# '값에 의한 호출'로 객체 전달



- 함수를 호출하는 쪽에서 객체 전달
  - 객체 이름만 사용
- 함수의 매개 변수 객체 생성
  - 매개 변수 객체의 공간이 스택에 할당
  - 호출하는 쪽의 객체가 매개 변수 객체에 그대로 복사됨
  - **매개 변수 객체의 생성자는 호출되지 않음**
- 함수 종료
  - **매개 변수 객체의 소멸자 호출**


- *값에 의한 호출 시 매개 변수 객체의 생성자가 실행되지 않는 이유?*
  - 호출되는 순간의 실인자 객체 상태를 매개 변수 객체에 그대로 전달하기 위함

# '값에 의한 호출' 방식으로 increase(Circle c) 함수가 호출되는 과정

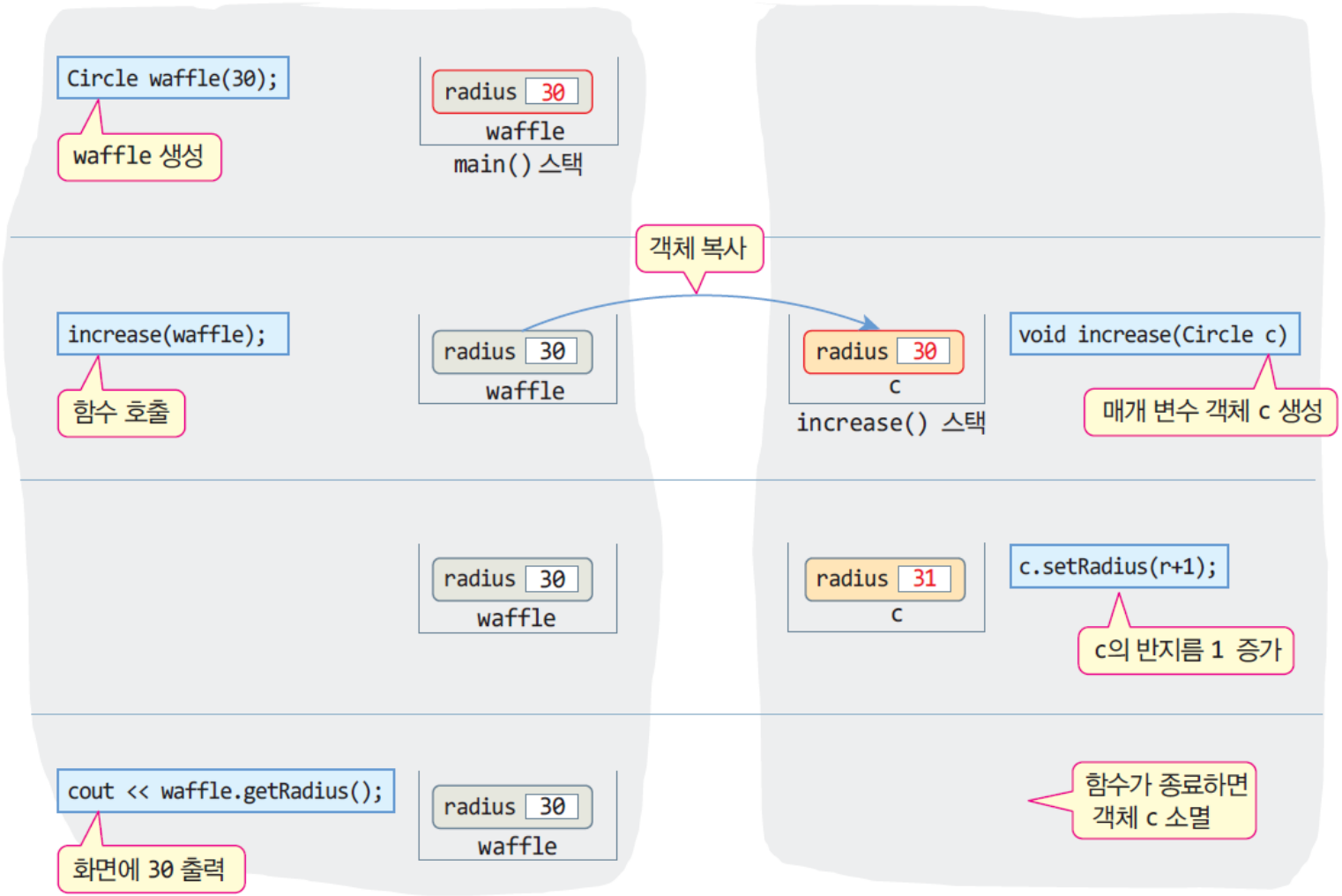
실행 결과

```
30
```

```
int main() {
    Circle waffle(30);
    increase(waffle);
    cout << waffle.getRadius() << endl;
}
```

call by value

```
void increase(Circle c) {
    int r = c.getRadius();
    c.setRadius(r+1);
}
```



# '주소에 의한 호출'로 increase(Circle *p) 함수가 호출되는 과정



```
int main() {
    Circle waffle(30);
    increase(&waffle);
    cout << waffle.getRadius() ;
}
```

call by address

```
void increase(Circle *p) {
    int r = p->getRadius();
    p->setRadius(r+1);
}
```

