
배열의 선언과 사용

- Chapter 06 -

학습목차

I. 배열 개요 및 사용 방법

II. 배열과 반복문

III. 형변환

배열 개요

▶ 배열의 필요성

▶ 동일한 자료형(data type)을 가진 연속된 메모리 공간으로 이루어진 자료구조(data structure)

▷ 같은 자료형의 변수들이 여러 개 필요할 때 사용

▶ 많은 양의 데이터를 처리할 때 유용

▶ 배열을 사용하면 코드의 길이가 짧아지고 가독성이 향상됨

▷ 예) 100명의 시험 점수를 관리하는 프로그램을 만들 경우 100개의 변수가 필요

```
#include <stdio.h>
int main(void)
{
    // int형 변수 100개
    int student1, student2, ... , student100;
    ...
    return 0;
}
```



```
#include <stdio.h>
int main(void)
{
    // int형 배열
    int student[100];
    ...
    return 0;
}
```

배열의 사용 방법

▶ 배열의 선언과 초기화

▶ 배열의 선언

- ▶ 자료형 : 배열의 자료형을 지정(char, int, float ...)
- ▶ 배열 이름 : 변수 이름과 마찬가지로 배열을 구분하는 이름
 - ▶ 배열 요소(element)는 배열의 이름을 공유하고 첨자(index number)로 구분
- ▶ 배열 길이(=요소 수) : 배열 요소의 총 개수(=변수의 총 개수)

자료형	배열 이름	[배열 길이]
↓	↓	↓
int	array	[10];

▶ 배열의 선언과 요소의 사용

- ▶ 배열 요소의 일괄적인 초기화는 반드시 선언과 동시에 중괄호{ }로 데이터를 묶어서 대입

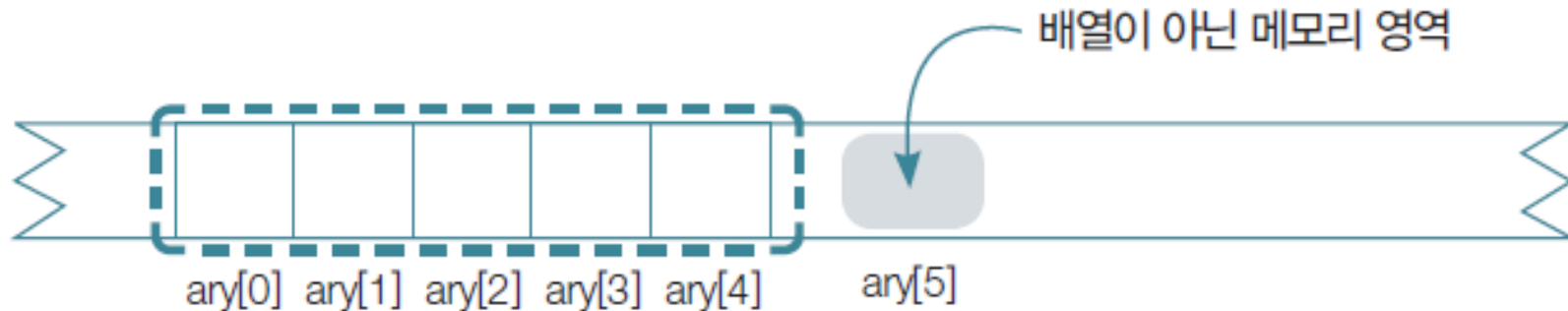
구분	사용 예	기능
배열 선언	int ary[5];	int형 변수 5개를 한 번에 확보한다.
요소 사용	ary[0], ary[1], ary[2], ary[3], ary[4],	배열 요소를 사용할 때는 첨자를 0부터 시작하여 '요소 수 - 1'까지 쓴다.
초기화	int ary[5] = {1, 2, 3, 4, 5};	초기화는 중괄호 안에 값을 나열한다.

