C프로그래밍

7 함수

중간고사 공지

- **시험일자 : 4/26(토) 오후 13시**
- **시험장소 : 참빛관 지하 106호**
- **시험범위 : 1~7주차 강의내용**
  - ✓ C언어와 문법에 대한 이해
- **4/28(월) 강의 진행**
  - ✓ 5/5(월) 어린이날 휴강 고려

01
함수의 이해

함수의 개념 (1/11)

- **함수(Function) : “무엇을 넣으면, 어떤 것을 돌려주는 요술상자”**
- **C 언어 프로그램 자체에서 제공하거나, 직접 만들어서 사용 가능**

```
함수_이름( );
```

✓ printf( ) 함수 : C 언어에서 자체 제공

```
printf("Basic-C");
```

➔ 출력: Basic-C

함수의 개념 (2/11)

- **함수를 자판기에 비교**
  - ✓ 자판기가 없는 경우 → 매번 같은 작업을 반복

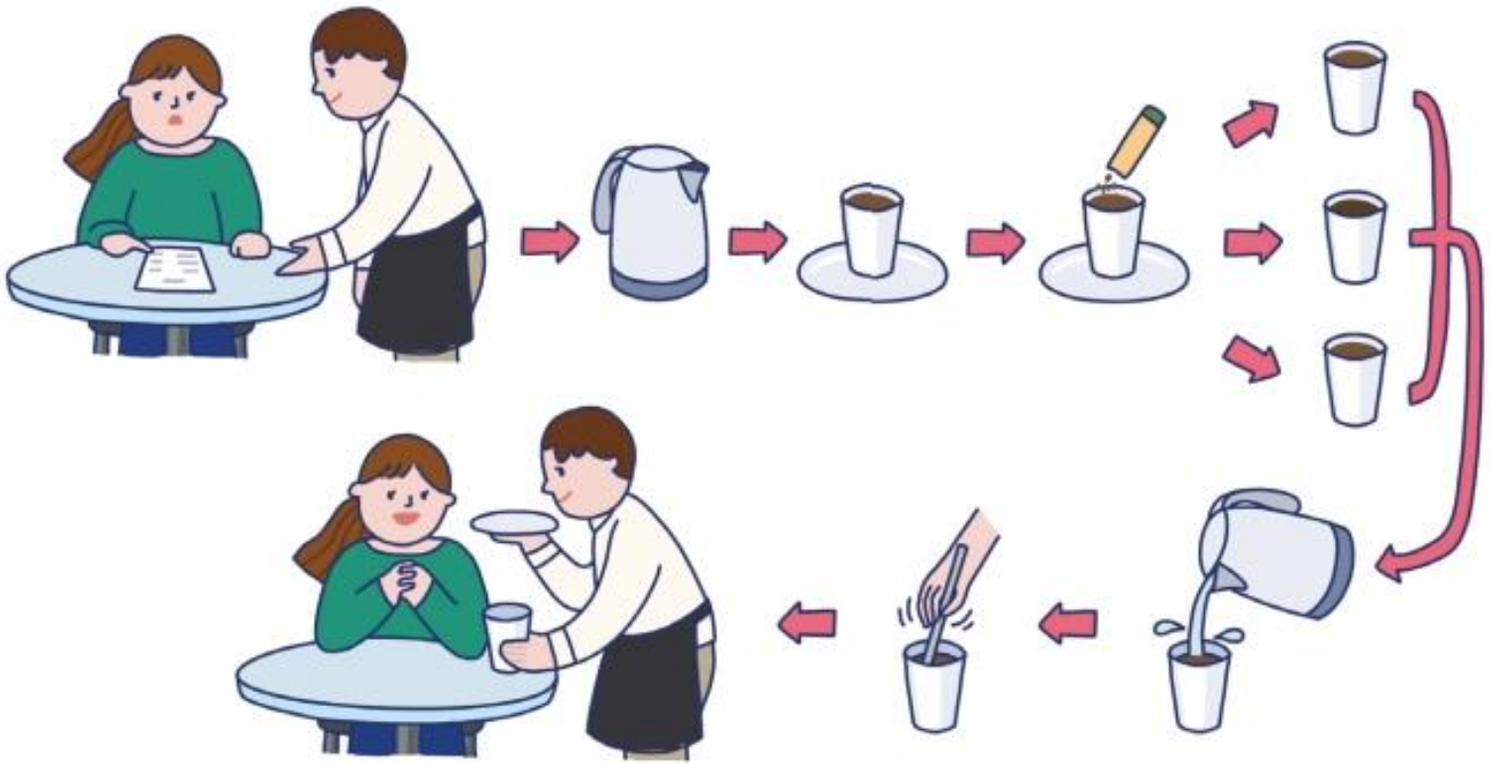


그림 10-1 직접 커피를 서비스하는 과정

함수의 개념 (3/11)

- **직접 커피를 만들어서 서비스 하는 예**

기본 10-1 직접 커피를 서비스하는 과정의 예 10-1.c

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int coffee;                                  ----- 커피의 종류를 선택하는 변수이다.
06
07    printf("어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
08    scanf("%d", &coffee);                        ----- 커피의 종류를 입력받는다.
09
10    printf("\n# 1. 뜨거운 물을 준비한다\n");
11    printf("# 2. 종이컵을 준비한다\n");
12
13    switch(coffee)                               ----- 커피의 종류에 따라 안내문을
14    {                                                  출력한다.
15    case 1: printf("# 3. 보통커피를 탄다\n"); break;
16    case 2: printf("# 3. 설탕커피를 탄다\n"); break;
17    case 3: printf("# 3. 블랙커피를 탄다\n"); break;
18    default: printf("# 3. 아무거나 탄다\n"); break;
19    }
```

함수의 개념 (4/11)

- **직접 커피를 만들어서 서비스 하는 예 (cont'd)**

```
20
21      printf("# 4. 물을 붓는다\n");
22      printf("# 5. 스푼으로 저어서 녹인다\n\n");
23
24      printf("손님~ 커피 여기 있습니다.\n\n");
25  }
```

실행 결과

