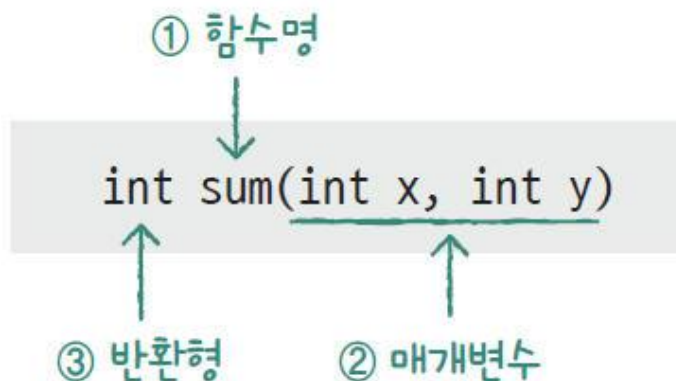


7장 함수



❖ 함수 정의 (1/3)

- 함수명, 매개변수, 반환형으로 함수원형을 만든다.
 - 함수명 : 함수의 기능에 맞는 이름
 - 매개변수 : 함수가 기능을 수행하는 데 필요한 데이터
 - 반환형 : 함수가 수행된 후의 결과



❖ 함수 정의 (2/3)

- 함수원형에 함수의 실행코드를 중괄호 안에 넣어 완성한다.

```
int sum(int x, int y)
{
    int temp;

    temp = x + y;

    return temp;
}
```

실제 함수의 내용
함수가 수행하는 명령

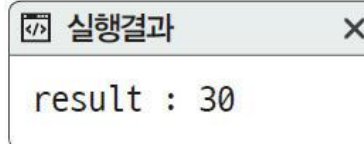
❖ 함수 정의 (3/3)

2개의 함수로 만든 프로그램

소스 코드 예제 7-1.c

```
01 #include <stdio.h>
02
03 int sum(int x, int y);
04
05 int main(void)
06 {
07     int a = 10, b = 20;
08     int result;
09
10     result = sum(a, b);
11     printf("result : %d\n", result);
12
13     return 0;
14 }
15
```

```
16 int sum(int x, int y)
17 {
18     int temp;
19
20     temp = x + y;
21
22     return temp;
23 }
```

