컴퓨터 프로그래밍2

배열

장서윤 *pineai@cnu.ac.kr*

배열이란?

- 각 저장 공간을 이름과 첨자(index)로 구분
- 메모리에 저장 공간 한꺼번에 확보
- 사용할 때는 하나씩 떼어 쓰는 방식으로 구현

배열의 선언과 초기화

표 8-1 배열의 선언과 요소의 사용

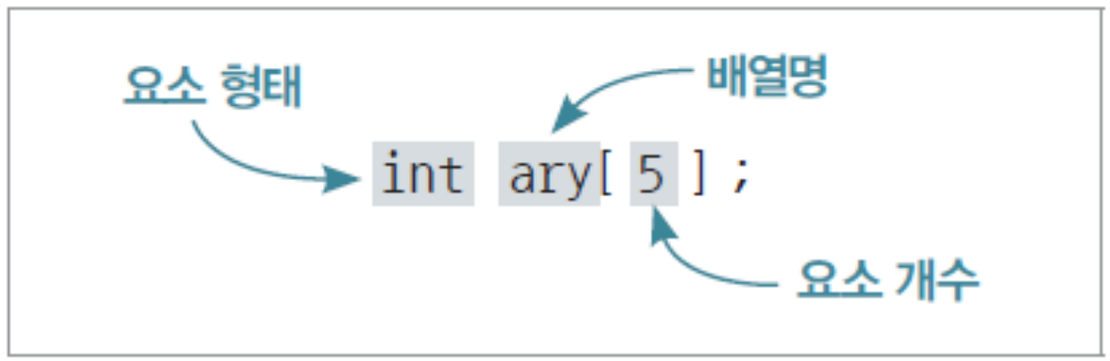
구분	사용 예	기능
배열 선언	int ary[5];	int형 변수 5개를 한 번에 확보한다.
요소 사용	ary[0], ary[1], ary[2], ary[3], ary[4],	배열 요소를 사용할 때는 첨자를 0부터 시작하여 '요소 수 – 1'까지 쓴다.
초기화	int ary[5] = {1, 2, 3, 4, 5};	초기화는 중괄호 안에 값을 나열한다.

배열 선언과 배열 요소 사용

- 배열은 사용하기 전에 먼저 선언

  - 변수 선언과 마찬가지로 메모리에 저장 공간 확보
  - 저장 공간의 개수와 관계없이 이름은 하나만 사용
  - 요소의 자료형에 이름 붙이고 필요한 요소 수 표시

- ex) int형의 요소가 5개인 배열 선언하는 예



배열 선언과 배열 요소 사용

실행 결과

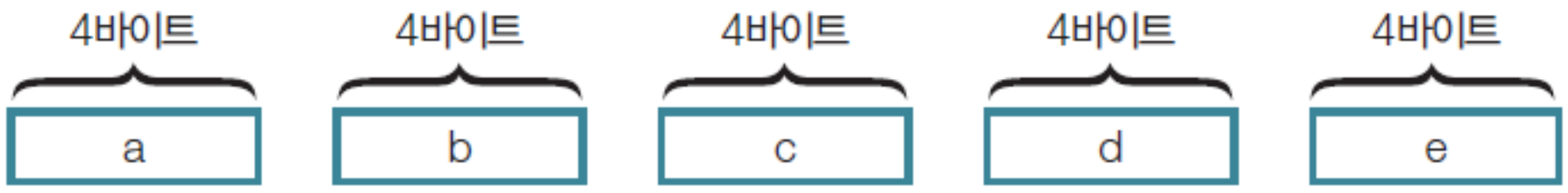
```
50
30
50
-858993460
```

예제 8-1 다섯 명의 나이를 저장할 배열

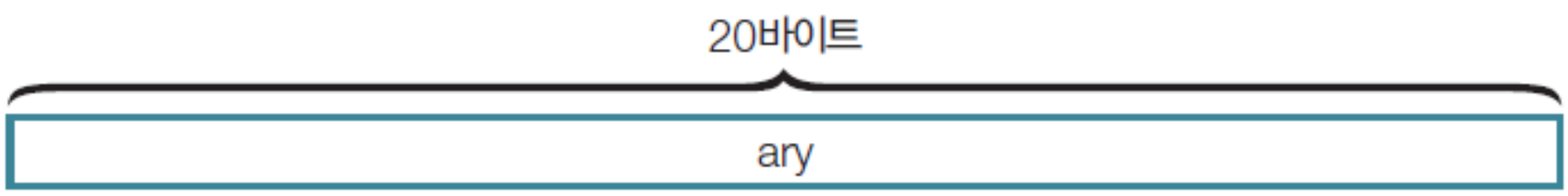
```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int ary[5];                     // int형 요소 5개의 배열 선언
6.
7.      ary[0] = 10;                    // 첫 번째 배열 요소에 10 대입
8.      ary[1] = 20;                    // 두 번째 배열 요소에 20 대입
9.      ary[2] = ary[0] + ary[1];       // 첫 번째와 두 번째 요소를 더해 세 번째 저장
10.     scanf("%d", &ary[3]);           // 키보드로 네 번째 요소에 입력
11.
12.     printf("%d\n", ary[2]);         // 세 번째 배열 요소 출력
13.     printf("%d\n", ary[3]);
14.     printf("%d\n", ary[4]);         // 마지막 배열 요소는 쓰레기값
15.
16.     return 0;
17. }
```

배열 선언과 배열 요소 사용

```
int a, b, c, d, e;      // 각 공간에 이름이 있다.
```




```
int ary[5];         // 메모리에 공간이 연속적으로 할당되며 이름은 하나다.
```

