C프로그래밍

9 배열

휴강 공지

- **5/5(월) 어린이날 휴강**

01
배열의 이해

배열을 사용하는 이유 (1/4)

- **배열의 개념**
  - ✓ 여러 개의 변수를 나란히 연결하여 하나의 자료구조로 저장, 관리
  - ✓ 박스(변수)를 한 줄로 붙이고, 박스의 이름(aa)을 지정
  - ✓ 각각의 박스는 aa[0], aa[1], … 과 같이 첨자를 붙임

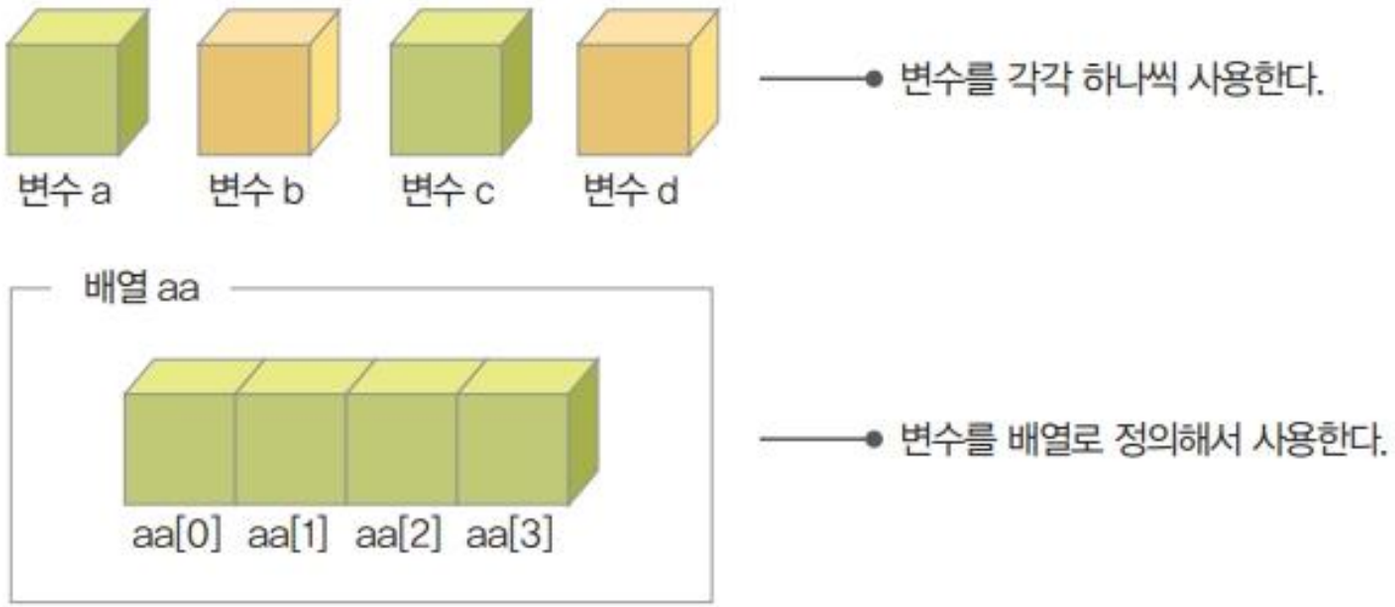

그림 8-1 배열의 개념

배열을 사용하는 이유 (2/4)

- **개별적인 변수들을 사용하는 예**

기본 8-1 변숫값 여러 개를 선언하여 출력하는 예 8-1.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int a, b, c, d;                  ---- 각 입력 변수와 합계 변수를 선언한다.
06    int hap;
07
08    printf("1번째 숫자를 입력하세요 : ");
09    scanf("%d", &a);                 ---- 변수에 숫자를 입력한다.
10    printf("2번째 숫자를 입력하세요 : ");
11    scanf("%d", &b);                 ---- 변수에 숫자를 입력한다.
12    printf("3번째 숫자를 입력하세요 : ");
13    scanf("%d", &c);                 ---- 변수에 숫자를 입력한다.
14    printf("4번째 숫자를 입력하세요 : ");
15    scanf("%d", &d);                 ---- 변수에 숫자를 입력한다.
16
17    hap = a + b + c + d;             ---- 입력받은 숫자의 합계 결과이다.
18
19    printf(" 합계 ==> %d \n", hap);
20 }
```

실행 결과

```
1번째 숫자를 입력하세요 : 10
2번째 숫자를 입력하세요 : 20
3번째 숫자를 입력하세요 : 30
4번째 숫자를 입력하세요 : 40
 합계 ==> 100
```

배열을 사용하는 이유 (3/4)

- **배열의 선언 방법**

```
데이터_형식  배열_이름[개수];
```

✓ 변수 4개를 담은 정수형 배열을 선언의 예

```
int aa[4];
```

✓ 배열을 사용하지 않는다면 각각의 변수를 int a, b, c, d;와 같이 선언해야 함 (기본 8-1)

✓ 배열의 경우에는 첨자를 사용하여 aa[0], aa[1], aa[2], aa[3]과 같이 각 요소를 식별

✓ 배열을 4개 선언할 때는 첨자를 1~4가 아닌 0~3을 사용

구분	변수	배열
선언 예	int a, b, c, d;	int aa[4];
사용할 수 있는 변수 형식 예	a, b, c, d	aa[0], aa[1], aa[2], aa[3]

배열을 사용하는 이유 (4/4)

- **배열을 사용하여 기본 8-1을 변경시킨 예**

기본 8-2 배열에 값을 선언하여 출력하는 예 — 8-2.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int aa[4];                  ---- 정수형 배열을 선언한다.
06    int hap;
07
08    printf("1번째 숫자를 입력하세요 : ");
09    scanf("%d", &aa[0]);        ---- aa[0]에 숫자를 입력한다.
10    printf("2번째 숫자를 입력하세요 : ");
11    scanf("%d", &aa[1]);        ---- aa[1]에 숫자를 입력한다.
12    printf("3번째 숫자를 입력하세요 : ");
13    scanf("%d", &aa[2]);        ---- aa[2]에 숫자를 입력한다.
14    printf("4번째 숫자를 입력하세요 : ");
15    scanf("%d", &aa[3]);        ---- aa[3]에 숫자를 입력한다.
16
17    hap = aa[0] + aa[1] + aa[2] + aa[3];  ---- 입력받은 배열에 저장된 숫자의 합계 결과이다.
18
19    printf(" 합계 ==> %d \n", hap);
20 }
```

실행 결과

```
1번째 숫자를 입력하세요 : 10
2번째 숫자를 입력하세요 : 20
3번째 숫자를 입력하세요 : 30
4번째 숫자를 입력하세요 : 40
 합계 ==> 100
```

배열의 활용 범위 (1/10)

- **반복문을 이용하여 데이터들을 차례대로 검색할 수 있음**
  - ✓ 배열의 첨자가 순서대로 변할 수 있도록 반복문과 함께 활용해야만 배열의 효율성이 극대화

