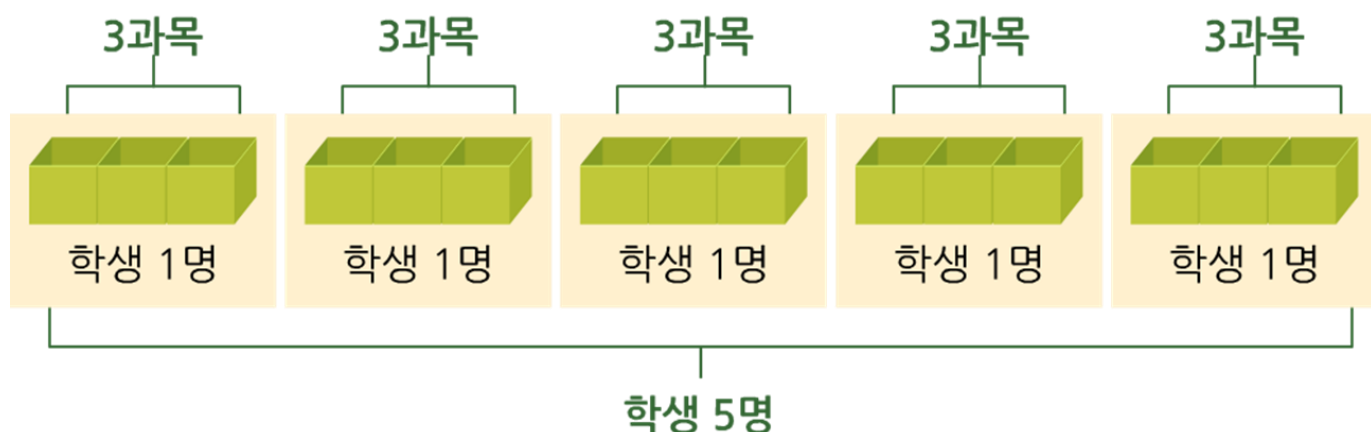
1 2차원 배열

01 배열을 이차원 형태로 확장

02 2차원은 index가 2개

03 index의 수가 배열의 차수

04 형식 : 데이터 타입 배열명 [행 크기][열 크기] ;



다차원 배열

1 2차원 배열

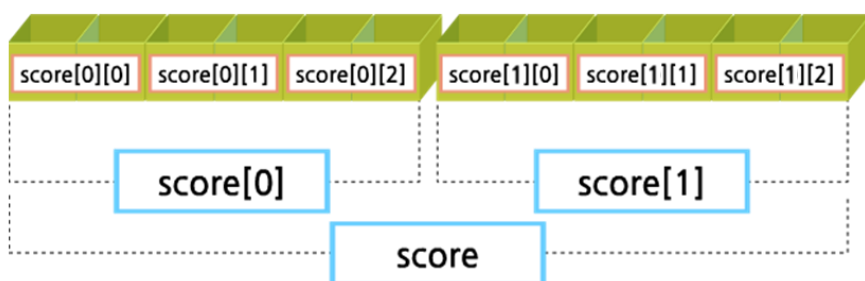
05

프로그래머는 2차원 형태이나 컴퓨터는 일차원 구조

score[0]	[0][0]	[0][1]	[0][2]
score[1]	[1][0]	[1][1]	[1][2]

short score [2] [3]

3개씩 묶음 2개



다차원 배열





1 2차원 배열

50명 학생의 국어, 영어, 수학 성적을 처리하는 프로그램

```
int kor[50], eng[50], mat[50];
```



```
int score[50][3];
```

번호	학생	score	국어	영어	수학
#1		score[0]	[0][0]	[0][1]	[0][2]
#2		score[1]	[1][0]	[1][1]	[1][2]
#3		score[2]	[2][0]	[2][1]	[2][2]
...
#50		score[49]			

다차원 배열

1 2차원 배열

50명 학생의 국어, 영어, 수학 성적을 처리하는 프로그램

```
int i, score[50][3];
for(i=0;i<50;i++)
    scanf("%d %d %d", &score[i][0], &score[i][1], &score[i][2] );
for(i=0;i<50;i++)
    printf("%d 번 학생의 성적 : %d  %d  %d", i+1,
    score[i][0], score[i][1], score[i][2] );
```

이차원 배열의 초기화

행 단위로 { }로 묶어서 초기화(열 크기만큼 묶음)

