Kumoh National Institute of Technology

C Programming

Ch.6-1 1차원 배열

컴퓨터공학과

C프로그래밍

Contents

1. 배열의 이해와 배열의 선언 및 초기화
2. 배열을 이용한 문자열 변수의 표현
3. 배열을 매개변수로 전달해 보자 (추가)

- 추가 : 교재 외 추가 자료

배열이란 무엇인가?

- 학생 100명에 대한 점수를 저장해야 한다면?

```
int main(void)
{
    int score1, score2, score3, ???? ;
}
```

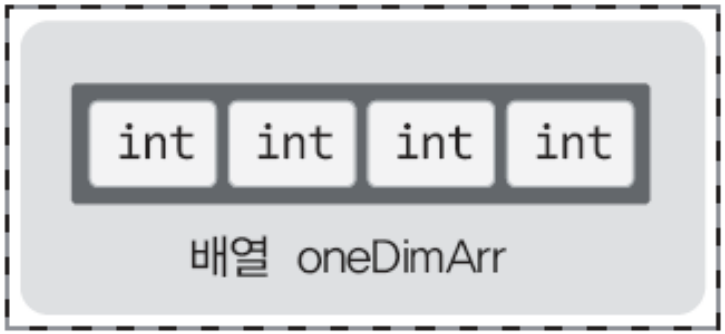
  - 변수 100개를 간단하게 표현하는 방법 → 배열
- 배열
  - 배열을 사용하면 하나의 선언을 통해 다수의 변수 선언 가능
  - 많은 양의 데이터를 일괄적으로 처리할 때 유용
  - 1차원 배열, 2차원 배열, 3차원 배열, ...

1차원 배열 선언에 필요한 세 가지

- 1차원 배열의 선언

```
int oneDimArr[4];
```

int	배열을 이루는 요소(변수)의 자료형
oneDimArr	배열의 이름
[4]	배열의 길이

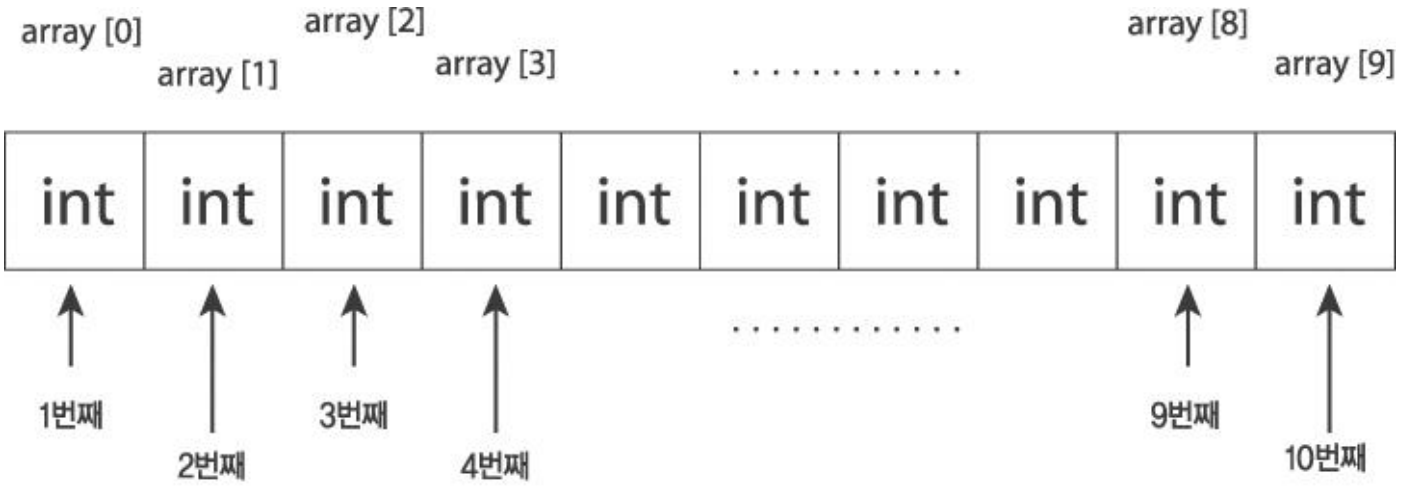


생성되는 배열의 형태

- 배열 요소 자료형 : 배열을 구성하는 변수들의 자료형
  - 변수들의 자료형은 모두 동일
- 배열의 이름 : 배열의 원소에 접근할 때 사용하는 이름
- 배열의 길이 : 배열을 구성하는 변수의 개수
  - 상수만 사용 가능

1차원 배열의 각 변수에 대한 접근

- int array[10];인 경우
  - 각 변수에 대한 표현 : array[index]
  - index는 0부터 시작 : array[0], array[1], ..., ary[9]




```
array[0] = 10; // 첫 번째 변수에 10 저장
int i = 3;     // index 값으로 변수 사용 가능
array[i] = 20; // (i + 1)번째 변수에 20 저장
```

int : int형 정수 저장을 위한 4바이트 메모리 블록

1차원 배열의 사용 예

```
int main(void)
{
    int arr[5];
    int sum = 0, i;

    arr[0] = 10, arr[1] = 20, arr[2] = 30, arr[3] = 40, arr[4] = 50;

    for (i = 0; i < 5; i++)
        sum += arr[i];

    printf("배열요소에 저장된 값의 합: %d \n", sum);
}
```

```
배열요소에 저장된 값의 합: 150
```

arr[0], ary[1], ..., arr[4] 각각이 int 변수이다.
int 변수 사용 방법은 이미 다 알고 있다!
한 가지 더, ary[i]와 같이 변수 index를 통해 접근할 수 있으니 매우 편하다.