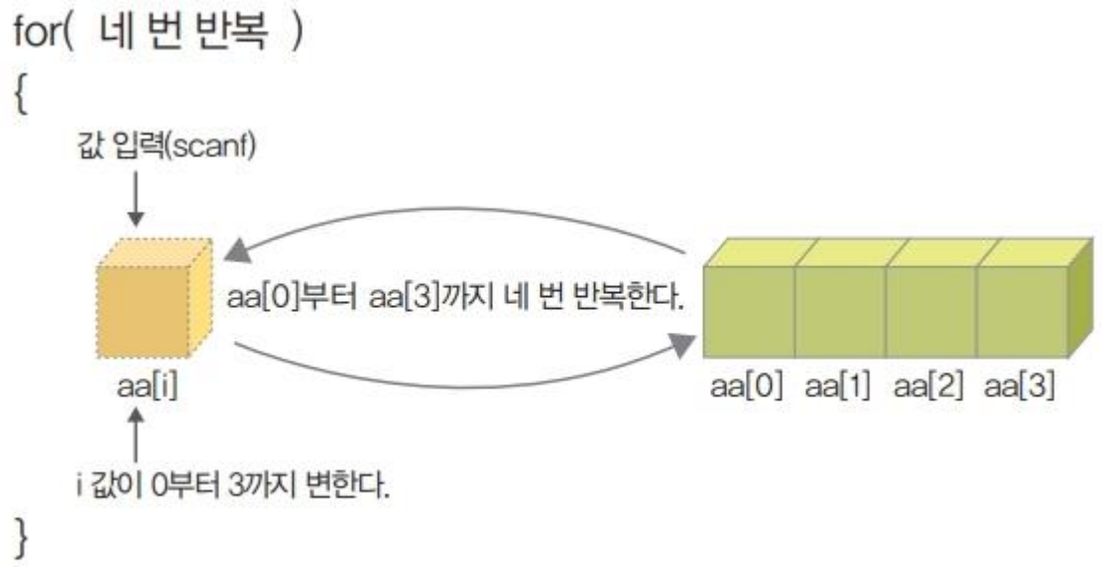

그림 8-2 for문을 활용한 배열값 입력

  - ✓ for문을 네 번 돌면서 aa[i]의 첨자가 aa[0]~aa[3]으로 변하게 하면 변수 4개를 차례대로 검색할 수 있음

배열의 활용 범위 (2/10)

- **반복문을 이용하여 기본 8-2를 개선한 예**

응용 8-3 for문으로 배열의 첨자를 활용하는 예 8-3.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int aa[4];                          ----- 배열과 합계 변수, 첨자를 선언한다.
06    int hap=0;
07    int i;
08
09    for(i=0; i <= 3; i++)               ----- 배열 aa[0]~[3]에 숫자 4개를 입력받는다.
10    {
11       printf("%d번째 숫자를 입력하세요 : ", i+1);
12       scanf("%d", __1__ );
13    }
14
15    hap = aa[0] + aa[1] + aa[2] + aa[3];   ----- 배열에 저장된 숫자 4개를 더한다.
16
17    printf(" 합계 ==> %d \n", hap);
18 }
```

&aa[i]

실행 결과

```
1번째 숫자를 입력하세요 : 10
2번째 숫자를 입력하세요 : 20
3번째 숫자를 입력하세요 : 30
4번째 숫자를 입력하세요 : 40
 합계 ==> 100
```

배열의 활용 범위 (3/10)

- **반복문을 이용하여 기본 8-2를 개선한 예 (cont'd)**
  - ✓ 9행에서 i가 0부터 3까지 네 번 실행
  - ✓ 12행에서도 첨자 i가 0부터 3까지 네 번 변경되므로 변수 aa[0], aa[1], aa[2], aa[3]에 값을 차례대로 입력 받음
  - ✓ 15행에서는 변수 4개를 더했는데, 만약 배열이 100개라면 'hap = aa[0] + aa[1] + … aa[99]'로 코딩을 해야 함
  - ➔ 따라서 다음과 같이 for문으로 변경하는 것이 효율적

```
15  for(i=0; i <= 3; i++)
16  {
17      hap = hap + aa[i];
18  }
```

배열의 활용 범위 (4/10)
- 배열의 초기화

- **배열을 정의하는 동시에 값을 대입하는 것**
  - ✓ 4개의 값을 담은 배열 aa의 초기화 : 블록({ })과 콤마(,)를 사용

```
int aa[4] = {100, 200, 300, 400} ;
```

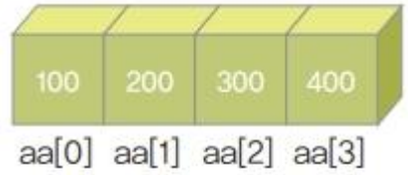

그림 8-3 배열의 초기화 1

- **배열의 개수(첨자)를 반드시 지정하지 않아도 됨**
  - ✓ 개수를 지정하지 않으면 블록({ }) 안의 초깃값 개수에 따라 자동으로 배열의 개수가 정해짐

```
int aa[] = {100, 200, 300, 400};
```

- **배열을 선언하기만 하고 초기화하지 않을 경우 : 각 배열에 아무것도 넣지 않았기 때문에 쓰레기 값이 저장되어 있음**

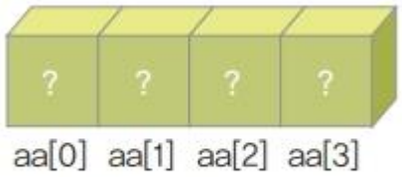

그림 8-4 배열의 초기화 2

배열의 활용 범위 (5/10)
– 배열의 초기화

- **배열의 개수보다 초기화 값의 개수가 적은 경우**
  - ✓ 초깃값이 주어진 aa[0]과 aa[1]에는 각각 100과 200이 들어가고 나머지 aa[2]와 aa[3] 에는 0이 들어감

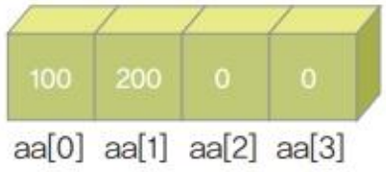

그림 8-5 배열의 초기화 3

  - ✓ 초깃값을 0이라고 직접 써도 되고 초기화할 부분을 비워 놓아도 됨

```
int aa[4] = {100, 200, 0, 0};   =   int aa[4] = {100, 200};
```

  - ✓ ex) 배열 1000개를 모두 0으로 초기화하고 싶다면

```
int aa[1000] = {0}
```

- **배열의 개수보다 초기화할 값의 개수가 많다면 컴파일 오류가 발생**

int aa[4] = {100, 200, 300, 400, 500}

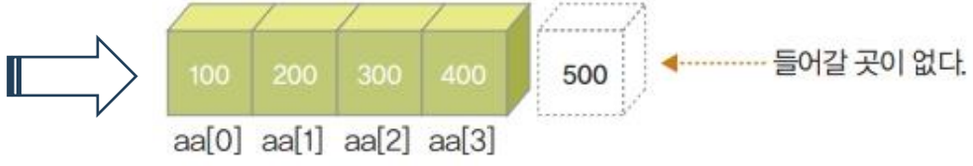

그림 8-6 배열의 초기화 4

배열의 활용 범위 (6/10)
– 배열의 초기화

- **배열 초기화의 예**

기본 8-4 배열의 초기화 예 1 — 8-4.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int aa[4] = {100, 200, 300, 400};      ---- 배열의 개수를 지정하고 초기화한다.
06    int bb[]  = {100, 200, 300, 400};      ---- 배열의 개수를 지정하지 않고 초기화한다.
07    int cc[4] = {100, 200};                ---- 배열의 일부만 초기화한다.
08    int dd[4] = {0};                       ---- 배열 전체를 0으로 초기화한다.
09    int i;
10
11    for(i=0; i <= 3; i++)                  ---- 4회 반복하며 배열 aa[0]~aa[3] 값을 출력한다.
12       printf("aa[%d]==>%d\t", i, aa[i]);
13    printf("\n");
14
15    for(i=0; i <= 3; i++)                  ---- 4회 반복하며 배열 bb[0]~bb[3] 값을 출력한다.
16       printf("bb[%d]==>%d\t", i, bb[i]);
17    printf("\n");
18
```

