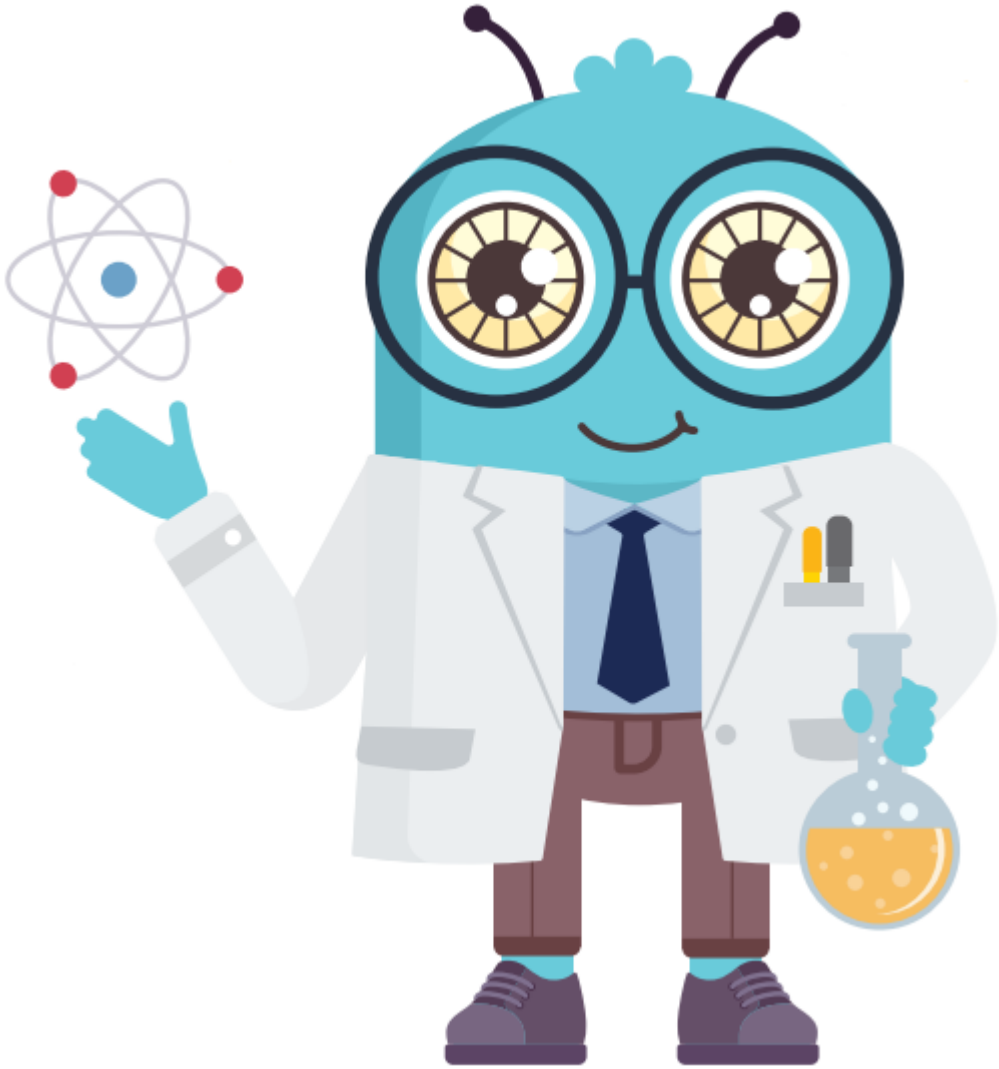

Chapter 08

배열

목차

01
배열의 이해

1. 배열의 이해

1. 배열을 사용하는 이유

▪ 배열의 개념

- 여러 개의 변수를 나란히 연결하는 개념
- 박스(변수)를 한 줄로 붙이고, 박스의 이름(aa)을 지정
- 각각의 박스는 aa[0], aa[1], ... 과 같이 첨자를 붙임

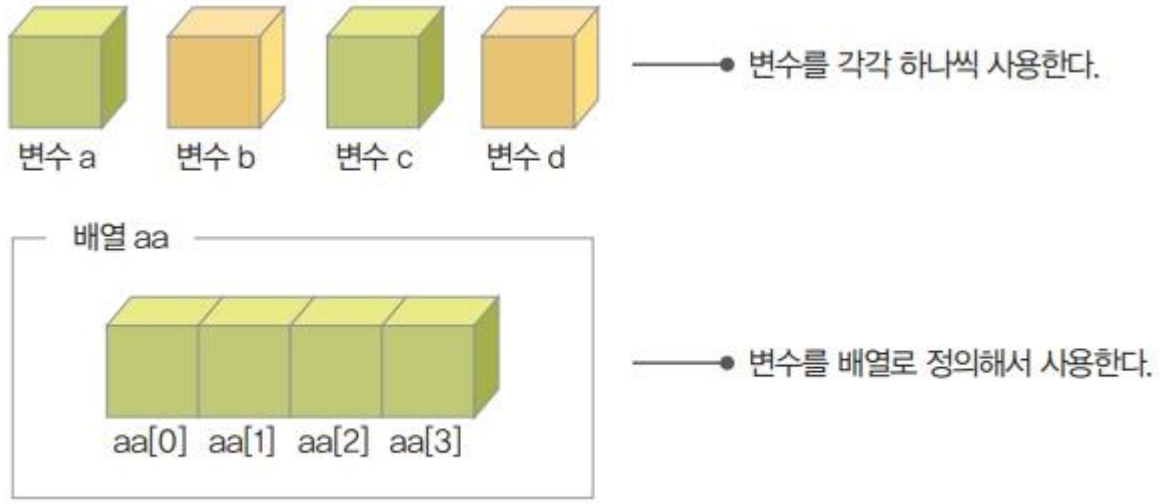

그림 8-1 배열의 개념

1. 배열의 이해

1. 배열을 사용하는 이유

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
```

1. 배열의 이해

1. 배열을 사용하는 이유

▪ 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
- 배열의 경우에는 첨자를 사용하여 aa[0], aa[1], aa[2], aa[3]과 같이 선언
- 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]

1. 배열의 이해

1. 배열을 사용하는 이유

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
```

