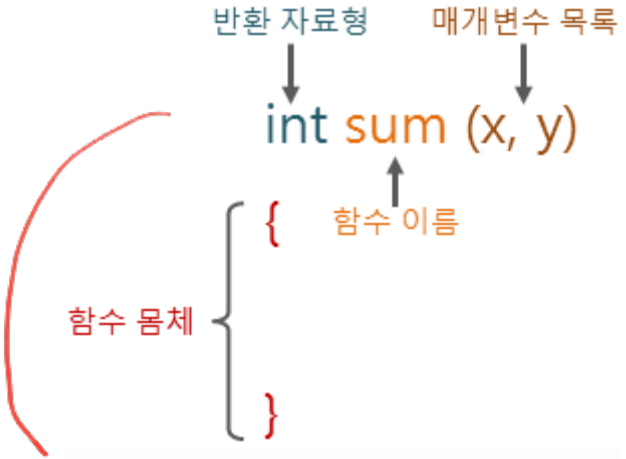
Function

Section 1 함수: 사용자 정의 함수
Section 2 전역 변수와 지역 변수
Section 3 함수의 입력과 출력

함수란?

● 함수의 개념

- 함수 (Function) : '입력에 대한 출력을 제공하는 기능'



- C 언어에서 자주 쓰일 수 있는 함수를 제공

```
함수_이름( );
```

- 예1) printf( ): C 언어에서 자체 제공하는 표준 출력 함수

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

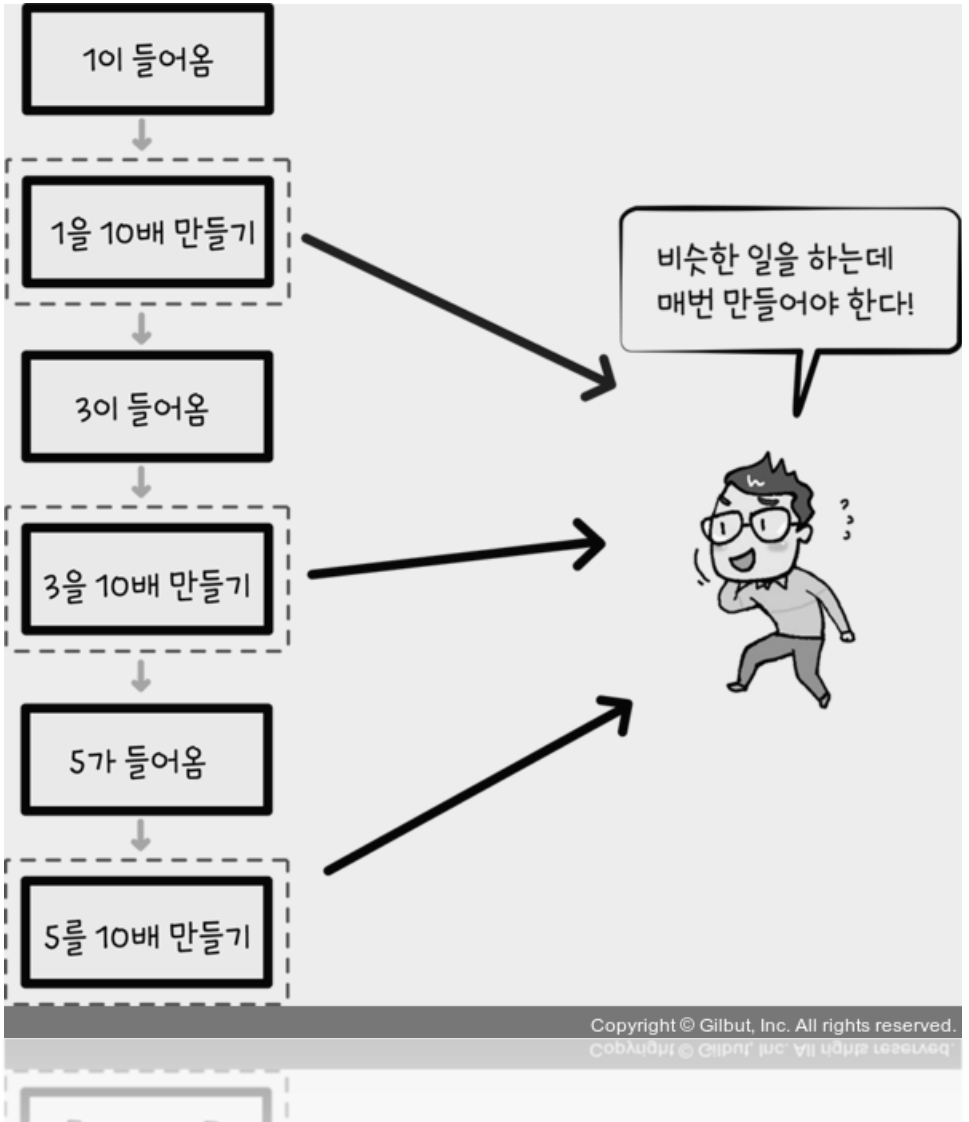
→ 'Basic-C' 출력

- 예2) round( ): C 언어에서 자체 제공하는 반올림 함수

함수란?

