2010-1학기 프로그래밍입문(1)

chapter 06-3 참고자료

배열과포인터

박 종 혁

Tel: 970-6702
Email: jhpark1@snut.ac.kr

출처: 뇌를 자극하는 C프로그래밍, 한빛미디어

❑ 포인터로 배열 다루기

- 배열의 모든 값을 출력하는 함수를 만들 때 배열요소의 값을 일일이 전달인자로 주는 것은 한계가 있다.

배열의 선언

```
int ary[5] = {10, 20, 30, 40, 50};
```

함수의 호출

```
ary_prn(ary[0], ary[1], ary[2], ary[3], ary[4]);
```

모든 배열요소를 일일이 전달인자로 줘야 한다.

함수의 정의

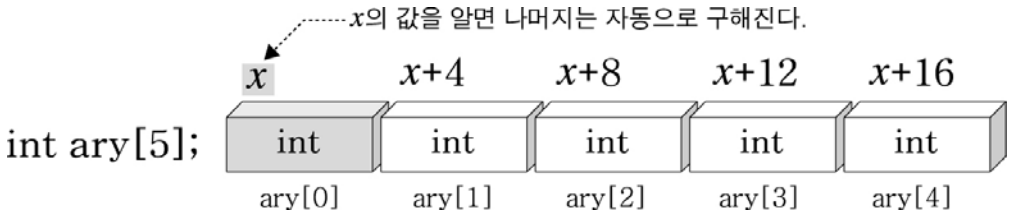
```
void ary_prn(int a, int b, int c, int d, int e)
{
    printf("%d, %d, %d, %d, %d₩n", a, b, c, d, e);
}
```

매개변수도 배열요소의 개수만큼 있어야 한다!

- 포인터를 사용하면 배열요소의 값을 간단히 처리할 수 있다.

▶ 포인터로 배열요소를 참조하자.

- 배열은 첫번째 배열요소의 포인터만 알면 나머지 배열요소의 포인터도 쉽게 알 수 있다.



- 포인터에 정수값을 더할 때는 포인터가 가리키는 자료형의 크기를 곱해서 더해준다. 예를 들어 4를 더하면 마지막 배열요소의 포인터가 구해진다.

포인터 + 정수값

X의 값이 36번지라고 할 때

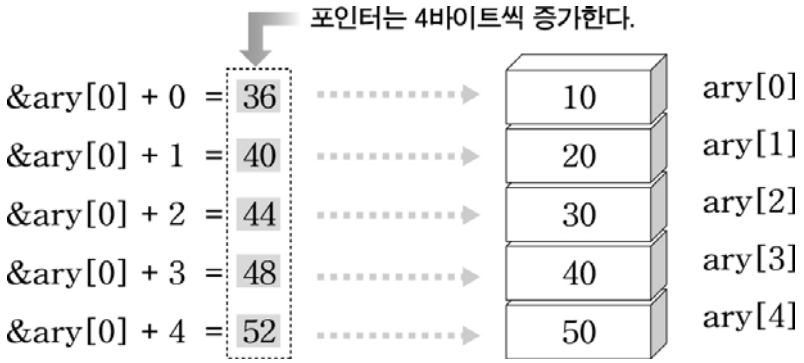
포인터 + (정수값 * 포인터가 가리키는 자료형의 크기)

&ary[0] + 4 = &ary[0] + (4 * sizeof(int)) = 36 + 16 = 52번지

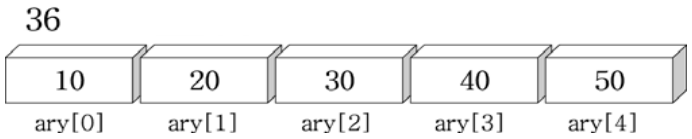
▶ 포인터로 배열요소를 참조하자.

- 모든 배열요소의 포인터는 첫번째 배열요소의 포인터에 정수값을 차례로 더하면 구해진다.