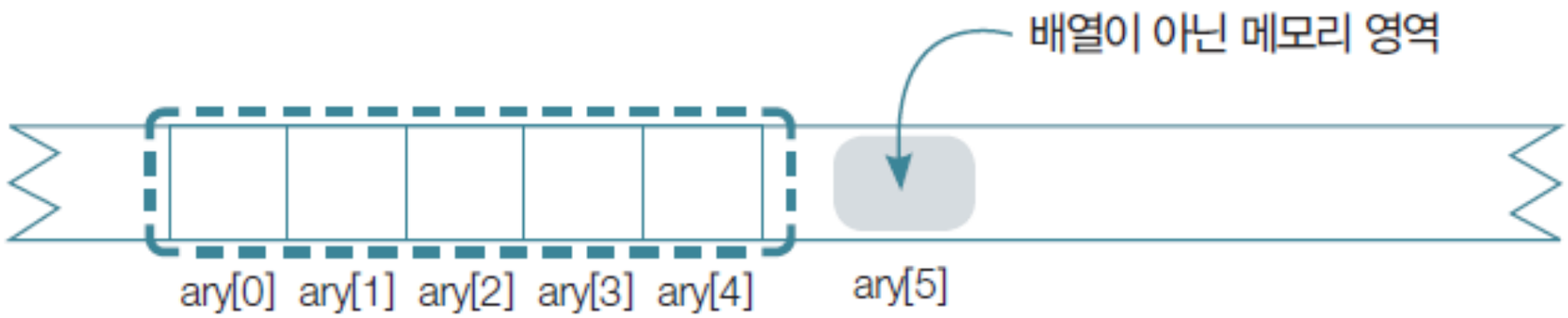


배열 선언과 배열 요소 사용

- 다섯 명의 나이를 저장하는 배열 예제 풀이
  - 배열 요소(element)
    - 배열의 나누어진 조각
    - int형 변수 크기는 4바이트 * 5개 연속 할당 = 총 20바이트
    - 각각의 배열 요소는 int형 변수와 똑같이 쓰임
    - 배열명에 첨자(index) 붙여 표현
    - 첨자는 0부터 시작
    - Ex) 세 번째 배열 요소 -> ary[2]

배열 선언과 배열 요소 사용

- 배열 요소(element) 의 사용
  - 배열의 첨자는 0부터 시작
    - 최대 '배열 요소 수 – 1'까지만 사용
    - 첨자의 사용범위 벗어나는 경우 할당된 메모리 벗어남
    - 사용범위 벗어나지 않도록 주의



배열 초기화

예제 8-2 배열의 초기화 방법

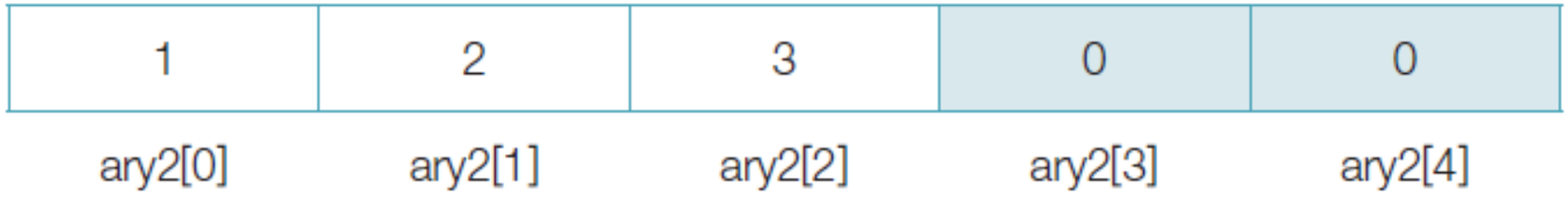
```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int ary1[5] = {1,2,3,4,5};                  // int형 배열 초기화
6.      int ary2[5] = {1,2,3};                      // 초깃값이 적은 경우
7.      int ary3[] = {1,2,3};                        // 배열 요소 개수가 생략된 경우
8.      double ary4[5] = {1.0, 2.1, 3.2, 4.3, 5.4};   // double형 배열 초기화
9.      char ary5[5] = {'a','p','p','l','e'};         // char형 배열 초기화
10.
11.     ary1[0] = 10;
12.     ary1[1] = 20;
13.     ary1[2] = 30;
14.     ary1[3] = 40;
15.     ary1[4] = 50;
16.
17.     return 0;
18. }
```

배열 초기화

- 최초 할당된 저장 공간에는 쓰레기값 존재
  - 배열이 선언과 동시에 원하는 값 갖도록 하려면 초기화

- 배열은 중괄호로 묶어 초기화
  - 반드시 선언과 동시에 초기화
  - 선언하고 난 후에 값 저장하려면 배열 요소에 하나씩 값 대입

- 왼쪽에서 차례로 초기화되고 남는 배열 요소는 모두 0으로 채움
  - 아무리 큰 배열도 쉽게 모든 요소를 0으로 초기화 가능




```
int ary[1000] = {0};   // 남는 배열 요소는 0으로 자동 초기화된다.
```

배열 초기화

- 선언 이후의 초기화
  - 배열 요소 사용해 일일이 값 바꿔야 함
  - 대입 연산으로 한 번에 값 바꾸는 것 불가능

```
int ary1[5] = {1, 2, 3, 4, 5};  // ary1을 선언과 동시에 중괄호로 초기화한다.
 ary1 = {10, 20, 30, 40, 50}; // 선언부가 아니므로 중괄호로 배열 요소값을 한꺼번에 바꿀 수 없다.
```

※ 주의 : 이렇게 대입할 수 없습니다!