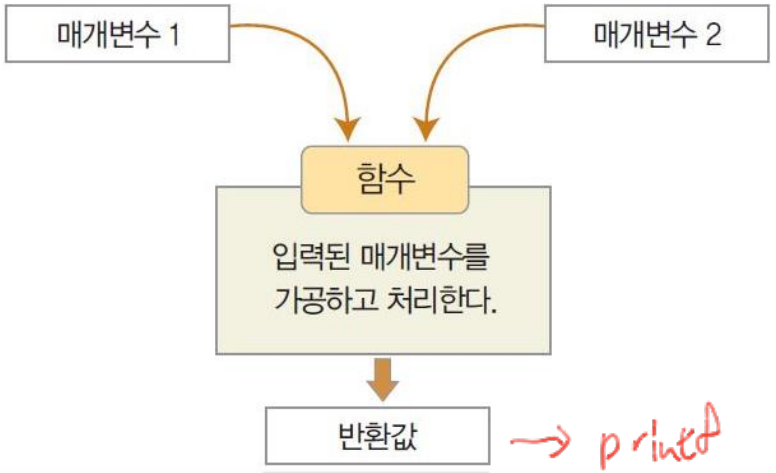
- **함수의 유용성**
  - 함수를 잘 작성해 놓으면,
  - 비슷한 일을 매번 수행하는데
  - 그것을 또 코드로 작성하지 않아도 된다.

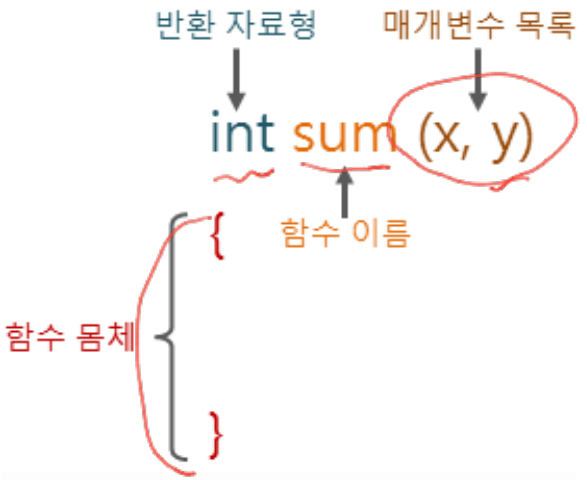
- **함수의 의미** → 재활용
  - 일종의 작은 프로그램 (subprogram)
    입력을 받아 출력을 내는 작은 프로그램
  - 내가 (나: int main() 함수)가 할 일을
    대신 처리해주는 작은 프로그램
  - C가 제공하지 않는 다른 기능을
    직접 스스로에게 제공하도록 작성하는
    작은 프로그램



함수의 정의 방법

● 함수의 기본 형태





- 함수는 '매개변수(또는 '인수')'를 입력 받아 가공하고 처리한 후 '반환값'을 돌려줌.

● 예시

- 커피 자판기에 비유하면 '동전 넣기'과 '버튼 입력'이라는 매개변수를 받아서
- 커피를 탄 후
- 반환값으로 '커피'를 돌려줌.

