C프로그래밍

12 포인터와 배열, 함수

공지 사항

- **코딩테스트**
  - ✓ 6/2(월) 문제 공지, 6/9(월) 강의시간에 코딩테스트 수행

- **기말고사**
  - ✓ 6/14(토) 13:00 ~ 15:00, 장소 추후공지
  - ✓ 시험범위: 1주차 ~ 13주차 강의내용

01
포인터 review

포인터 review (1/3)

❶ 포인터 변수가 어떤 데이터 타입을 저장하는 곳의 주소를 나타내는지 확인

❷ 포인터 변수를 선언할 때는 변수 앞에 *만 붙임

```
int *a;
char *str;
float *b;
```

❸ 포인터 변수에는 꼭 주솟값을 넣어야 함

- ✓ 일반 변수 이름 앞에 '&' 를 붙임
- ✓ 배열의 이름은 그 자체가 주소이므로 '&'를 붙이지 않음

❶ 변수인 경우

```
int a;
int *p;


p = &a;
```

❷ 배열인 경우

```
int a[10];
int *p;


p = a;
```

포인터 review (2/3)

❹ 포인터가 가리키는 곳의 실제값을 구하려면 *를 붙임 (1/2)

✓ 포인터 변수 p가 변수 a가 들어있는 주소인 1016번지를 가리킨다고 가정

```
int a=123;
int *p;
p = &a;
```

`printf("%d", p)`	실행결과 ▶	a의 주소인 1016이 출력된다(주소 자체는 중요하지 않다).
`printf("%d", *p)`	실행결과 ▶	p가 가리키는 곳의 실제값, 즉 a값인 123이 출력된다.

그림 9-16 포인터가 가리키는 실제 값

포인터 review (3/3)

❹ 포인터가 가리키는 곳의 실제값을 구하려면 *를 붙임 (2/2)

✓ *p에는 임의의 값을 대입할 수 있지만, p에는 오직 주소만 들어간다는 점에 주의

```
int a=123;
int *p;
```

`p=&a;`	실행결과 ▶	p에 a의 주솟값을 대입한다.(O)
`*p = 1`	실행결과 ▶	p가 가리키는 주소의 실제값을 정수 1로 바꾼다.(O)
`p = 1`	실행결과 ▶	p에 1번지라는 주솟값을 직접 넣으라는 의미다. 그런데 과연 컴퓨터의 메모리에 1번지라는 주소가 있을까? 또 있다고 해도 전혀 엉뚱한 위치가 된다.(×)

02
포인터와 배열

문자형 배열과 포인터 (1/3)

- **문자형 배열과 포인터의 관계 예제**

기본 9-8 문자형 배열과 포인터의 관계 1 9-8.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    char s[8]= "Basic-C";        ------ 문자형 배열을 선언하고 초깃값을 대입한다.
06    char  *p;                    ------ 문자형 포인터 변수를 선언한다.
07
08    p = s;                       ------ p에 배열 s의 주소를 대입한다.
09
10    printf("&s[3] ==> %s\n", &s[3]);   ------ 문자열과 포인터의 주솟값을 %s로 출력한다.
11    printf("p+3   ==> %s\n\n", p+3);
12
13    printf("s[3]   ==> %c\n", s[3]);   ------ 문자와 포인터의 실제 값을 %c로 출력한다.
14    printf("*(p+3) ==> %c\n", *(p+3));
15 }
```

실행 결과

```
&s[3] ==> ic-C
p+3   ==> ic-C

s[3]   ==> i
*(p+3) ==> i
```

문자형 배열과 포인터 (2/3)

- **기본 9-8 review (1/2)**
  - ✓ 5행에서 여덟 자리 문자형 배열 s를 선언하고 “Basic-C”라는 문자열로 초기화
  - ✓ 6행에서 문자형 포인터 변수 p를 선언하고 8행에서 p에 배열 s의 주솟값인 s를 대입
  - ✓ [기본 9-8]의 변수와 포인터의 관계를 그림으로 나타내면 다음과 같음

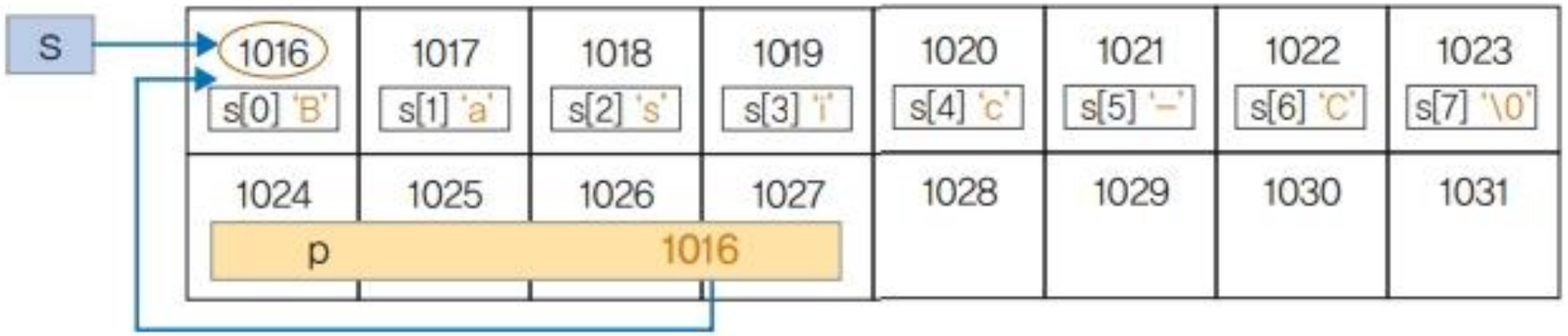


그림 9-14 [기본 9-8]의 변수와 포인터의 관계

문자형 배열과 포인터 (3/3)

- **기본 9-8 review (2/2)**
  - ✓ 현재 포인터 변수 p에는 1016이 들어 있으므로 11행의 p+3은 거기서 세 칸을 건너뛴 1019
  - ✓ 1019번지에는 i가 들어 있으므로 10행과 마찬가지로 'ic-C'가 출력
  - ✓ 13행에서는 s[3]을 printf("%c")로 출력하니 당연히 'i'가 출력
  - ✓ 14행의 *(p+3) 역시 p에서 세 칸을 건너뛴 주소의 실제 값을 의미하므로 1019번지의 실제 값인 'i'가 출력

표 9-2 배열의 포인터 표현