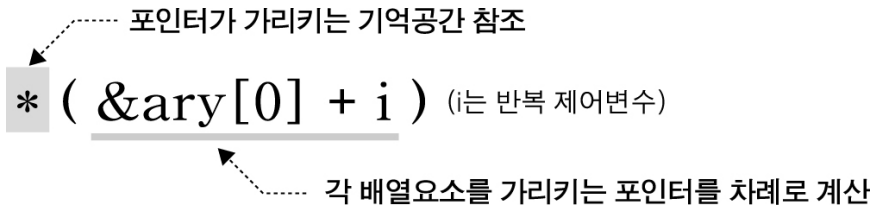
int ary[5] = {10, 20, 30, 40, 50};



첫번째 배열요소의 포인터 값이 36번지 일 때



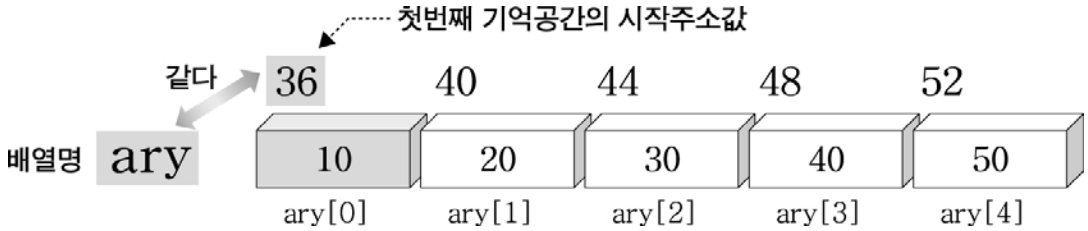
- 각 배열요소의 포인터에 참조연산자를 사용하면 모든 값을 참조할 수 있다.




```
for(i=0; i<5; i++){
    printf("%d₩n", *(&ary[0]+i));
}
```

▶ 배열명은 포인터이다!

- 배열명은 첫 번째 배열요소를 가리키는 포인터를 기호화한 것이다.



- 따라서 배열명으로 주소값을 계산하여 모든 배열요소를 참조할 수 있으며 의미상 이해하기 쉽게 배열표현을 주로 사용하는 것이다.

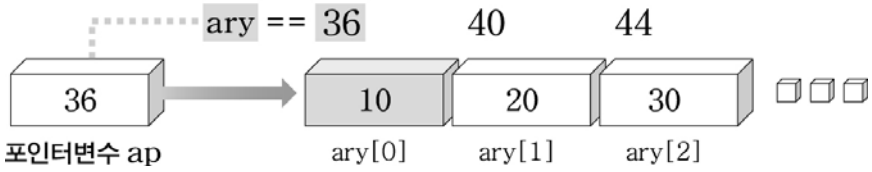
```
for(i=0; i<5; i++){
    printf("%d₩n", *(ary+i));
}
```

배열표현		포인터표현
ary[0]		*(ary+0)
ary[1]		*(ary+1)
ary[2]	= =	*(ary+2)
ary[3]		*(ary+3)
ary[4]		*(ary+4)

▶ 포인터변수로 배열요소를 참조하자.

- 배열명을 포인터변수에 저장하면 포인터변수도 배열명처럼 사용할 수 있다. 이 때 포인터변수는 첫번째 배열요소를 가리킨다.

```
int *ap = ary;  // 배열명을 포인터변수에 저장
```



* (ap + 1) → 40번지에 있는 배열요소를 참조한다.

36 + (1 * sizeof(int)) = 36 + 4 = 40번지

```
int ary[5]={10,20,30,40,50};
int *ap=ary;
int i;

for(i=0; i<5; i++){
    printf("%5d", *(ap+i));  // ap[i]도 가능
}
```

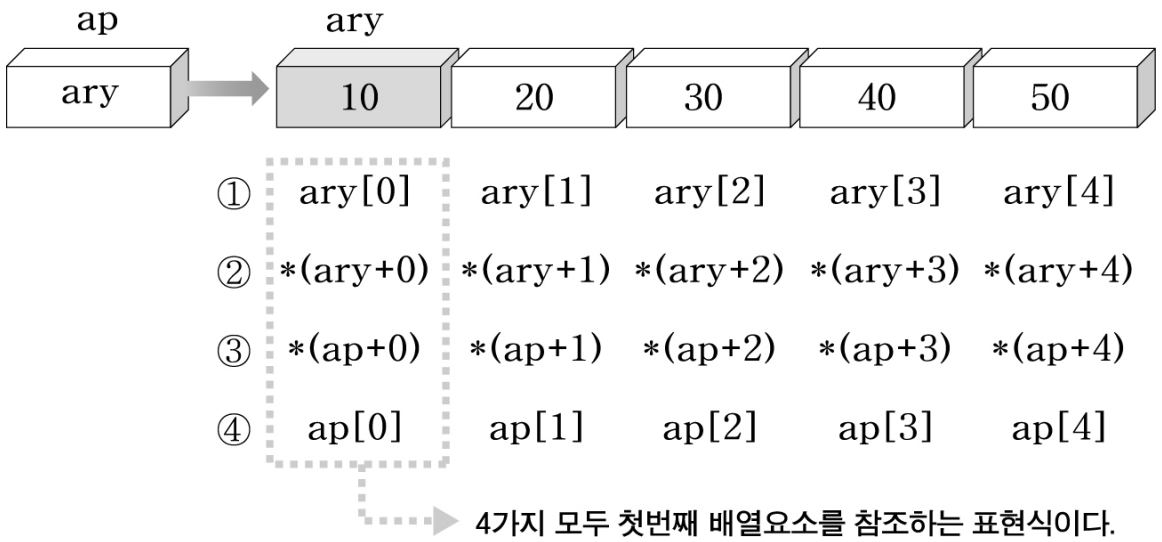
배열표현		포인터표현
ap[0]	= =	*(ap+0)
ap[1]		*(ap+1)
ap[2]		*(ap+2)
ap[3]		*(ap+3)
ap[4]		*(ap+4)