배열과 반복문

- 배열은 같은 형태의 변수가 많이 필요할 때 쉽게 저장 공간 할당, 초기화 가능한 장점 가짐
- 모든 배열 요소를 일일이 하나씩 사용하는 것은 번거로움

```
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

예제 8-3 배열과 반복문을 사용한 성적처리 프로그램

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int score[5];                   // 다섯 과목의 성적을 입력할 int형 배열 선언
6.      int i;                          // 반복 제어 변수
7.      int tot = 0;                    // 총점을 누적할 변수
8.      double avg;                     // 평균을 저장할 변수
9.
10.     for(i = 0; i < 5; i++)          // i가 0부터 4까지 다섯 번 반복
11.     {
12.         scanf("%d", &score[i]);     // 각 배열 요소에 성적 입력
13.     }
14.
15.     for(i = 0; i < 5; i++)
16.     {
17.         tot += score[i];            // 성적을 누적하여 총점 계산
18.     }
19.     avg = tot / 5.0;                // 평균 계산
20.
21.     for(i = 0; i < 5; i++)
22.     {
23.         printf("%5d", score[i]);    // 성적 출력
24.     }
25.     printf("\n");
26.
27.     printf("평균 : %.1lf\n", avg);  // 평균 출력
28.
29.     return 0;
30. }
```

실행 결과

```
80 95 77 84 100
   80   95   77   84  100
평균 : 87.2
```

배열과 반복문

- 배열에 성적 입력하는 반복문
- 제어 변수 i가 0부터 하나씩 증가
  - 이 값을 배열 요소의 첨자로 사용
  - 반복 과정에서 모든 배열 요소에 값 입력 가능

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

i를 배열의 첨자로 활용

SIZEOF 연산자를 활용한 배열 처리

- 배열은 보통 많은 양의 데이터 처리
  - 반복문 사용 필수

- 배열 요소 수가 바뀌는 경우?
  - 배열 처리 반복문을 모두 수정해야 하는 부담
  - 배열 요소 수를 직접 계산하여 반복문에 사용하면 편리

- 배열 요소 수 = ‘sizeof (배열명) / sizeof (배열 요소)’

SIZEOF 연산자를 활용한 배열 처리 (CONT')

예제 8-4 sizeof 연산자를 사용한 배열

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int score[5];
6.      int i;
7.      int tot = 0;
8.      double avg;
9.      int cnt;                                    // 배열 요소 수를 저장할 변수
10.
11.     cnt = sizeof(score) / sizeof(score[0]);     // 배열 요소 수 계산
12.
13.     for(i = 0; i < cnt; i++)                    // 11행에서 계산한 cnt만큼 반복
14.     {
15.         scanf("%d", &score[i]);
16.     }
17.
```

SIZEOF 연산자를 활용한 배열 처리

```
18.     for(i = 0; i < cnt; i++)                    // 11행에서 계산한 cnt만큼 반복
19.     {
20.         tot += score[i];
21.     }
22.     avg = tot / (double)cnt;            // 총합을 cnt로 나누어 평균 계산
23.
24.     for(i = 0; i < cnt; i++)            // 11행에서 계산한 cnt만큼 반복
25.     {
26.         printf("%5d", score[i]);
27.     }
28.     printf("\n");
29.
30.     printf("평균 : %.1lf\n", avg);
31.
32.     return 0;
33. }
```

실행 결과

```
80 95 77 84 100
   80   95   77   84  100
평균 : 87.2
```

SIZEOF 연산자를 활용한 배열 처리

- sizeof 연산자에 배열명에 사용
  - 배열 전체의 크기를 바이트 단위로 계산
  - 배열 요소 하나의 크기로 나누면 배열 요소 수

```
cnt   =   sizeof(ary)  /  sizeof(ary[0]) ;
```

- cnt: 배열 요소 수 (5)
- sizeof(ary): 배열 전체 크기 (20바이트)
- sizeof(ary[0]): 배열 요소 하나의 크기 (4바이트)

문자를 저장하는 배열

- char형 배열
  - 문자열을 저장하는 변수와 같은 역할
  - char 배열 사용 방법

표 8-2 문자열 처리

구분	사용 예	기능
char형 배열 초기화	char str[80] = "apple";	char형 배열은 문자열로 초기화한다. 문자열의 끝에는 널문자가 있다.
문자열 대입	char str[80]; strcpy(str, "apple");	문자열 대입은 strcpy 함수를 쓴다. str 배열에 문자열 "apple" 저장

문자열 입출력	char str[80];
	scanf("%s", str);
	gets(str);
	printf("%s", str);
	puts(str);
		
		scanf 함수는 하나의 단어만 입력
		gets 함수는 한 줄 입력
		printf 함수는 문자열 출력
		puts 함수는 문자열 출력 후 줄 바꿈

문자를 저장하는 배열

- ❖ char형 배열의 선언과 초기화
  - ➤문자열은 문자의 연속
    - ➤문자열 저장할 때는 char형 배열 사용
    - ➤char형 배열 선언 시 주의점
      - ➤저장할 문자열의 길이보다 최소한 하나 이상 크게 배열 선언

➤Ex) 문자열 “apple”저장할 배열

➤배열 요소 수가 최소한 6 이상이어야

➤여분의 공간 필요한 이유

➤ 널문자(null character) 저장 위함

문자를 저장하는 배열

예제 8-5 문자열을 저장하는 char형 배열

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      char str[80] = "applejam";          // 문자열 초기화
6.
7.      printf("최초 문자열 : %s\n", str);     // 초기화 문자열 출력
8.      printf("문자열 입력 : ");
9.      scanf("%s", str);                   // 새로운 문자열 입력
10.     printf("입력 후 문자열 : %s\n", str);  // 입력된 문자열 출력
11.
12.     return 0;
13. }
```