배열 개요

▶ 배열의 사용 방법

- ▶ 각 저장 공간을 이름과 첨자(색인:index)로 구분
 - ▷ 저장 공간의 개수와 관계없이 이름은 하나만 사용
 - ▷ 배열로 선언된 변수는 번호(첨자)로 구분
 - ▷ student[0], student[1], student[2]...
- ▶ 변수 선언과 마찬가지로 메모리에 저장 공간 확보
 - ▷ 메모리에 연속된 저장 공간을 한꺼번에 확보
- ▶ 사용할 때는 하나씩 따로 사용하는 방식으로 구현
 - ▷ 인덱스(배열의 번호)는 0부터 시작하고 최대 '배열 요소 수 - 1'까지만 사용 가능

```
#include <stdio.h>
int main(void)
{
    // int형 배열
    int student[30];
    student[0] = 10;
    student[1] = 20;
    student[2] = student[1] + student[0];
    ...
    return 0;
}
```



배열의 사용 방법

▶ 배열의 초기화

- ▶ 배열의 길이는 변수가 아닌 상수로만 설정가능(리터럴 상수, 심볼릭 상수 둘 다 가능)
- ▶ 일괄적인 초기화는 배열의 선언과 동시에 데이터를 중괄호{ }로 묶어서 대입해야 함
- ▶ 선언과 동시에 대입(초기화)하지 않는 경우, 이후에는 배열 변수를 각각 초기화해야 함

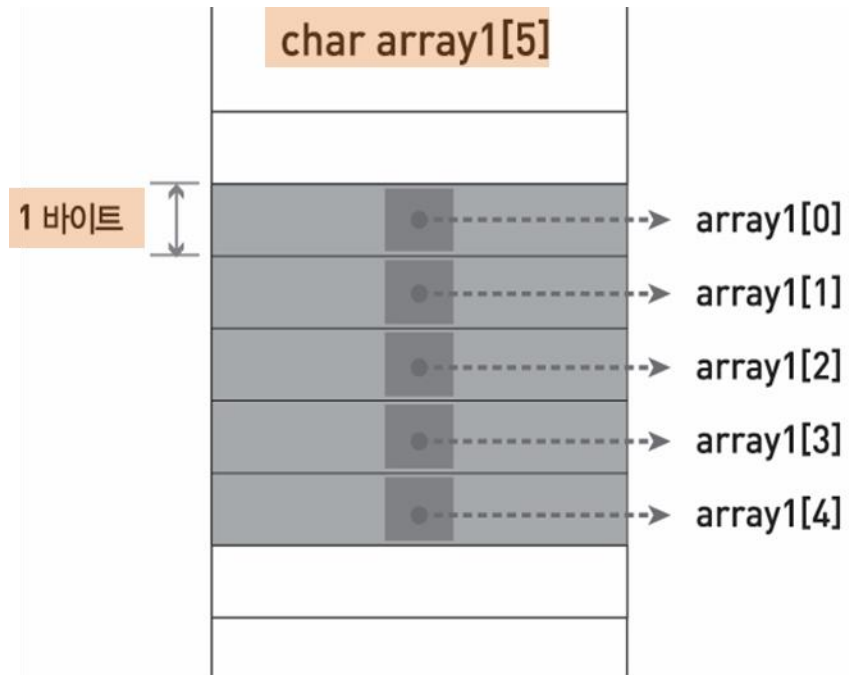
```
int max = 5;
const int SIZE = 5;           //심볼릭(symbolic) 상수 선언
int array0[max];              //오류 발생
int array0[SIZE];
int array1[5] = {1, 2, 3, 4, 5};
int array2[5] = {1, 2, 3};    //맨 처음 요소부터 대입하고 나머지는 0으로 초기화
int array3[ ] = {1, 2, 3, 4, 5}; //배열의 크기(요소수)는 데이터의 개수에 의해 자동 정의
int array4[5];
array4 = {1, 2, 3, 4, 5};     //오류 발생
array4[0] = 1;
array4[1] = 2;
array4[2] = 3;
array4[3] = 4;
array4[4] = 5;
```