1. 배열의 이해

2. 배열의 활용 범위

- 배열의 첨자가 순서대로 변할 수 있도록 반복문과 함께 활용해야만 배열의 효율성이 극대화

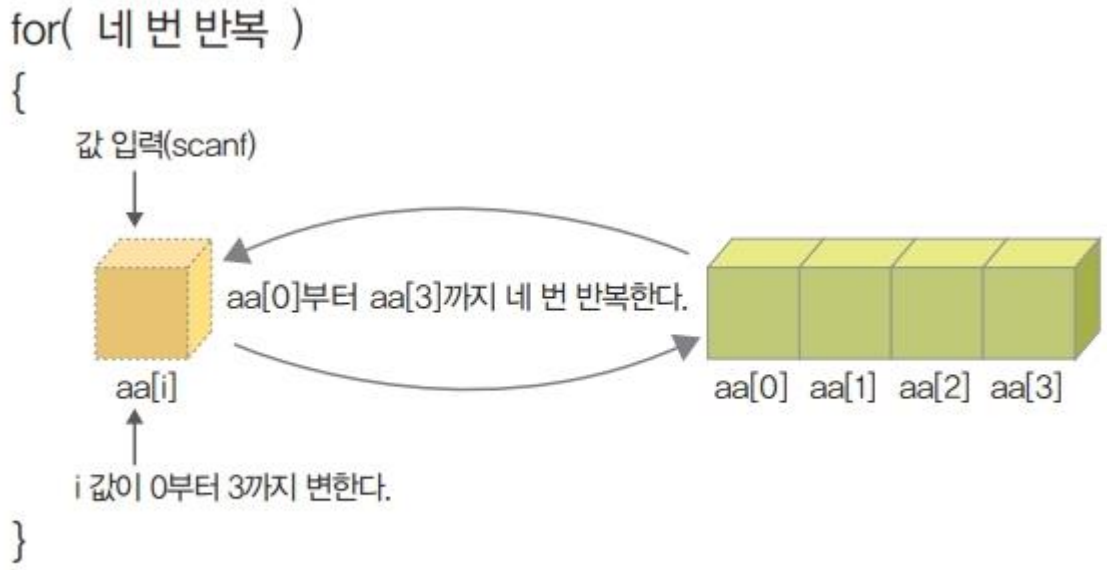

그림 8-2 for문을 활용한 배열값 입력

- for문을 네 번 돌면서 aa[i]의 첨자가 aa[0]~aa[3]으로 변하게 하면 변수 4개에 자동으로 값이 입력

1. 배열의 이해

2. 배열의 활용 범위

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 }
```

실행 결과

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

2. 배열의 활용 범위

- 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 차 례대로 입력받음
- 15행에서는 변수 4개를 더했는데, 만약 배열이 100개라면 'hap = aa[0] + aa[1] + … aa[99]'로 코딩을 해야 함
- 이럴 때는 다음과 같이 for문으로 변경하는 것이 바람직

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

1. 배열의 이해

- **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} ;
```

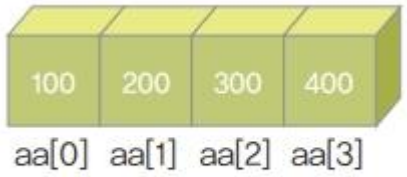

그림 8-3 배열의 초기화 1

  - 이때 배열의 개수(첨자)를 반드시 지정하지 않아도 됨
  - 개수를 지정하지 않으면 블록({ }) 안 의 초깃값 개수에 따라 자동으로 배열의 개수가 정해지기 때문

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

  - 배열을 선언하기만 하고 초기화하지 않으면 각 배열에 아무것도 넣지 않았기 때문에 쓰레기 값이 들어감

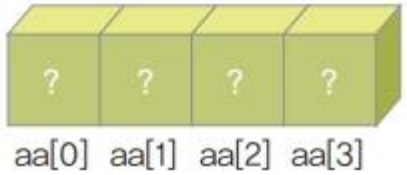

그림 8-4 배열의 초기화 2

1. 배열의 이해

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

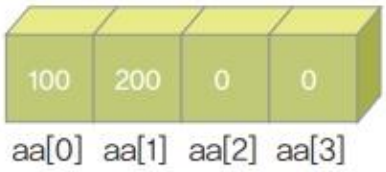

그림 8-5 배열의 초기화 3

- 초깃값을 0이라고 직접 써도 되고 초기화할 부분을 비워놓아도 됨

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

- 배열 1000개를 모두 0으로 초기화하고 싶다면

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

- 배열의 개수보다 초기화할 값의 개수가 많다면 마지막의 500이 들어갈 곳이 없기 때문에 컴파일 오류가 발생

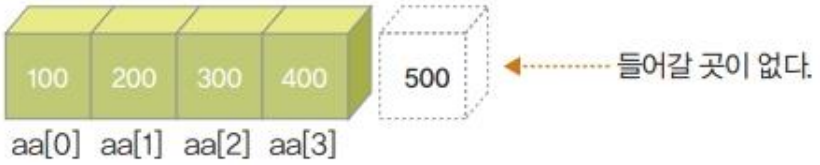

그림 8-6 배열의 초기화 4

1. 배열의 이해

2. 배열의 활용 범위

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
```

1. 배열의 이해

2. 배열의 활용 범위