```
char s[3];
char* p;
p = s;
```

배열 첨자	포인터 상수	포인터 변수
s[0]	*(s+0)	*(p+0)
s[1]	*(s+1)	*(p+1)
s[2]	*(s+2)	*(p+2)

문자열 배열과 포인터 응용 (1/2)

- **포인터를 이용하여 배열에 저장되어 있는 문자열을 역순으로 출력**

응용 9-9 문자형 배열과 포인터의 관계 2 9-9.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    char s[8]= "Basic-C";
06    char  *p;
07    int i;
08
09    p = s;                          ---- 포인터 변수에 배열 주소를 대입한다.
10
11    for( i=sizeof(s)-2; i >= 0; i-- )   ---- 문자형 배열의 끝부터 배열의 개수만큼 반복한다.
12       printf("%c", *(p+i) );           ---- 포인터 변수가 가리키는 곳의 문자 하나를 출력한다.
13
14    printf("\n");
15 }
```

배열의 크기는 8byte이다.
for문에서 i의 초기값을 (sizeof(s) -1)을 하지 않고
(sizeof(s) -2)를 하는 이유는?

실행 결과

```
C-cisaB
```

문자열 배열과 포인터 응용 (2/2)

- **응용 9-9 review**
  - ✓ 5행 : 문자열 배열 선언, 'Basic-C'로 초기화
  - ✓ 6행 : 포인터 변수 p 선언
  - ✓ 9행 : 배열 s의 이름을 p에 대입
  - ✓ 11행 : i의 초기값을 '배열크기-2'로 대입

    i가 0보다 크거나 같은 동안 반복(6~0까지 7회 반복)
  - ✓ 12행 : (p+i)의 실제값 출력 → *(p+i)

    (p+6) → (p+5) → (p+4) → (p+3) → (p+2) → (p+1) → (p+0)

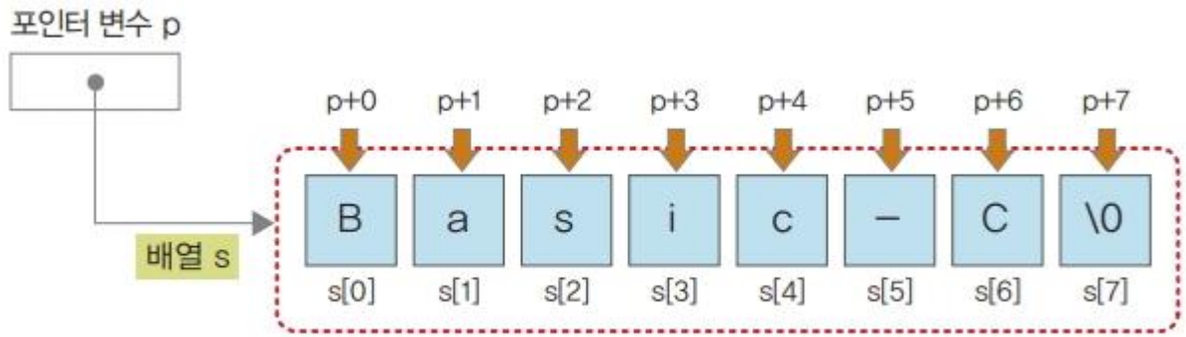


그림 9-15 [응용 9-9]의 변수와 포인터의 관계

주소와 포인터 정리

- int a; //정수형 값을 저장하는 변수 a
- int *p; //정수형 값을 저장하는 메모리 공간의 주소를 저장하는 변수 p
- 정수형 값을 저장하는 메모리 공간의 주소 대입 : p = &a;
- 특정 주소에 저장되어 있는 값 : *(&a) == *p == a

- char s[10]; //문자의 집합인 문자열을 저장하는 배열 s
- char *p; //문자 값을 저장하는 메모리 공간의 주소를 저장하는 변수 p
- 문자형 값을 저장하는 메모리 공간의 주소 대입 : p = s; or p = &s[0];
- 배열의 첫번째 칸에 저장되어 있는 값 :

  *(&s[0]) == *p == *(s+0) == s[0]
- 배열의 두번째 칸에 저장되어 있는 값 :

  *(&s[1]) == *(p+1) == s[1]