▶ 포인터를 사용한 배열요소의 참조 정리

- 포인터(변수)로 배열요소를 참조하는 방법은 다음과 같다.

① 배열명을 사용한 배열표현
② 배열명을 사용한 포인터표현
③ 배열명을 저장한 포인터변수를 사용한 포인터표현
④ 배열명을 저장한 포인터변수를 사용한 배열표현

```
int ary[5] = {10, 20, 30, 40, 50};
int *ap=ary;
```



▶ 배열명은 포인터변수가 아니다.

- 배열명은 포인터상수이므로 자신의 값을 바꿀 수 없다.

```
int ary[5] = {10, 20, 30, 40, 50};
```

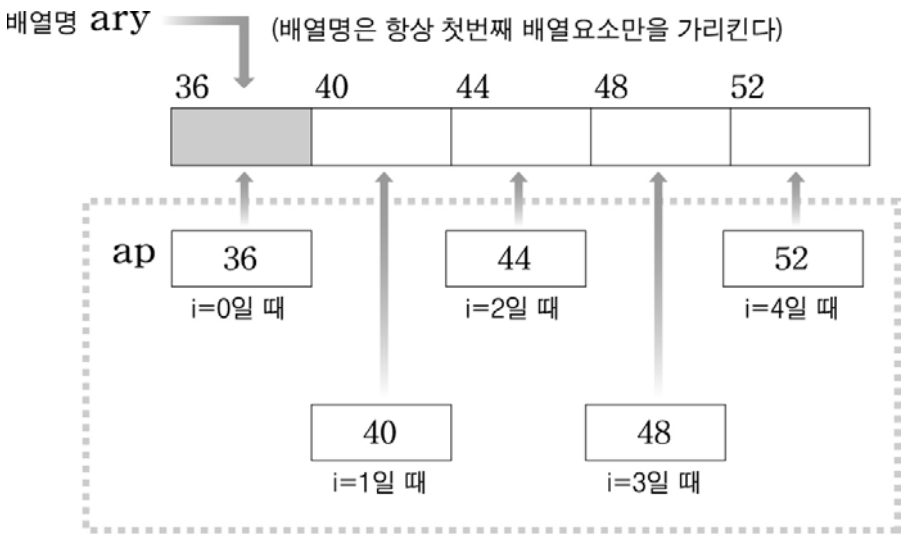
```
ary = ary + 2;
ary++;
```

"배열명은 변수가 아니므로 자신의 값을 바꿀 수 없다"

- 포인터변수는 기억공간이므로 자신의 값을 바꿀 수 있다.

```
int ary[5]={10,20,30,40,50};
int *ap=ary;
int i;

for(i=0; i<5; i++){
    printf("%5d", *ap);
    ap++;
}
```



(포인터변수는 자신의 값이 바뀌면서 다음 배열요소를 가리킨다.)

❑ 배열을 처리하는 함수

- 배열의 모든 요소는 포인터로 참조할 수 있으므로 배열을 처리하는 함수에는 그 시작위치인 배열명을 전달인자로 준다.
- 배열의 값을 출력하는 함수
- 배열에 값을 입력하는 함수
- 배열의 평균을 구하는 함수

▶ 배열의 값을 출력하는 함수

- 배열명을 전달인자로 받으므로 매개변수는 포인터변수를 선언한다.

```
#include <stdio.h>
void ary_prn(int *);                // 함수의 선언
int main()
{
    int ary[5]={10,20,30,40,50};    // 배열의 선언과 초기화
    ary_prn(ary);                   // 배열명을 전달인자로 주고 호출한다.
    return 0;
}
void ary_prn(int *ap)               // 배열명을 저장할 포인터변수 선언
{
    int i;
    for(i=0; i<5; i++){
        printf("%5d", ap[i]);       // 포인터변수를 마치 배열명처럼 사용한다.
    }
}
```