```
어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 2

# 1. 뜨거운 물을 준비한다
# 2. 종이컵을 준비한다
# 3. 설탕커피를 탄다
# 4. 물을 붓는다
# 5. 스푼으로 저어서 녹인다

손님~ 커피 여기 있습니다.
```

함수의 개념 (5/11)

- **손님 3명이 연속해서 온다고 가정 → 같은 동작을 3번 반복**

```
⋮
07    printf("어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
08    scanf("%d", &coffee);
09
10    printf("\n# 1. 뜨거운 물을 준비한다\n");
11    printf("# 2. 종이컵을 준비한다\n");
12
13    switch(coffee)
14    {
15    case 1: printf("# 3. 보통커피를 탄다\n"); break;
16    case 2: printf("# 3. 설탕커피를 탄다\n"); break;
17    case 3: printf("# 3. 블랙커피를 탄다\n"); break;
18    default: printf("# 3. 아무거나 탄다\n"); break;
19    }
20
22    printf("# 5. 스푼으로 저어서 녹인다\n\n");
23
24    printf("손님~ 커피 여기 있습니다.\n\n");
25
26 [두 번째 손님을 위해 07~24행을 다시 반복한다.]
27
28 [세 번째 손님을 위해 07~24행을 다시 반복한다.]
```

함수의 개념 (6/11)

- **작업을 효율화 하기 위해 커피자판기를 설치**

그림 10-2 커피 자판기를 사용하는 과정

함수의 개념 (7/11)

- **커피자판기(커피 머신)을 이용한 서비스 예**

기본 10-2 함수를 사용하여 변경한 [기본 10-1] 10-2.c

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 int coffee_machine(int button)
04 {
05    printf("\n# 1.(자동으로) 뜨거운 물을 준비한다\n");
06    printf("# 2. (자동으로) 종이컵을 준비한다\n");
07
08    switch(button)
09    {
10    case 1: printf("# 3. (자동으로) 보통커피를 탄다\n"); break;
11    case 2: printf("# 3. (자동으로) 설탕커피를 탄다\n"); break;
12    case 3: printf("# 3. (자동으로) 블랙커피를 탄다\n"); break;
13    default: printf("# 3. (자동으로) 아무거나 탄다\n"); break;
14    }
15
16    printf("# 4. (자동으로) 물을 붓는다\n");
17    printf("# 5. (자동으로) 스푼으로 저어서 녹인다\n\n");
18
```

커피 자판기 함수를 구현한다.

선택한 버튼에 따라 안내문을 출력한다.

함수의 개념 (8/11)

- **커피자판기(커피 머신)을 이용한 서비스 예 (cont'd)**

```
19    return 0;                                          ------ 30행으로 돌아간다.
20 }
21
22 void main( )
23 {
24    int coffee;                                        ----- 커피 종류 변수와
25    int ret;                                                 반환값 변수를
26                                                             선언한다.
27    printf("어떤 커피를 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
28    scanf("%d", &coffee);                              ----- 커피를 주문받는다.
29
30    ret = coffee_machine(coffee);                      ----- 커피 자판기의 버튼을 누른다
31                                                             (coffee_machine( ) 함수를 호출한다).
32    printf("손님~ 커피 여기 있습니다.\n\n");
33 }
```

실행 결과

```
어떤 커피를 드릴까요? (1:보통, 2:설탕, 3:블랙) 2

# 1.(자동으로) 뜨거운 물을 준비한다
# 2. (자동으로) 종이컵을 준비한다
# 3. (자동으로) 설탕커피를 탄다
# 4. (자동으로) 물을 붓는다
# 5. (자동으로) 스푼으로 저어서 녹인다

손님~ 커피 여기 있습니다.
```

직원은 주문을 받고 서빙을 하며,
커피를 만드는 과정은
커피 자판기에게 맡김

함수의 개념 (9/11)

- **다수에게 커피자판기(커피 머신)을 이용하여 서비스하는 예**

기본 10-3 여러 명에게 주문을 받도록 변경한 [기본 10-2] 10-3.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 int coffee_machine(int button)
04 {
05    printf("\n# 1.(자동으로) 뜨거운 물을 준비한다\n");
06    printf("# 2. (자동으로) 종이컵을 준비한다\n");
07
08    switch(button)
09    {
10    case 1: printf("# 3. (자동으로) 보통커피를 탄다\n"); break;
11    case 2: printf("# 3. (자동으로) 설탕커피를 탄다\n"); break;
12    case 3: printf("# 3. (자동으로) 블랙커피를 탄다\n"); break;
13    default: printf("# 3. (자동으로) 아무거나 탄다\n"); break;
14    }
15
16    printf("# 4. (자동으로) 물을 붓는다\n");
17    printf("# 5. (자동으로) 스푼으로 저어서 녹인다\n\n");
18
```

커피 자판기 함수를 구현한다.

선택한 버튼에 따라 안내문을 출력한다.

함수의 개념 (10/11)

- **다수에게 커피자판기(커피 머신)을 이용하여 서비스하는 예 (cont'd)**

```
19    return 0;                                  ------ 각각을 호출한 곳으로
20 }                                                   돌아간다(29행, 34행,
21                                                     39행).
22 void main( )
23 {                          직원은 주문과 서빙에 집중,
24    int coffee;                 효율을 높일 수 있음
25    int ret;
26
27    printf("A님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
28    scanf("%d", &coffee);                      ------ 주문을 받고 커피 자판기의 버튼을 누른다
29    ret = coffee_machine(coffee);                     (함수를 호출한다).
30    printf("A님 커피 여기 있습니다.\n\n");
31
32    printf("B님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
33    scanf("%d", &coffee);                      ------ 주문을 받고 커피 자판기의 버튼을 누른다
34    ret = coffee_machine(coffee);                     (함수를 호출한다).
35    printf("B님 커피 여기 있습니다.\n\n");
36
37    printf("C님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) ");
38    scanf("%d", &coffee);                      ------ 주문을 받고 커피 자판기의 버튼을 누른다
39    ret = coffee_machine(coffee);                     (함수를 호출한다).
40    printf("C님 커피 여기 있습니다.\n\n");
41 }
```

실행 결과

```
A님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 1

# 1.(자동으로) 뜨거운 물을 준비한다
# 2. (자동으로) 종이컵을 준비한다
# 3. (자동으로) 보통커피를 탄다
# 4. (자동으로) 물을 붓는다
# 5. (자동으로) 스푼으로 저어서 녹인다

A님 커피 여기 있습니다.

B님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 2

# 1.(자동으로) 뜨거운 물을 준비한다
# 2. (자동으로) 종이컵을 준비한다
# 3. (자동으로) 설탕커피를 탄다
# 4. (자동으로) 물을 붓는다
# 5. (자동으로) 스푼으로 저어서 녹인다

B님 커피 여기 있습니다.