여기서 잠깐

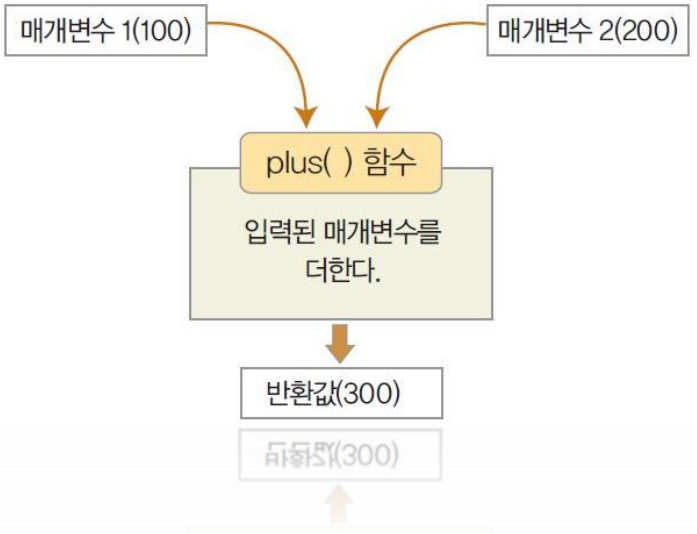
함수 사용의 장점

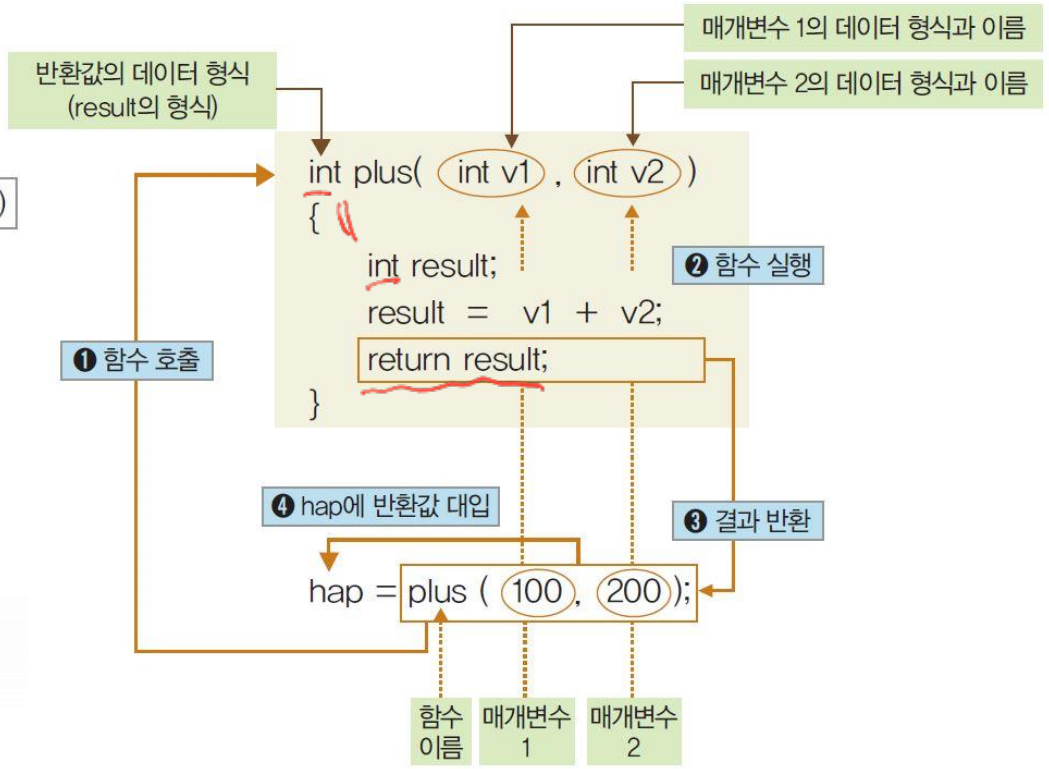
함수를 공부하는 이유는 다음과 같은 장점 때문이다. 지금은 좀 이해하기 어려울 테니 가볍게 훑어보자. 이 장을 마치고 다시 읽어보면 더 공감하게 될 것이다.

- 코드의 모듈화: 함수를 기능별로 작성하여 필요한 기능만 조합할 수 있다.
- 코드의 간략화: 반복되는 문장을 밖으로 빼냄으로써 C 소스를 간결하게 만든다.
- 코드의 재사용화: 한 번 작성한 함수를 다시 사용할 수 있다.
- 코드의 쉬운 수정: 프로그램의 오류를 수정하기가 쉽다.

함수의 정의 방법

- **더하기 함수 plus()의 정의와 호출**
  - 오른쪽 그림: plus()의 정의
  - 왼쪽 그림: 간단한 개념적 표현





- **함수의 더 큰 장점**
  - 예) printf 함수가 실제 내부적으로 어떻게 구현이 되었는지 아는가?
  - 즉, 내부에 어떻게 구현이 되어있는지를 모르더라도
    입력에 따른 결과를 정확히 이해한다면, 함수 형식만으로도 사용할 수 있다.

2 전역변수와 지역변수

변수의 종류

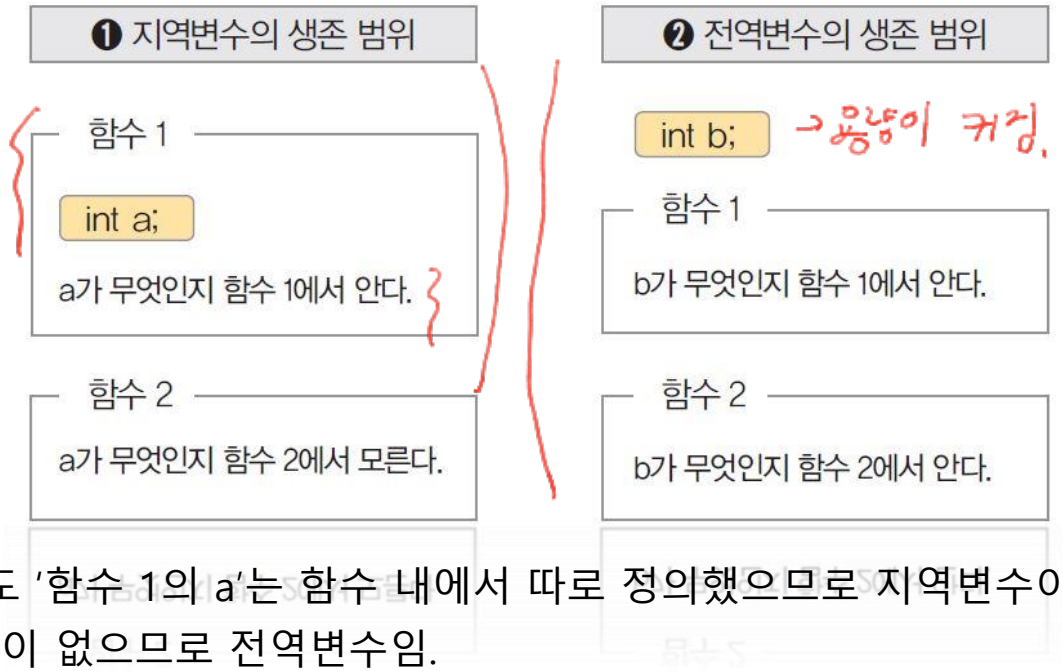
- **지역변수**
  - 한정된 지역(local)에서만 사용되는 변수
  - 함수 블록 { } 안에서만 생존하며, 블록이 끝나면 사라진다.
  - 함수 뿐 아니라 블록을 사용하는 모든 표현(for, while, if)의 블록 등에도 해당한다.