배열의 활용 범위 (7/10)
– 배열의 초기화

- **배열 초기화의 예 (cont'd)**

```
19    for(i=0; i <= 3; i++)                  ----- 4회 반복하며 배열 cc[0]~cc[3] 값을 출력한다.
20      printf("cc[%d]==>%d\t", i, cc[i]);
21    printf("\n");
22
23    for(i=0; i <= 3; i++)                  ----- 4회 반복하며 배열 dd[0]~dd[3] 값을 출력한다.
24      printf("dd[%d]==>%d\t", i, dd[i]);
25    printf("\n");
26  }
```

실행 결과

```
aa[0]==>100     aa[1]==>200     aa[2]==>300     aa[3]==>400
bb[0]==>100     bb[1]==>200     bb[2]==>300     bb[3]==>400
cc[0]==>100     cc[1]==>200     cc[2]==>0       cc[3]==>0
dd[0]==>0       dd[1]==>0       dd[2]==>0       dd[3]==>0
```

배열의 활용 범위 (8/10)
– 배열의 초기화

- **배열 초기화 응용 예**

응용 8-5 배열의 초기화 예 2 — 8-5.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int aa[100], bb[100];          ---- 배열 aa와 bb를 선언한다.
06    int i;
07
```

// 배열 aa를 짝수로 초기화

for(i = 0; i < 100; i++)

*aa[i] = i * 2;*

//배열 bb에 역순으로 대입

for(i = 0; i < 100; i++)

bb[i] = a[99-i];

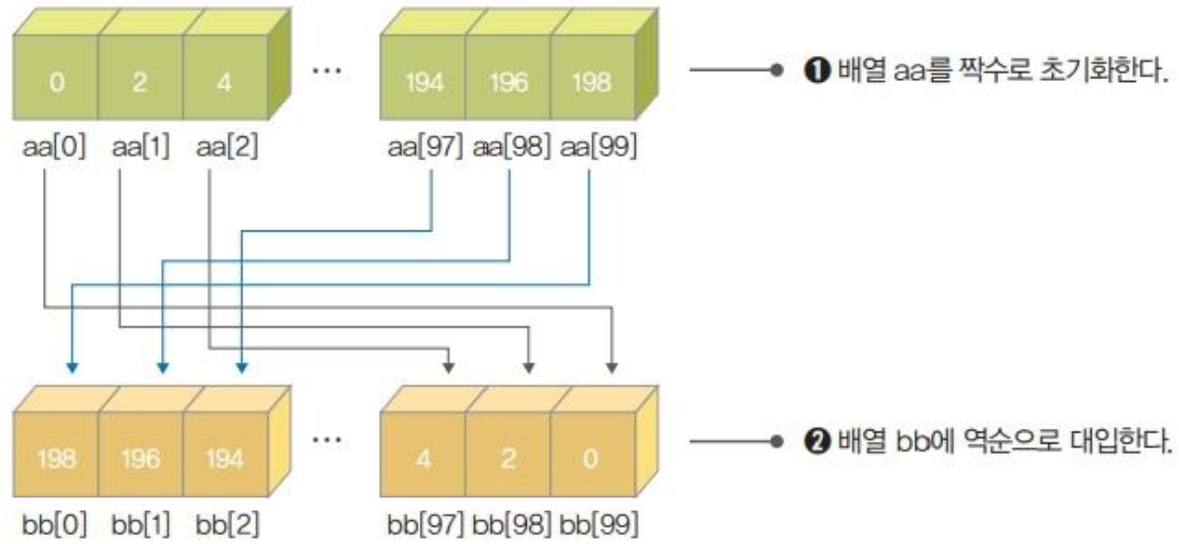

그림 8-7 배열의 초기화 5

배열의 활용 범위 (9/10)
- 배열의 크기

- **배열의 크기 알아내기**

✓ sizeof( ) 사용

```
배열의 크기(요소 개수) = sizeof(전체 배열 이름) / sizeof(배열의 데이터 형식);
```

✓ int aa[4]; 배열의 크기 알아내기

```
배열의 크기(요소 개수) = sizeof(aa) / sizeof(int);
```

① aa 배열이 메모리에서 차지하고 있는 크기(4바이트×4개=16바이트)를 알아냄

② 선언한 배열의 데이터 형식의 크기(4바이트)로 나눔

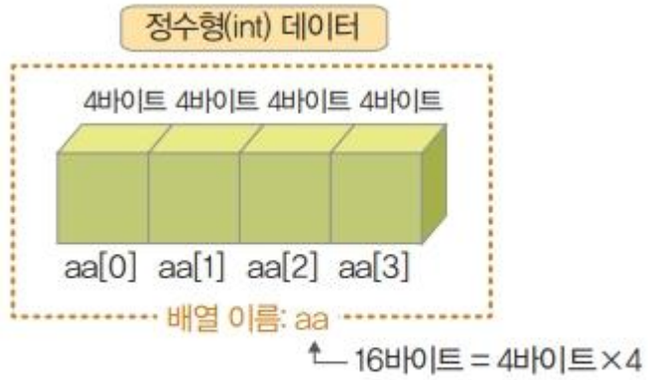

그림 8-8 배열의 크기

배열의 활용 범위 (10/10)
- 배열의 크기

- **배열의 크기 예**

기본 8-6 배열의 크기를 계산하는 예 8-6.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int aa[] = {10, 20, 30, 40, 50};      ---- 배열을 선언한다. 변수의 개수를 지정하지 않았다.
06    int count;                            ---- 배열 크기를 저장할 변수이다.
07
08    count = sizeof(aa) / sizeof(int);     ---- 배열 크기를 계산한다.
09
10    printf("배열 aa[]의 요소의 개수는 %d 입니다.\n", count);
11 }
```

실행 결과

```
배열 aa[]의 요소의 개수는 5 입니다.
```

02
배열과 문자열

문자형 배열 (1/13)

- **정수형 배열과 문자형 배열**
  - ✓ 정수형 배열: 각각의 배열 요소에 정수(100, 200, 300, 400)를 입력
  - ✓ 문자형 배열: 각 각의 배열 요소에 문자('X', 'Y', 'Z', '\0')를 입력
  - ✓ {'X', 'Y', 'Z', '\0'} 대신 "XYZ"와 같은 문자열을 대입하면 편리
  - ✓ 문자열은 문자형 배열에 입력하는 '문자의 집합'