```
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
```

1. 배열의 이해

2. 배열의 활용 범위

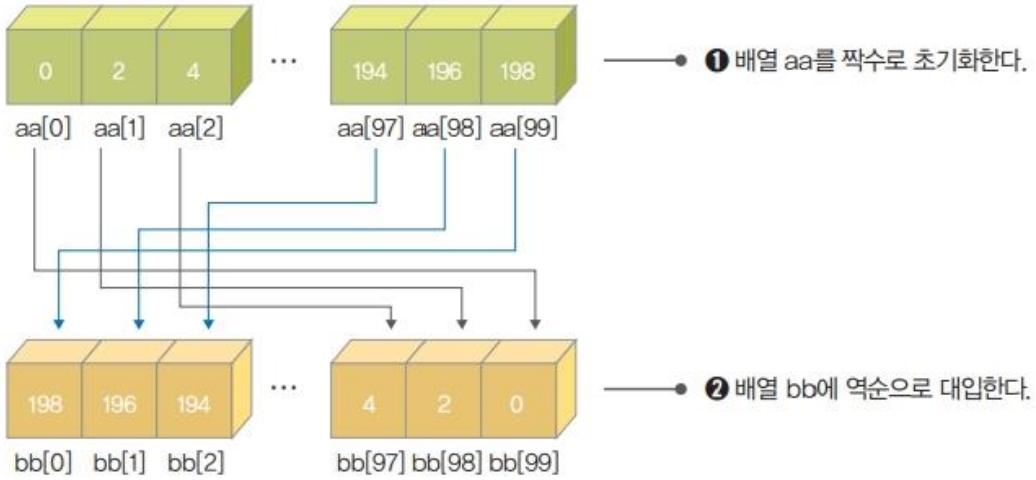

그림 8-7 배열의 초기화 5

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
```

1. 배열의 이해

- **배열의 크기 알아내기**
  - sizeof( ) 함수 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

1. 배열의 이해

- 배열의 크기 알아내기

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

2. 배열과 문자열

1. 정수형 배열과 문자형 배열

- 정수형 배열은 각각의 배열 요소에 정수(100, 200, 300, 400)를 입력하고, 문자형 배열은 각각의 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'이라고 할 수 있음

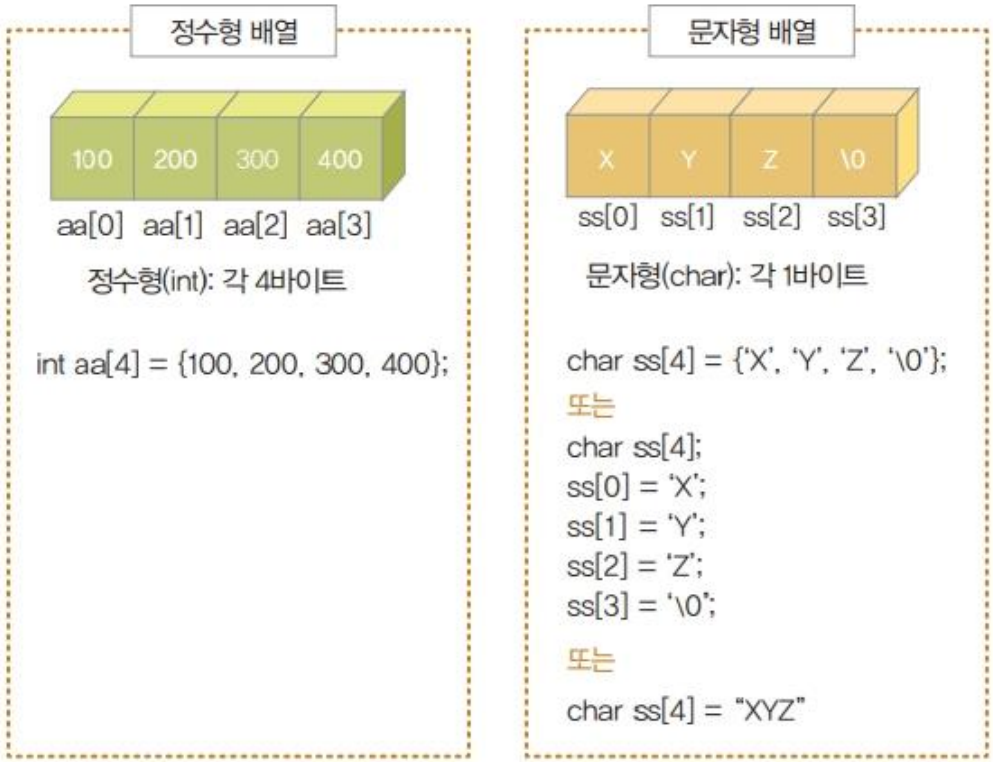

그림 8-9 정수형 배열과 문자형 배열

2. 배열과 문자열

1. 정수형 배열과 문자형 배열

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 }
```

05 ----- 크기가 8인 문자형 배열을 선언하고 초기화한다.

08 ----- 여섯 번째 문자를 바꾼다.

10 ----- 여덟 번 반복하면서 배열 ss의 각 문자를 출력한다.

15 ----- 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
```