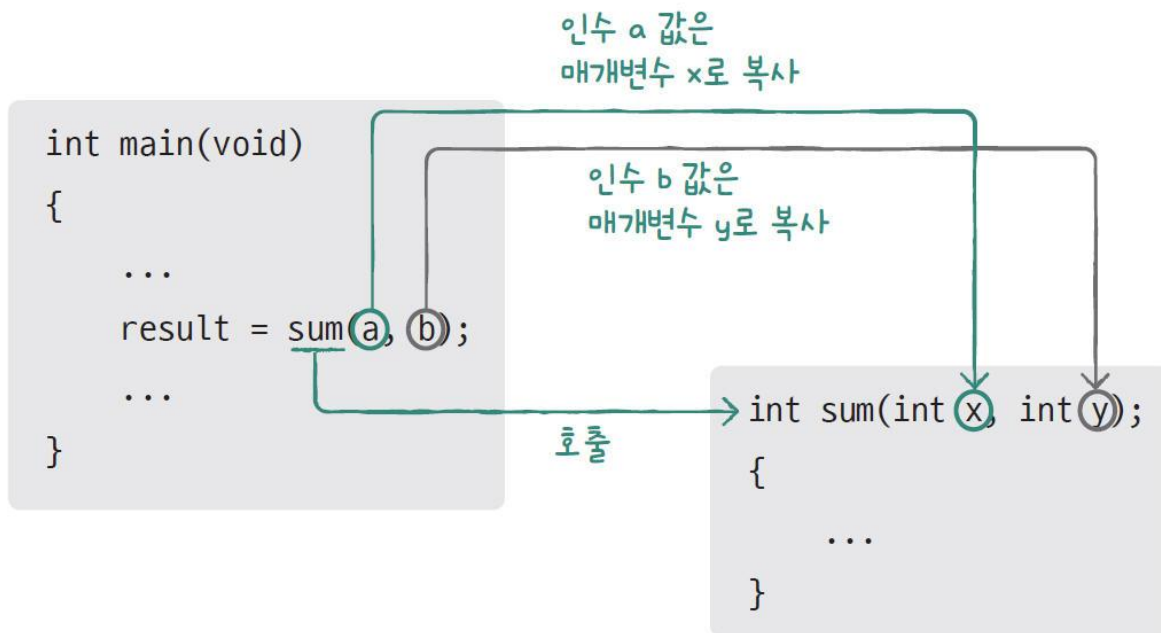


실행결과

result : 30

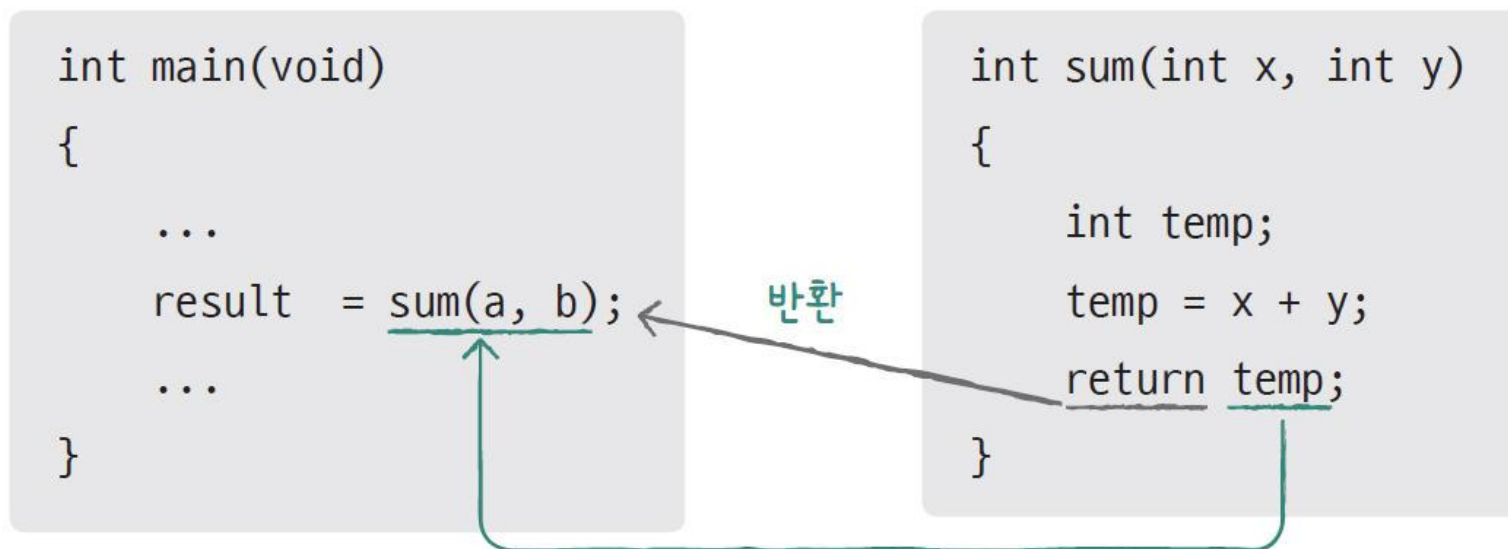
❖ 함수 호출

- 정의한 함수는 이름으로 호출하여 사용한다.
- 호출할 때는 함수에 필요한 인수를 준다.



❖ 함수 반환

- 실행을 끝낸 함수는 호출한 곳으로 값을 반환한다.



❖ 함수 선언

- 원형에 세미콜론을 붙여 선언하며, 매개변수명 생략 가능

```
int sum(int, int);    // 매개변수 이름을 생략한 형태
```

- 호출 전에 선언한다.
 - 컴파일러에 함수의 원형에 대한 정보 제공
 - 호출한 곳에 반환값의 형태에 맞는 임시공간 확보
 - 호출할 때 인수의 형태와 개수 검사
-

키워드로 끝내는 핵심 포인트

- ❖ 함수 선언을 하면 컴파일러에 함수의 형태를 미리 알릴 수 있다.
- ❖ 함수 정의는 원하는 기능의 함수를 직접 만드는 것이다.
- ❖ 함수 호출은 만든 함수를 사용하는 것이다.
- ❖ `return`은 함수를 실행한 다음 값을 반환하는 제어문이다.

표로 정리하는 핵심 포인트

표 7-1 함수의 3가지 상태

구분	예	설명
함수 선언	<code>int sum(int a, int b);</code>	함수의 형태를 알린다. 함수 원형에 세미콜론을 붙인다.
함수 정의	<code>int sum(int a, int b) { return a + b; }</code>	함수를 만든다. 반환값의 형태, 이름, 매개변수를 표시하고 블록 안에 기능을 구현한다.
함수 호출	<code>sum(10, 20);</code>	함수를 사용한다. 함수에 필요한 값을 인수로 준다.

