05

함수와 참조, 복사 생성자

학습 목표

1. 값에 의한 호출과 주소에 의해 호출을 복습한다.
2. 함수 호출 시 객체가 전달되는 과정을 이해한다.
3. 객체 치환과 객체 리턴을 이해한다.
4. 참조에 대한 개념을 이해하고, 참조 변수를 선언할 수 있다.
5. 참조에 의한 호출과 참조 리턴에 대해 이해하고 코드를 작성할 수 있다.
6. 복사생성자의 필요성과 활용을 이해하고, 작성할 수 있다.

함수의 인자 전달 방식 리뷰

- 인자 전달 방식
  - 값에 의한 호출, call by value
    - 함수가 호출되면 매개 변수가 스택에 생성됨
    - 호출하는 코드에서 값을 넘겨줌
    - 호출하는 코드에서 넘어온 값이 매개 변수에 복사됨
  - 주소에 의한 호출, call by address
    - 함수의 매개 변수는 포인터 타입
      - 함수가 호출되면 포인터 타입의 매개 변수가 스택에 생성됨
    - 호출하는 코드에서는 명시적으로 주소를 넘겨줌
      - 기본 타입 변수나 객체의 경우, 주소 전달
      - 배열의 경우, 배열의 이름
    - 호출하는 코드에서 넘어온 주소 값이 매개 변수에 저장됨

값에 의한 호출과 주소에 의한 호출

(a) 값에 의한 호출

(b) 주소에 의한 호출

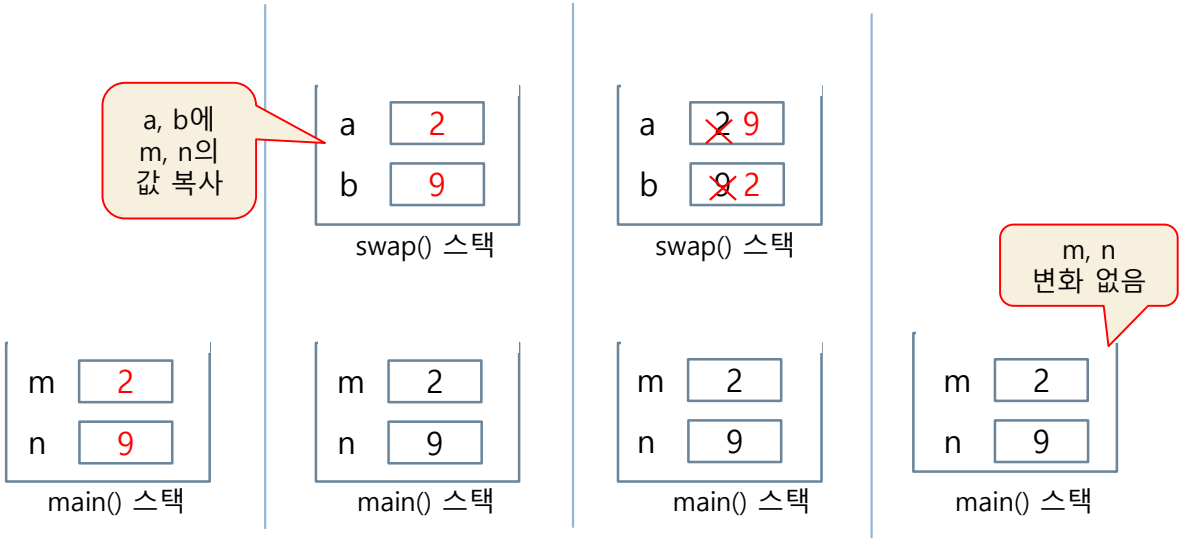
```
#include <iostream>
using namespace std;

void swap(int a, int b) {
    int tmp;

    tmp = a;
    a = b;
    b = tmp;
}

int main() {
    int m=2, n=9;
    swap(m, n);
    cout << m << ' ' << n;
}
```

2 9



(1) swap() 호출 전 (2) swap() 호출 직후 (3) swap() 실행 (4) swap() 리턴 후

값에 의한 호출

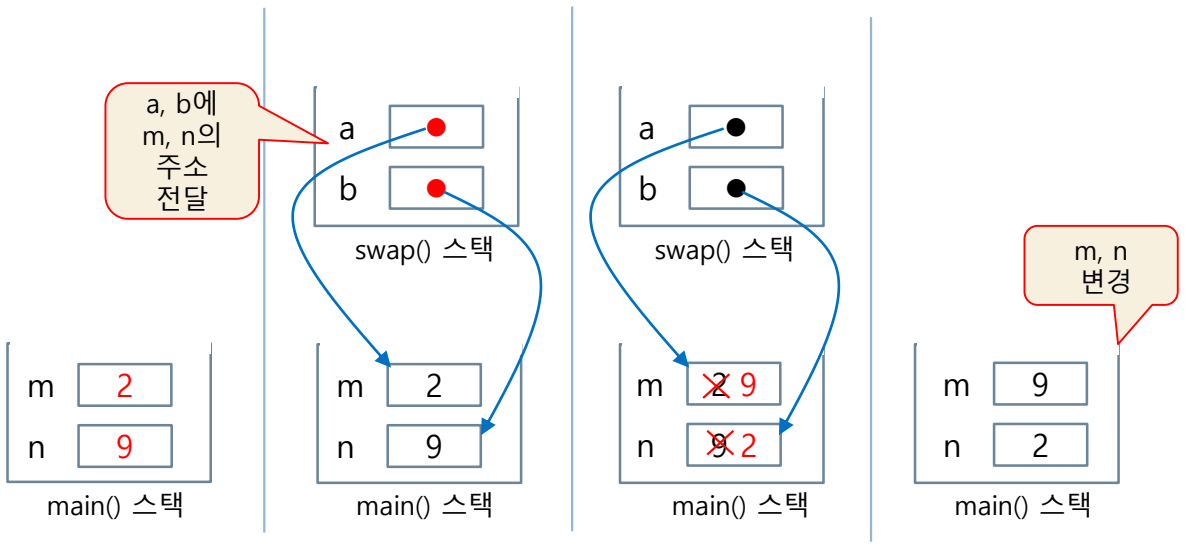
```
#include <iostream>
using namespace std;

void swap(int *a, int *b) {
    int tmp;

    tmp = *a;
    *a = *b;
    *b = tmp;
}

int main() {
    int m=2, n=9;
    swap(&m, &n);
    cout << m << ' ' << n;
}
```