2. 배열과 문자열

1. 정수형 배열과 문자형 배열

- [기본 8-7]의 5행에서 "Basic-C"라는 일곱 글자를 넣기 위해 널 문자('₩0') 자리까지 포함해서 여덟 자리의 배열을 정의

- [그림 8-10]의 ❸은 문자 1개를 출력하는 방식, ❹는 전체 문자열을 출력하는 방식

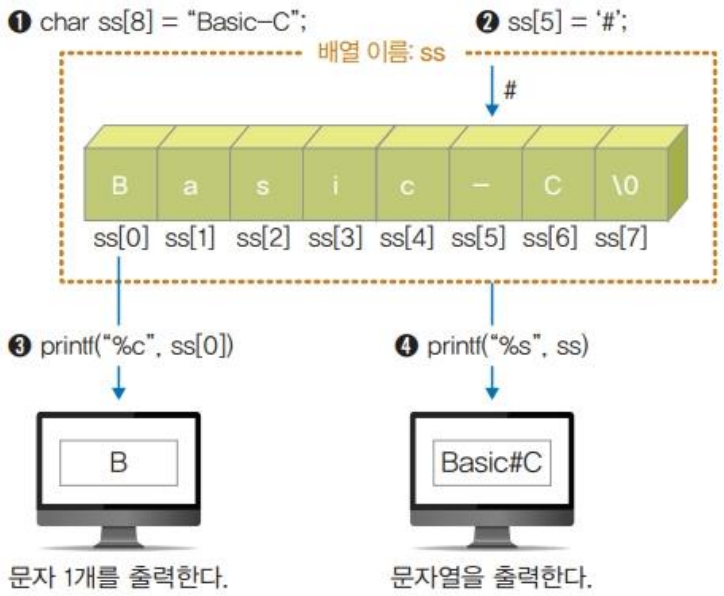

그림 8-10 문자 출력과 문자열 출력

2. 배열과 문자열

1. 정수형 배열과 문자형 배열

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
```

2. 문자열 함수로 문자열 다루기

▪ 문자열 처리 함수

- 문자열 함수를 사용하려면 소스를 시작하는 부분에 다음과 같은 구문을 써야 함

```
#include <string.h>
```

- 이는 문자열 함수의 목록이 정의된 string.h 파일을 포함하라는 의미

2. 문자열 함수로 문자열 다루기

- 문자열의 길이를 알려주는 함수: strlen( )

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);
                                       └----- 큰따옴표의 내용을 출력하기 위해 \" 문자를 사용한다.
12 }
```

실행 결과

```
문자열 "XYZ"의 길이 ==> 3
```

2. 문자열 함수로 문자열 다루기

- **문자열의 길이를 알려주는 함수: strlen( )**
  - 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

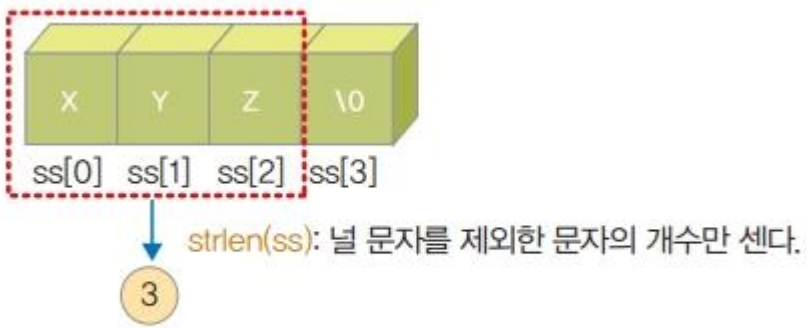

그림 8-11 strlen( ) 함수

2. 문자열 함수로 문자열 다루기

▪ 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
```

2. 문자열 함수로 문자열 다루기

- **문자열을 복사하는 함수: strcpy( )**
  - 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

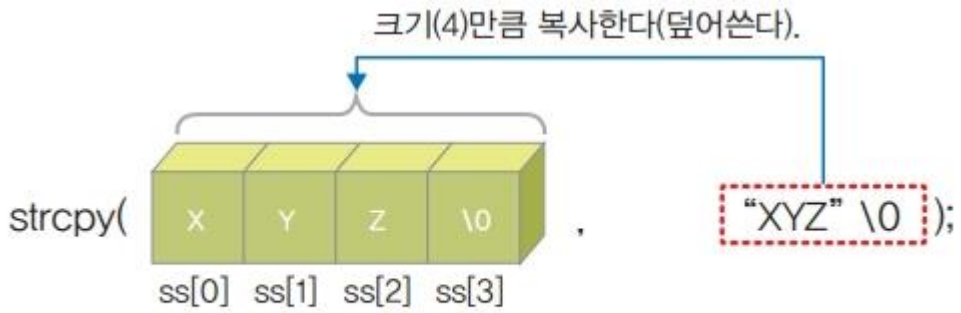

그림 8-12 strcpy( ) 함수

2. 배열과 문자열

- strcpy( ) 함수는 이미 선언된 문자열 배열에 다른 문자열을 대입하고 싶을 때 주로 사용
  - 오류

```
ss = "XYZ"
```

실행결과 ▶ 오류

  - 올바른 사용(문자열을 바로 배열에 대입할 수 없으므로 strcpy( ) 함수를 사용)