❖ 매개변수가 없는 함수 (1/2)

- 괄호 안에 매개변수 대신에 void를 쓰거나 비워 둔다.
- 호출할 때 인수를 주지 않으나 괄호는 사용한다.
- 반환값은 return문과 함께 반환한다.

```
int get_num(void)
{
    ...
    return num;
}
```

← 매개변수가 없다

← return문 사용

❖ 매개변수가 없는 함수 (2/2)

매개변수가 없는 함수

소스 코드 예제7-2.c

```
01 #include <stdio.h>
02
03 int get_num(void);
04
05 int main(void)
06 {
07     int result;
08
09     result = get_num();
10     printf("반환값 : %d\n", result);
11     return 0;
12 }
13
```

```
14 int get_num(void)
15 {
16     int num;
17
18     printf("양수 입력 : ");
19     scanf("%d", &num);
20
21     return num;
22 }
```

❖ 반환값이 없는 함수 (1/2)

- 반환형에 void를 적는다.
- return문이 없으며 함수의 실행을 끝내면 자동으로 반환한다.
- 함수의 중간에서 반환할 때는 값 없이 return문만 사용한다.
- 호출문을 수식과 함께 사용할 수 없다.

```
print_char('@', 5) + 10
```

07-2

여러 가지 함수 유형

❖ 반환값이 없는 함수 (2/2)

반환값이 없는 함수

소스 코드 예제 7-3.c

```
01 #include <stdio.h>
02
03 void print_char(char ch, int count);
04
05 int main(void)
06 {
07     print_char('@', 5);
08
09     return 0;
10 }
11
```

```
12 void print_char(char ch, int count)
13 {
14     int i;
15
16     for (i = 0; i < count; i++)
17     {
18         printf("%c", ch);
19     }
20
21     return;
22 }
```

실행결과

@@@@@

❖ 매개변수와 반환값이 모두 없는 함수

반환값과 매개변수가 모두 없는 함수

소스 코드 예제 7-4.c

```

01 #include <stdio.h>
02
03 void print_line(void); // 함수 선언
04
05 int main(void)
06 {
07     print_line();      // 함수 호출
08     printf("학번      이름      전공      학점\n");
09     print_line();      // 함수 호출
10
11     return 0;
12 }
13
14 void print_line(void)
15 {
16     int i;
17
18     for (i = 0; i < 50; i++)
19     {
20         printf("-");
21     }
22     printf("\n");
23 }

```

실행결과

```

-----
학번      이름      전공      학점
-----

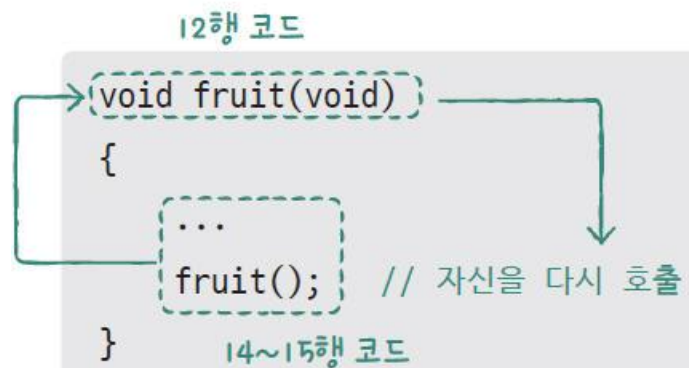
```

❖ 재귀호출 함수 (1/2)

재귀호출 함수

소스 코드 예제 7-5.c

```
01 #include <stdio.h>
02
03 void fruit(void);    // 함수 선언
04
05 int main(void)
06 {
07     fruit();          // 함수 호출
08
09     return 0;
10 }
11
12 void fruit(void)      // 재귀호출 함수 정의
13 {
14     printf("apple\n");
15     fruit();           // 자신을 다시 호출
16 }
```



실행결과

```
apple
apple
apple
:
```

07-2

여러 가지 함수 유형

❖ 재귀호출 함수 (2/2)

3번 실행되는 재귀호출 함수

소스 코드 예제7-6.c

```
01 #include <stdio.h>
02
03 void fruit(int count);
04
05 int main(void)
06 {
07     fruit(1);
08     // 처음 호출하므로 1을 인수로 준다.
09     return 0;.
10 }
```

③ count를 1 증가하여
다시 count에 대입

```
void fruit(int count) // 12행
{
    printf("apple\n");
    if (count == 3) return;
    fruit(count + 1);
}
```

① 최초 1부터 시작

② 호출 횟수가
3이면 반환

```
12 void fruit(int count)          // 호출 횟수를 매개변수에 저장
13 {
14     printf("apple\n");
15     if (count == 3) return;    // 호출 횟수가 3이면 반환하고 끝낸다.
16     fruit(count + 1);         // 재호출할 때 호출 횟수를 1 증가
17 }
```