배열의 사용 방법

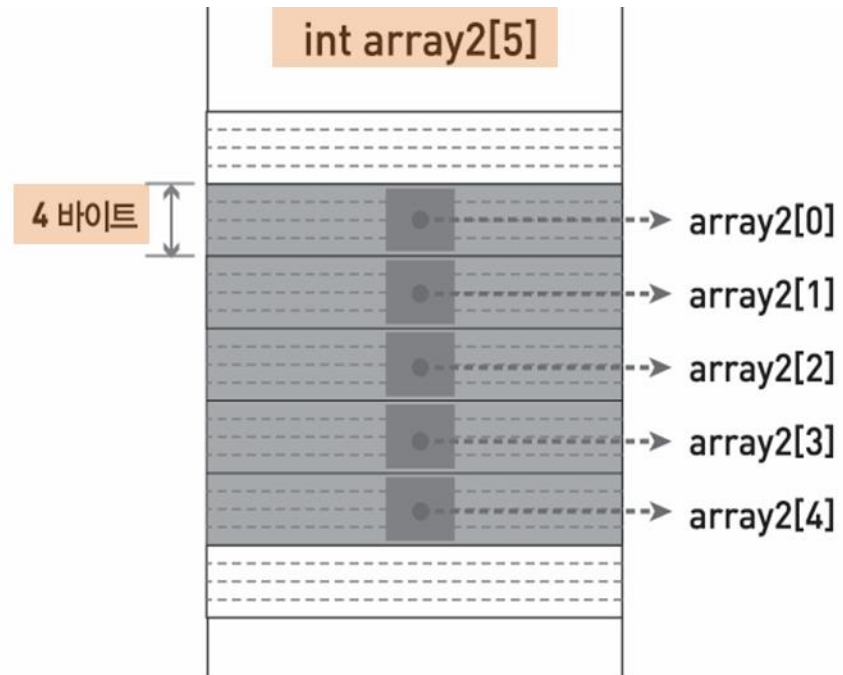
▶ 배열의 크기

- ▶ 배열의 자료형에 따라 같은 요소수라도 메모리 크기가 다름

```
char array1[5];  
int array2[5];
```



- ① 총 5바이트 크기의 연속된 메모리 공간을 할당하며 배열 요소는 0부터 시작



- ② 총 20바이트 크기의 연속된 메모리 공간을 할당하며 배열 요소는 0부터 시작

배열의 사용 방법

▶ sizeof 연산자를 활용한 배열 처리

▶ 배열은 보통 많은 양의 데이터 처리

▷ 반복문 사용 필수

▶ 반복문에서 처리해야 하는 배열 요소 수가 바뀌는 경우?