```
char ss[4] = "XYZ"
```

실행결과 ▶ 가능

  - 한 글자씩 대입할 수도 있음

```
char ss[4];
ss[0] = 'X';
ss[1] = 'Y';
ss[2] = 'Z';
ss[3] = '\0';
```

2. 배열과 문자열

- **두 문자열을 이어주는 함수 : strcat_s( )**
  - '문자열 배열 A'와 '문자열 B'를 이어 다시 '문자열 배열 A'에 넣음

    (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
```

2. 배열과 문자열

- **두 문자열을 이어주는 함수 : strcat_s( )**
  - strcat( ) 함수는 두 문자열을 그냥 이어주는 원리
  - 단, 이어주는 자리는 널 문자 자리부터 시작
  - 결과적으로 ss는 "XYZABC"

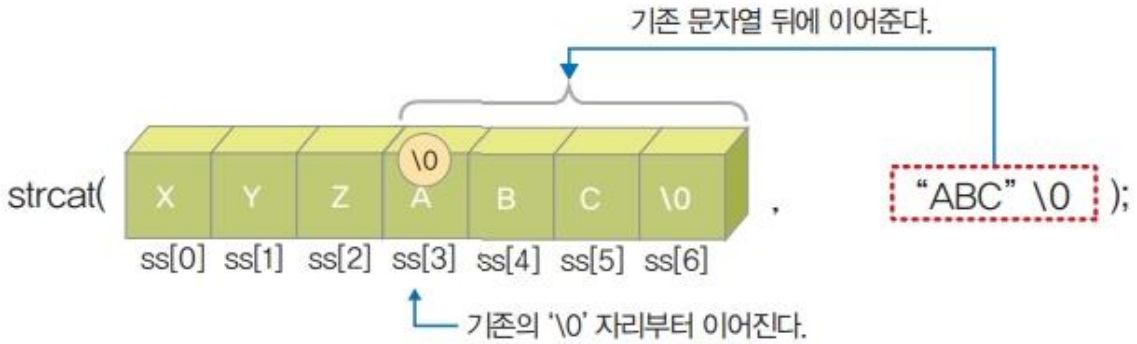

그림 8-13 strcat( ) 함수

2. 배열과 문자열

- **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

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)    ⇨ (×)
```

2. 배열과 문자열

- **두 문자열을 비교하는 함수 : strcmp( )**
  - strcmp( ) 함수는 두 문자열을 비교해서 같으면 0, 다르면 그 외의 숫자를 돌려줌
  - 주로 두 문자열이 같은지 확인할 때 사용

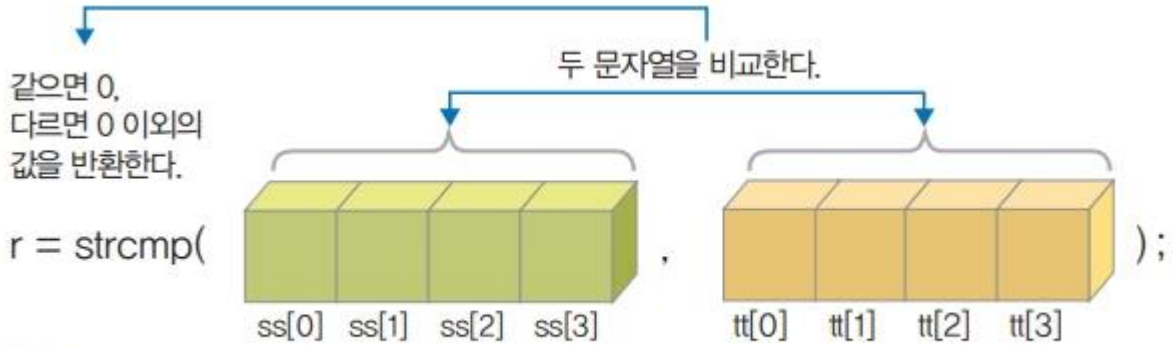

그림 8-14 strcmp( ) 함수

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

2. 배열과 문자열

- **문자열 입출력 함수**
- **문자열을 입력받는 함수 : gets( )**
  - scanf()와 비슷한 기능으로, 문자열 입력 시 상대적으로 유용
  - 최대 입력 문자는 널 문자를 고려해서 '배열크기 –1'까지 입력
  - Enter 키를 입력할 때까지 ss에 문자열을 받아들임

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

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

```
char ss[10] = "XYZ";
puts(ss);
```

2. 배열과 문자열

- **문자열 입출력 함수**

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(ss);
12
13    puts("두 번째 문자열을 입력하세요.");
14    __1__
15
16    r1 = strlen(ss);                      ----- 배열 ss와 tt의 문자열 길이를
17    r2 = strlen(tt);                            저장한다.
18
```

2. 배열과 문자열

- **문자열 입출력 함수**

```
19    printf("첫 번째 문자열의 길이 ==> %d \n", r1);
20    printf("두 번째 문자열의 길이 ==> %d \n", r2);
21
22    if( __2__ == 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.
```

03
2차원 배열

3. 2차원 배열

1. 2차원 배열의 기본 개념

- 1차원 배열을 여러 개 연결한 것으로, 두 개의 첨자 사용
- 'int aa[3];'으로 정의했다면 aa [0], aa[1], aa[2]라는 요소가 생성