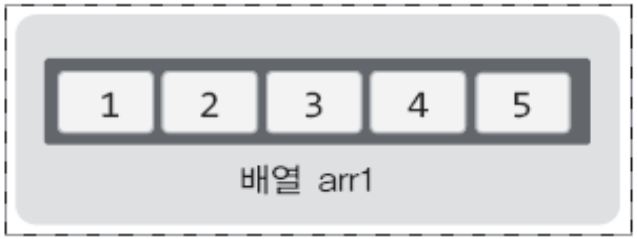
배열 선언과 동시에 초기화하기

- 배열 선언 시 초기화 : { 값1, 값2, ... }

```
int main(void)
{
    int arr1[5] = { 1, 2, 3, 4, 5 };
    int arr2[] = { 1, 2, 3, 4, 5, 6, 7 };
    int arr3[5] = { 1, 2 };
}
```

```
int arr1[5]={1, 2, 3, 4, 5};
```

초기화 결과



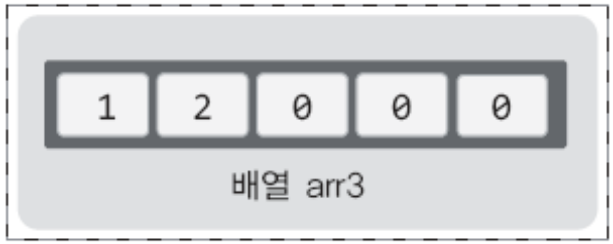

```
int arr2[ ]={1, 2, 3, 4, 5, 6, 7};
```

컴파일러가 배열의 길이정보 채움

```
int arr2[7]={1, 2, 3, 4, 5, 6, 7};
```

```
int arr3[5]={1, 2};
```

부족한 부분 0으로 채워짐



배열 선언 시 초기화의 예

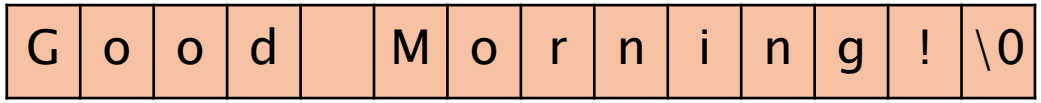
```
int main(void)
{
    int arr1[5] = { 1, 2, 3, 4, 5 };
    int arr2[] = { 1, 2, 3, 4, 5, 6, 7 };
    int arr3[5] = { 1, 2 };
    int ar1Len, ar2Len, ar3Len, i;

    printf("배열 arr1의 크기: %d \n", sizeof(arr1));
    printf("배열 arr2의 크기: %d \n", sizeof(arr2));
    printf("배열 arr3의 크기: %d \n", sizeof(arr3));

    ar1Len = sizeof(arr1) / sizeof(int);   // 배열 arr1의 길이 계산
    ar2Len = sizeof(arr2) / sizeof(int);   // 배열 arr2의 길이 계산
    ar3Len = sizeof(arr3) / sizeof(int);   // 배열 arr3의 길이 계산

    for (i = 0; i < ar1Len; i++)
        printf("%d ", arr1[i]);
    printf("\n");

    for (i = 0; i < ar2Len; i++)
        printf("%d ", arr2[i]);
    printf("\n");

    for (i = 0; i < ar3Len; i++)
        printf("%d ", arr3[i]);
    printf("\n");
}
```

sizeof(배열이름)
: 배열 전체의 바이트 수

```
배열 arr1의 크기: 20
배열 arr2의 크기: 28
배열 arr3의 크기: 20
1 2 3 4 5
1 2 3 4 5 6 7
1 2 0 0 0
```

char형 배열을 이용한 문자열 저장

- 문자열 상수 : 문자열이면서 상수의 특징을 가짐
  - “Good Morning!”
  - 문자(char)의 집합
  - 항상 널 문자(\0)로 끝남



- 문자열 변수 : char 배열로 표현

```
int main(void)
{
    char str[] = "Good morning!";
    printf("배열 str의 크기: %d \n", sizeof(str));
    printf("널 문자 문자형 출력: %c \n", str[13]);
    printf("널 문자 정수형 출력: %d \n", str[13]);

    str[12] = '?';
    printf("문자열 출력: %s \n", str);
}
```

```
배열 str의 크기: 14
널 문자 문자형 출력:
널 문자 정수형 출력: 0
문자열 출력: Good morning?
```