03
함수 review

함수의 모양과 활용

- **함수의 기본 형태**

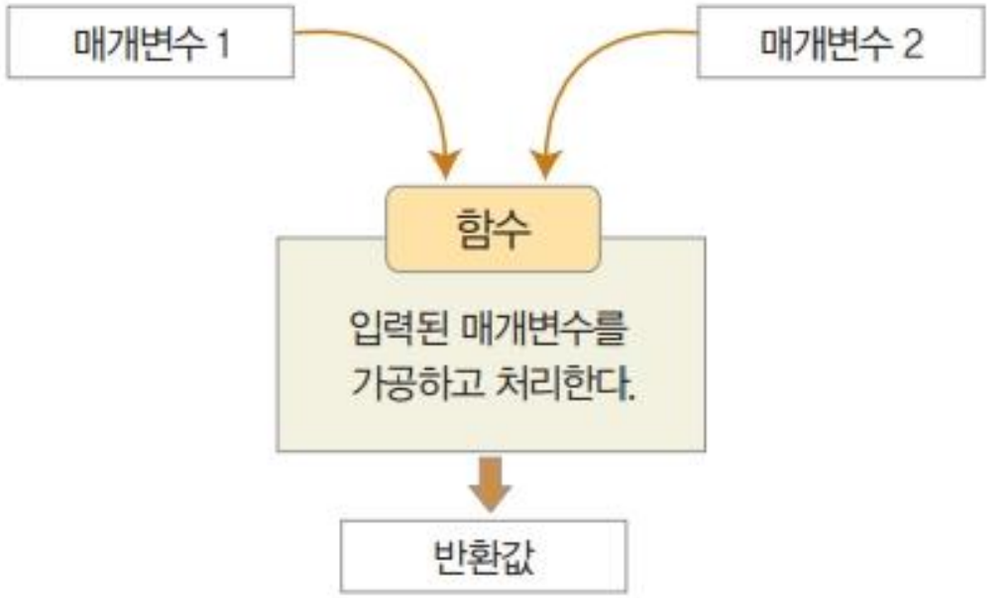


그림 10-3 함수의 형태

지역변수와 전역변수

- **지역변수 : 한정된 지역(local)에서만 사용되는 변수**
- **전역변수 : 프로그램 전체(global)에서 사용되는 변수**

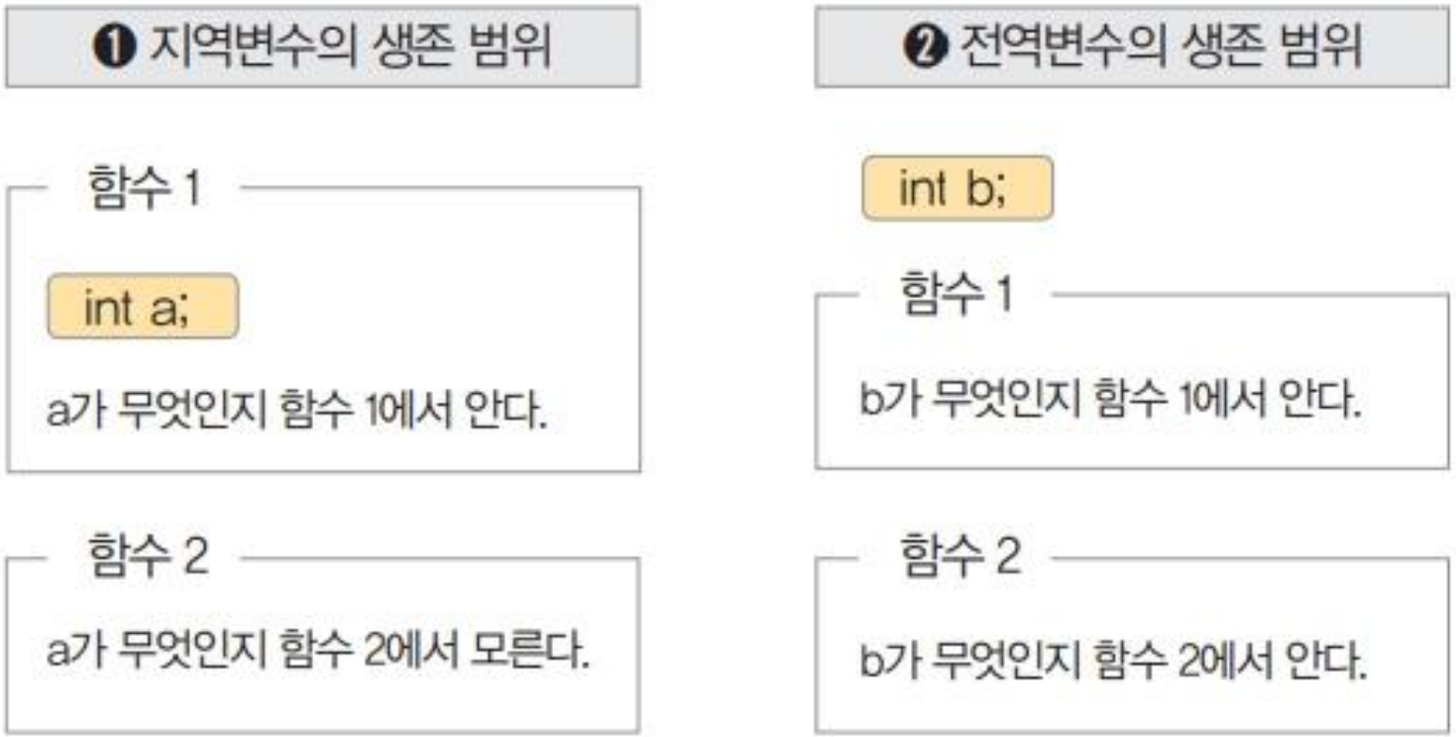


그림 10-6 지역변수와 전역변수의 생존 범위

반환값 유무에 따른 함수 구분 (1/2)