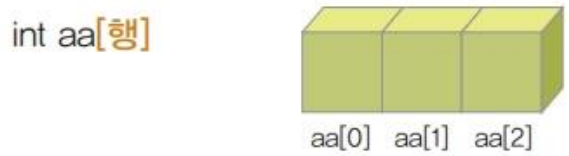

그림 8-15 1차원 배열의 개념

- 이를 확장해서 2차원 배열 'int aa[3][4]'를 정의
- 이때 앞의 3은 가로줄 수를, 뒤의 4는 세로줄 수를 의미(3행 4열짜리 배열이 생성)

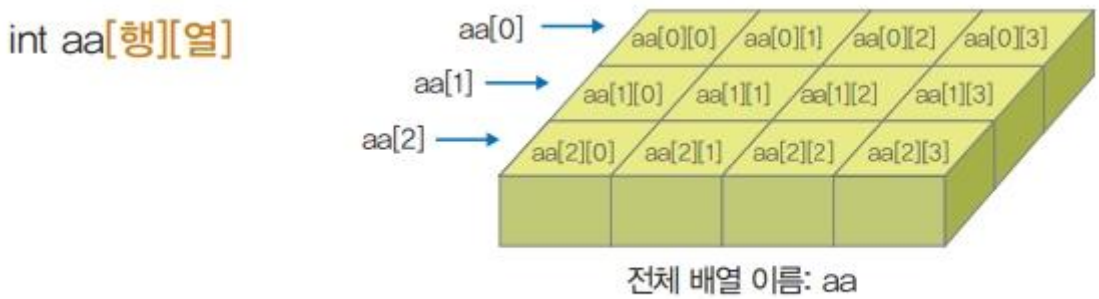

그림 8-16 2차원 배열의 개념

3. 2차원 배열

1. 2차원 배열의 기본 개념

기본 8-14 2차원 배열 사용 예 1 — 8-14.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int aa[3][4];                                          ---- 2차원 배열을 선언한다.
06
07    aa[0][0] = 1; aa[0][1] = 2; aa[0][2] = 3; aa[0][3] = 4;   ---- 각 요소에 값을 대입한다.
08    aa[1][0] = 5; aa[1][1] = 6; aa[1][2] = 7; aa[1][3] = 8;
09    aa[2][0] = 9; aa[2][1] =10; aa[2][2] =11; aa[2][3] =12;
10
11    printf("aa[0][0]부터 aa[2][3]까지 출력 \n");
12
13    printf("%3d %3d %3d %3d\n", aa[0][0], aa[0][1], aa[0][2], aa[0][3]);
14    printf("%3d %3d %3d %3d\n", aa[1][0], aa[1][1], aa[1][2], aa[1][3]);
15    printf("%3d %3d %3d %3d\n", aa[2][0], aa[2][1], aa[2][2], aa[2][3]);
16 }                                                          ---- 배열의 내용을 출력한다.
```

실행 결과

```
aa[0][0]부터 aa[2][3]까지 출력
  1   2   3   4
  5   6   7   8
  9  10  11  12
```

3. 2차원 배열

1. 2차원 배열의 기본 개념

응용 8-15 2차원 배열 사용 예 2 — 8-15.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int aa[3][4];
06    int i, k;
07
08    int val=1;
09
10    for( i=0; i < 3; i++ )
11    {
12       for( __1__ )
13       {
14          aa[i][k] = val;
15          val++;
16       }
17    }
```

- 05–06: 2차원 배열과 첨자 변수를 선언한다.
- 08: 배열에 들어갈 값을 초기화한다.
- 10–11: 바깥 for문을 세 번 반복한다. 즉 앞 첨자가 행 단위로 변경된다.
- 12–13: 안쪽 for문을 네 번 반복한다. 즉 뒤 첨자가 열 단위로 변경된다.
- 14–15: 배열에 val 값을 입력한 후 1 증가시킨다.

3. 2차원 배열

1. 2차원 배열의 기본 개념

```
18
19    printf("aa[0][0]부터 aa[2][3]까지 출력 \n");
20
21    for( i=0; i < 3; i++ )                    ---- 입력과 동일한 개념으로 12회 출력한다.
22    {
23       for( k=0; k < 4; k++ )
24       {
25          printf("%3d ", __2__ );
26       }
27       printf("\n");                          ------ 한 행을 출력한 후 줄을 넘긴다.
28    }
29 }
```

정답_ 1 k=0; k<4; k++ 2 aa[i][k]

실행 결과

```
aa[0][0]부터 aa[2][3]까지 출력
  1   2   3   4
  5   6   7   8
  9  10  11  12
```

3. 2차원 배열

2. 2차원 배열의 초기화

기본 8-16 2차원 배열의 초기화 예 8-16.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int aa[3][4] = {                  ----- 2차원 배열을 초기화한다.
06       { 1, 2, 3, 4 },
07       { 5, 6, 7, 8 },
08       { 9,10,11,12 }
09    };
10
11    int i, k;
12    printf("aa[0][0]부터 aa[2][3]까지 출력 \n");
13
```

3. 2차원 배열

2. 2차원 배열의 초기화

```
14    for( i=0; i < 3; i++ )
15    {
16       for( k=0; k < 4; k++ )
17       {
18          printf("%3d", aa[i][k]);
19       }
20       printf("\n");
21    }
22 }
```

----- 2차원 배열에 저장된 값을 출력한다.

실행 결과