- **전역변수**
  - 프로그램 전체(global)에서 사용되는 변수

2 전역변수와 지역변수

변수의 종류

- **변수의 생존 시기 – 예시**
  - ❶에서 a가 '함수 1' 안에 선언됨. 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.

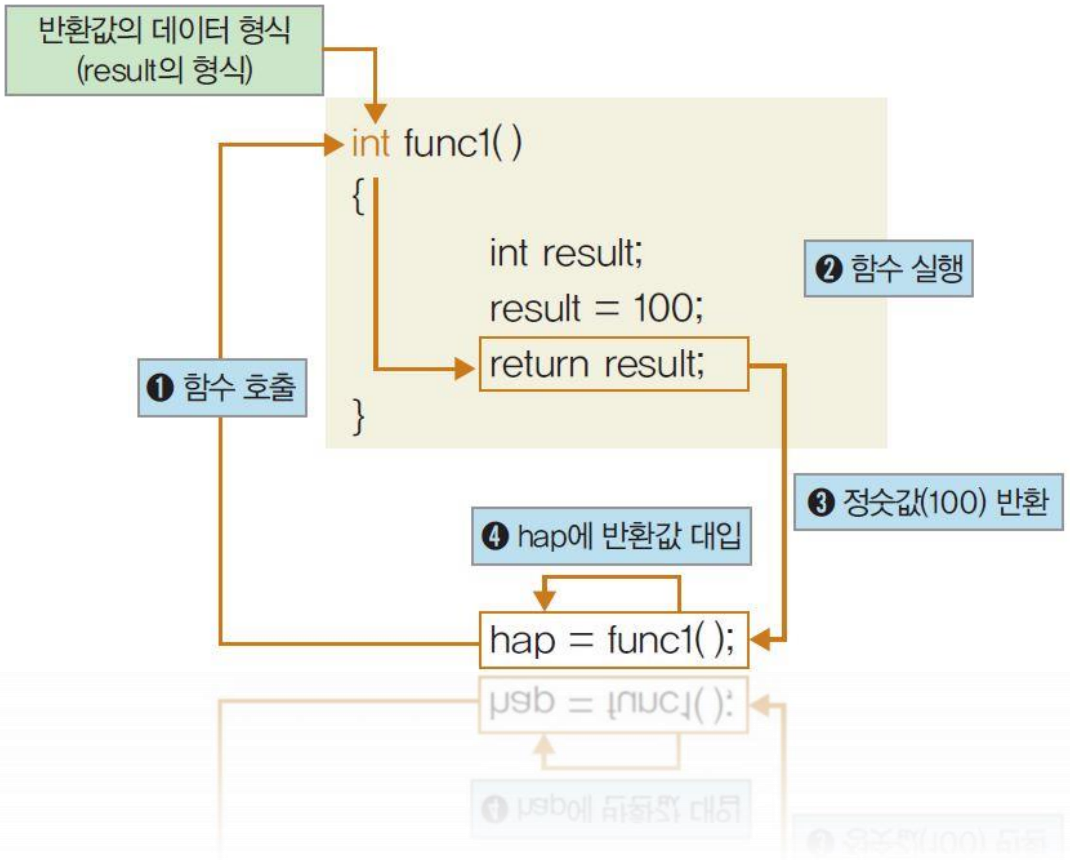


  - 같은 a라고 해도 '함수 1의 a'는 함수 내에서 따로 정의했으므로 지역변수이고 '함수 2의 a'는 함수 안에 정의된 것이 없으므로 전역변수임.

반환 값에 따른 함수의 유형 구분

● 반환 값이 있는 함수

- 함수를 실행한 결과값은 함수의 데이터형을 따름
- 'int 함수 이름( )'으로 정의했다면 결과도 정수형 변수나 정수값이어야 함.
  - 'return 정수형 변수;' 또는 'return 정수;'로 표현해야 함