C님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 3
⋮
```

함수의 개념 (11/11)

- **기본 10-3 복기 : 반복문을 사용하면 되지 않는가?**
  - ➔ 커피 서비스만을 반복적으로 한다면 반복문으로 처리 가능
  - ➔ but, 종업원은 커피 서비스 뿐만 아니라 청소, 대화 등 다양한 일을 해야함
  - ➔ 종업원이 커피 서비스만을 계속 하고 있다면…

- **함수의 용도**
  - ✓ 프로그램의 요구 기능을 작은 기능들로 분할하고 하나씩 구현
  - ✓ Divide and Conquer!

- **함수의 필요성**
  - ✓ 문제의 발생 또는 요구사항 변경 시 대응하기가 쉬움 ➔ 변경 범위 축소
  - ✓ 재사용성 증대 : printf( ), scanf( )를 생각해보라

함수의 모양과 활용 (1/6)

- **함수의 기본 형태**
  - ✓ 함수는 '매개변수(또는 '인수')'를 입력 받고 그 매개변수를 가공하고 처리한 후 '반환값'을 돌려줌

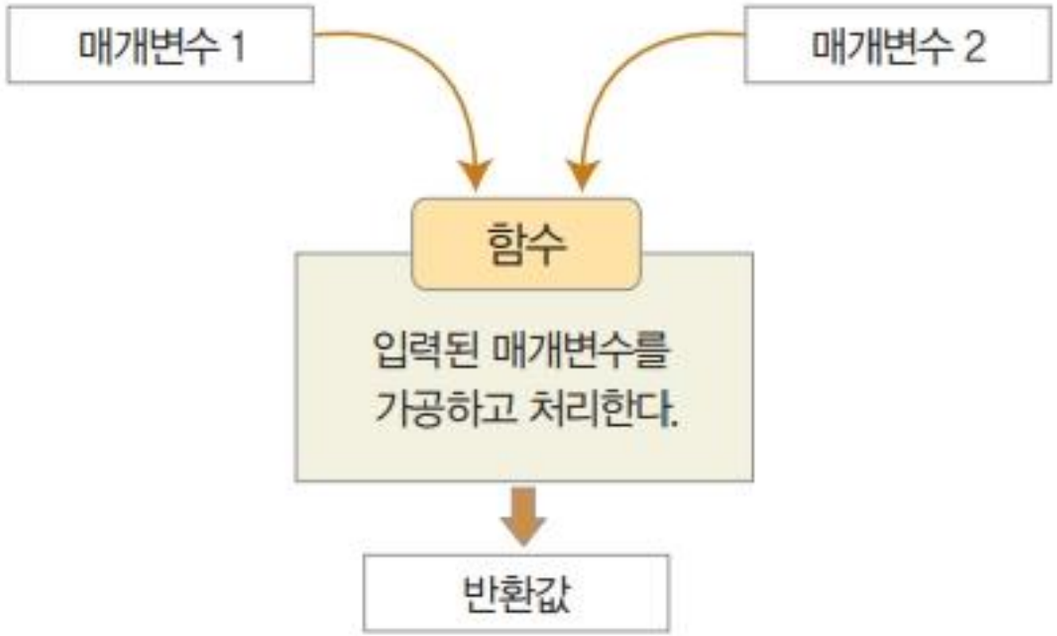


그림 10-3 함수의 형태

함수의 모양과 활용 (2/6)

- **값을 더하고 그 결과를 반환하는 함수 사용 예**

기본 10-4 본격적인 함수 사용 예 1 — 10-4.c

```
01 #include <stdio.h>
02
03 int plus(int v1, int v2)          ---- plus( ) 함수를 정의한다.
04 {
05    int result;
06    result = v1 + v2;              ------ 3행에서 받은 두 매개변수의 합을 구한다.
07    return result;                 ------ plus( ) 함수를 호출한 곳에 result 값을 반환한다.
08 }
09
10 void main( )
11 {
12    int hap;
13
14    hap = plus(100, 200);          ---- 매개변수 2개를 지정하여 plus( ) 함수를
                                          호출하고 반환값은 hap에 저장한다.
15
16    printf("100과 200의 plus( ) 함수 결과는 : %d\n", hap);
17 }
```

실행 결과

```
100과 200의 plus( ) 함수 결과는 : 300
```

함수의 모양과 활용 (3/6)

- **기본 10-4 복기**

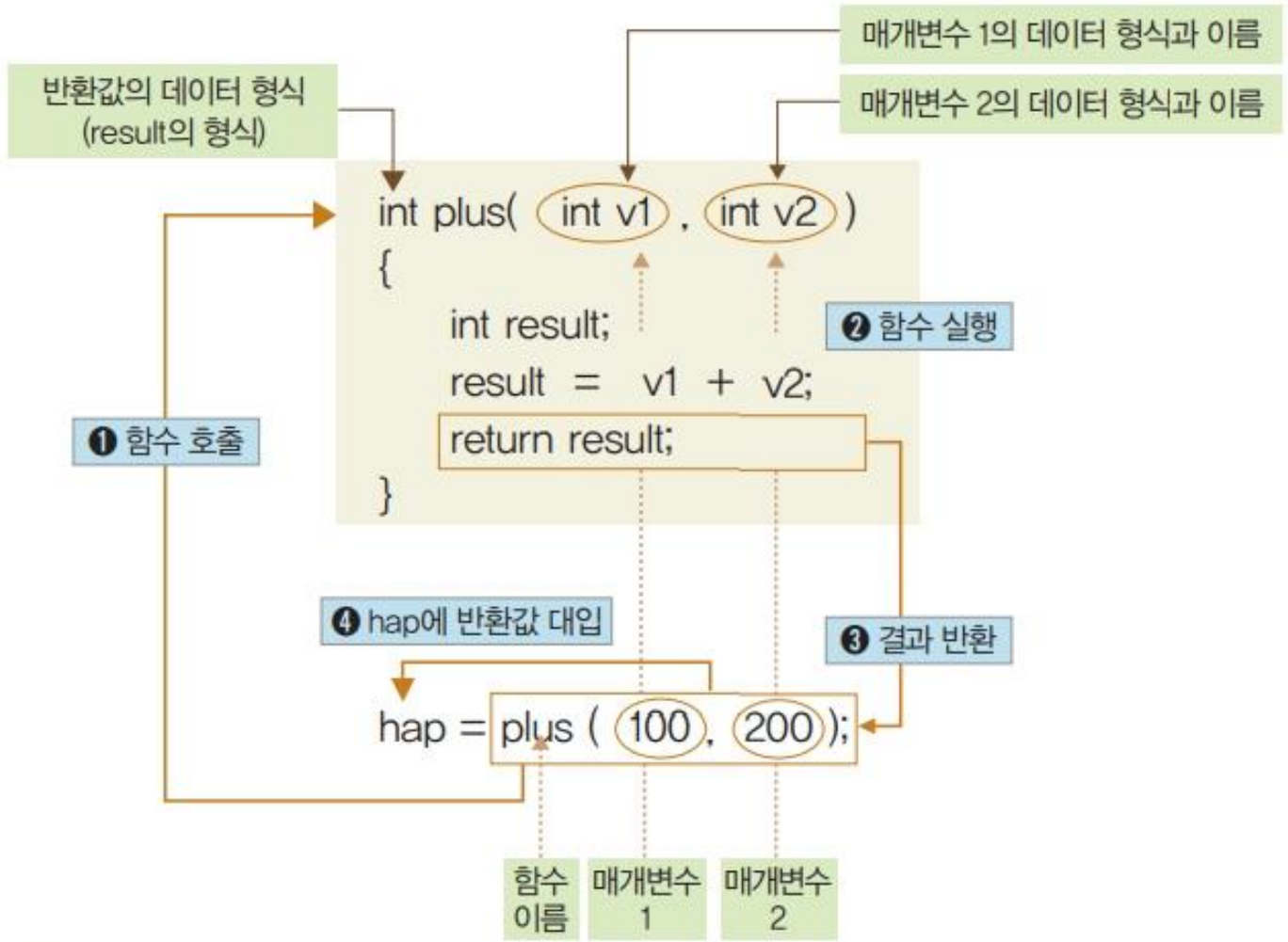


그림 10-4 plus( ) 함수의 형태와 호출 순서

함수의 모양과 활용 (4/6)

- **기본 10-4 복기 (cont'd)**

  ❶ 함수 호출 : plus(100, 200); 으로 함수 호출

  ❷ 함수 실행 : v1과 v2를 더해 result에 대입시킨 후 이 함수를 호출했던 곳으로 돌아감

  ❸ 결과 반환 : 결과 result값 300을 plus( ) 함수를 호출했던 곳으로 돌려줌

  ❹ hap에 반환값 대입 : result값 300을 변수 hap에 대입, plus(100, 200)의 결과를 hap에 넣어야 하므로 hap과 plus( ) 함수 반환값의 데이터 형식이 같아야 함

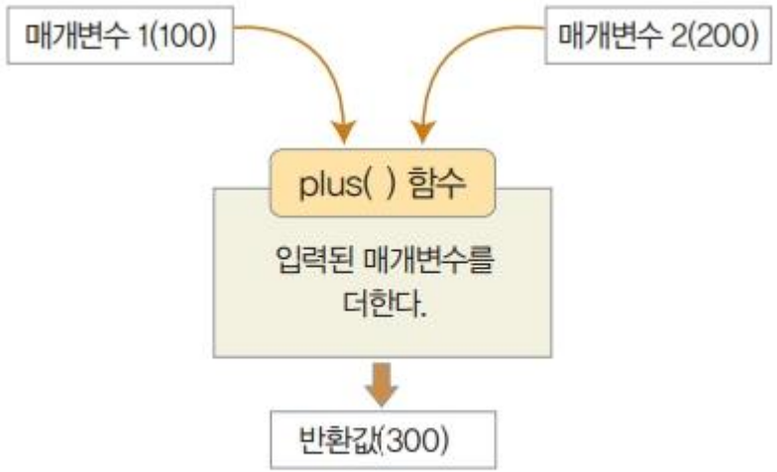


그림 10-5 plus( ) 함수의 호출을 간단하게 표현한 형태

함수의 모양과 활용 (5/6)

- **계산기 함수를 이용한 프로그램 예**

응용 10-5 본격적인 함수 사용 예 2 — 10-5.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 int calc(int v1, int v2, int op)
04 {
05    int result;
06
07    switch(__1__)
08    {
09    case 1:   result = v1 + v2; break;
10    case 2:   result = v1 - v2; break;
11    case 3:   result = v1 * v2; break;
12    case 4:   result = v1 / v2; break;
13    }
14
15    return result;
16 }
17
```