널 문자와 공백 문자의 비교

- 널 문자('\0') Vs. 공백 문자(' ')
  - 널 문자는 공백 문자와 다름
    - 널 문자의 아스키 코드값 : 0
    - 공백 문자의 아스키 코드값 : 32 // 키보드 스페이스바

```
int main(void)
{
    char nu = '\0';      // 널 문자
    char sp = ' ';       // 공백 문자

    printf("%d %d\n", nu, sp);
}
```

```
0 32
```

널 문자는 문자열의 끝과 같이 특수한 용도로 사용됨

scanf 함수를 이용한 문자열의 입력

- char str[50]; scanf("%s", str);
  - %s 사용 : 공백 문자가 나타나기 이전까지의 문자열 저장
  - 배열 변수명 앞에 &를 붙이지 않음 → 왜? (포인터 – 교재 7주차)
  - 널 문자가 마지막에 자동으로 들어가게 됨

```
int main(void)
{
    char str[50];   // 49자까지 저장 가능(+ 널문자)
    int idx = 0;

    printf("문자열 입력: ");
    scanf("%s", str);
    printf("입력 받은 문자열: %s \n", str);

    printf("문자 단위 출력: ");
    while (str[idx] != '\0')    // 이렇게 문자열의 마지막 감지
    {
        printf("%c", str[idx]); // 각 요소는 char
        idx++;
    }
    printf("\n");
}
```

```
문자열 입력: Hello C World
입력 받은 문자열: Hello
문자 단위 출력: Hello
```

한 줄 단위의 문자열 입력은?
- gets, fgets 함수 사용
- 이후 관련 주제에서 설명

문자열의 끝에 널 문자가 필요한 이유

- 유효한 문자열의 끝 판단
- 실제 문자열과 쓰레기값의 경계 의미
- printf 함수는 널 문자를 통해 출력의 범위를 알게 됨

```
int main(void)
{
    char str[50] = "I like C programming";
    printf("string: %s \n", str);

    str[8] = '\0';      // 9번째 요소에 널 문자 저장
    printf("string: %s \n", str);

    str[6] = '\0';      // 7번째 요소에 널 문자 저장
    printf("string: %s \n", str);

    str[1] = '\0';      // 2번째 요소에 널 문자 저장
    printf("string: %s \n", str);
}
```

```
string: I like C programming
string: I like C
string: I like
string: I
```

문자열과 char형 배열의 차이 (널 문자 유무)

- char arr1[] = “abc”; // 문자열
- char arr2[] = { ‘a’, ‘b’, ‘c’ }; // 배열
- char arr3[] = { ‘a’, ‘b’, ‘c’, ‘\0’ }; // 문자열

배열도 매개변수로 전달할 수 있을까?

- 배열 요소 값들의 합을 구해 반환하는 함수 Sum

```
int Sum(int arr[5])
{
    int result = 0;

    for (int i = 0; i < 5; i++)
        result += arr[i];

    return result;
}

int main(void)
{
    int arr[5] = { 1, 2, 3, 4, 5 };
    int result = Sum(arr);

    printf("합계 : %d \n", result);
}
```

그냥 요소 5개인 배열로 받았다.

각 요소의 합을 구함

실행이 잘 된다!
- 항상 요소 5개인 배열에만 적용됨?
→ 요소의 개수가 다른 경우에는?

배열 전달

합계 : 15

요소의 개수가 다른 경우의 매개변수 전달

- 요소의 개수도 함께 전달하자!

```
int Sum(int arr[], int count)
{
    int result = 0;

    for (int i = 0; i < count; i++)
        result += arr[i];

    return result;
}

int main(void)
{
    int result;
    int arr1[5] = { 1, 2, 3, 4, 5 };
    result = Sum(arr1, 5);
    printf("arr1 합계 : %d \n", result);

    int arr2[3] = { 10, 20, 30 };
    result = Sum(arr2, 3);
    printf("arr2 합계 : %d \n", result);
}
```

요소 개수는 비워두고, 요소 개수를 매개변수로 받자!

잘 되는데 이렇게 하면 되는건가?
일단 이렇게 사용해도 된다.
- 완전한 이해 → 포인터 (7주차)

배열과 요소 개수 전달

```
arr1 합계 : 15
arr2 합계 : 60
```

이번 장에서 배운 것

- 배열은 동일한 타입의 변수 여러 개를 한 번에 선언하는 방법이다.
- 배열을 사용하면 인덱스를 통해 많은 데이터를 쉽게 처리할 수 있다.
- 배열을 선언함과 동시에 초기값을 설정할 수 있다.
- 문자열은 문자들의 집합으로 마지막에 널 문자(₩0)를 가지고 있어야 한다.
- 문자열은 char 배열을 통해 저장할 수 있다.
- scanf 함수와 printf 함수를 통해 문자열을 저장하고 출력하기 위해서는 서식 문자 %s를 사용한다.
- 배열도 함수의 매개변수로 전달이 가능한 것처럼 보인다. 그러나 이에 대해 완벽하게 이해하기 위해서는 배열의 특징과 포인터에 대한 이해가 필요하다.