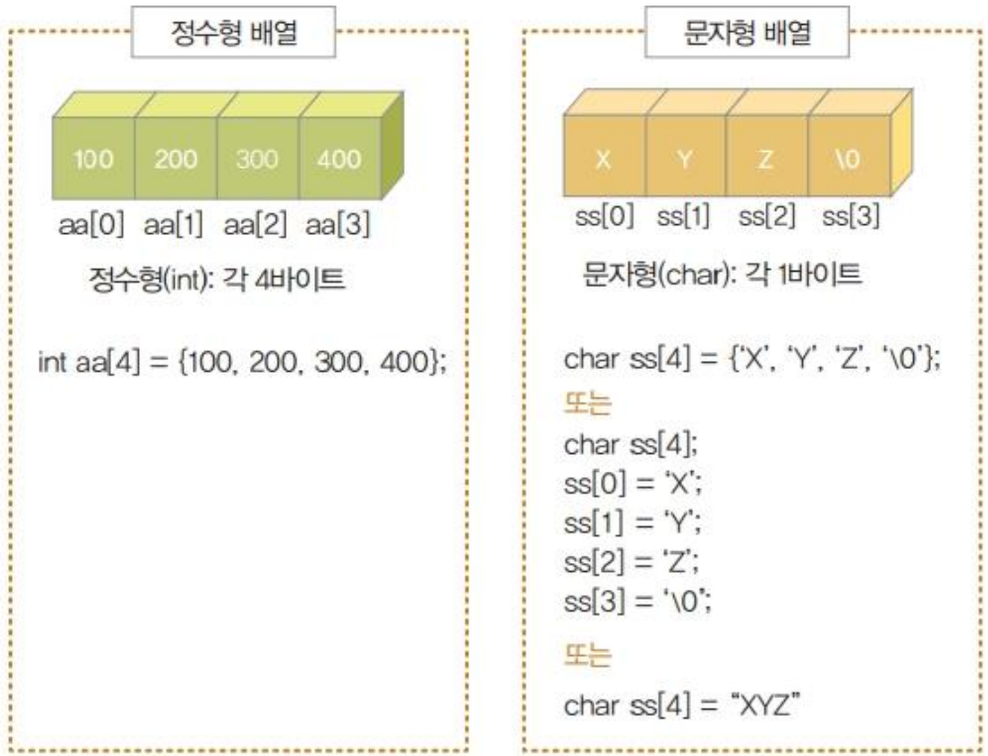

그림 8-9 정수형 배열과 문자형 배열

문자형 배열 (2/13)

- **문자열의 기본 형식**

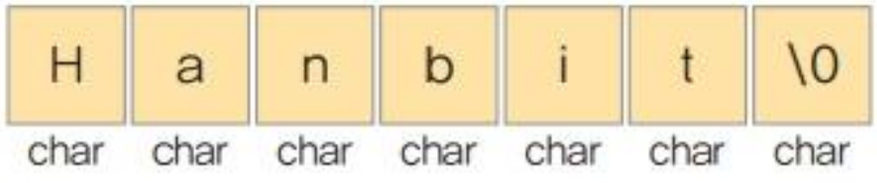

그림 3-24 문자열의 기본 구조

✓ 문자열은 문자형(char)의 집합이므로 “Hanbit”이라는 문자열을 저장하기 위해서는 각 문자들이 나란히 있으면 됨

✓ [그림 3-24]에서는 문자가 6개가 아니라 7개인데 이를 주의해야 함

- 문자와 문자열의 차이점은? ‘\0’ 문자 포함 여부

문자형 배열 (3/13)

- **문자열의 기본 형식 (cont'd)**

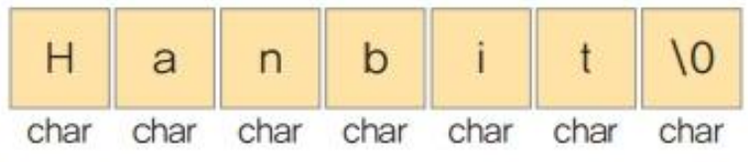

그림 3-24 문자열의 기본 구조

✓ 문자열을 저장하는 변수는 문자형을 연속적으로 나열한 것을 의미하는 배열 형태가 되어야 함

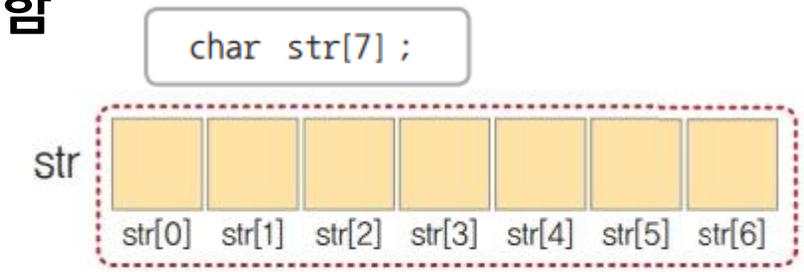

그림 3-25 배열로 표현한 문자열

문자형 배열 (4/13)

- **문자열의 기본 형식 (cont'd)**
  - ✓ 전체 배열의 이름은 str이고 각각의 이름은 str[0] ~ str[6]
  - ✓ 이번에는 str 배열에 "Basic"이라는 문자열을 대입
  - ✓ 다른 데이터 형식은 대입 연산자(=)를 사용해서 대입했으나 문자열은 특별히 strcpy( ) 함수를 사용해야 함
    - 초기화 할 때만 대입 연산자로 대입이 가능하고, 이후에는 strcpy( ) 사용
    - 개념적으로는 대입 연산자와 비슷

```
strcpy(str, "Basic");
```

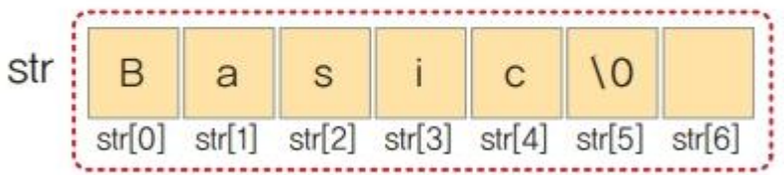

그림 3-26 str 배열에 문자열 대입

문자형 배열 (5/13)

- **문자열과 문자의 표현**

여기서 잠깐 올바른 문자 표현

- 문자는 반드시 ' '로 감싸야 하며 한 글자만 올 수 있으며 ❶~❸은 모두 틀린 표현
- char a;
- ❶ a = 'Ab'; ❷ a = "A"; ❸ a = "Ab";

문자형 배열 (6/13)

- **문자열 대입의 예**

기본 3-14 문자열 형식 사용 예 1 — 3-14.c

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <string.h>
03 #include <stdio.h>
04 void main( )
05 {
06    char str1[10];                  ----- 문자형 배열 str1과 str2를 선언한다.
07    char str2[10];
08    char str3[10] = "CookBook";     ----- 문자형 배열 str3을 선언함과 동시에 문자열을
09                                          대입한다.
10    strcpy(str1, "Basic-C");        ----- str1에 문자열을 대입한다.
11    strcpy(str2, str3);             ----- str3의 값을 str2에 복사한다.
12
13    printf("str1 ==> %s \n", str1); ----- 문자형 배열 str1, str2, str3을 출력한다.
14    printf("str2 ==> %s \n", str2);
15    printf("str3 ==> %s \n", str3);
16 }
```

실행 결과