- **반환값이 있는 함수**
  - ✓ 함수를 실행한 후에 나온 결과값은 함수의 데이터형을 따름

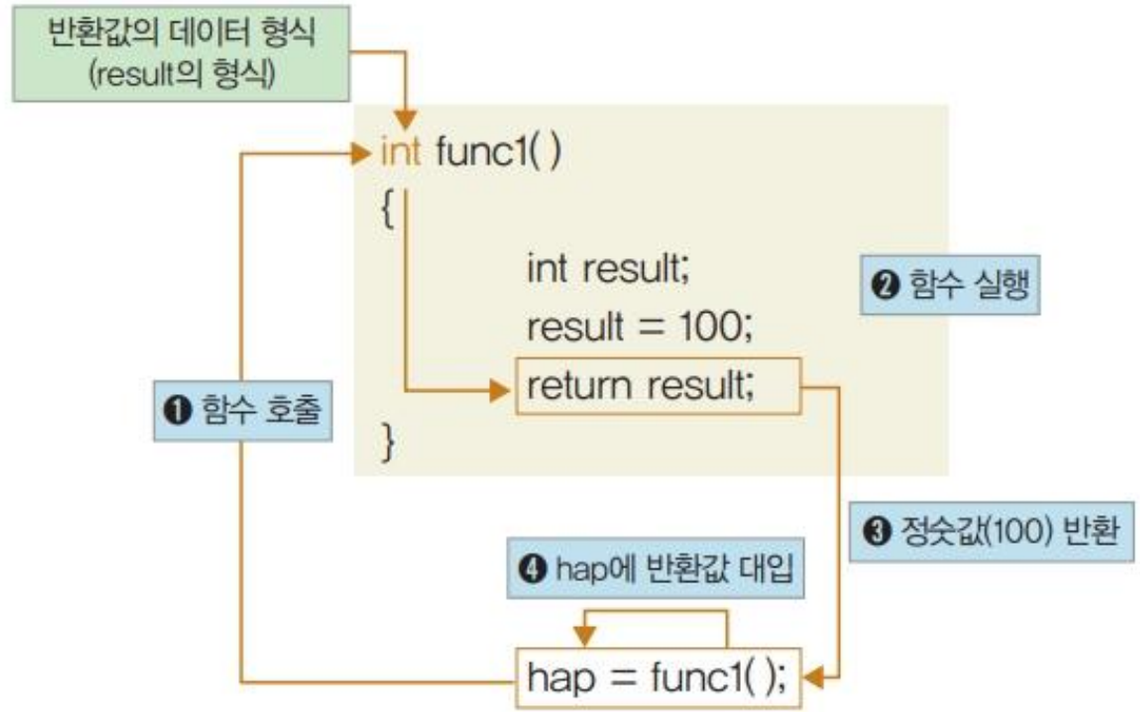


그림 10-8 int 형 값의 반환

반환값 유무에 따른 함수 구분 (2/2)

- **반환값이 없는 함수**
  - ✓ 함수를 실행한 결과, 돌려줄 것이 없는 경우
  - ✓ void로 함수 표시 : void 형 함수를 호출할 때는 함수 이름만 표시

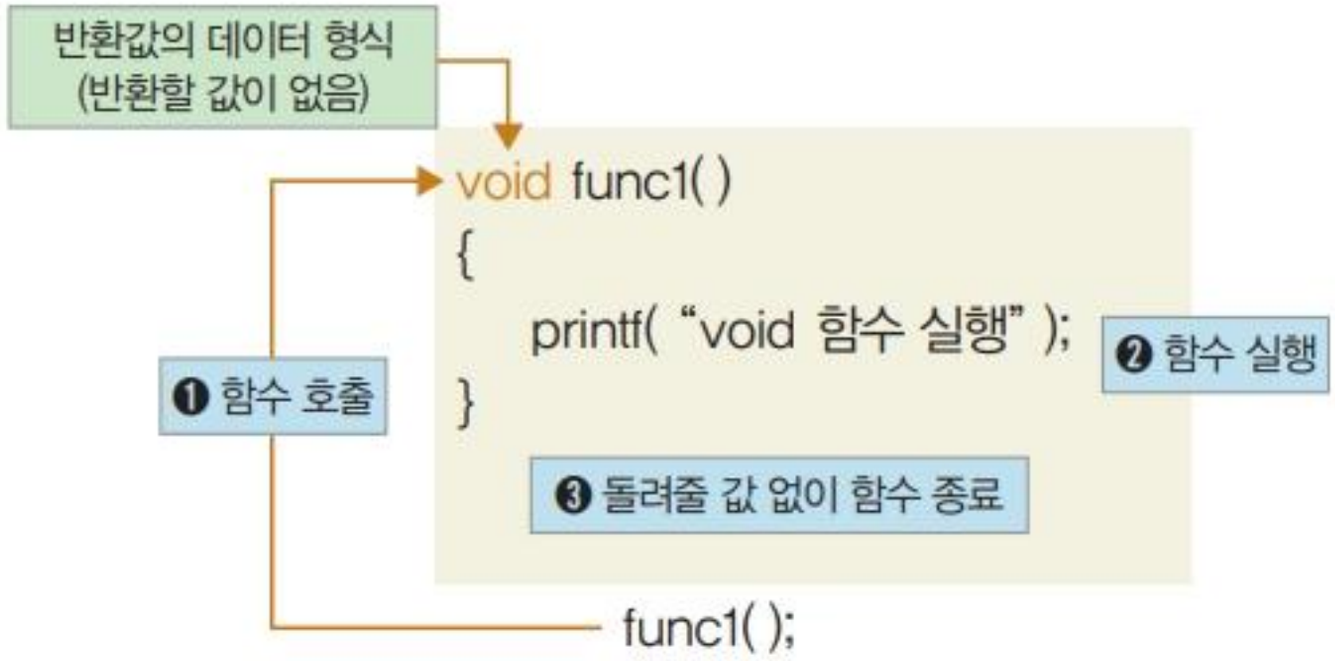


그림 10-9 void 형 함수의 작동

매개 변수 전달 방법

- **값으로 전달(call by value)**

```
void func1(int a){
      a = a +1; }
void main{
      int a = 10;
      func1(a);
      printf("%d", a); }
```