▶ 배열의 값을 출력하는 함수

- 포인터변수는 배열명이 아니므로 sizeof연산시 포인터변수의 크기만 계산된다. 따라서 포인터변수로 배열요소의 개수를 구할 수 없다.

```
for(i=0; i<sizeof(ap)/sizeof(ap[0]); i++){    // sizeof(ap)는 포인터변수의 크기만 계산
    printf("%5d", ap[i]);
}
```

- 배열의 크기가 바뀌어도 출력할 수 있는 함수를 만들 때는 배열요소의 개수를 전달인자로 받아야 한다.

호출할 때 → `ary_prn ( ary, sizeof(ary)/sizeof(ary[0]) );`

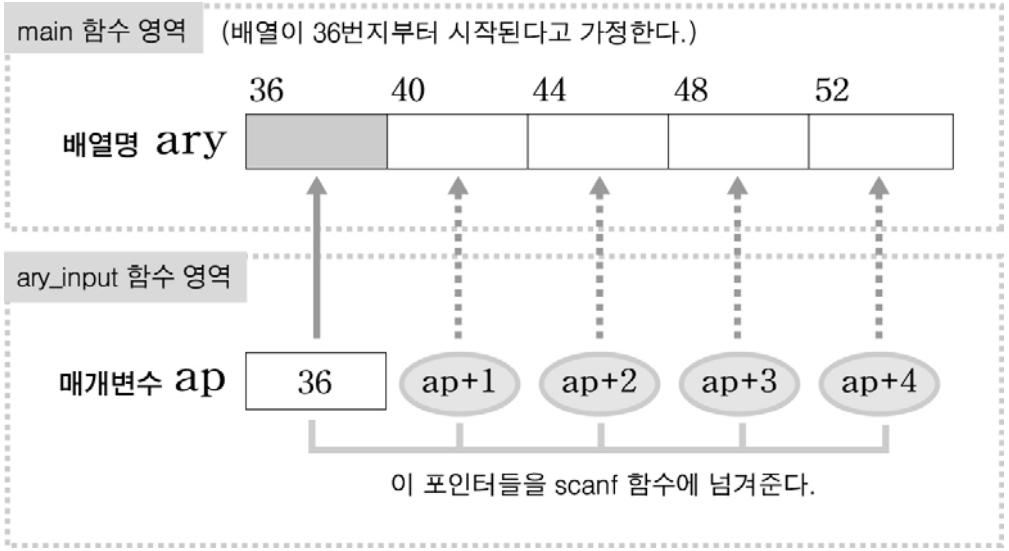
ary: 배열명 / sizeof(ary)/sizeof(ary[0]): 배열요소의 개수

```
void ary_prn(int *ap, int an)      // 배열요소의 개수를 받을 매개변수를 추가한다.
{
    int i;
    for(i=0; i<an; i++){           // 배열요소의 개수만큼 반복한다.
        printf("%5d", ap[i]);
    }
}
```

▶ 배열에 값을 입력하는 함수

- 배열에 값을 입력할 때는 scanf함수에 각 배열요소의 포인터만을 전달인자로 준다(즉, 참조연산자를 사용하지 않는다).

```
void ary_input(int *ap)          // 배열명을 저장할 포인터변수 선언
{
    int i;
    for(i=0; i<5; i++){          // 배열요소의 개수만큼 반복한다.
        scanf("%d", ap+i);       // 각 배열요소의 포인터를 구해서 전달인자로 준다.
    }
}
```



▶ 배열의 평균을 구하는 함수

- 모든 배열요소의 평균을 구해서 리턴하는 함수를 만들자.

```
#include <stdio.h>
double ary_avg(int *);                  // 함수의 선언
int main()
{
    int ary[5]={75,80,92,88,98};
    double res;                         // 리턴값을 저장할 변수
    res=ary_avg(ary);                   // 전달인자는 배열명, 리턴값은 res에 저장한다.
    printf("배열의 평균은 : %.2lf₩n", res);
    return 0;
}
double ary_avg(int *ap)                 // 매개변수는 포인터변수
{
    int i, tot=0;                       // 제어변수와 합을 저장할 변수
    double average;                     // 평균을 저장할 변수
    for(i=0; i<5; i++) tot+=ap[i];      // 배열요소의 개수만큼 반복하면서 tot에 누적한다.
    average=tot/5.0;                    // 평균 계산
    return average;                     // 계산된 평균값 리턴
}
```