03 ~~~ 매개변수 3개를 받아서 계산하는 함수이다.

07 ~~~ 매개변숫값에 따라서 '1: 덧셈, 2: 뺄셈, 3: 곱셈, 4: 나눗셈'을 실행한다.

15 ~~~ 계산 결과를 반환한다.

함수의 모양과 활용 (6/6)

- **계산기 함수를 이용한 프로그램 예 (cont'd)**

```
18 void main( )
19 {
20    int res;                          ----- 계산 결과, 연산자, 입력 숫자에 대한
21    int oper, a, b;                         변수이다.
22
23    printf("계산 입력 (1:+, 2:-, 3:*, 4:/) : ");
24    scanf("%d", &oper);               ----- 연산자를 입력한다.
25
26    printf("계산할 두 숫자를 입력 : ");
27    scanf("%d %d", &a, &b);           ----- 계산할 두 숫자를 입력한다.
28
29    res = calc( __2__ );    a, b, oper ----- 매개변수 3개를 넣고 calc( ) 함수를
30                                          호출한다. 계산 결과는 res에 저장한다.
31    printf("계산 결과는 : %d\n", res);
32 }
```

실행 결과

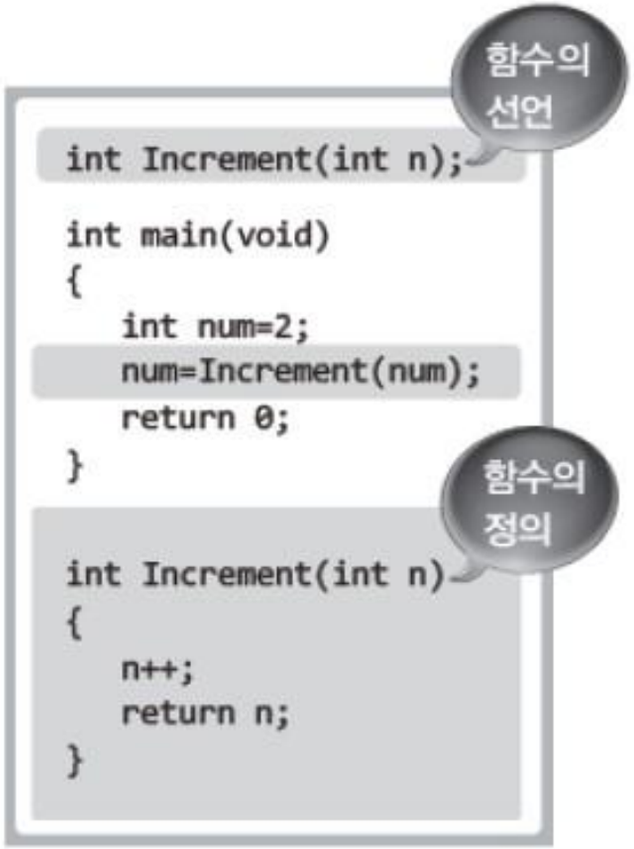
```
계산 입력 (1:+, 2:-, 3:*, 4:/) : 3
계산할 두 숫자를 입력 : 7  8
계산 결과는 : 56
```

정답_ 1 op 2 a, b, oper

02
함수의 선언

함수의 선언

- **컴파일은 위에서 아래로 진행 → 함수의 배치순서가 중요**
  - ✓ 정의되지 않은 함수는 사용할 수 없음
- **함수의 선언을 이용하면 나중에 정의하여도 사용 가능**



// int Increment(int); 으로도 선언 가능

03
지역변수와 전역변수

지역변수와 전역변수 (1/4)