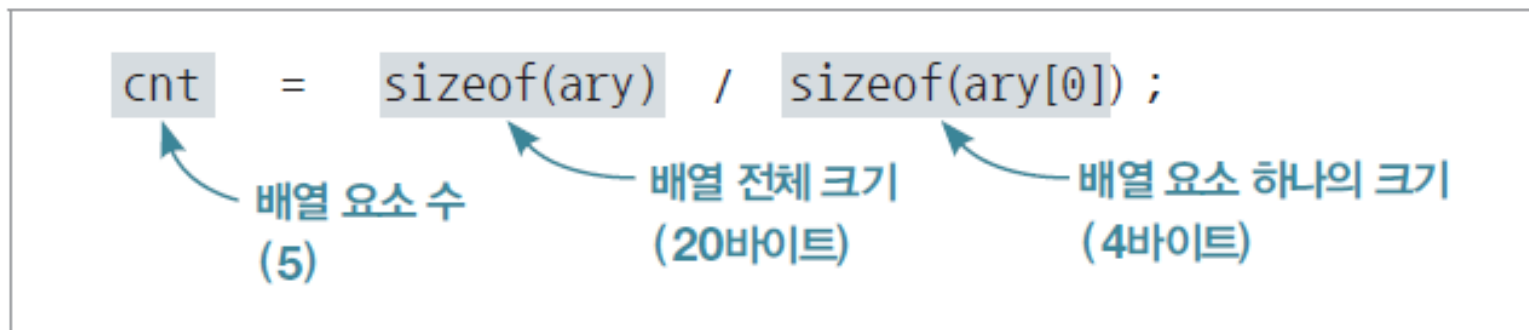
▷ 배열 처리 반복문의 조건을 매번 수정해야 하는 부담

▷ 배열 요소 수를 자동으로 계산하여 반복문에 사용하면 편리

▶ 배열 요소 수 = 'sizeof (배열명) / sizeof (배열 요소)'

▷ int ary[5]; 인 경우에는 배열의 총 크기는 20byte이고 요소 한 개의 크기는 4byte

▷ 결국 배열의 총 크기를 요소 한 개의 크기로 나누면 배열에 포함된 요소의 개수를 알 수 있음



배열의 사용 방법

▶ 배열과 반복문

▶ 모든 배열 요소를 일일이 하나씩 사용하는 것은 번거롭기 때문에 반복문과 같이 사용하는 것이 효율적

▶ 예) 배열 요소 다섯 개인 배열에 정수 모두 입력하는 예제

```
int score[5];           // 배열 선언
scanf("%d", &score[0]); // 첫 번째 배열 요소에 입력
scanf("%d", &score[1]);
scanf("%d", &score[2]);
scanf("%d", &score[3]);
scanf("%d", &score[4]);
```

▶ 바뀌는 것은 첨자 뿐 -> 반복문 사용가능

```
int score[5];
int i;
for(i = 0; i<sizeof(score)/sizeof(score[0]); i++)
    scanf("%d", &score[i]);
```

배열의 사용 방법

▶ 배열과 반복문

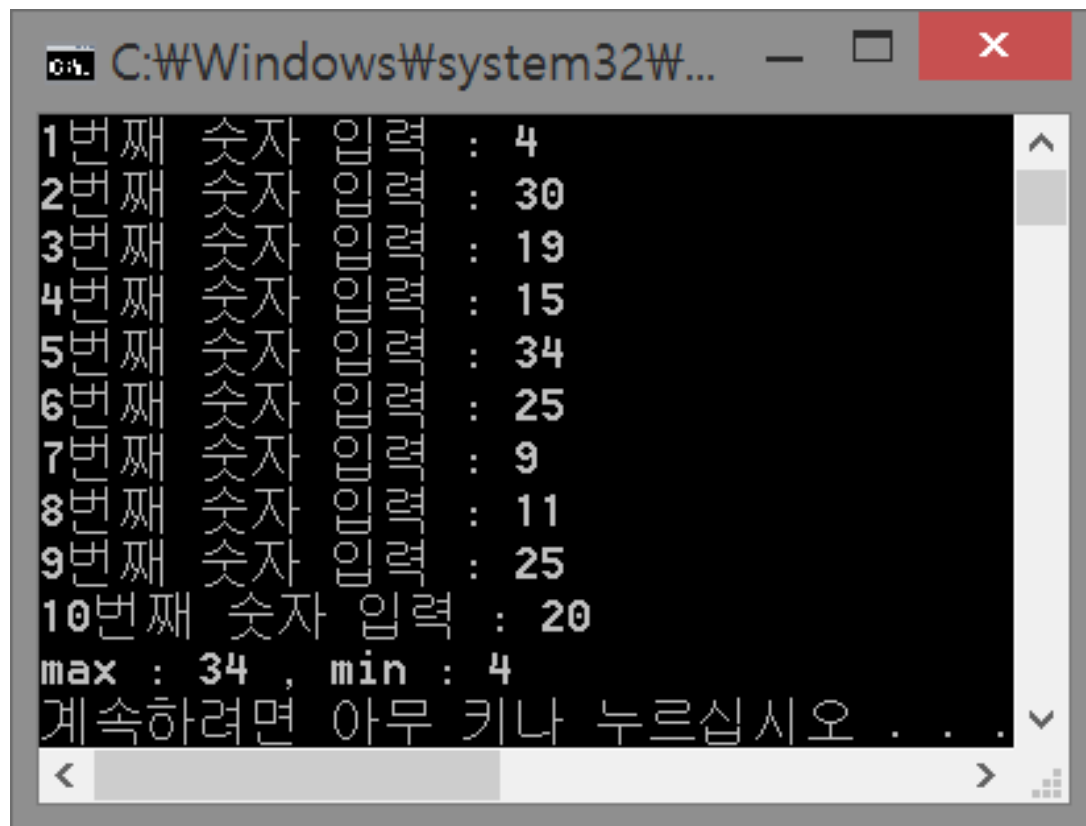
- ▶ 전체 요소를 출력할 경우에도 동일하게 사용
- ▶ 분모로 score[0]을 사용하는 이유는 배열이라면 0번째 요소가 반드시 존재하기 때문

```
int i;  
for(i = 0; i<sizeof(score)/sizeof(score[0]); i++)  
    scanf("%d", &score[i]);  
for(i = 0; i<sizeof(score)/sizeof(score[0]); i++)  
    printf("%d",score[i]);
```