```
str1 ==> Basic-C
str2 ==> CookBook
str3 ==> CookBook
```

문자형 배열 (7/13)

- **기본 3-14 복기**

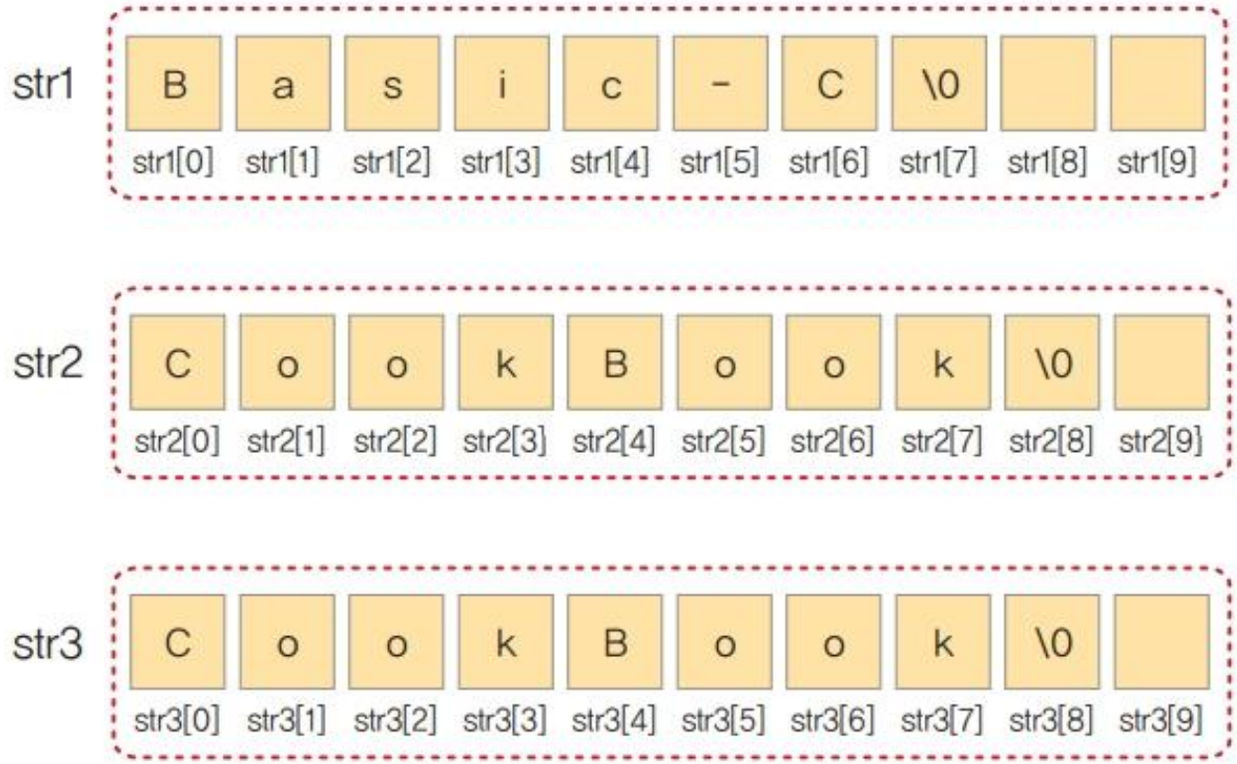

그림 3-27 [기본 3-14]의 문자열 선언 상태

문자형 배열 (8/13)

- **두번째 문자열 대입의 예**

응용 3-15 문자열 형식 사용 예 2 — 3-15.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    char str[10] = "0123456789";          ----- 열 자리의 str 배열에 글자 10개를 대입한다.
06
07    printf("str ==> %s \n", str);         ----- str의 내용을 출력한다.
08
09    str[0] = 'I';                         ----- str 배열에 글자 6개와 널 문자를 입력한다.
10    str[1] = 'T';
11    str[2] = 'C';
12    str[3] = 'o';
13    str[4] = 'o';
14    str[5] = 'k';
15    str[6] = '\0';
16
17    printf("str ==> %s \n", str);         ----- str의 내용을 출력한다.
18    printf("str[7] ==> %c \n", __1__ );   ----- str[7]의 한 글자를 출력한다.
19    printf("str[50] ==> %c \n", __2__ )   ----- str[50]의 한 글자를 출력한다.
20 }
```

실행 결과

```
str ==> 0123456789僦僦僦?x?缸
str ==> ITCook
str[7] ==> 7
str[50] ==> ?
```

정답_ 1 str[7] 2 str[50]

문자형 배열 (9/13)

- **응용 3-15 복기**
  - ✓ str 배열에 널 문자가 없으므로 그 이후에 무엇이 들어 있는지 알 수 없지만 일단은 널 문자를 만날 때까지 계속 출력
  - ✓ 널 문자는 문자열을 출력할 때 자동차의 브레이크와 같은 역할을 하므로, 이 경우는 브레이크가 없어서 계속 달릴 수밖에 없는 상황

그림 3-28 [응용 3-15]의 str 배열 내용 1

문자형 배열 (10/13)

- **응용 3-15 복기 (cont'd)**
  - ✓ 9~15행은 str 배열에 strcpy( ) 함수를 이용하여 값을 대입하지 않고 한 글자 씩 직접 대입

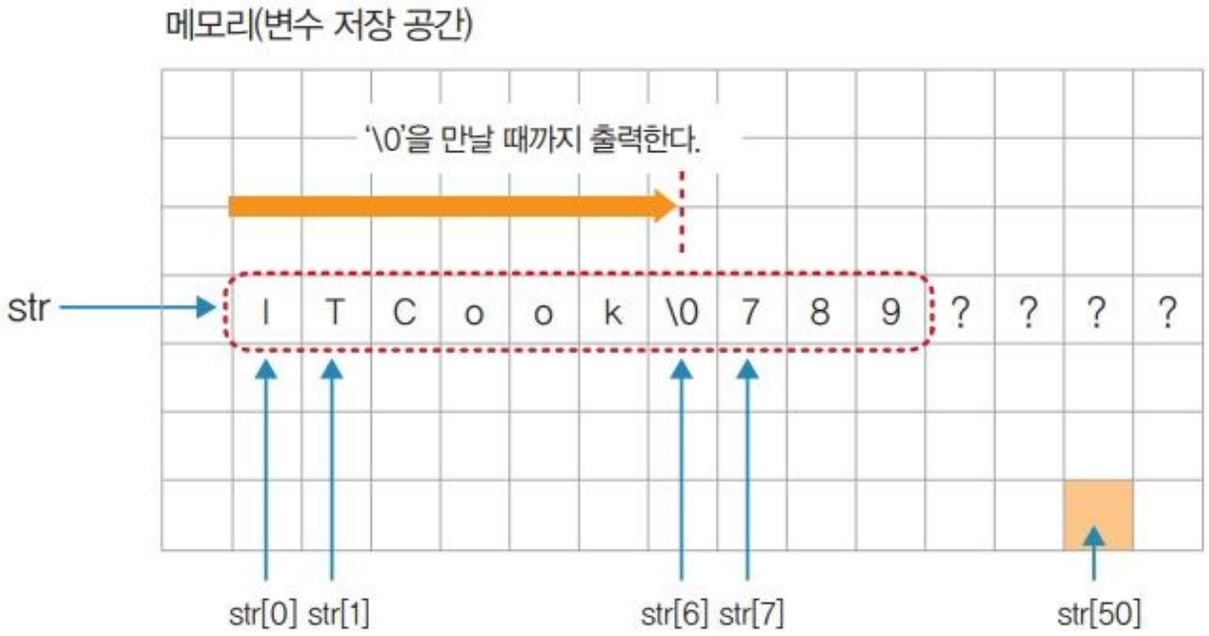

그림 3-29 [응용 3-15]의 str 배열 내용 2

문자형 배열 (111/13)