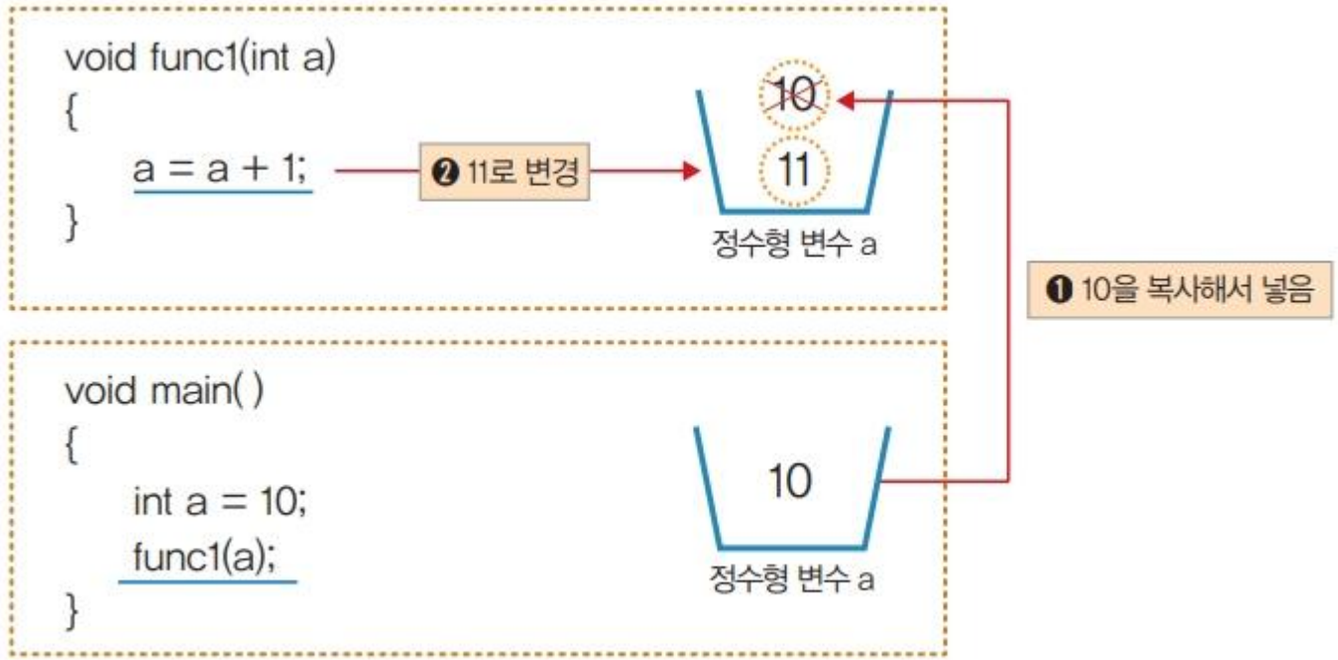


그림 10-10 매개변수 전달: 값으로 전달

- **출력 결과는? 10**

➔ 값이 복사되어 전달되기 때문에 main( )의 a값은 변하지 않음

04
함수와 포인터

주소로 매개변수 전달 (1/5)

- **포인터를 이용한 주소로 매개변수 전달 예제**

기본 10-9 매개변수 전달 방법: 주소로 전달 10-9.c

```
01 #include <stdio.h>
02
03 void func1(int *a)          ----- 매개변수로 주솟값(포인터)을 받는다.
04 {
05    *a = *a + 1;             ----- a가 가리키는 곳의 실제 값+1을 수행한다
06    printf("전달받은 a ==> %d\n", *a);   ----- a가 가리키는 곳의 실제 값을 출력한다.
07 }
08
09 void main( )
10 {
11    int a=10;                ----- a를 10으로 초기화한다.
12
13    func1(&a);               ----- 함수를 호출할 때 a의 주소를 전달한다.
14    printf("func1( ) 실행 후의 a ==> %d\n", a);   ----- 함수를 호출한 후 a 값을 출력한다.
15 }
```

같은 *가 사용되지만
서로 다른 의미임!!!!

실행 결과

```
전달받은 a ==> 11
func1( ) 실행 후의 a ==> 11
```

주소로 매개변수 전달 (2/5)