실습예제 01

▶ 10개의 숫자를 입력받아 int input[10]내에 모두 저장한 후 최댓값과 최솟값을 구하는 프로그램을 작성하세요.

▶ 음수, 0, 양수 등 모든 정수 입력 가능



```
C:\Windows\system32\cmd.exe
1번째 숫자 입력 : 4
2번째 숫자 입력 : 30
3번째 숫자 입력 : 19
4번째 숫자 입력 : 15
5번째 숫자 입력 : 34
6번째 숫자 입력 : 25
7번째 숫자 입력 : 9
8번째 숫자 입력 : 11
9번째 숫자 입력 : 25
10번째 숫자 입력 : 20
max : 34 , min : 4
계속하려면 아무 키나 누르십시오 . . .
```

실습예제 02

▶ 학생 5명의 시험점수 (score)를 입력 받아 총점(total)과 평균(avg)을 구하는 프로그램을 작성하시오.

▶ 0 ~ 100사이의 정수만 입력 받을 수 있으며 범위 이외의 값을 입력하면 재입력을 요구

▶ 5명의 학생 점수를 모두 입력하면 총점과 평균을 출력

▷ 정수형 데이터의 연산 결과를 실수형으로 출력하는 방법 - 강제형 변환을 이용

▷ 평균은 소수점 두 번째 자리까지 구하시오.(세 번째 자리에서 반올림)

배열의 복사

▶ for문을 이용한 배열의 복사

▶ 앞서 학습한 내용을 응용

```
int array1[5] = {1,2,3,4,5};  
int array2[5] = {0};  
int i;  
  
for(i = 0; i<sizeof(array2)/sizeof(array2[0]); i++)  
    printf("array[%d] : %d\n", i, array2[i]);  
  
for(i = 0; i<sizeof(array1)/sizeof(array1[0]); i++)  
    array2[i] = array1[i];  
  
for(i = 0; i<sizeof(array2)/sizeof(array2[0]); i++)  
    printf("array[%d] : %d\n", i, array2[i]);
```

```
array[0] : 0  
array[1] : 0  
array[2] : 0  
array[3] : 0  
array[4] : 0  
array[0] : 1  
array[1] : 2  
array[2] : 3  
array[3] : 4  
array[4] : 5
```

Q & A