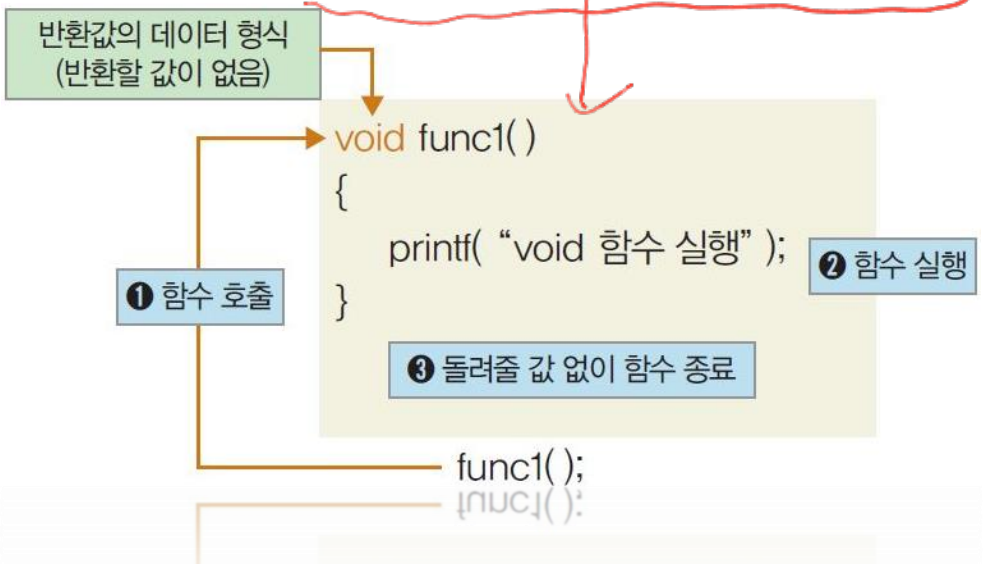
반환 값에 따른 함수의 유형 구분

● 반환 값이 포인터인 함수

- int가 아니라 float, char도 가능하며 심지어는 int*, double*, char*도 가능하다
- 단, 배열은 안됨 ➔ 배열이 아닌 배열의 포인터 상수는 가능
- 결과가 명시적인 숫자나 글자일 필요는 없다.
- 주소도 일종의 값.

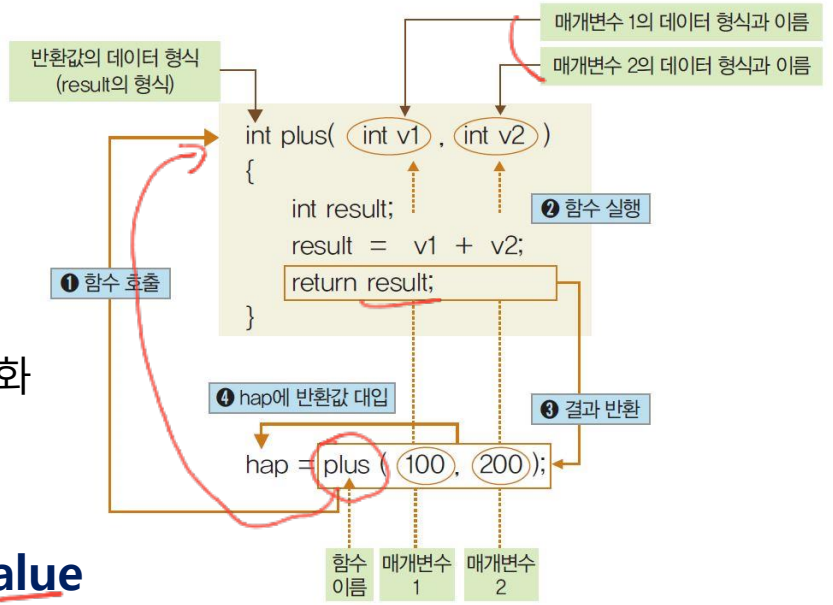
● 반환 값이 없는 함수 – void 함수

- 함수를 실행한 결과로 돌려줄 것이 없는 경우
- 함수의 데이터형을 void로 표시: void 형 함수를 호출할 때는 함수 이름만 쓴다.



매개변수 전달 방법

- **매개변수의 선언부를 주목**
  - 엄연히 int v1, int v2로 변수를 선언하고 있다.
  - 함수 실행 시 v1, v2인 변수를 선언한다.
  - v1와 v2의 초기화는 함수 호출 시 수행
  - 함수 호출 시 전달 받은 값을 v1, v2에 복사하여 초기화