실행 결과

```
최초 문자열 : applejam
문자열 입력 : grape
입력 후 문자열 : grape
```

문자를 저장하는 배열

❖ char형 배열의 선언과 초기화

➤ char형 배열은 배열 요소의 형태가 char일뿐 다른 배열 (int형 배열, double형 배열) 과 동일한 사용법

➤ 초기화 - 중괄호 사용하여 문자 차례로 나열

```
char str[80] = {'a', 'p', 'p', 'l', 'e', 'j', 'a', 'm'};   // 문자상수로 하나씩 초기화
char str[80] = "applejam";      // 문자열 상수로 한 번에 초기화
```

문자를 저장하는 배열

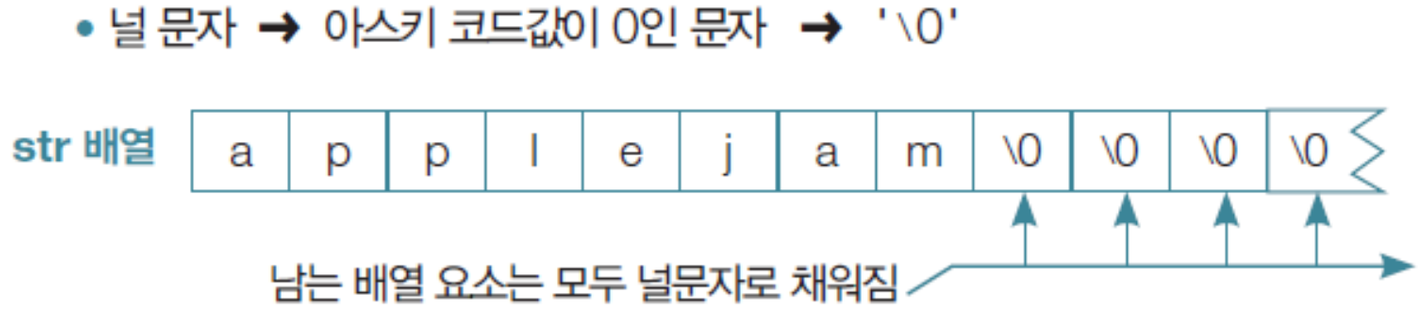
❖ char형 배열의 선언과 초기화

➤ 초기화한 문자들은 배열의 처음부터 차례로 저장 -> 문자열

➤ 남는 배열 요소에는 자동으로 0 채워짐

➤ char형 배열에 저장된 0 – 널문자

➤ 모든 문자는 아스키 코드값으로 저장 -> 널문자는 아스키 코드 값이 0인 문자



➤ printf 함수

➤ 배열의 크기와 관계없이 초기화된 문자열만 정확히 출력

➤ char형 배열에서 널문자가 나올 때까지만 출력

➤ 문자열 처리하는 모든 함수에 적용

문자를 저장하는 배열

❖ char형 배열의 선언과 초기화

➤char형 배열에 문자열 저장할 때

➤ 문자들을 하나씩 대입할 경우

➤ 마지막에 반드시 널문자 직접 채워야 한다

```
char str[80];                   // 배열 선언, 초기화하지 않음
str[0] = 'a';                   // 배열 요소에 직접 문자 대입
str[1] = 'p';
str[2] = 'p';
str[3] = 'l';
str[4] = 'e';
str[5] = '\0';                  // 마지막 문자 다음에 반드시 널문자 대입!
```

문자를 저장하는 배열

- char형 배열의 선언과 초기화
  - char형 배열 선언할 때 주의할 점
    - 문자열은 길이가 일정하지 않음
      - 예상 가능한 가장 긴 문자열도 저장할 수 있도록 선언
      - 널문자로 그 끝 표시하므로 최소한 널문자까지 저장할 수 있도록 선언

문자를 저장하는 배열

예제 8-6 널문자가 없는 문자열

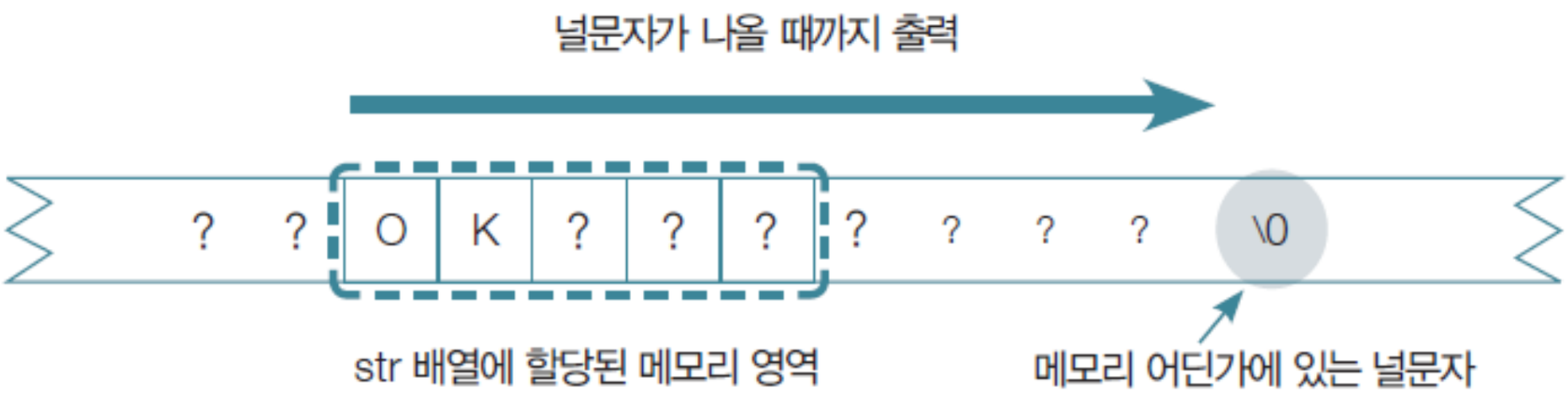
```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      char str[5];
6.
7.      str[0] = 'O';
8.      str[1] = 'K';
9.      printf("%s\n", str);
10.
11.     return 0;
12. }
```