```
aa[0][0]부터 aa[2][3]까지 출력
 1  2  3  4
 5  6  7  8
 9 10 11 12
```

2. 2차원 배열의 초기화

- 3행 4열의 배열이므로 [그림 8-17] 과 같이 초기화

그림 8-17 2차원 배열의 초기화

3. 2차원 배열

3. 3차원 이상의 배열

- 2차원 배열 위에 또 다른 2차원 배열을 쌓은 것

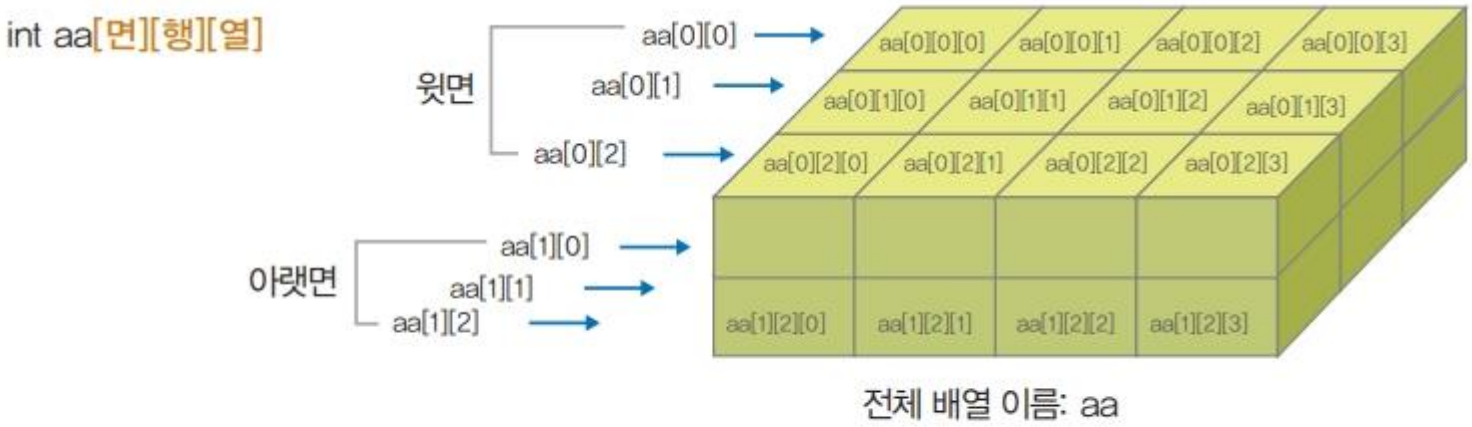

그림 8-18 3차원 배열의 개념

3. 2차원 배열

- **3차원 배열의 초기화**
  - 2차원 배열의 초기화를 한번 더 하는 개념
  - 콤마로 분리하고, 전체를 다시 블록으로 묶음

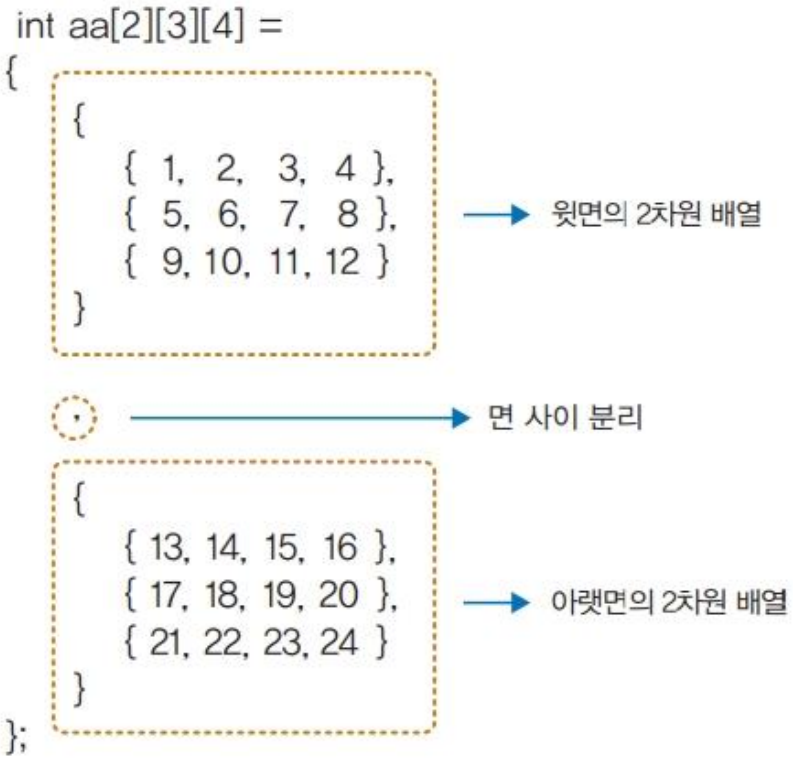

그림 8-19 3차원 배열의 초기화

*

예제 모음

[예제모음 20] 입력된 문자열을 반대 순서로 출력

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

실행 결과

```
문자열을 입력하세요 : Hanbit
내용을 거꾸로 출력 ==> tibnaH
```

[예제모음 20] 입력된 문자열을 반대 순서로 출력

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <string.h>
03 #include <stdio.h>
04 void main( )
05 {
06    char ss[100];          ----- 문자형 배열 ss를 선언한다.
07    char tt[100];          ----- 문자형 배열 tt를 선언한다.
08    int count, i;
09
10    printf("문자열을 입력하세요 : ");
11    scanf("%s", ss);       ----- 문자열을 입력받는다.
12
13    count = strlen(ss);    ----- 입력받은 문자열의 개수를 구한다.
14
15    for(i=0; i < count; i++)   ----- 문자열의 개수만큼 반복해서 tt 배열에
16    {                              문자열을 반대 순서로 저장한다.
17       tt[i] = ss[count-(i+1)];
18    }
19    tt[count] = '\0';      ------ tt 배열의 마지막에 널 문자를 입력한다.
20
21    printf("내용을 거꾸로 출력 ==> %s \n", tt);
22 }
```

[예제모음 21] 대문자와 소문자의 변환

예제 설명 입력된 문자열이 대문자이면 소문자로, 소문자이면 대문자로 변환하고 그 외의 문자는 그대로 출력하는 프로그램이다.