- **문자열 선언 및 출력 예**

기본 8-7 문자열을 선언하고 출력하는 예 8-7.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    char ss[8] = "Basic-C";
06    int i;
07
08    ss[5] = '#';
09
10    for(i=0; i < 8; i++)
11    {
12       printf("ss[%d] ==> %c \n", i, ss[i]);
13    }
14
15    printf("문자열 배열 ss ==> %s \n", ss);
16 }
```

----- 크기가 8인 문자형 배열을 선언하고 초기화한다.

----- 여섯 번째 문자를 바꾼다.

----- 여덟 번 반복하면서 배열 ss의 각 문자를 출력한다.

----- 배열 ss의 전체 문자열을 출력한다.

실행 결과

```
ss[0] ==> B
ss[1] ==> a
ss[2] ==> s
ss[3] ==> i
ss[4] ==> c
ss[5] ==> #
ss[6] ==> C
ss[7] ==>
문자열 배열 ss ==> Basic#C
```

문자형 배열 (12/13)

- **기본 8-7 복기**
  - ✓ 5행에서 "Basic-C"라는 일곱 글자를 넣기 위해 널 문자('\0') 자리까지 포함해서 여덟 자리의 배열을 정의
  - ✓ 아래에서 ❸은 문자 1개를 출력하는 방식, ❹는 전체 문자열을 출력하는 방식

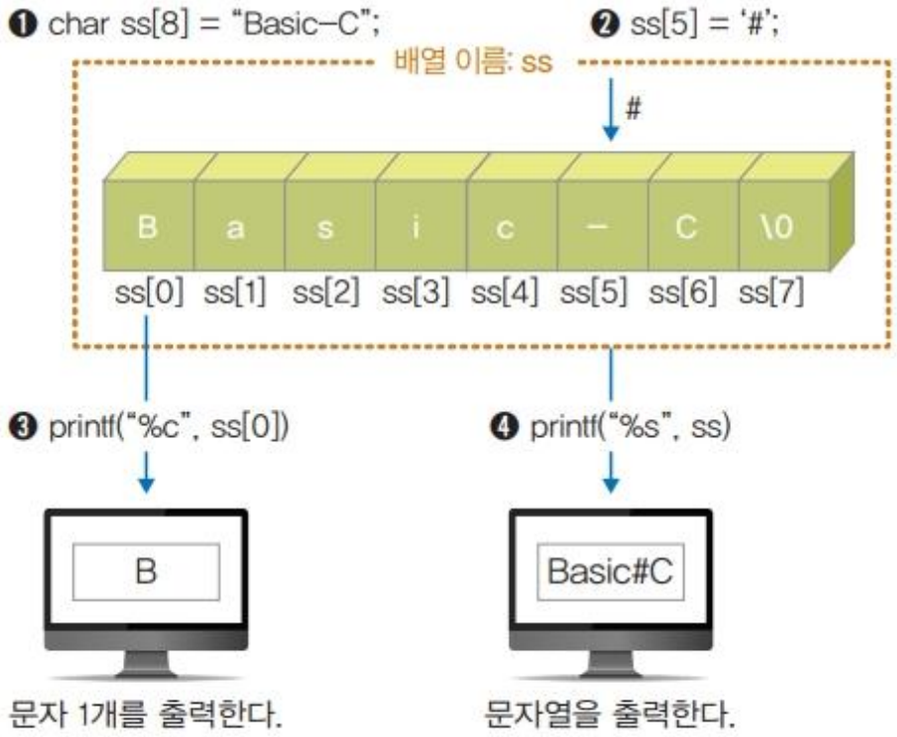

그림 8-10 문자 출력과 문자열 출력

문자형 배열 (13/13)

- **문자열 반대로 출력하기 예**

응용 8-8 문자열을 반대 순서로 출력하는 예 8-8.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    char ss[5] = "abcd";          ----- 크기 5의 문자형 배열 ss와 변환해서 저장할 배열 tt이다.
06    char tt[5];
07    int  i;
08
09    for(i=0; i < 4; i++)          ----- 4회 반복해서 각 배열에 문자를 반대 순서로 대입한다.
10    {
11       tt[i] = __1__ ;
12    }
13    tt[4] = '\0';                 ----- 마지막에 널 문자를 삽입한다.
14
15    printf("거꾸로 출력한 결과==> %s \n", tt);
16 }
```

정답_ 1 ss[3-i]

실행 결과

```
거꾸로 출력한 결과==> dcba
```

문자열 함수로 문자열 다루기 (1/13)

- **문자열의 길이를 알려주는 함수: strlen( )**

  ➔ string.h를 include해야 사용 가능

기본 8-9 문자열 처리 함수 strlen( ) 사용 예 8-9.c

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <string.h>          ----- 문자열 함수의 목록이 있는 string.h를 포함한다.
03 #include <stdio.h>
04 void main( )
05 {
06    char ss[] = "XYZ";         ----- 문자열 배열과 길이를 저장할 변수이다.
07    int len;
08
09    len = strlen(ss);          ----- 문자열 배열 ss의 길이를 구한다.
10
11    printf("문자열 \"%s\"의 길이 ==> %d \n", ss, len);
12 }                                   └---- 큰따옴표의 내용을 출력하기 위해 \" 문자를 사용한다.
```

실행 결과

```
문자열 "XYZ"의 길이 ==> 3
```

문자열 함수로 문자열 다루기 (2/13)

- **문자열의 길이를 알려주는 함수: strlen( ) (cont'd)**
  - ✓ 6행에서 선언한 배열 ss의 크기는 널 문자를 포함하므로 4로 설정
  - ✓ 9행에서는 strlen( ) 함수를 사용하여 ss의 길이를 구함
  - ✓ 11행에서는 문자열을 "XYZ" 형식으로 출력하고 그 길이도 출력
  - ✓ strlen( ) 함수로 길이를 구할 때는 [그림 8-11]과 같이 널 문자를 제외
  - ✓ 그러므로 배열의 크기는 4이지만 문자열의 길이인 3이 출력

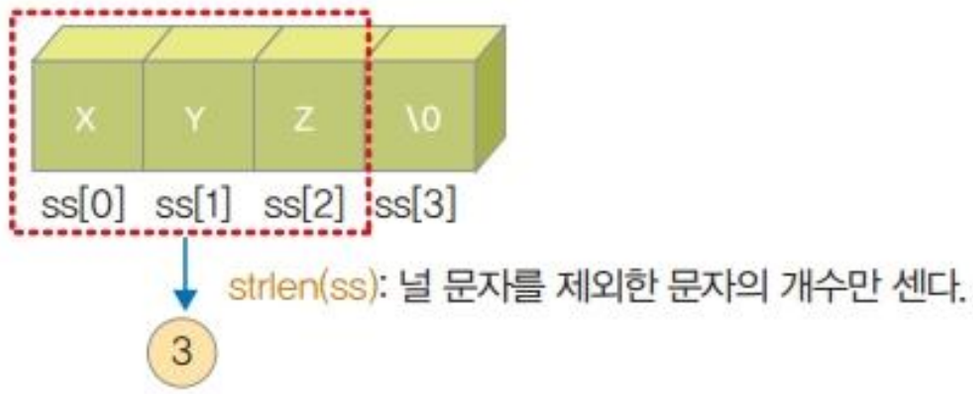

그림 8-11 strlen( ) 함수

문자열 함수로 문자열 다루기 (3/13)

- **문자열을 복사하는 함수: strcpy( )**
  - ✓ strcpy(문자열 배열 A, 문자열 B) 함수는 '문자열 배열 A'에 '문자열 B'를 복사

기본 8-10 문자열 처리 함수 strcpy( ) 사용 예 8-10.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <string.h>
03 #include <stdio.h>
04 void main( )
05 {
06    char ss[4];                    ------ 문자열 배열을 선언한다.
07
08    strcpy(ss, "XYZ");             ------ 배열 ss에 문자열 "XYZ"를 복사한다.
09
10    printf("문자열 ss의 내용 ==> %s \n", ss);
11 }
```