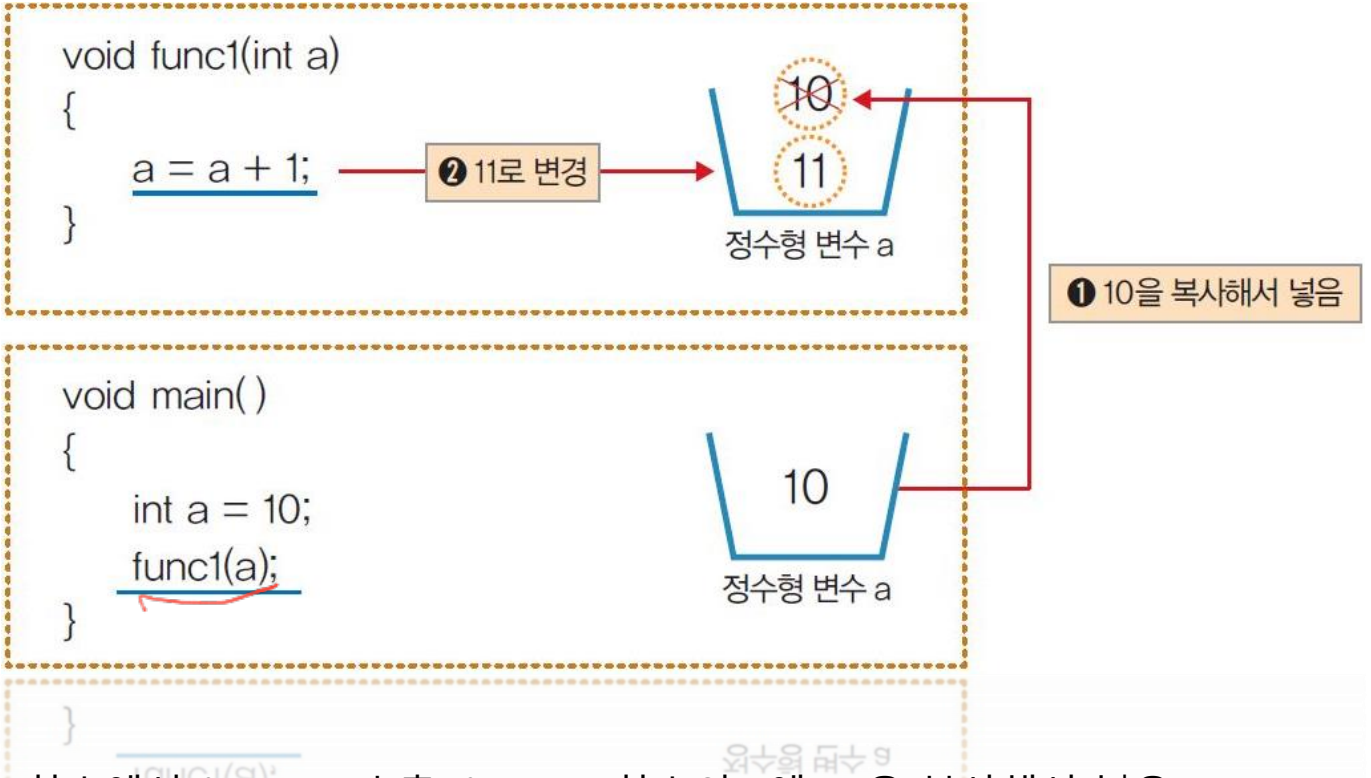


- **복사하는 값이 원래 값: 값으로 전달 – call by value**
  - 숫자나 문자 등의 값 자체를 함수에 넘겨주는 방법
  - 원래 값을 전달한 곳에는 아무런 영향을 미치지 않음

- **복사하는 값이 주소: 주소(또는 참조)로 전달 – call by reference**
  - 주소 값(address)을 함수에 넘겨주는 방법
  - 원래 값을 전달한 곳에 영향을 줌