- **기본 10-9 review**
  - ✓ 주소를 매개변수로 전달하면, func( )내에서 변경한 값이 main( )에도 영향을 끼침

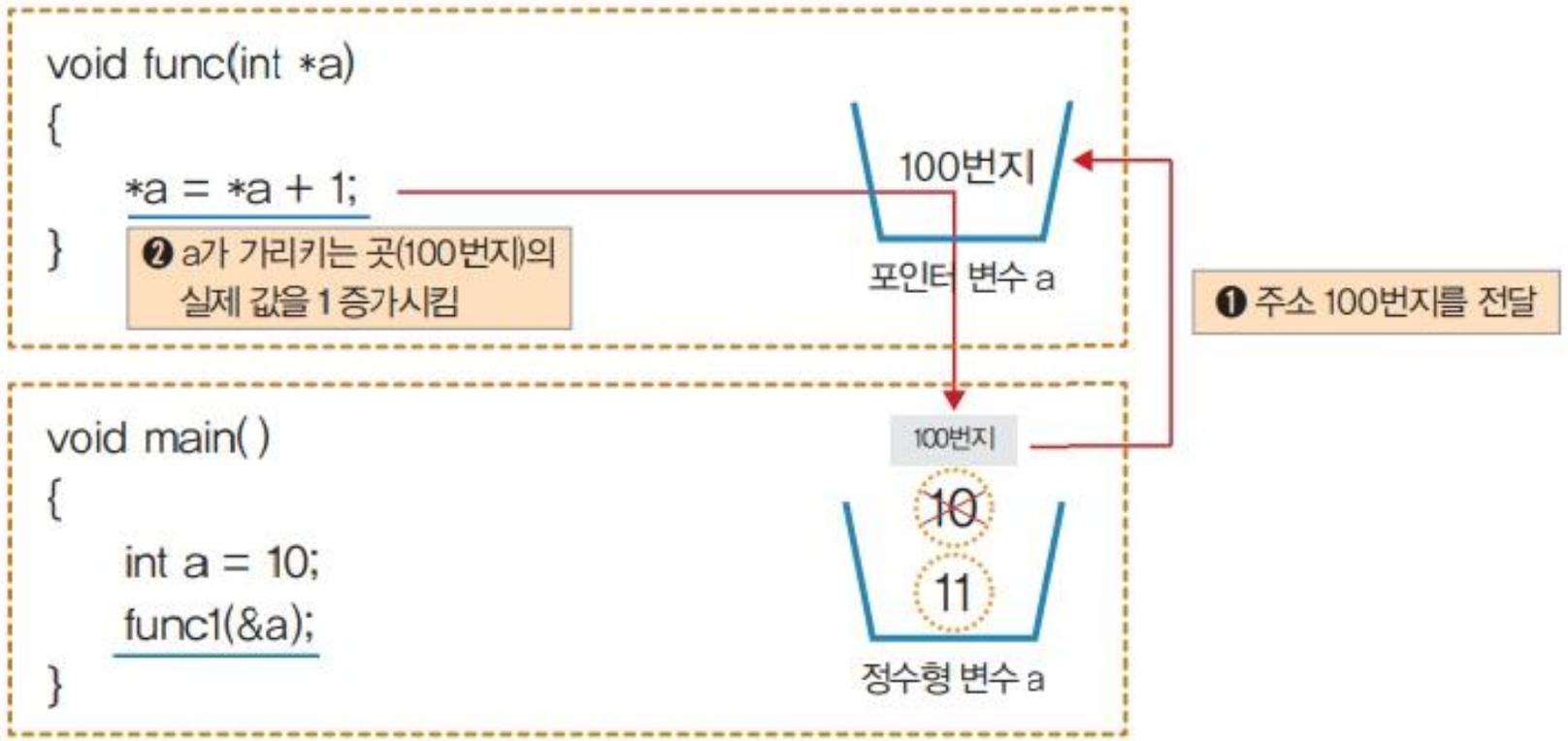


그림 10-11 매개변수 전달: 주소로 전달

주소로 매개변수 전달 (3/5)

- **포인터를 이용한 주소로 매개변수 전달 예제 #2**

응용 10-10 매개변수 전달 방법 비교 10-10.c

```
01 #include <stdio.h>
02
03 void func1(char a, char b)
04 {
05    int imsi;
06
07    imsi = a;
08    a = b;
09    b = imsi;
10 }
11
12 void func2(char *a, char *b)
13 {
14    int imsi;
15
16    imsi = *a;
17    *a = *b;
18    *b = imsi;
19 }
```

----- 매개변수가 값인 함수이다.