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

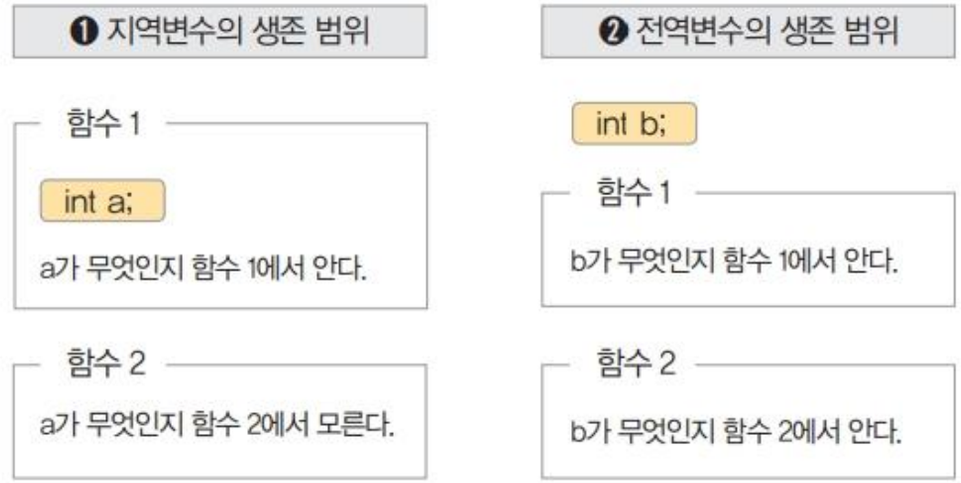


그림 10-6 지역변수와 전역변수의 생존 범위

❶에서 a는 현재 '함수 1' 안에 선언, 그러므로 a는 '함수 1' 안에서만 사용될 수 있고, '함수 2'에서는 a의 존재를 모름

❷는 전역변수 b를 보여줌. b는 함수(함수 1, 함수 2) 안이 아니라 함수 바깥에 선언되어 있으므로 모든 함수에서 b의 존재를 알게 됨

지역변수와 전역변수 (2/4)

- **같은 a라고 해도 '함수 1의 a'는 함수 내에서 따로 정의했으므로 지역변수, '함수 2의 a는 함수 안에 정의된 것이 없으므로 전역변수**

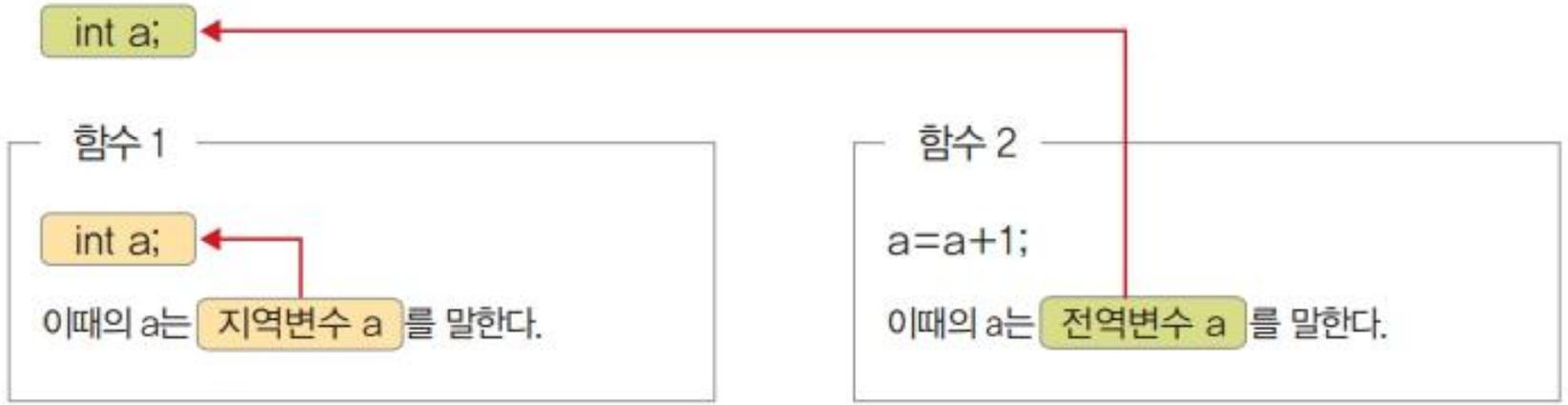


그림 10-7 이름이 같은 지역변수와 전역변수의 구분

지역변수와 전역변수 (3/4)

- **지역변수와 전역변수를 사용하는 예**

기본 10-6 지역변수와 전역변수의 구분 10-6.c

```
01 #include <stdio.h>
02
03 int a = 100;                          ----- 전역변수 a를 선언하고 초깃값을 대입한다.
04
05 void func1( )
06 {
07    int a = 200;                       ----- 지역변수 a를 선언하고 초깃값을 대입한다.
08    printf("func1( )에서 a의 값==> %d\n", a);   ----- 지역변수를 출력한다.
09 }
10
11 void main( )
12 {
13    func1( );                          ----- 함수를 호출한다.
14    printf("main( ) 에서 a의 값==> %d\n", a);   ----- 전역변수를 출력한다.
15 }
```

실행 결과

```
func1( )에서 a의 값==> 200
main( ) 에서 a의 값==> 100
```

지역변수와 전역변수 (4/4)

- **다양한 형태의 지역변수 → { }로 구별**

여기서 잠깐 다양한 형태의 지역변수

- 반복문, 조건문 등에서 중괄호를 사용 시 해당 중괄호 내에 선언된 변수도 지역변수
- 따라서, 반복문, 조건문의 중괄호를 빠져나가면 소멸

→ for(int i = 0; i<10; i++)를 생각해보자! i는 for문을 빠져나가면 소멸됨

프로그램 전역변수

- **프로그램 전체(global)에서 사용되는 변수**
  - ✓ 전역변수는 함수 외부에 선언
  - ✓ 프로그램의 시작과 동시에 메모리 공간에 할당되어 종료 시 까지 존재
  - ✓ 별도의 값으로 초기화 하지 않으면 0으로 초기화
  - ✓ 프로그램 전체 영역 어디서든 접근이 가능

```
void Add(int val);
int num;    // 전역변수는 기본 0으로 초기화됨

int main(void)
{
    printf("num: %d \n", num);
    Add(3);
    printf("num: %d \n", num);
    num++;   // 전역변수 num의 값 1 증가
    printf("num: %d \n", num);
    return 0;
}

void Add(int val)
{
    num += val;    // 전역변수 num의 값 val만큼 증가
}
```

```
num: 0
num: 3
num: 4
```

static 변수

- **선언된 함수 내에서만 접근이 가능 : 지역변수의 특성**
  - ✓ 1회만 초기화되고, 프로그램 종료 시까지 메모리 공간에 존재 : 전역변수 특성
  - ✓ 선언 시 초기화 하지 않으면 0으로 초기화 : 전역변수 특성
  - ✓ 전역변수에 비해 접근 범위가 좁기 때문에 안정적

```
void SimpleFunc(void)
{
    static int num1=0;    // 초기화하지 않으면 0 초기화
    int num2=0;     // 초기화하지 않으면 쓰레기 값 초기화
    num1++, num2++;
    printf("static: %d, local: %d \n",num1, num2);
}

int main(void)
{
    int i;
    for(i=0; i<3; i++)
        SimpleFunc();
    return 0;
}
```

```
static: 1, local: 1
static: 2, local: 1
static: 3, local: 1
```

