1차원 배열

- 배열의 선언과 접근, 문자열 변수 -

성공회대학교 IT융합자율학부
소프트웨어공학전공
홍 성 준

배열이란 무엇인가?

◎ 배열 (array)

- 연관된 데이터를 모아 통으로 관리하기 위해 사용하는 데이터 구조
- 여러 개의 데이터를 하나의 변수에 저장하기 위해 사용

```
int main(void)
{
    int floor101, floor102, floor103, floor104;   // 1층 101호부터 104호까지
    int floor201, floor202, floor203, floor204;   // 2층 201호부터 204호까지
    int floor301, floor302, floor303, floor304;   // 3층 301호부터 304호까지
    . . . .
}
```

- 선언하는 방식에 따라 1차원 배열, 2차원 배열 등 다차원으로 선언이 가능

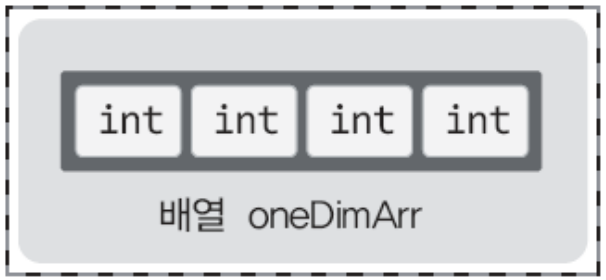
배열의 선언

◎ 1차원 배열의 선언

- 배열의 자료형, 배열의 이름, 배열의 길이(상수)

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

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




```
int arr1[7];        // 길이가 7인 int형 1차원 배열 arr1
float arr2[10];     // 길이가 10인 float형 1차원 배열 arr2
double arr3[12];    // 길이가 12인 double형 1차원 배열 arr3
```

배열의 접근

◎ 1차원 배열의 접근

- 배열의 이름과 [ ] 연산자에 위치 정보(index)를 사용하여 배열 요소에 접근
- C 언어에서 배열의 인덱스는 0에서부터 시작

```
arr[0]=10;     // 배열 arr의 첫 번째 요소에 10을 저장해라!
arr[1]=12;     // 배열 arr의 두 번째 요소에 12를 저장해라!
arr[2]=25;     // 배열 arr의 세 번째 요소에 25를 저장해라!
```

1차원 배열 접근의 예

일반화

```
arr[idx]=20; → "배열 arr의 idx+1번째 요소에 20을 저장해라!"
```

배열의 접근

◎ ArrayAccess.c

```
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);
    return 0;
}
```

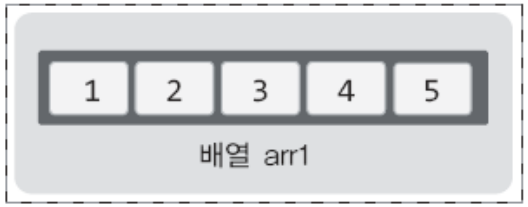
```
배열요소에 저장된 값의 합: 150
```

배열의 선언과 초기화

◎ 리스트를 이용한 초기화

- 배열 선언과 동시에 { }를 이용하여 배열의 원소를 나열하면 순서대로 저장됨

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

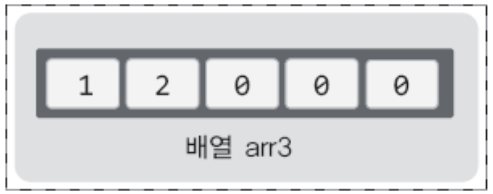


- 배열의 길이 정보 생략 시

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

- 리스트 원소의 개수가 배열의 길이보다 작을 시

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



배열의 선언과 초기화

◎ ArrayInt.c (feat. 배열의 길이 계산하기)

```
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");
    return 0;
}
```