값으로 전달하는 경우

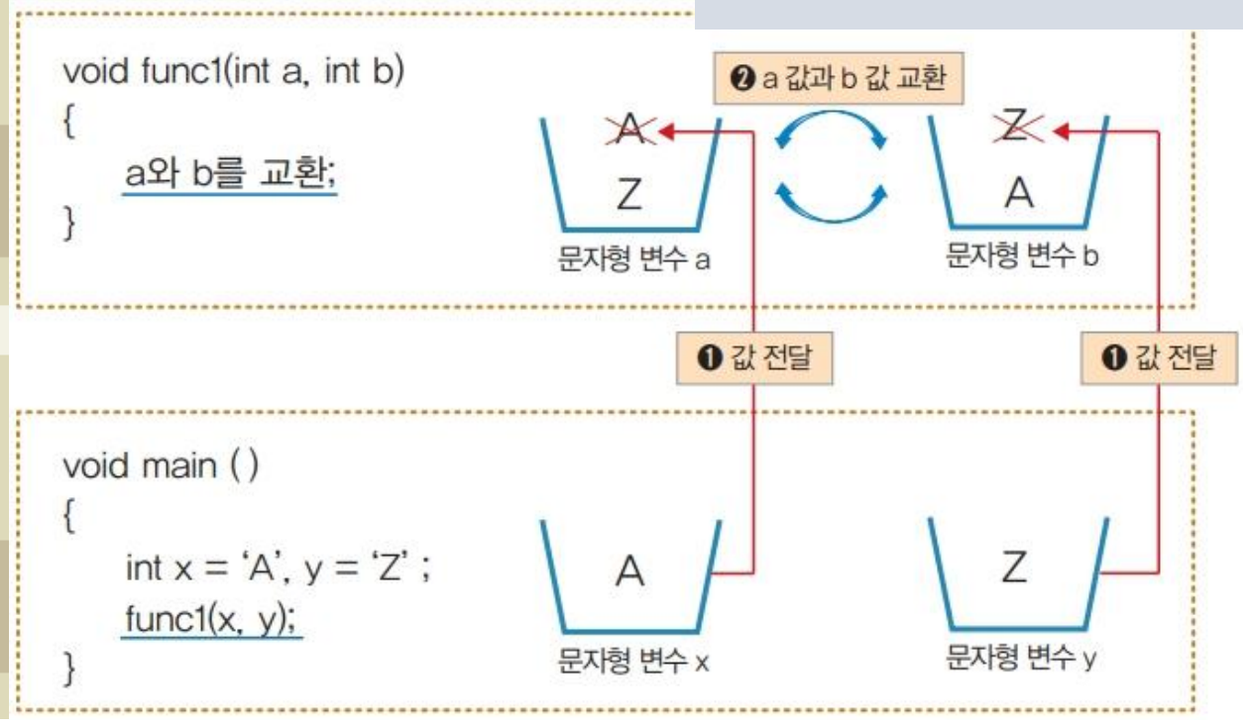


그림 10-12 값으로 전달을 통한 교환

주소로 매개변수 전달 (4/5)

- **포인터를 이용한 주소로 매개변수 전달 예제 #2**

응용 10-10 매개변수 전달 방법 비교

```
01 #include <stdio.h>
02
03 void func1(char a, char b)
04 {
05    int imsi;
06
07    imsi = a;
08    a = b;
09    b = imsi;
10 }
11
12 void func2(char *a, char *b)
13 {
14    int imsi;
15
16    imsi = *a;
17    *a = *b;
18    *b = imsi;
19 }
```

주소로 전달하는 경우

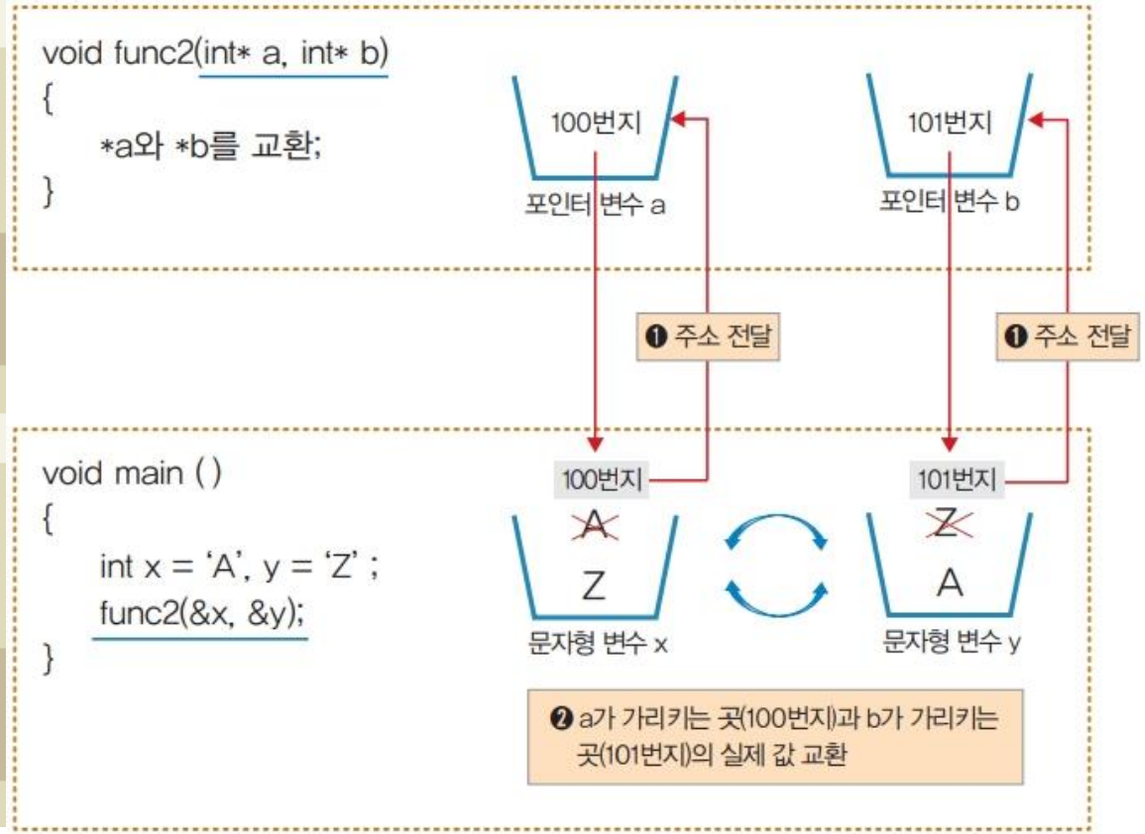


그림 10-13 주소로 전달을 통한 교환

주소로 매개변수 전달 (5/5)

- **포인터를 이용한 주소로 매개변수 전달 예제 #2**

```
20
21 void main( )
22 {
23    char x = 'A', y = 'Z';
24
25    printf("원래 값        : x=%c, y=%c\n", x, y);
26
27    func1( __3__ );
28    printf("값을 전달한 후  : x=%c, y=%c\n", x, y);
29
30    func2( __4__ );
31    printf("주소를 전달한 후: x=%c, y=%c\n", x, y);
32 }
```

25행 ---- 원래 문자를 출력한다.

27행 ---- 값을 전달해서 func1( ) 함수를 호출한다.

30행 ---- 주소를 전달해서 func2( ) 함수를 호출한다.