04
함수의 반환값과 매개변수

반환값 유무에 따른 함수 구분 (1/4)

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

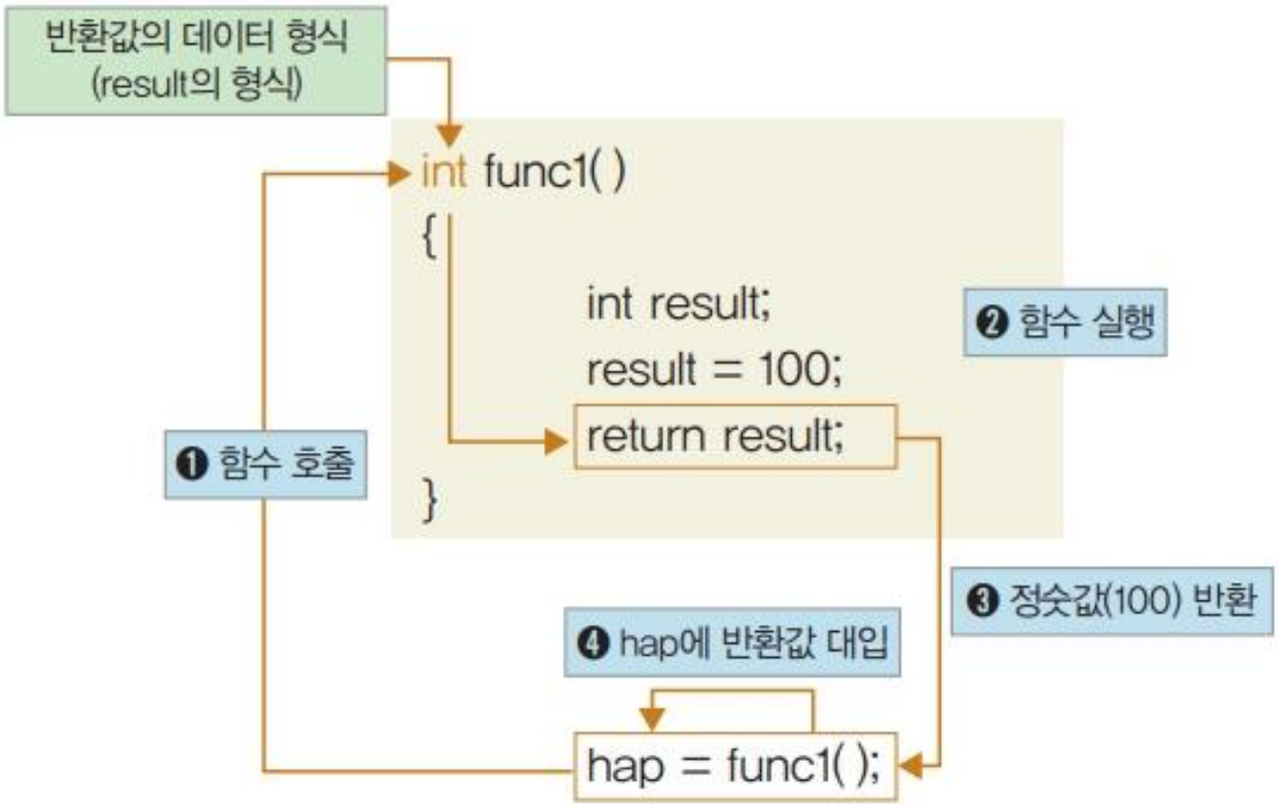


그림 10-8 int 형 값의 반환

반환값 유무에 따른 함수 구분 (2/4)

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

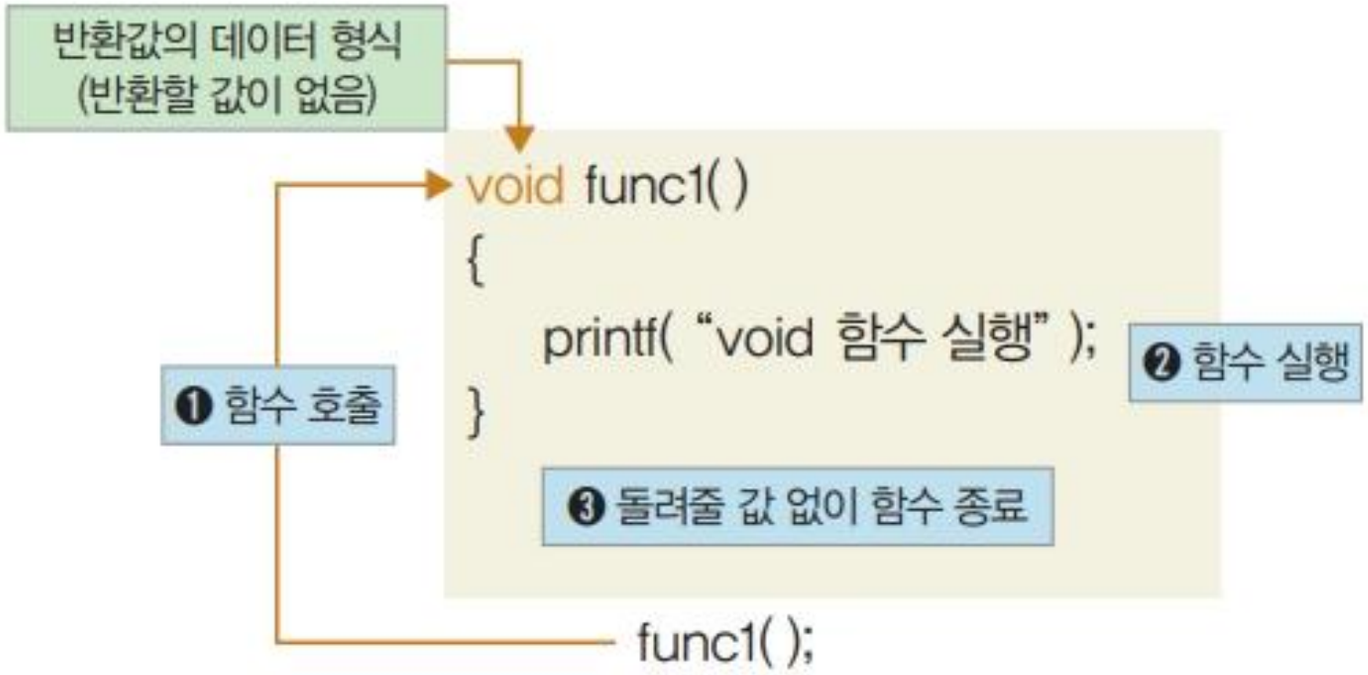


그림 10-9 void 형 함수의 작동

반환값 유무에 따른 함수 구분 (3/4)