매개변수 전달 방법

- **값으로 전달**

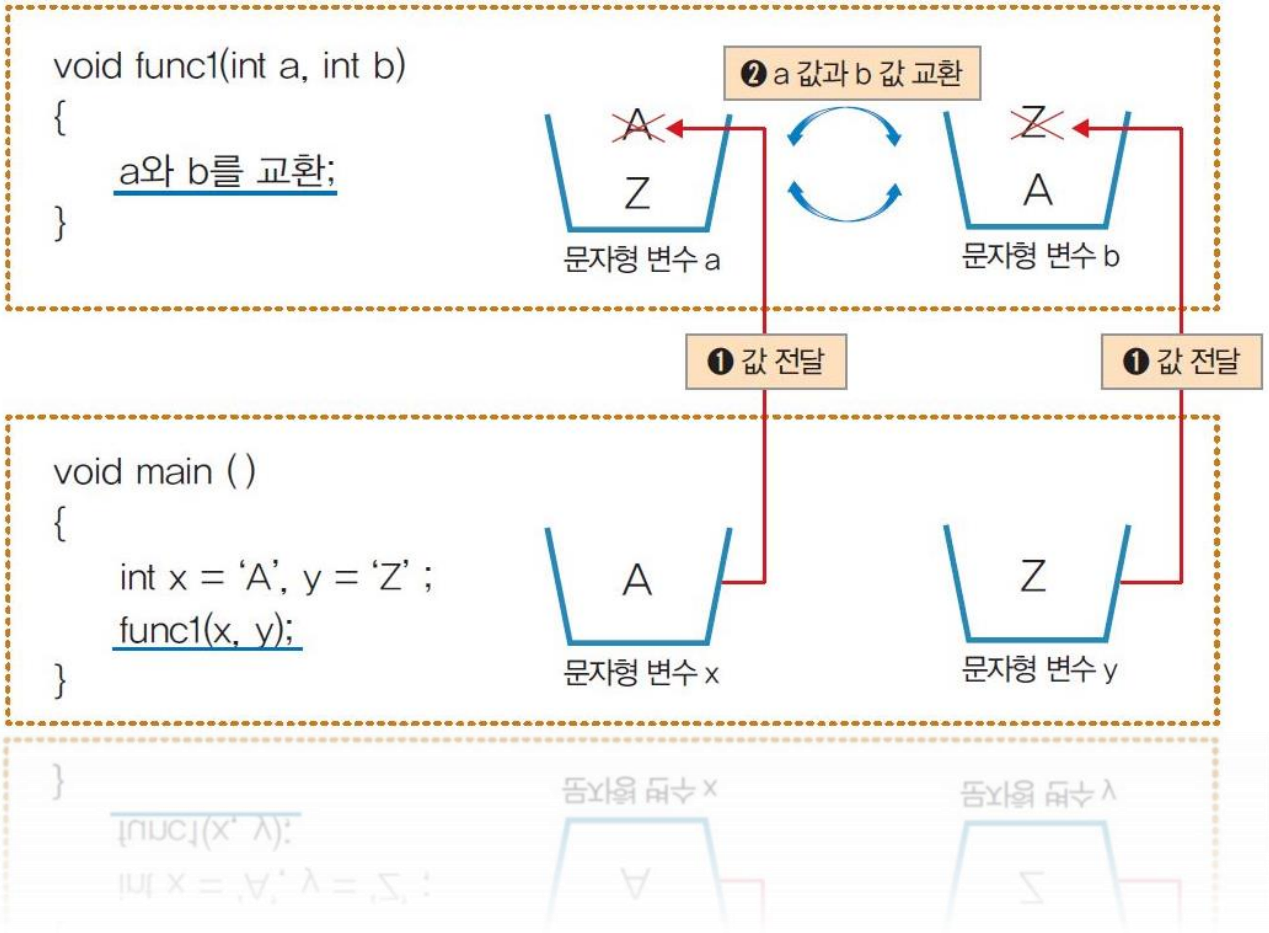


- main( ) 함수에서 func1(a) 호출. func1( ) 함수의 a에 10을 복사해서 넣음.
- func1( ) 함수에서는 a 값을 1 증가시켜서 11로 바꿈.
- main( )의 a는 변경되지 않고 10을 유지함.

매개변수 전달 방법

● 값으로 전달

- func1( )을 호출한 후에도 main() 함수에서 출력되는 값에 영향을 주지 않음



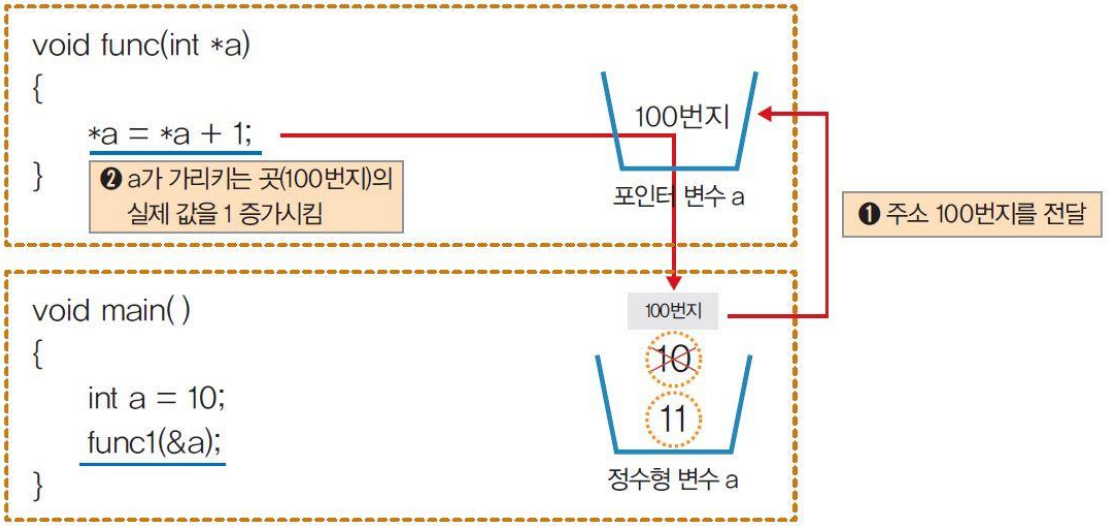
매개변수 전달 방법

● 주소(또는 참조)로 전달

- 포인터 변수는 주소를 저장
- 주소는 다른 변수를 가리킴

- 주소를 복사하여 사용하면
- 원본과 사본이 가리키는 변수는
- 사실상 동일하다.

- 따라서 *a로 수행한 결과가
- 원본의 값이 바뀔 수 있다.



여기서 잠깐

포인터에 대한 기억을 잠깐 되살려 정리해보자.

```
int *a;        ⇨ 주소를 저장할 수 있는 포인터 변수를 선언한다.
int b = 10;    ⇨ 정수형 변수이다.
a = &b;        ⇨ a에 b의 주소를 대입한다.
*a = 20;       ⇨ a가 가리키는 곳의 실제 값을 20으로 변경한다.
```