실행 결과

```
OK倣倣倣倣倣?差□*      // 쓰레기값을 문자열로 출력
```

문자를 저장하는 배열

❖ char형 배열의 선언과 초기화



문자열 대입

- char형 배열 - 문자열 저장하는 변수의 역할
  - 초기화된 이후에도 새로운 문자열 저장 가능
  - 문자열의 길이가 다를 수 있음
    - 일반 변수처럼 대입 연산자 사용하는 것은 불가능
      - strcpy 함수
        - char형 배열에 새 문자열 저장할 때 사용
        - 저장할 문자열의 길이 파악해 그 길이 만큼 char형 배열에 복사

```
char *strcpy(char *dest, const char *src)
```

문자열 대입

예제 8-7 문자열을 대입하는 strcpy 함수

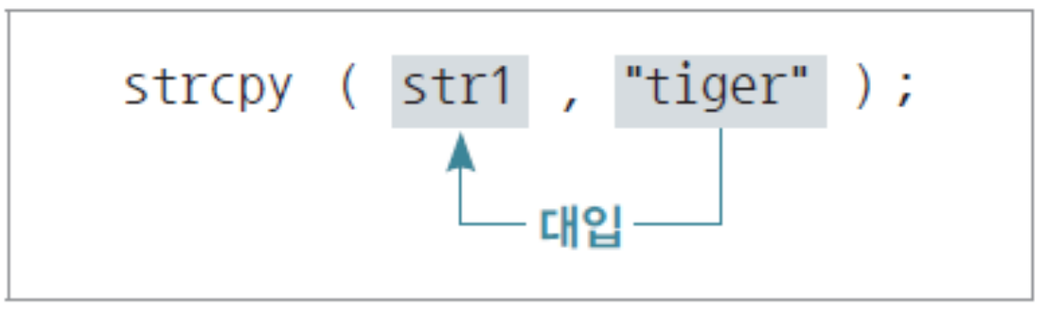
```
1.  #include <stdio.h>
2.  #include <string.h>
3.
4.  int main(void)
5.  {
6.      char str1[80] = "cat";
7.      char str2[80];
8.
9.      strcpy(str1, "tiger");          // str1 배열에 "tiger" 복사
10.     strcpy(str2, str1);             // str2 배열에 str1 배열의 문자열 복사
11.     printf("%s, %s\n", str1, str2);
12.
13.     return 0;
14. }
```

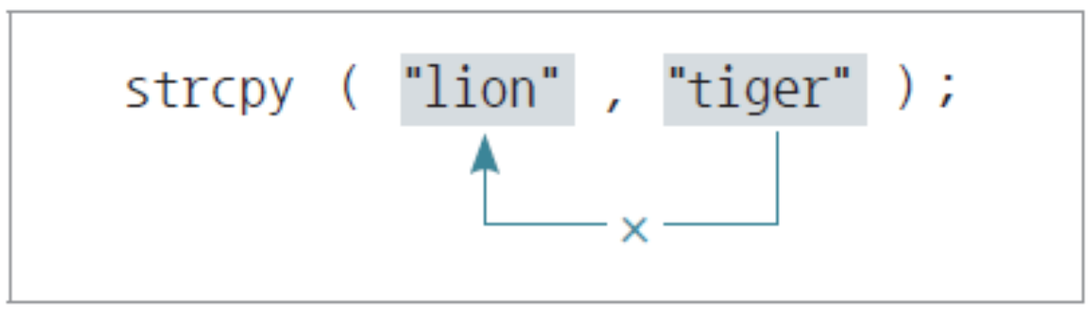
실행 결과

```
tiger, tiger
```

문자열 대입

- 문자열을 대입하는 strcpy 함수
  - 새로운 헤더 파일 포함
  - string.h은 문자열 다루는 함수들의 원형 모아놓음
  - strcpy 함수에 접근하기 위해 사용
  - 첫 번째 인수 - 저장될 곳의 배열명
  - 두 번째 인수 - 저장할 문자열
  - 오른쪽 값을 왼쪽 변수에 대입하는 연산으로 이해





문자열 전용 입출력 함수(GETS, PUTS)

- gets 함수
  - 빈 칸을 포함하여 한 줄 전체를 문자열로 입력
  - 문자열 출력 함수 puts와 같이 쓰임
    - 문자열 입력 중간에 빈칸이나 탭 문자 사용 가능
    - 엔터 입력 전까지 전체를 하나의 문자열로 배열에 저장
    - 마지막에 널문자 붙여 문자열의 끝 표시
  - 두 함수의 함수 원형

```
char *gets(char *str)
int puts(const char *str)
```

문자열 전용 입출력 함수(GETS, PUTS)

예제 8-8 빈칸을 포함한 문자열 입력

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      char str[80];
6.
7.      printf("문자열 입력 : ");    // 입력 안내 메시지 출력
8.      gets(str);                  // 빈칸을 포함한 문자열 입력
9.      puts("입력된 문자열 : ");    // 문자열 상수 출력
10.     puts(str);                  // 배열에 저장된 문자열 출력
11.
12.     return 0;
13. }
```

실행 결과