- **반환값 유무에 따른 사용 예**

기본 10-7 반환값 유무에 따른 함수 비교 10-7.c

```
01 #include <stdio.h>
02
03 void func1( )                          ----- void 형 함수이므로 반환값이 없다.
04 {
05    printf("void 형 함수는 돌려줄 게 없음.\n");
06 }
07
08 int func2( )                           ----- int 형 함수이므로 반환값이 있다.
09 {
10    return 100;
11 }
12
13 void main( )
14 {
15    int a;
16
17    func1( );                           ----- void 형 함수를 호출한다.
18
19    a = func2( );                       ----- int 형 함수를 호출한다.
20    printf("int 형 함수에서 돌려준 값 ==> %d\n", a);
21 }
```

실행 결과

```
void 형 함수는 돌려줄 게 없음.
int 형 함수에서 돌려준 값 ==> 100
```

반환값 유무에 따른 함수 구분 (4/4)

- **값을 반환하지 않는 return**
  - ✓ 함수를 실행한 결과, 돌려줄 것이 없는 경우
  - ✓ “함수의 탈출(또는 종료)” 이라는 용도로 사용

```
void NoReturnType(int num)
{
    if(num<0)
        return;    // 값을 반환하지 않는 return문!
    . . . .
}
```

매개변수 전달 방법 (1/5)

- **값의 전달 (call by value)**

  ✓ 값 자체를 함수에 넘겨주는 방법 ➔ 값을 복사해서 전달

기본 10-8 매개변수 전달 방법: 값으로 전달 10-8.c

```
01 #include <stdio.h>
02
03 void func1(int a)
04 {
05    a = a + 1;                  ---- 전달받은 a 값을 1 증가시킨 후 출력한다.
06    printf("전달받은 a ==> %d\n", a);
07 }
08
09 void main( )
10 {
11    int a=10;                   ---- 변수 a를 선언한다.
12
13    func1(a);                   ---- a 값을 매개변수로 넘겨 함수를 호출한다.
14    printf("func1( ) 실행 후의 a ==> %d\n", a);  ---- 함수를 호출한 후 a 값을 출력한다.
15 }
```

실행 결과

```
전달받은 a ==> 11
func1( ) 실행 후의 a ==> 10
```

매개변수 전달 방법 (2/5)

- **값의 전달 (call by value) (cont'd)**
  - ✓ 11행에서 a에 10을 입력하고 13행에서 func1(a)를 호출
  - ✓ 3행의 a에 10이 들어가면 5행에서는 a 값을 1 증가시킨 후 출력
  - ✓ func1( ) 함수가 종료된 후 14행에서 a를 출력하니 원래의 10이 출력

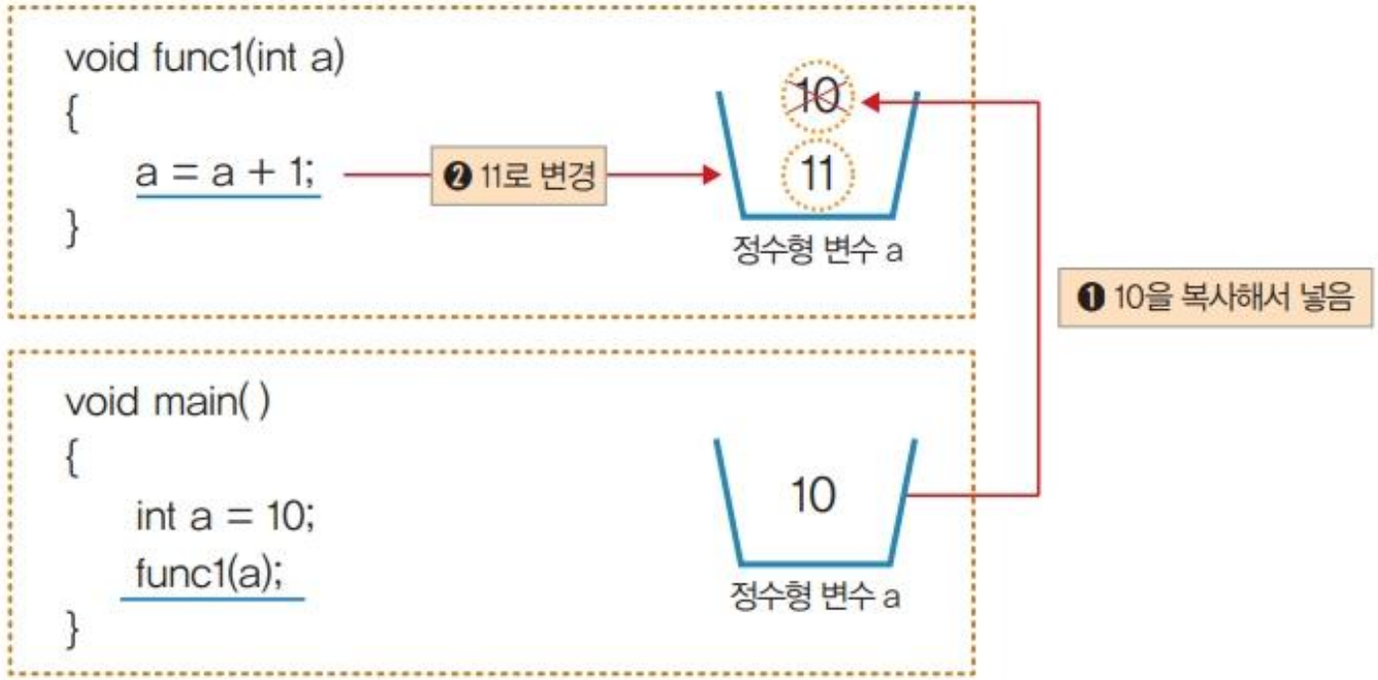
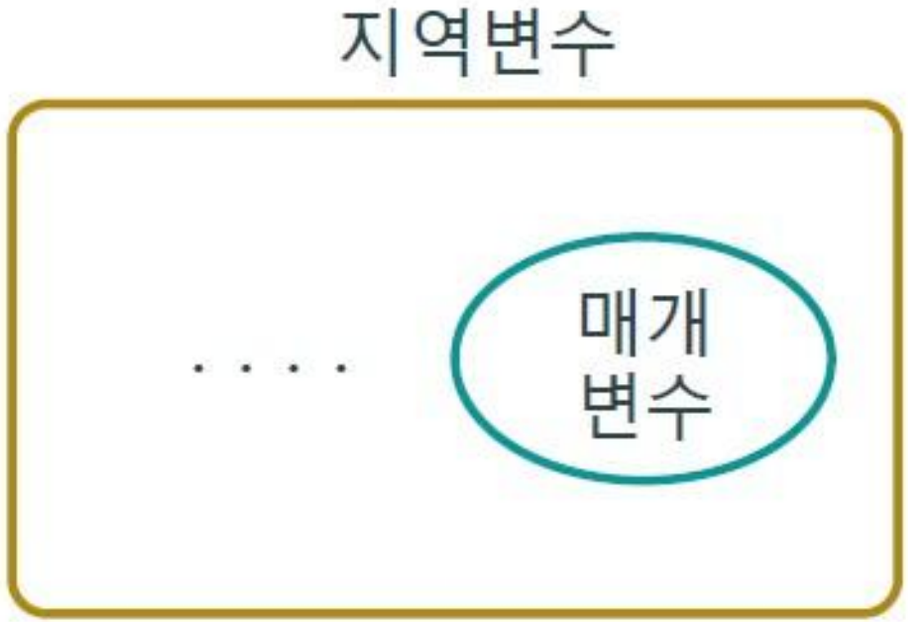


