동일한 자료형 여러 개를 묶어 사용하기 위해 배열을 사용합니다.

```
int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

위의 배열은 정수 자료를 아래처럼 보관하고 있습니다.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

혹은 세로로 볼 수도 있겠죠?

1
2
3
4
5
6
7
8
9
10

다중 배열

그렇다면 2차원 구조는 배열로 어떻게 나타낼 수 있을까요?

1	**2**	**3**	**4**
5	6	7	8
9	10	11	12

'다중 배열'을 선언하면 됩니다.

```
int[][] multiArray;
```

위 표의 내용으로 초기값을 바로 설정하기 위해서는 이렇게 써야 합니다.

```
int[][] multiArray = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};
```

' **int[4]** 배열 세 개가 묶인 배열'이라고 볼 수 있습니다. 즉, **multiArray[0]** 의 자료형은 **int[4]** 이고 내용은 **{1, 2, 3, 4}** 인 것이죠.

생성

위에서는 배열의 초기값을 바로 설정해주었는데요. 만약 선언과 생성만 하기 위해서는 어떻게 해야 할까요?

3 x 4 사이즈의 빈 배열을 만들어봅시다.

```
int[][] multiArray = new int[3][4];
```

각 대괄호 사이에 사이즈를 넣어줍니다. 일반적으로 '행(줄)'을 첫 번째 대괄호에, '열(칸)'을 두 번째 대괄호에 넣습니다.

사용

앞서 보셨듯, **multiArray[0]** 은 이제 **int[4]** 의 자료형을 갖게 됩니다. 마찬가지로 **multiArray[1]** , **multiArray[2]** 모두 **int[4]** 의 자료형을 갖게 되죠. 그렇기 때문에 **multiArray[0]** 을 일반적인 배열 탐색법으로 탐색할 수 있습니다.

```
for (int i = 0; i < multiArray[0].length; i++) {
    multiarray[0][i] = 1 + i;
}
```

마찬가지 방법으로 위의 표 내용처럼 **multiArray** 를 이렇게 채울 수 있습니다.

```
for (int i = 0; i < multiArray[0].length; i++) {
    multiarray[0][i] = 1 + i;
}

for (int i = 0; i < multiArray[1].length; i++) {
    multiarray[1][i] = 5 + i;
}

for (int i = 0; i < multiArray[2].length; i++) {
    multiarray[2][i] = 9 + i;
}
```

중첩 반복문 (Nested Loops)

중첩 반복문 (Nested Loops)

하지만 위의 방법도 너무 반복적이죠? 중첩 반복문을 사용하면 깔끔하게 쓸 수 있습니다!

```
for (int i = 0; i < multiArray.length; i++) {
    for (int j = 0; j < multiArray[i].length; j++) {
        multiArray[i][j] = (i * 4 + 1) + j;
    }
}
```

여기서 **`multiArray.length`** 는 전체 자리 수 **`12`** 가 아닌, 행(줄)의 수인 **`3`** 입니다.

☑ 수업을 완료하셨으면 체크해주세요.

수강생 Q&A 보기 (/questions?assignment_id=487&sort_by=popular)

질문하기 (/questions/new?assignment_id=487&op1=%EA%B0%9D%EC%B2%B4+%EC%A7)

이전 강의
DNA 염기 서열 분석 (해설) (/assignments/326)

다음 강의
콘솔 입력 받기 (Scanner) (/assignments/343)