```
문자열 입력 : Love is belief...↵
입력된 문자열 :
Love is belief...
```

문자열 전용 입출력 함수(GETS, PUTS)

- gets 함수의 단점
  - 입력할 배열의 크기 검사하지 않음
  - 배열의 크기보다 긴 문자열 입력 시
    - 배열 벗어난 메모리 영역 침범 가능성
  - 컴파일러는 안전성 문제를 경고 메시지로 출력
- puts 함수
  - 문자열 상수, char형 배열의 배열명 주면 문자열을 화면에 출력
  - printf 함수의 문자열 출력 기능과 흡사
  - 문자열 출력 후에 자동으로 줄 바꾸는 차이가 있음

배열과 포인터

➤배열은 시작 주소를 알면 포인터로 모든 요소에 접근 가능

표 10-1 배열과 포인터

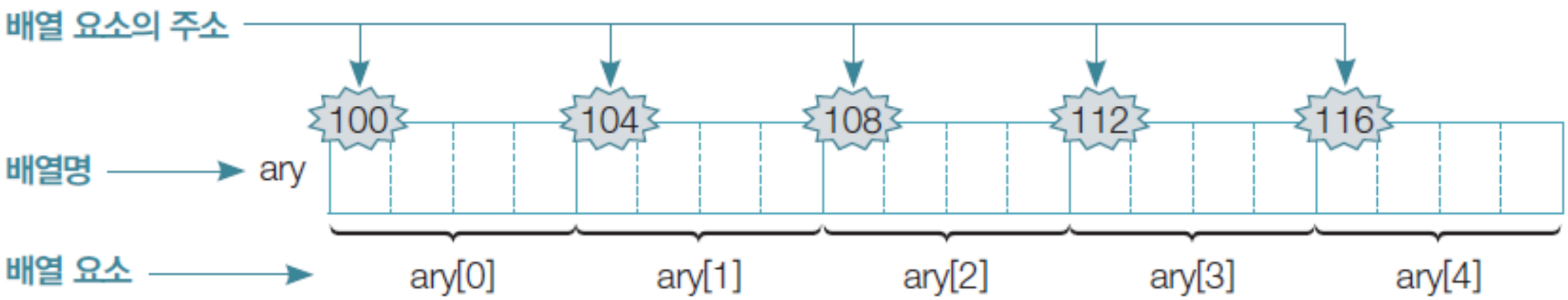
구분	사용 예	기능
배열명	int ary[3]; ary == &ary[0];	배열명은 첫 번째 요소의 주소
배열명 + 정수	int ary[3]; ary + 1;	가리키는 자료형의 크기를 곱해서 더한다. ary + (1 * sizeof(*ary))

배열명과 포인터는 같다.	int ary[3];
	int *pa = ary;
	pa[1] = 10;
		포인터가 배열명을 저장하면 배열명처럼 쓸 수 있다.
		두 번째 배열 요소에 10 대입

배열명과 포인터는 다르다.	ary++; ➡ (×) pa++; ➡ (○)	배열명은 상수이므로 그 값을 바꿀 수 없지만 포인터는 가능하다.

배열명의 정체

- 배열명은 컴파일 과정에서 첫 번째 배열 요소 주소로 변환
  - 배열 - 자료형이 같은 변수 메모리에 연속 할당
    - 각 배열 요소는 일정한 간격으로 주소 가짐
    - Ex) int ary[5];의 배열이 메모리 100번지부터 할당
      - int형 변수의 크기가 4바이트라면?



배열명의 정체

예제 10-1 배열명이 주소인지 확인

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int ary[5] = {10, 20, 30, 40, 50};
6.
7.      printf("배열명 자체의 값 : %u\n", ary);
8.      printf("첫 번째 배열 요소의 주소 : %u\n", &ary[0]);
9.      printf("배열명이 가리키는 요소의 값 : %d\n", *ary);
10.     printf("첫 번째 배열 요소의 값 : %d\n", ary[0]);
11.
12.     return 0;
13. }
```

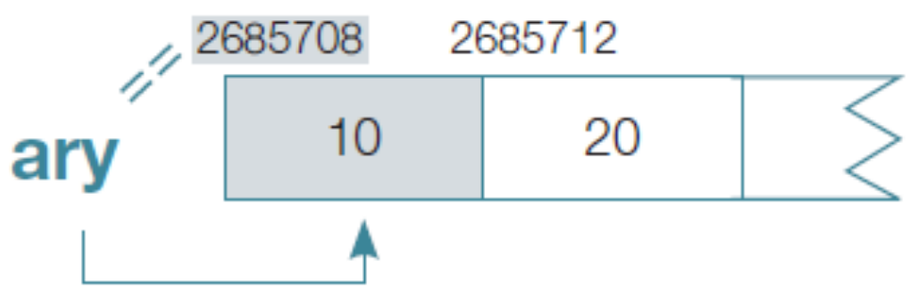
실행 결과

```
배열명 자체의 값 : 2685708
첫 번째 배열 요소의 주소 : 2685708
배열명이 가리키는 요소의 값 : 10
첫 번째 배열 요소의 값 : 10
```

배열명의 정체

➤배열명에 간접참조 연산 수행

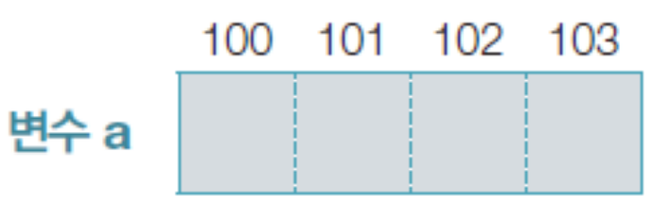
➤첫 번째 배열 요소도 사용 가능



- 첫 번째 배열 요소의 주소값이다.
- 첫 번째 배열 요소를 가리킨다.
- *ary는 첫 번째 배열 요소가 된다.