그림 10-10 매개변수 전달: 값으로 전달

매개변수 전달 방법 (3/5)

- **값의 전달 시 매개변수도 일종의 지역변수**
  - ✓ 선언된 함수가 반환하면, 매개변수도 소멸



매개변수 전달 방법 (4/5)

- **주소(또는 참조)로 전달**

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 }
```

실행 결과

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

매개변수 전달 방법 (5/5)

- **주소(또는 참조)로 전달 (cont'd)**

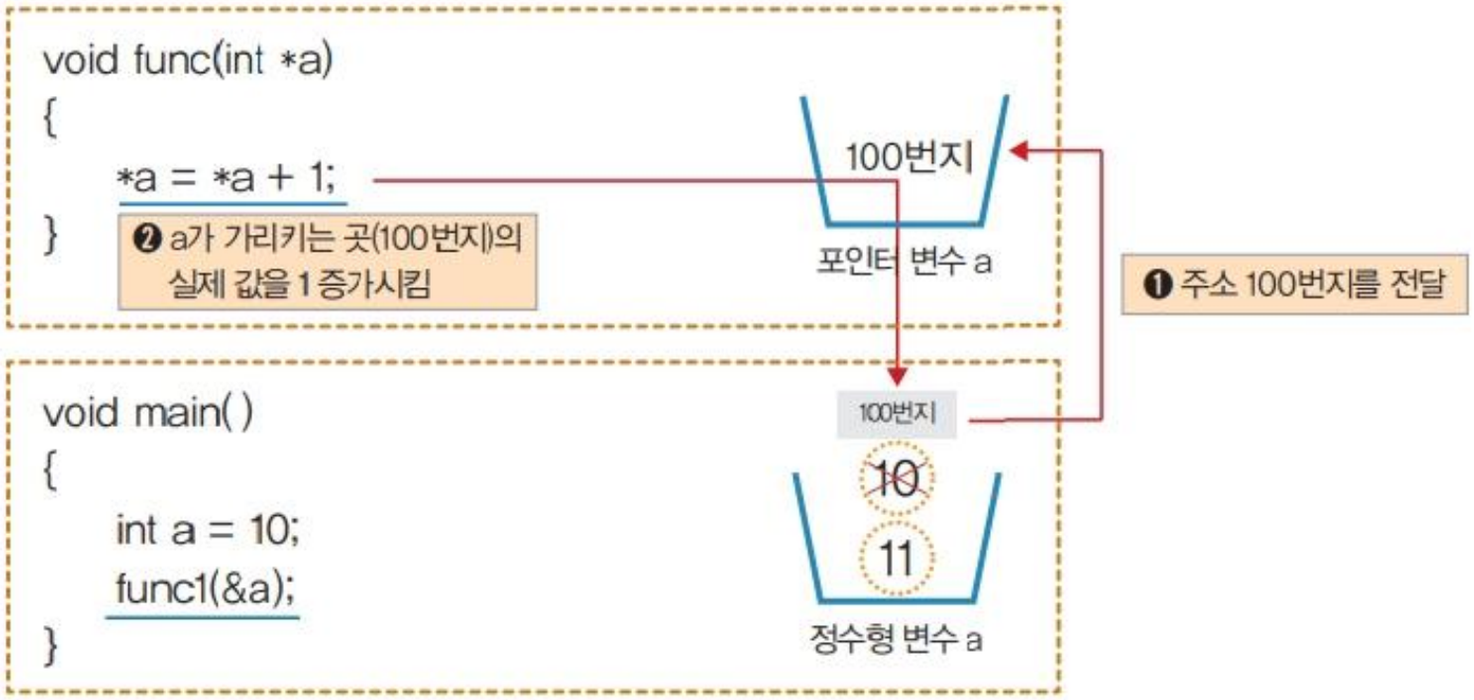


그림 10-11 매개변수 전달: 주소로 전달

예 제

[예제 01] 함수를 이용한 구구단 프로그램

예제 설명 함수를 사용하여 구구단을 출력하는 프로그램이다.

실행 결과

```
출력하고 싶은 단을 입력 : 7
7 X  1=  7
7 X  2= 14
7 X  3= 21
7 X  4= 28
7 X  5= 35
7 X  6= 42
7 X  7= 49
7 X  8= 56
7 X  9= 63
```

❶ 구구단을 출력하는 void gugu(int dan)함수를 구현

❷ 출력하고 싶은 단을 입력 받아서 구구단을 출력하는 프로그램(main( ))을 구현

[예제 02] 사용자 입력값의 누적 합계 출력 프로그램

1. 사용자로부터 숫자를 입력 받고, 받을 때마다 그때까지의 합계를 출력
2. 사용자가 0을 입력하면 프로그램을 종료
3. 합계를 계산해서 반환하는 int addToTotal(int num) 함수 구현
4. 전역변수를 사용하지 않고, addToTotal( )에 static 변수를 사용하여 합계를 저장

[예제 03] 함수들을 이용한 계산기 프로그램 v2.0

1. 사용자로부터 2개의 숫자와 연산자를 선택 받아 계산을 해주는 계산기 프로그램 작성
2. 제공하는 연산은 4칙 연산과 나머지 연산
3. 각 연산은 main( )가 아닌 각각 다른 함수로 처리

➔ +, -, *, /, % 연산을 처리하는 각각 별도의 함수가 존재해야 함

Q & A