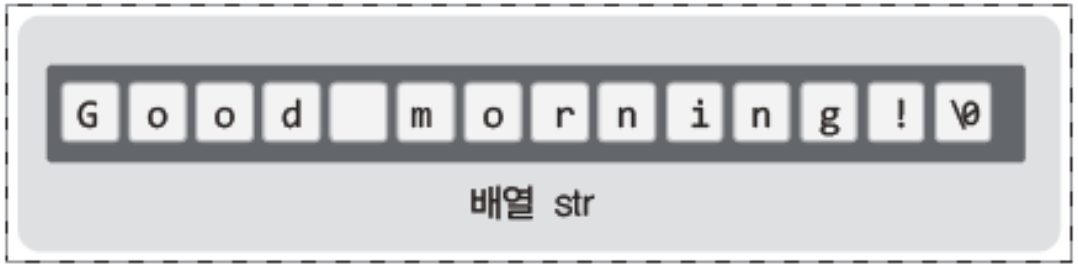
```
배열 arr1의 크기: 20
배열 arr2의 크기: 28
배열 arr3의 크기: 20
1 2 3 4 5
1 2 3 4 5 6 7
1 2 0 0 0
```

배열을 이용한 문자열 변수의 표현

◎ char 형 배열의 문자열 저장

```
char str[14]="Good morning!";
```

```
char str[  ]="Good morning!";
```



- 문자열의 끝엔 문자열의 끝을 알리는 보이지 않는 널(null, '₩0') 문자가 존재
- 문자 배열과 문자열을 구분하기 위해 널 문자가 필요

```
char arr1[ ] = {'H', 'i', '~'};
char arr2[ ] = {'H', 'i', '~', '\0'};
```

배열을 이용한 문자열 변수의 표현

◎ ArrayString.c


```
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]='?';    // 배열 str에 저장된 문자열 데이터는 변경 가능!
    printf("문자열 출력: %s \n", str);
    return 0;
}
```

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

배열을 이용한 문자열 변수의 표현

◎ 널 문자(null, '₩0')와 공백 문자(space, ' ')의 비교

- 널 문자를 %c를 이용해서 출력하면 아무것도 출력되지 않지만, 널 문자가 공백을 의미하는 것이 아님

```
int main(void)
{
	char nu = '\0';    // 널 문자 저장
	char sp = ' ';     // 공백 문자 저장
	printf("%d %d", nu, sp);    // 0과 32 출력
	return 0;
}
```

- 널 문자의 ASCII 코드 값은 0 이며, 공백 문자의 ASCII 코드 값은 32
- nu = 0; // nu = '0';

◎ 문자열 "", " "는 어떻게 다를까?

문자열의 끝에 널 문자가 필요한 이유

◎ StartEndString.c

```
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);
    return 0;
}
```

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

- 문자열의 시작위치는 명시적이지만 문자열의 끝을 판단할 수 없기 때문에 문자열의 끝을 알리는 널 문자가 필요
- printf() 함수도 배열 str의 시작 위치를 기준으로 널 문자를 만날 때까지 출력을 진행

scanf() 함수를 이용한 문자열의 입력

◎ ReadString.c

```
int main(void)
{
    char str[50];
    int idx=0;

    printf("문자열 입력: ");
    scanf("%s", str);  // 문자열을 입력 받아서 배열 str에 저장!
    printf("입력 받은 문자열: %s \n", str);

    printf("문자 단위 출력: ");
    while(str[idx] != '\0')
    {
        printf("%c", str[idx]);
        idx++;
    }
    printf("\n");
    return 0;
}
```

```
문자열 입력: Simple
입력 받은 문자열: Simple
문자 단위 출력: Simple
```

scanf() 함수를 이용한 문자열의 입력

◎ ReadString.c

```
문자열 입력: He is my friend
입력 받은 문자열: He
문자 단위 출력: He
```

- scanf() 함수는 공백을 기준으로 입력 데이터를 구분
  => 공백을 포함하는 문자열은 입력 받을 수 없음

```c
int main(void)
{
    char str[50];
    int idx=0;

    printf("문자열 입력: ");
    scanf("%s", str);  // 문자열을 입력 받아서 배열 str에 저장!
    printf("입력 받은 문자열: %s \n", str);

    printf("문자 단위 출력: ");
    while(str[idx] != '\0')
    {
        printf("%c", str[idx]);
        idx++;
    }
    printf("\n");
    return 0;
}
```