Circle waffle(30);

waffle 생성

radius 30

waffle

main() 스택

waffle의 주소가 p에 전달

increase(&waffle);

함수호출

radius 30

waffle

p

increase() 스택

void increase(Circle *p)

매개 변수 포인터 p 생성

radius 31

waffle

p

p->setRadius(r+1);

waffle의 반지름 1 증가

함수가 종료하면 포인터 p 소멸

cout << waffle.getRadius();

31이 화면에 출력됨

radius 31

waffle

# 객체 치환 및 객체 리턴



- 객체 치환
  - 동일한 클래스 타입의 객체끼리 치환 가능
  - 객체의 모든 데이터가 복사

```
Circle c1(5);
Circle c2(30);
c1 = c2; // c2 객체를 c1 객체에 비트 단위 복사. c1의 반지름 30됨
```

  - 치환된 두 객체는 현재 내용물만 같을 뿐 독립적인 공간 유지
- 객체 리턴
  - 객체의 복사본 리턴

```
Circle getCircle() {
   Circle tmp(30);
   return tmp; // 객체 tmp 리턴
}
```

```
Circle c; // c의 반지름 1
c = getCircle(); // tmp 객체의 복사본이 c에 치환. c의 반지름은 30이 됨
```

# 참조 변수



- 참조 변수 선언
  - 참조자 &의 도입
  - 이미 존재하는 변수에 대한 다른 이름(별명)을 선언
    - 참조 변수는 이름만 생기며
    - 참조 변수에 새로운 공간을 할당하지 않는다.
    - 초기화로 지정된 기존 변수를 공유한다.

```
int n=2;
int &refn = n; // 참조 변수 refn 선언. refn은 n에 대한 별명

Circle circle;
Circle &refc = circle; // 참조 변수 refc 선언. refc는 circle에 대한 별명
```

# 참조에 의한 호출



- 참조를 가장 많이 활용하는 사례
  - 함수의 매개 변수를 참조 타입으로 선언
    - 참조 매개 변수(reference parameter)라고 부름
      - 참조 매개 변수는 실인자 변수를 참조함
    - 참조매개 변수의 이름만 생기고 공간이 생기지 않음
    - 참조 매개 변수는 실인자 변수 공간 공유
    - 참조 매개 변수에 대한 조작은 실인자 변수 조작 효과

# 참조 리턴



- C 언어의 함수 리턴
  - 함수는 반드시 값만 리턴
    - 기본 타입 값 : int, char, double 등
    - 포인터 값
- C++의 함수 리턴
  - 함수는 값 외에 참조 리턴 가능
  - 참조 리턴
    - 변수 등과 같이 현존하는 공간에 대한 참조 리턴
      - 변수의 값을 리턴하는 것이 아님

# 값을 리턴하는 함수 vs. 참조를 리턴하는 함수



문자 리턴

```
char c = 'a';

char get() { // char 리턴
   return c; // 변수 c의 문자('a') 리턴
}

char a = get(); // a = 'a'가 됨

get() = 'b'; // 컴파일 오류
```

(a) 문자 값을 리턴하는 get()

char 타입의 공간에 대한 참조 리턴

find()가 리턴한 공간에 'b' 문자 저장

```
char c = 'a';

char& find() { // char 타입의 참조 리턴
   return c; // 변수 c에 대한 참조 리턴
}

char a = find(); // a = 'a'가 됨

char &ref = find(); //  ref는 c에 대한 참조
ref = 'M'; // c= 'M'

find() = 'b'; // c = 'b'가 됨
```

(b) char 타입의 참조(공간)을 리턴하는 find()

# 디폴트 복사 생성자

- 디폴트 복사 생성자
  - 자신과 같은 클래스형의 레퍼런스를 인자로 갖는 생성자이다.
  - 복사 생성자는 오브젝트를 새 오브젝트에 복사한다.
  - 복사 생성자는 **오브젝트가 초기화될 때 호출**된다.
    (복사생성자는 반드시 참조형으로 받아야 한다)

  - 사용자 정의 복사  생성자가 없을 때 자동 삽입
  - 멤버 변수 대 멤버 변수의 복사를 수행한다.

# 복사 생성자



- 복사 생성자(copy constructor)란?
  - 객체의 복사 생성시 호출되는 특별한 생성자
- 특징
  - 한 클래스에 오직 한 개만 선언 가능
    - 클래스에 대한 참조 매개 변수를 가지는 독특한 생성자
- 복사 생성자 선언

```
class Circle {
    .....
    Circle(const Circle& c); // 복사 생성자 선언
    .....
};

Circle::Circle(const Circle& c) { // 복사 생성자 구현
    .....
}
```