배열명으로 배열 요소 사용하기

- 주소는 정수처럼 보이지만 자료형 대한 정보를 갖고 있는 특별한 값
- 정해진 연산만 가능
  - 정수 덧셈이 대표적인 예시
  - 주소 + 정수 ? 주소 + (정수 * 주소를 구한 변수의 크기)



&a
- a의 주소값(100)
- a의 포인터
- a를 가리킨다.
- int형을 가리킨다(a가 int형이므로).

&a +1 → 100 + (1 * sizeof (int)) → 104
- int형을 가리키는 주소 104번지
- 104번지부터 107번지까지 4바이트 저장 공간의 주소값

배열명으로 배열 요소 사용하기

예제 10-2 배열명에 정수 연산을 수행하여 배열 요소 사용

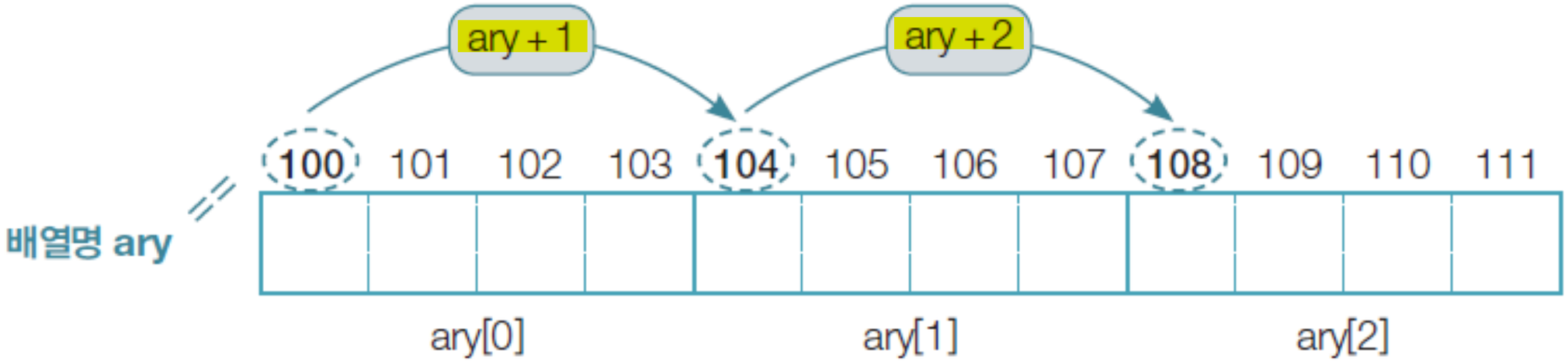
```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      int ary[3];
6.      int i;
7.
8.      *(ary + 0) = 10;                    // ary[0] = 10
9.      *(ary + 1) = *(ary + 0) + 10;       // ary[1] = ary[0] + 10
10.
11.     printf("세 번째 배열 요소에 키보드 입력 : ");
12.     scanf("%d", ary + 2);               // &ary[2]
13.
14.     for(i = 0; i < 3; i++)              // 모든 배열 요소 출력
15.     {
16.         printf("%5d", *(ary + i));      // ary[i]
17.     }
18.
19.     return 0;
20. }
```