정답_ 1 a = b; 2 *a = *b; 3 x, y 4 &x, &y

실행 결과

```
원래 값        : x=A, y=Z
값을 전달한 후  : x=A, y=Z
주소를 전달한 후: x=Z, y=A
```

예 제

[예제 01] 포인터를 이용해 문자열을 거꾸로 출력

예제 설명 8장의 [예제모음 20]에서처럼 입력한 문자열을 반대 순서로 출력해보자. 이번에는 포인터를 활용하여 작성한 프로그램이다.

실행 결과

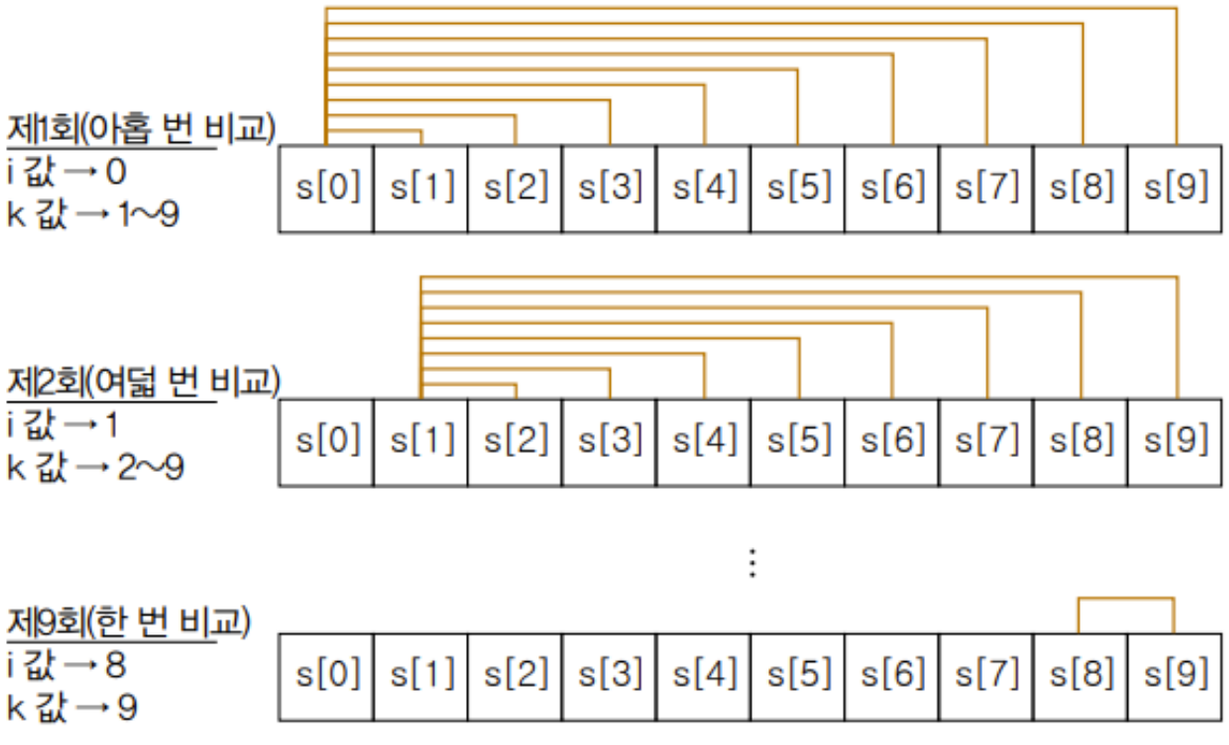
```
문자열을 입력하세요 : CookBook
내용을 거꾸로 출력 ==> kooBkooC
```

(1) 입력 받은 문자열의 길이를 측정하기 위해서 strlen( )를 사용

➔ strlen( )는 문자열의 \0을 제외한 문자열의 길이를 반환

[예제 02] 포인터를 이용한 배열의 정렬

예제 설명 포인터를 이용하여 배열에 들어 있는 값 10개를 정렬하는 프로그램이다. 다음 그림을 참고하여 두 값을 비교하고 작은 것을 앞쪽으로 옮기는 선택 정렬을 사용한다.



1. s[0]의 값을 s[1] ~ s[9]의 값과 비교하고, s[0]보다 값이 작으면 s[0]과 교환
2. s[1]의 값을 s[2] ~ s[9]의 값과 비교하고, s[1]보다 값이 작으면 s[1]과 교환
3. s[2] ~ s[9]까지 반복

※ 배열의 값을 읽을 때 포인터를 이용해야 함

실행 결과

```
정렬 전 배열 s ==> 1 0 3 2 5 4 7 6 9 8
정렬 후 배열 s ==> 0 1 2 3 4 5 6 7 8 9
```

[예제 03] 문자열을 반대로 출력하는 프로그램

예제 설명

사용자로부터 문자열을 입력 받아 반대로 출력하는 프로그램을 만드세요.

실행 결과

```
문자열을 입력하세요 : CookBook
내용을 거꾸로 출력 ==> kooBkooC
```

(1) 매개변수로 받은 문자열을 거꾸로 저장하는 void reverse_string(char* ss)를 만들어 사용

(2) 문자열의 입력과 출력은 모두 main( )에서 수행

[예제 04] 입력 받은 숫자를 정렬하는 프로그램

예제 설명

사용자로부터 10개의 숫자열을 입력 받아 정렬하여 출력하는 프로그램을 만드세요.

실행 결과

```
입력 숫자열 : 6 2 4 5 8 2 0 9 1 5
정렬 숫자열 : 9 8 6 5 5 4 2 2 1 0
```

(1) 매개변수로 받은 숫자열의 값들을 내림차순으로 정렬하는 void sort_num(int* num)를 만들어 사용

(2) 숫자열의 입력과 출력은 모두 main( )에서 수행

Q & A