자기 클래스에 대한 참조 매개 변수

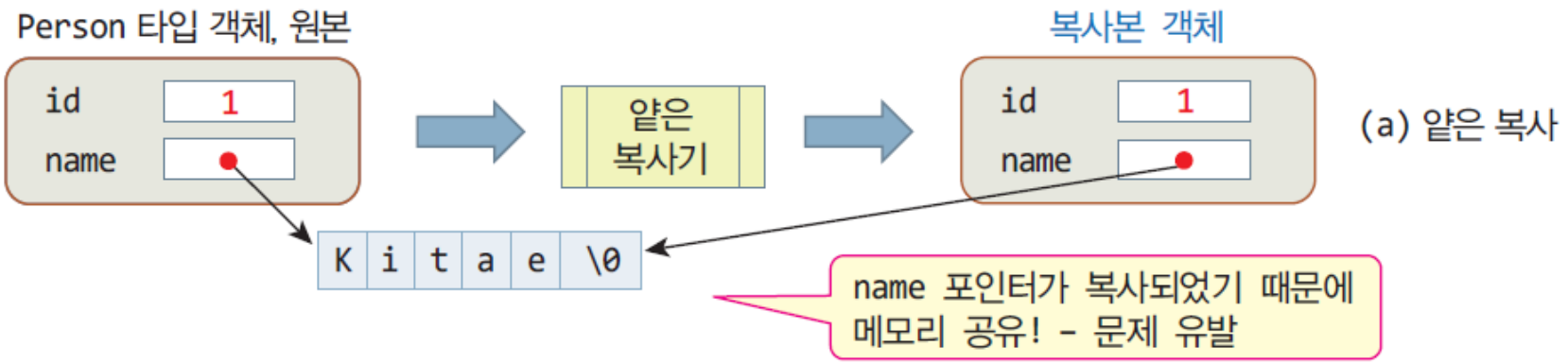
# C++에서 얕은 복사와 깊은 복사

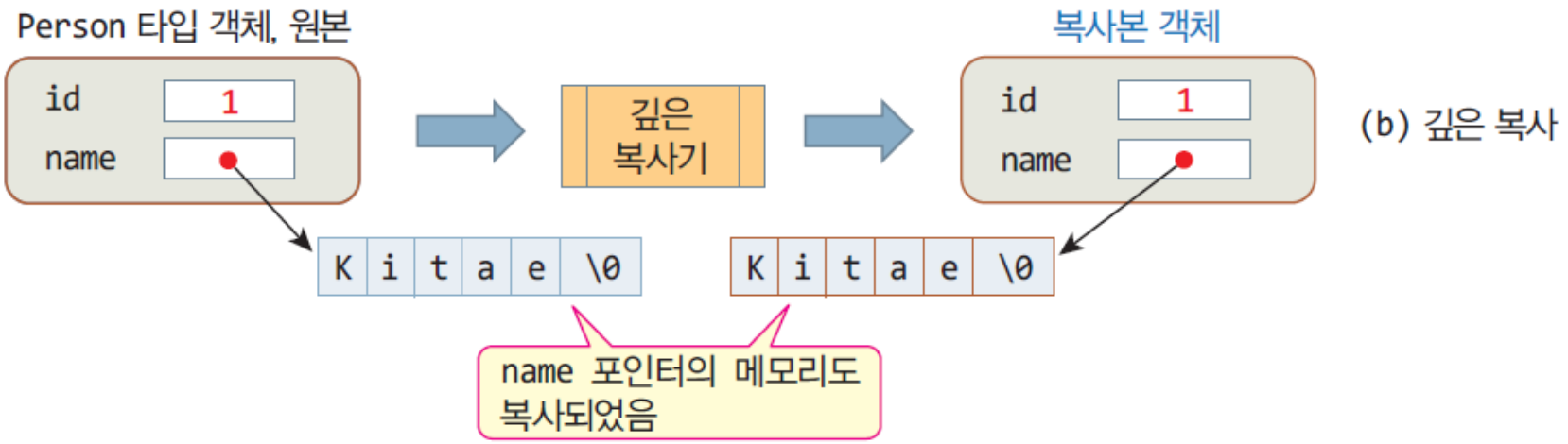


- 얕은 복사(shallow copy)
  - 객체 복사 시, 객체의 멤버를 1:1로 복사
  - 객체의 멤버 변수에 동적 메모리가 할당된 경우
    - 사본은 원본 객체가 할당 받은 메모리를 공유하는 문제 발생
- 깊은 복사(deep copy)
  - 객체 복사 시, 객체의 멤버를 1:1대로 복사
  - 객체의 멤버 변수에 동적 메모리가 할당된 경우
    - 사본은 원본이 가진 메모리 크기 만큼 별도로 동적 할당
    - 원본의 동적 메모리에 있는 내용을 사본에 복사
  - 완전한 형태의 복사
    - 사본과 원본은 메모리를 공유하는 문제 없음

# C++에서 객체의 복사

```
class Person {
   int id;
   char *name;
   ..................
};
```



# 복사 생성자의 호출 시기

- **복사 생성자 호출 형태 3가지**
  - **Case 1**
    - 기존에 생성된 객체로 새로운 객체 초기화
  - **Case 2**
    - 함수 호출 시 객체를 값에 의해 전달
  - **Case 3**
    - 함수 내에서 객체를 값에 의해 리턴