`short data[2][3] = { {1, 2, 3}, {4, 5, 6} };` — 3개씩 묶어서 초기화함

일차원 배열처럼 { } 안에 초기값만 나열할 수도 있음

`short data[2][3] = { 1, 2, 3, 4, 5, 6 };` — 위의 코드와 같은 의미

다차원 배열

1 2차원 배열

이차원 배열의 초기화

초기값을 생략하면 나머지 원소를 0으로 초기화

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

 → {{1, 2, 0}, {3, 4, 0}}으로 초기화됨

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

 → {1, 2, 3, 4, 0, 0}으로 초기화됨

이차원 배열에서 행 크기는 생략 가능, 열 크기는 생략 불가

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

 → `int a[3][2];`로 할당됨

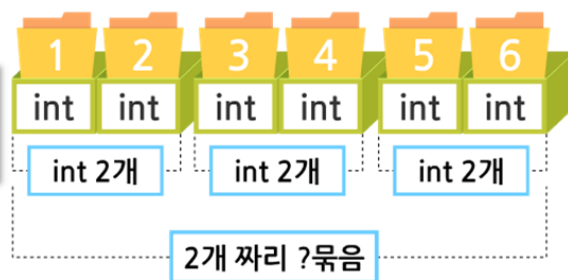
```
int b [3] [ ] = { 1, 2, 3, 4, 5, 6 };
```

 → 컴파일 에러

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

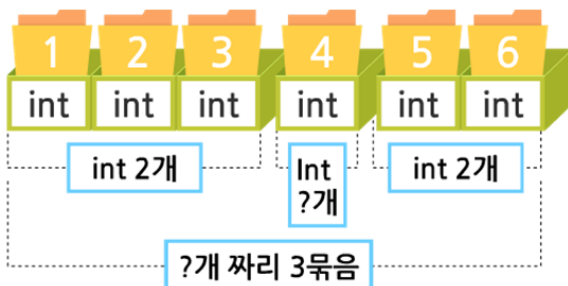
 (초기값으로부터 2개짜리가 3묶음이라는 것을 알 수 있음)

초기값으로부터 2개짜리가
3묶음이라는 것을 알 수 있음



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

몇 개씩 묶을 지 알 수 없음



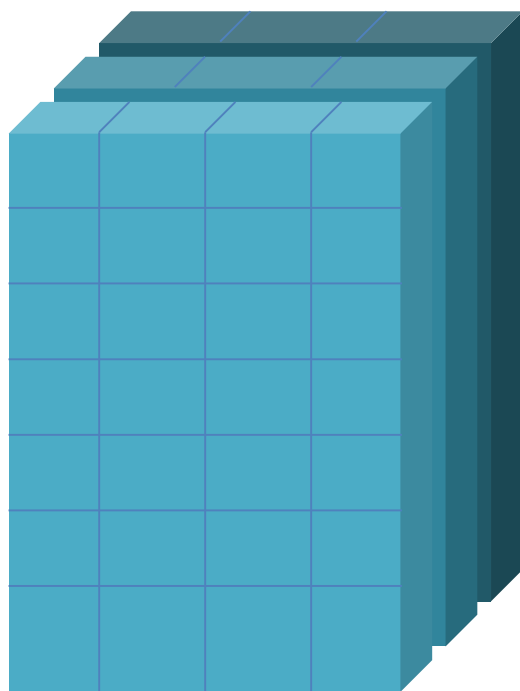
다차원 배열

2 3차원 배열

1 필요에 따라 다차원 배열 선언이 가능

2 데이터 타입 배열명 [면][행][열]

3 동일한 초기화 방법 사용



`int a[3][7][4]`

다차원 배열

2 3차원 배열

5개 학급의 학생별 3과목 성적 처리

```
#include <stdio.h>

int main()
{
    int i,j,score[5][50][3];
    for(i=0;i<5;i++)
    {
        printf("%d반 성적입력", i+1);
        for(j=0;j<50;j++)
            scanf("%d %d %d", &score[i][j][0], &score[i][j][1], &score[i][j][2]);
    }

    return 0;
}
```

학습정리

1. 선언과 초기화

- 배열 : 같은 데이터형의 변수들을 메모리에 연속적으로 할당하고 같은 이름으로 사용하는 자료 구조
- 배열의 선언
 - 배열 원소의 데이터형, 배열 이름, 배열의 크기가 필요함
 - 배열의 크기는 상수로만 지정함
- 배열의 사용
 - 배열의 각 원소에 접근하려면 인덱스를 사용함
 - 인덱스는 항상 0~(배열의 크기-1) 사이의 값임
- 배열의 초기화
 - 배열을 초기화하려면 { } 안에 초기값을 나열함
 - 배열을 초기화할 때는 배열의 크기를 생략할 수 있음

2. 다차원 배열

- 필요 시 2차원 이상의 배열 형태를 구현하는 것이 가능함
- 실제 메모리 구조는 인접한 메모리의 연속임