실행 결과

```
세 번째 배열 요소에 키보드 입력 : 30
   10   20   30
```

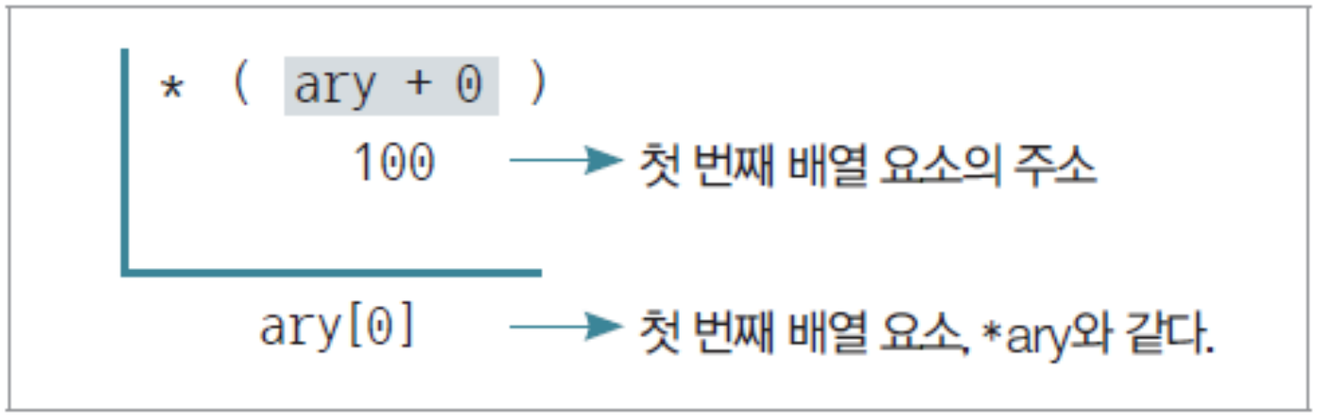
배열명으로 배열 요소 사용하기

➤선언된 배열이 메모리 100번지부터 할당되었다고 가정

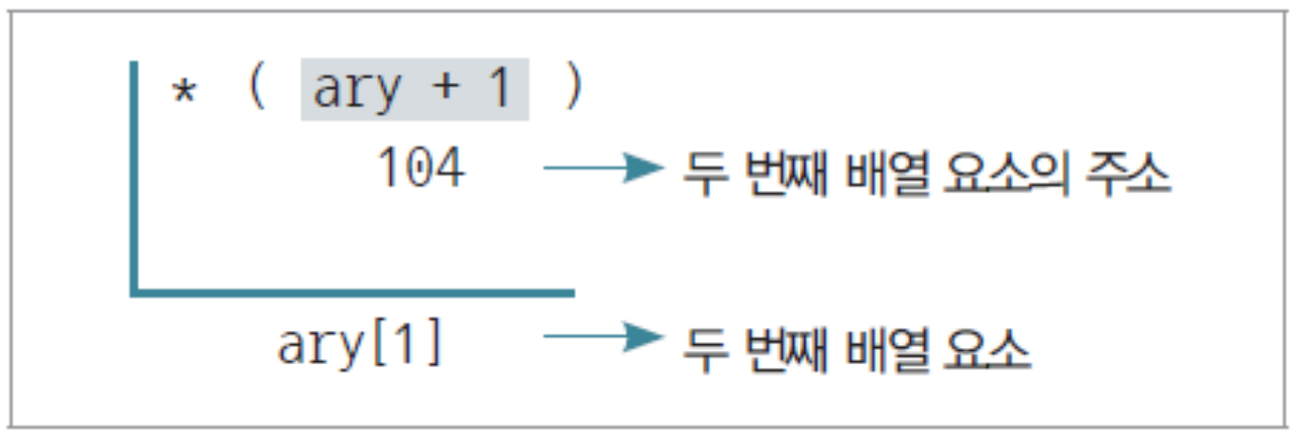


배열명으로 배열 요소 사용하기

➤ ary에 0 더하지 않고 바로 간접참조 연산 수행한 결과



➤ ary에 1 더하면 104번지 - 두 번째 배열 요소의 주소이며 이 값에 간접참조 연산 수행하면 두 번째 배열 요소 사용가능



배열명으로 배열 요소 사용하기

➤세 번째 배열 요소는 키보드로 값 입력

➤배열 요소의 표현 방법 써서 쉽게 작성

```
scanf("%d", & ary[2]);  // ary[2]가 세 번째 배열 요소이므로 &를 사용하여 입력
```

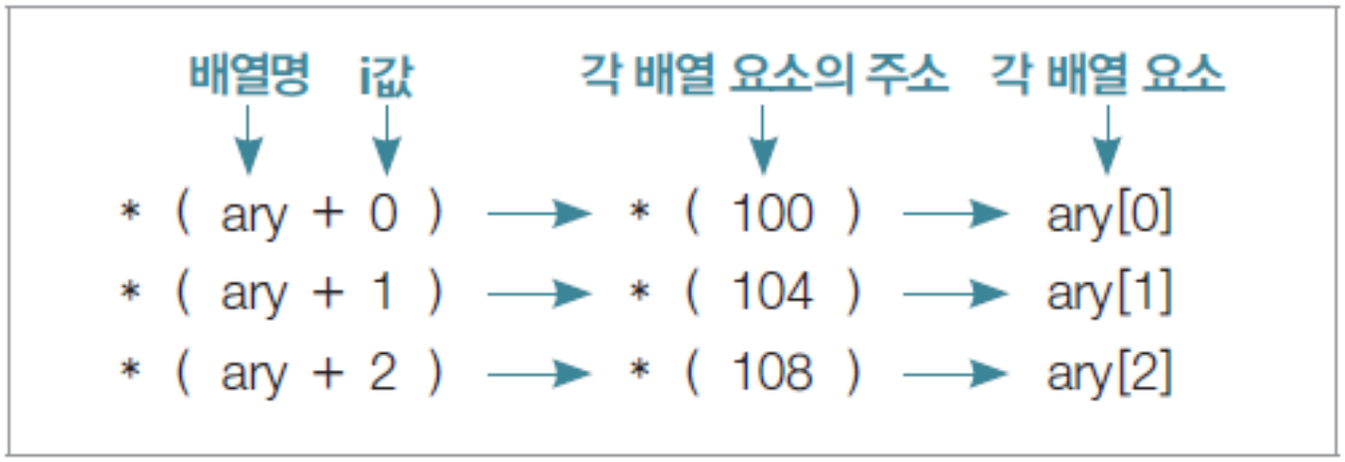
➤배열 요소 ary[2]의 주소를 scanf 함수에 주는 것 ->배열명 ary에 2를 더한 값

```
scanf("%d", ary + 2);   // 배열명에 2를 더해 ary[2]의 주소 계산
```

배열명으로 배열 요소 사용하기

➤반복 과정에서 배열명에 i 더해 각 배열 요소의 주소 구함

➤간접참조 연산으로 모든 배열 요소의 값 출력



➤배열요소에 사용하는 대괄호는 연산자

➤포인터 연산의 '간접참조, 괄호, 더하기' 연산 기능을 가짐

➤상황에 따라 대괄호나 포인터 연산식 중 골라 쓸 것

➤특별한 경우가 아니면 대괄호 사용하는 것이 쉬움.

➤ary[1] = *(ary+1)

➤&ary[2] = ary+2

배열명으로 배열 요소 사용하기

- 배열 할당 영역 벗어나는 포인터 연산식은 사용이 가능한가?
  - 문법적으로 문제 없으므로 컴파일 되나 결과예상불가
    - 배열 요소의 개수가 3개인 ary 배열에서 ary+3은 네 번째 배열 요소의 주소가 되고 *(ary+3)은 네 번째 배열 요소
    - 할당된 값이 아니므로 값이 바뀌거나 강제 종료 가능