실행 결과

```
문자열 ss의 내용 ==> XYZ
```

문자열 함수로 문자열 다루기 (4/13)

- **문자열을 복사하는 함수: strcpy( ) (cont'd)**
  - ✓ 8행에서는 문자열 "XYZ"의 내용을 배열 ss에 복사
  - ✓ 문자열 상수인 "XYZ"의 맨 뒤에는 문자열의 끝을 나타내는 널 문자가 있으므로 ss의 크기는 4 이상이어야 함

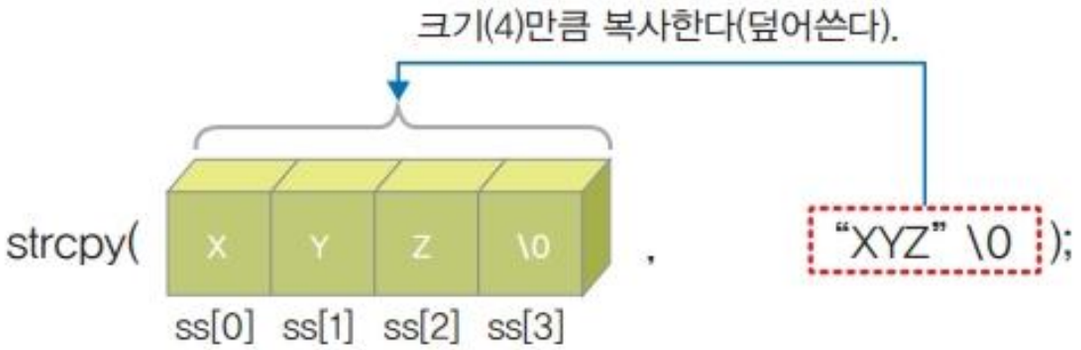

그림 8-12 strcpy( ) 함수

문자열 함수로 문자열 다루기 (5/13)

- **문자열을 복사하는 함수: strcpy( ) (cont'd)**
  - ✓ 다음과 같이 코드를 수정해도 동작할 것인가? YES!

```
char ss[4];
char xx[ ] = "XYZ"
strcpy(ss, xx);
```

- **주의사항**
  - ✓ 문자열이 되기 위해서는 꼭 끝에 널 문자가 들어가야 함
  - ✓ 초기화를 제외하고는 배열에 바로 문자열을 대입할 수 없고, strcpy( ) 함수를 사용해야 함

```
ss = "XYZ"
```

실행결과 ▶ 오류

문자열 함수로 문자열 다루기 (6/13)

- **두 문자열을 이어주는 함수 : strcat( )**
  - ✓ '문자열 배열 A'와 '문자열 B'를 이어 다시 '문자열 배열 A'에 넣음
  - ✓ '문자열 배열 A'의 최대 길이는 '문자열 배열 A와 문자열 B를 합친 길이 +1 이상' 이어야 함

기본 8-11 문자열 처리 함수 strcat( ) 사용 예 8-11.c

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <string.h>
03 #include <stdio.h>
04 void main( )
05 {
06    char ss[7] = "XYZ";                ----- 문자열 배열을 선언하고 초기화한다.
07
08    strcat(ss, "ABC");                 ----- 배열 ss의 내용("XYZ")에 문자열 "ABC"를 이어서 다시
09                                             ss에 대입한다.
10    printf("이어진 문자열 ss의 내용 ==> %s \n", ss);
11 }
```

실행 결과

```
이어진 문자열 ss의 내용 ==> XYZABC
```

문자열 함수로 문자열 다루기 (7/13)

- **두 문자열을 이어주는 함수 : strcat( ) (cont'd)**
  - ✓ strcat( ) 함수는 두 문자열을 그냥 이어주는 원리
  - ✓ 단, 이어주는 자리는 널 문자 자리부터 시작 ➔ 앞 문자 배열의 '\0'은 삭제
  - ✓ 결과적으로 ss는 "XYZABC"

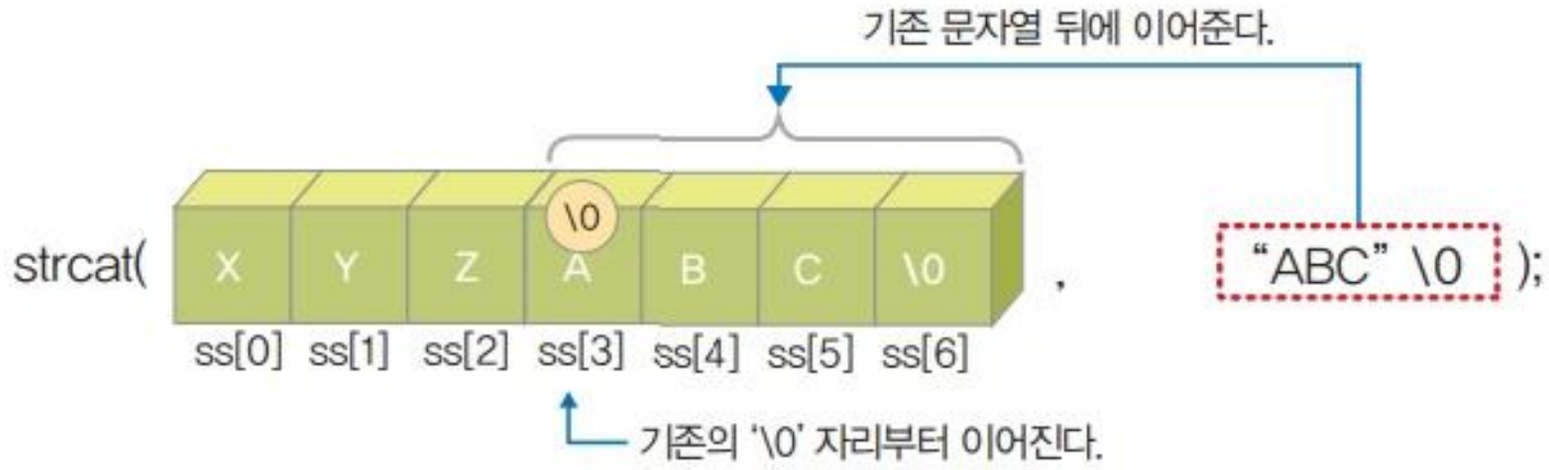

그림 8-13 strcat( ) 함수

문자열 함수로 문자열 다루기 (8/13)

- **두 문자열을 이어주는 함수 : strcat( ) (cont'd)**

여기서 잠깐 strcat( ) 함수 사용 시 주의점

strcat(A, B) 함수의 경우에 A는 꼭 문자형 배열이어야 함

A와 B를 이은 결과를 다시 A(배열)에 넣어야 하기 때문