실행 결과

```
문자 입력 : Hello, C Language is Funny ~~~
변환된 문자 =>hELLO, c lANGUAGE IS fUNNY ~~~
```

[예제모음 21] 대문자와 소문자의 변환

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <string.h>
03 #include <stdio.h>
04 void main( )
05 {
06    char in[50], out[50];          ----- 입력 문자형 배열 in과 출력 문자형 배열 out이다.
07    int i, len;
08    int diff = 'a' - 'A';          ----- 대문자와 소문자의 값 차이를 diff에 저장한다.
09
10    printf(" 문자 입력 : ");
11    gets(in);                      ----- 문자열을 입력받는다. 실제 최대 입력 문자는
12                                         '배열 크기-1'이다.
13    len = strlen(in);              ----- 입력된 문자열의 길이를 구한다.
14
15    for(i=0; i < len; i++)
16    {
17       if( ('A' <= in[i]) && (in[i] <= 'Z') )   ----- 문자가 대문자이면 대·소문자
18          out[i] = in[i] + diff;                      차이값을 더한다.
19       else if( ('a' <= in[i]) && (in[i] <= 'z') )  ----- 문자가 소문자이면 대·소문자
20          out[i] = in[i] - diff;                      차이값을 뺀다.
21       else                        ----- 영문자가 아닌 기호, 숫자 등은 그대로 둔다.
22          out[i] = in[i];
23    }
24    out[i] = '\0';                 ----- 마지막에 널 문자를 입력한다.
25
26    printf(" 변환된 문자 =>%s \n", out);
27 }
```

[예제모음 22] 구구단의 결과를 2차원 배열에 저장

예제 설명 구구단의 결과를 2차원 배열에 저장한 후 출력하는 프로그램이다.

실행 결과

```
1X1= 1  2X1= 2  3X1= 3  4X1= 4  5X1= 5  6X1= 6  7X1= 7  8X1= 8  9X1= 9
1X2= 2  2X2= 4  3X2= 6  4X2= 8  5X2=10  6X2=12  7X2=14  8X2=16  9X2=18
1X3= 3  2X3= 6  3X3= 9  4X3=12  5X3=15  6X3=18  7X3=21  8X3=24  9X3=27
1X4= 4  2X4= 8  3X4=12  4X4=16  5X4=20  6X4=24  7X4=28  8X4=32  9X4=36
1X5= 5  2X5=10  3X5=15  4X5=20  5X5=25  6X5=30  7X5=35  8X5=40  9X5=45
1X6= 6  2X6=12  3X6=18  4X6=24  5X6=30  6X6=36  7X6=42  8X6=48  9X6=54
1X7= 7  2X7=14  3X7=21  4X7=28  5X7=35  6X7=42  7X7=49  8X7=56  9X7=63
1X8= 8  2X8=16  3X8=24  4X8=32  5X8=40  6X8=48  7X8=56  8X8=64  9X8=72
1X9= 9  2X9=18  3X9=27  4X9=36  5X9=45  6X9=54  7X9=63  8X9=72  9X9=81
```

[예제모음 22] 구구단의 결과를 2차원 배열에 저장

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int gugu[9][9];
06    int i, k;
07
08    for(i=0; i < 9; i++)
09       for(k=0; k < 9; k++)
10          gugu[i][k] = (i+1) * (k+1);
11
12    for(i=0; i < 9; i++)
13    {
14       for(k=0; k < 9; k++)
15       {
16          printf("%dX%d=%2d  ", k+1, i+1, gugu[i][k]);
17       }
18       printf("\n");
19    }
20 }
```

----- 문자형 2차원 배열 gugu와 첨자 변수 i, k를 선언한다.

----- 구구단을 곱한 결과를 2차원 배열에 저장한다. i, k가 0부터 시작되므로 1을 더해서 곱한다.

----- 구구단 결과를 출력한다.

------ 한 행을 출력한 후 줄을 넘긴다.

[예제모음 23] 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
```

[예제모음 23] 문자열 내 특정 문자의 변환

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <string.h>
03 #include <stdio.h>
04 void main( )
05 {
06    char str[100];          ----- 문자형 배열 str을 선언한다.
07    char ch1, ch2;          ----- 기존 문자와 새 문자를 위한 문자형 변수이다.
08    int i;
09
10    printf("여러 글자를 입력 : ");
11    gets(str);              ------ 최대 99자를 입력받는다.
12
13    printf("기존 문자와 새로운 문자 : ");
14    scanf("%c %c", &ch1, &ch2);   ------ 기존 문자(ch1)와 새 문자(ch2)를 한 글자씩
15                                       입력받는다(띄어쓰기로 구분).
16    for(i=0; i < strlen(str); i++)
17    {                       ------ 문자열의 길이만큼 반복하면서 기존 문자(ch1)가
18       if(str[i] == ch1)            있으면 새 문자(ch2)로 교체한다.
19          str[i] = ch2;
20    }
21
22    printf("변환된 결과 ==> %s \n", str);
23 }
```

감사합니다!