포인터 다시 짚어보기

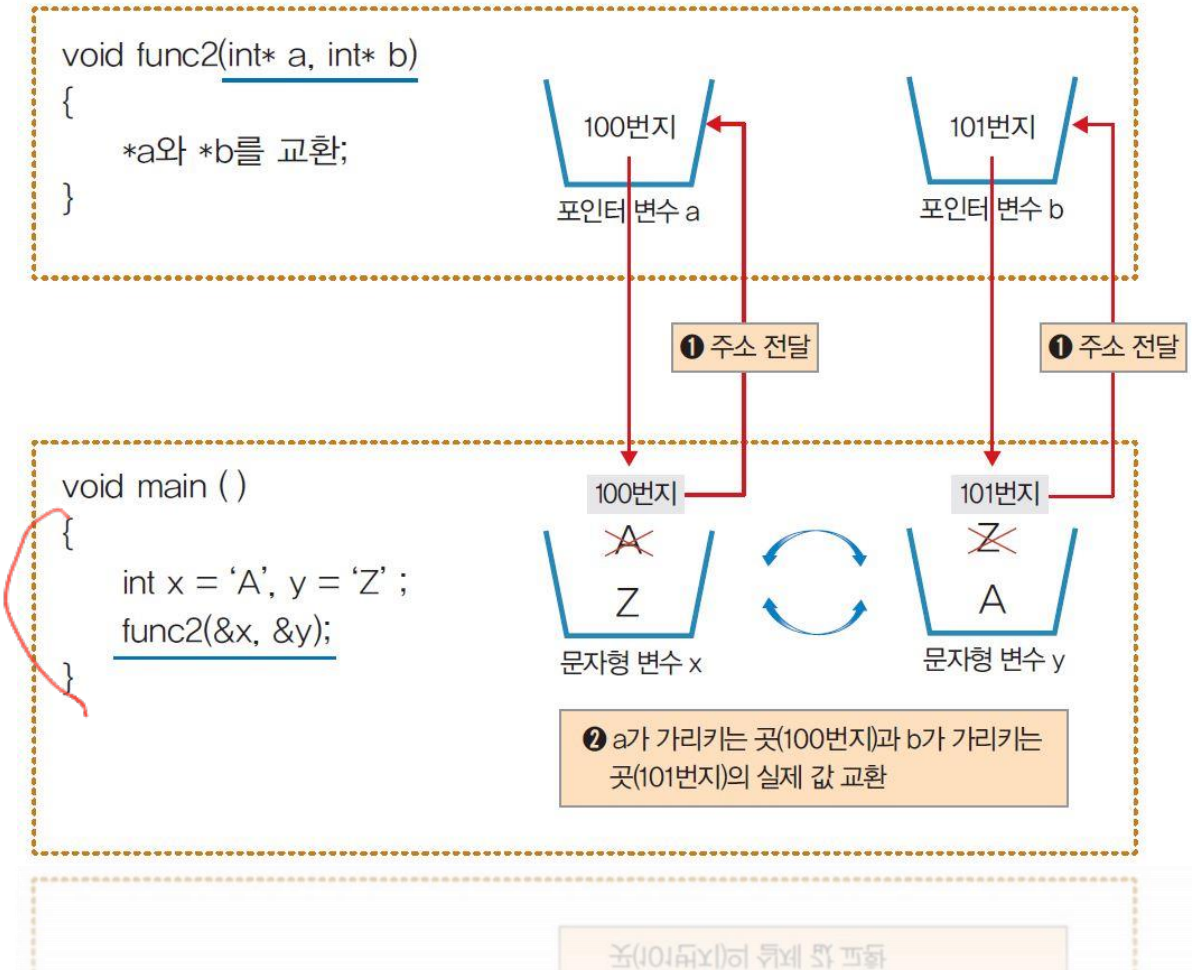
● 매개변수로 전달하는 원본이 수정 가능하다.

- 반환 값 대신에 매개변수에 전달하여 원본을 수정할 수도 있다.

매개변수 전달 방법

- **주소(또는 참조)로 전달**
  - 주소값을 매개변수로 주었기 때문에
  - func2( )함수를 호출하면 main( )함수에서 출력되는 값에도 영향을 줌

포인트



Summary

컴퓨터 공학에서 익혀야 하는 핵심 개념 중 하나

1. 함수의 이해

❶ 어떤 값이 들어가면 그것을 처리한 후 하나의 결과값을 돌려준다.

❷ 간단히 '함수 이름( )' 형식으로 사용한다.

❸ 함수는 반복적인 것을 처리할 때 유용하다

2. 함수를 정의하고 호출하는 예

```
int plus (int v1, int v2)
{
   int result;
   result = v1 + v2;
   return result;
}
hap = plus (100, 200);
```

3. 지역변수와 전역변수

지역변수는 선언된 블록 안에서만 유효한 변수이고,

전역변수는 모든 범위에서 유효한 변수이다.

Summary

컴퓨터 공학에서 익혀야 하는 핵심 개념 중 하나

4. 함수의 반환 값

❶ 함수에서 값을 돌려주기 위해서는 return문을 사용한다.

❷ 함수가 돌려줄 값에 따라 함수 이름 앞에 데이터 형식이 붙는다.
즉 정수값을 반환하려면 'int 함수 이름( )'처럼 사용한다.

❸ 돌려줄 값이 없다면 함수를 void 형으로 선언한다.

5. 매개변수 전달 방법

매개변수의 선언도 형태를 자세히 살펴보면 변수의 선언이다.

이 변수의 초기화는 함수를 호출할 때 전달 받은 값을 복사하는 것으로 이루어진다.

❶ 값으로 전달 : 값을 복사해서 해당 함수에서 사용하는 것이므로 기존의 변수에 들어가는 값은 변하지 않는다.

❷ 주소로 전달 : 값이 들어있는 주소를 복사하기 때문에 연산 결과에 따라 기존의 값이 변한다.