```
char ss[10] = "XYZ";
char tt[4] = "ABC";
```

```
strcat(ss, tt)       ⇨ (○)
strcat(ss, "ABC")    ⇨ (○)
strcat("ABC", "XYZ") ⇨ (×)
strcat("ABC", ss)    ⇨ (×)
```

문자열 함수로 문자열 다루기 (9/13)

- **두 문자열을 비교하는 함수 : strcmp( )**
  - ✓ strcmp(문자열 A, 문자열 B)는 'A-B'의 결과를 돌려줌
  - ✓ 결과가 0이 나오면 A와 B가 같은 문자열이라는 뜻, 그 외의 값은 두 문자열이 다르다는 의미

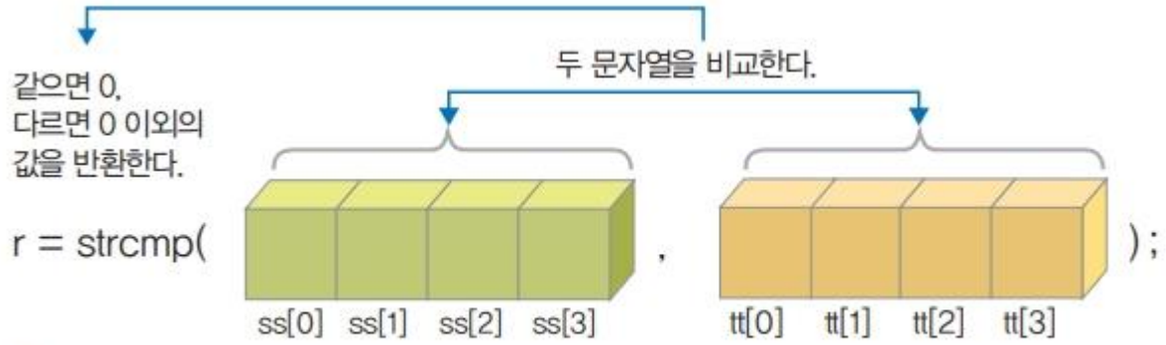

그림 8-14 strcmp( ) 함수

여기서 잠깐 strcmp( ) 함수의 의미

strcmp(ss, tt) 함수는 ss의 아스키코드 값에서 tt의 아스키코드 값을 뺌

0 이외의 값은 두 문자열의 아스키코드 값 차이를 나타내는데, 그다지 활용할 일은 없고 단지 두 문자열이 다르다는 뜻

문자열 함수로 문자열 다루기 (10/13)

- **문자열을 입력받는 함수 : gets_s( )**
  - ✓ scanf()와 비슷한 기능으로, 문자열 입력 시 상대적으로 유용
    - scanf( )는 공백이 나타나면 문자열 입력을 종료
    - gets_s( )는 "\n"이 나올때 까지(즉, 사용자가 Enter 키를 입력할 때 까지) 입력 받음

    ➔ 문자열 중간에 있는 공백도 문자열로 입력 받을 수 있음
  - ✓ gets( )는 메모리 오버플로우 문제때문에 사용하지 않음
  - ✓ 최대 입력 문자는 널 문자를 고려해서 '배열크기 –1'까지 입력
    - 마지막의 '\n'은 널 문자로 대체됨

```
char ss[10];
gets_s(ss, 10);
//gets_s(ss, sizeof(ss));
```

문자열 함수로 문자열 다루기 (11/13)

- **문자열을 출력하는 함수 : puts( )**
  - ✓ printf()와 비슷한 기능으로, 문자열 출력 시 상대적으로 유용
  - ✓ '\n'이 없어도 출력한 후 자동으로 줄을 넘김

```
char ss[] = "XYZ"
puts(ss);
```

문자열 함수로 문자열 다루기 (12/13)

- **문자열 입출력 함수 사용 예**

응용 8-13 문자열 입출력 함수 gets( ), puts( ) 사용 예 8-13.c

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <string.h>
03 #include <stdio.h>
04 void main( )
05 {
06    char ss[20];                          ----- 문자형 배열 ss와 tt를 선언한다.
07    char tt[20];
08    int r1, r2;
09
10    puts("첫 번째 문자열을 입력하세요.");    ----- 배열 ss와 tt에 문자열을 입력한다.
11    gets_s(ss, sizeof(ss));
12
13    puts("두 번째 문자열을 입력하세요.");
14    gets_s(tt, sizeof(tt));
15
16    r1 = strlen(ss);                      ----- 배열 ss와 tt의 문자열 길이를
17    r2 = strlen(tt);                            저장한다.
18
```

문자열 함수로 문자열 다루기 (13/13)

- **문자열 입출력 함수 사용 예 (cont'd)**

```
19    printf("첫 번째 문자열의 길이 ==> %d \n", r1);
20    printf("두 번째 문자열의 길이 ==> %d \n", r2);
21
22    if (strcmp(ss, tt) == 0)
23       puts("두 문자열의 내용이 같습니다.\n");
24    else
25       puts("두 문자열의 내용이 다릅니다.\n");
26 }
```

----- 각 배열의 문자열 길이를 출력한다.

----- ss와 tt의 문자열이 같은지 비교한다.

정답_ 1 gets(tt); 2 strcmp(ss, tt)

실행 결과

```
첫 번째 문자열을 입력하세요.
IT CookBook
두 번째 문자열을 입력하세요.
Hanbit
첫 번째 문자열의 길이 ==> 11
두 번째 문자열의 길이 ==> 6
두 문자열의 내용이 다릅니다.
```

예 제

[예제 01] 사용자가 입력한 정수를 역순으로 출력

1. 사용자로부터 숫자들을 입력 받고, 입력 받은 숫자들을 배열(크기가 10)에 저장
2. 사용자가 0을 입력하거나 배열에 정수가 다 찾을 경우 입력을 종료 (0도 배열에 저장)
3. sizeof( )를 이용하여 배열의 크기 계산
4. 반복문을 이용하여 배열에 저장된 정수들을 역순으로 출력

→ 0 또는 배열의 크기까지 출력

[예제 02] 입력된 문자열을 반대 순서로 출력

예제 설명 문자열 배열을 이용해서 입력받은 문자열을 반대 순서로 출력하는 프로그램이다.

```
문자열을 입력하세요: Hanbit
tibnaH
```

1. 입력과 출력에서는 gets_s( )와 puts( )를 사용
2. 두 개의 문자열 배열을 생성하고, 반복문을 이용하여 글자를 거꾸로 저장하고 출력
3. 문자열의 길이를 파악하기 위해서는 strlen( )를 사용

[예제 03] 문자열 내 특정 문자의 변환

예제 설명 문자열을 입력받고 그 문자열에서 변환할 기존 문자와 새로운 문자를 각각 입력받은 뒤 변환된 문자열을 반환하는 프로그램이다.

실행 결과

```
여러 글자를 입력 : Microsoft Visual Studio Community
기존 문자와 새로운 문자 : i  #
변환된 결과 ==> M#crosoft V#sual Stud#o Commun#ty
```

1. 입력에서는 gets_s( ), scanf( )를 사용
2. 하나의 문자열 배열을 생성하고, 해당 배열에 변환된 결과를 저장하고 출력
3. 문자열의 길이를 파악하기 위해서는 strlen( )를 사용

Q & A