실행결과

```
apple
apple
apple
```


07- 2

여러 가지 함수 유형

❖ 재귀호출과 반복문의 차이점

3번 실행되는 재귀호출 함수 소스 코드 예제7-6.c

```

01 #include <stdio.h>
02
03 void fruit(int count);
04
05 int main(void)
06 {
07     fruit(1);
08
09     return 0;
10 }
11
12 void fruit(int count)
13 {
14     printf("apple\n");
15     if (count == 3) return;
16     fruit(count + 1);
17     printf("jam\n");
18 }

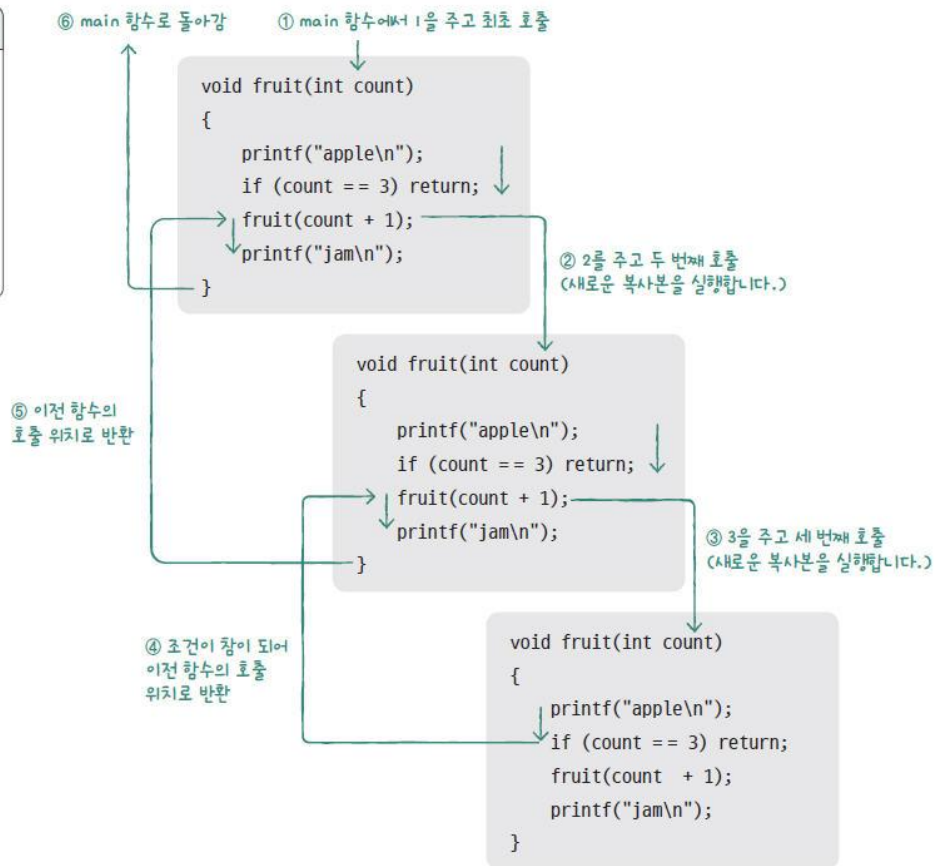
```

실행결과

```

apple
apple
apple
jam
jam

```



키워드로 끝내는 핵심 포인트

- ❖ 값을 스스로 입력하는 함수에는 **매개변수가 없어도 된다.**
- ❖ 전달받은 값을 출력하는 함수는 **반환형을 쓰지 않아도 된다.**
- ❖ 단지 화면에 출력만하는 함수는 **둘 다 쓰지 않아도 된다.**
- ❖ 매개변수와 반환값이 없을 때 빈 공간은 **void**를 적는다.
- ❖ **재귀호출 함수**는 자기 자신을 다시 호출한다.

표로 정리하는 핵심 포인트

표 7-2 다양한 함수 형태

형태	구분	설명
매개변수가 없는 경우	선언	<code>int get_num(void);</code> 또는 <code>int get_num();</code>
	특징	호출할 때 인수 없이 괄호만 사용한다.
반환형이 없는 경우	선언	<code>void print_char(char ch, int count);</code>
	특징	반환할 때 <code>return</code> 문을 쓰지 않거나 <code>return</code> 문만 사용한다. 호출 문장을 수식의 일부로 쓸 수 없다.
반환형이 매개변수와 모두 없는 경우	선언	<code>void print_title(void);</code>
	특징	두 가지 경우의 특징을 모두 포함한다.

표 7-3 재귀호출 함수

형태	구분	설명
재귀호출 함수	선언	<code>void fruit() { ... fruit(); ... }</code>
	특징	함수 안에 재귀호출을 멈추는 조건이 있어야 한다.