9 2



(1) swap() 호출 전 (2) swap() 호출 직후 (3) swap() 실행 (4) swap() 리턴 후

주소에 의한 호출

'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

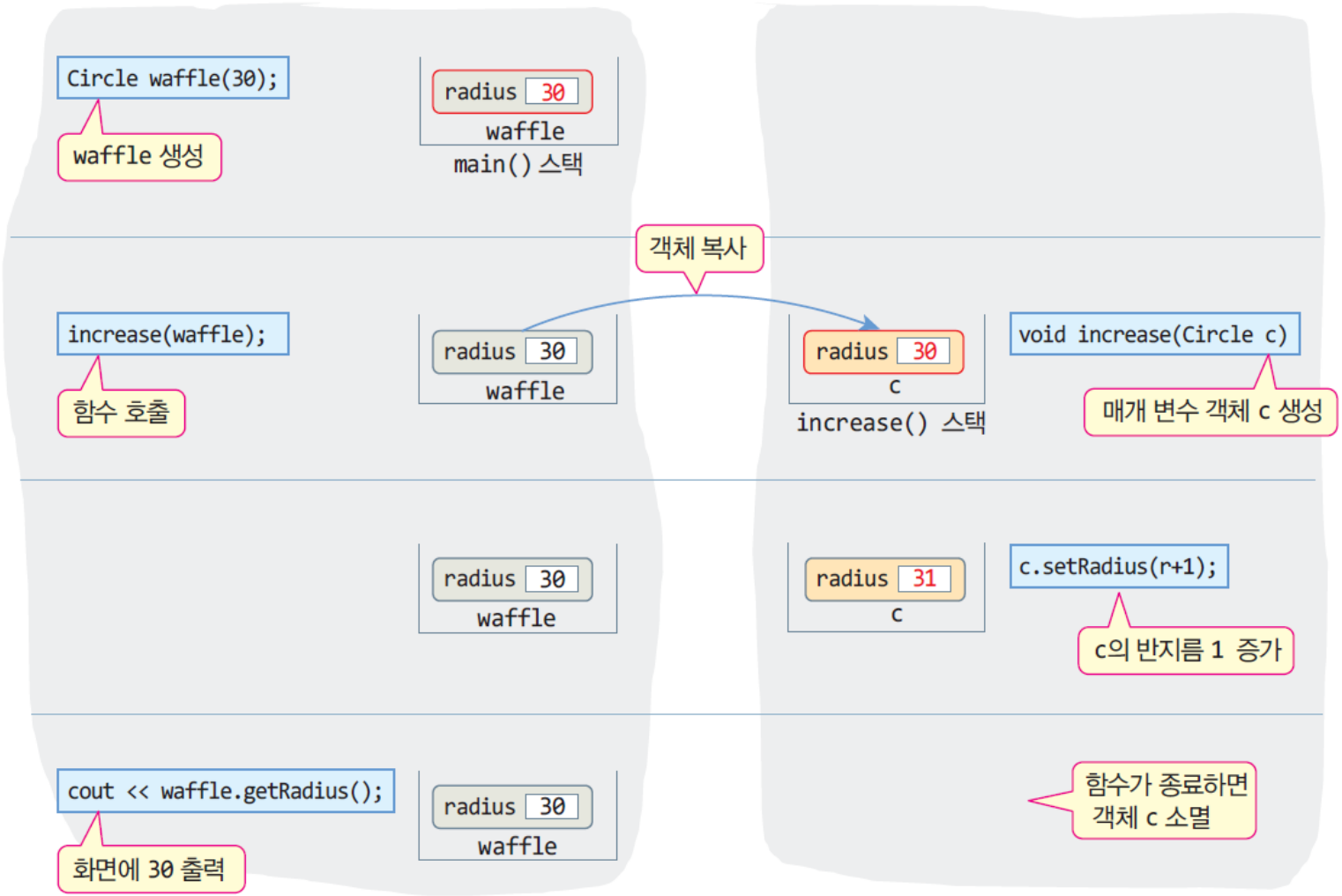
실행 결과

```
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);
}
```



예제 5-1 '값에 의한 호출'시 매개 변수의 생성자 실행되지 않음


```
#include <iostream>
using namespace std;

class Circle {
private:
    int radius;
public:
    Circle();
    Circle(int r);
    ~Circle();
    double getArea()  { return 3.14*radius*radius; }
    int getRadius() { return radius; }
    void setRadius(int radius) { this->radius = radius
};

Circle::Circle() {
    radius = 1;
    cout << "생성자 실행 radius = " << radius << endl;
}

Circle::Circle(int radius) {
    this->radius = radius;
    cout << "생성자 실행 radius = "  << radius << endl;
}

Circle::~Circle() {
    cout << "소멸자 실행 radius = " << radius << endl;
}
```

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

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

waffle의 내용이 그대로 c에 복사

```
생성자 실행 radius = 30
소멸자 실행 radius = 31
30
소멸자 실행 radius = 30
```

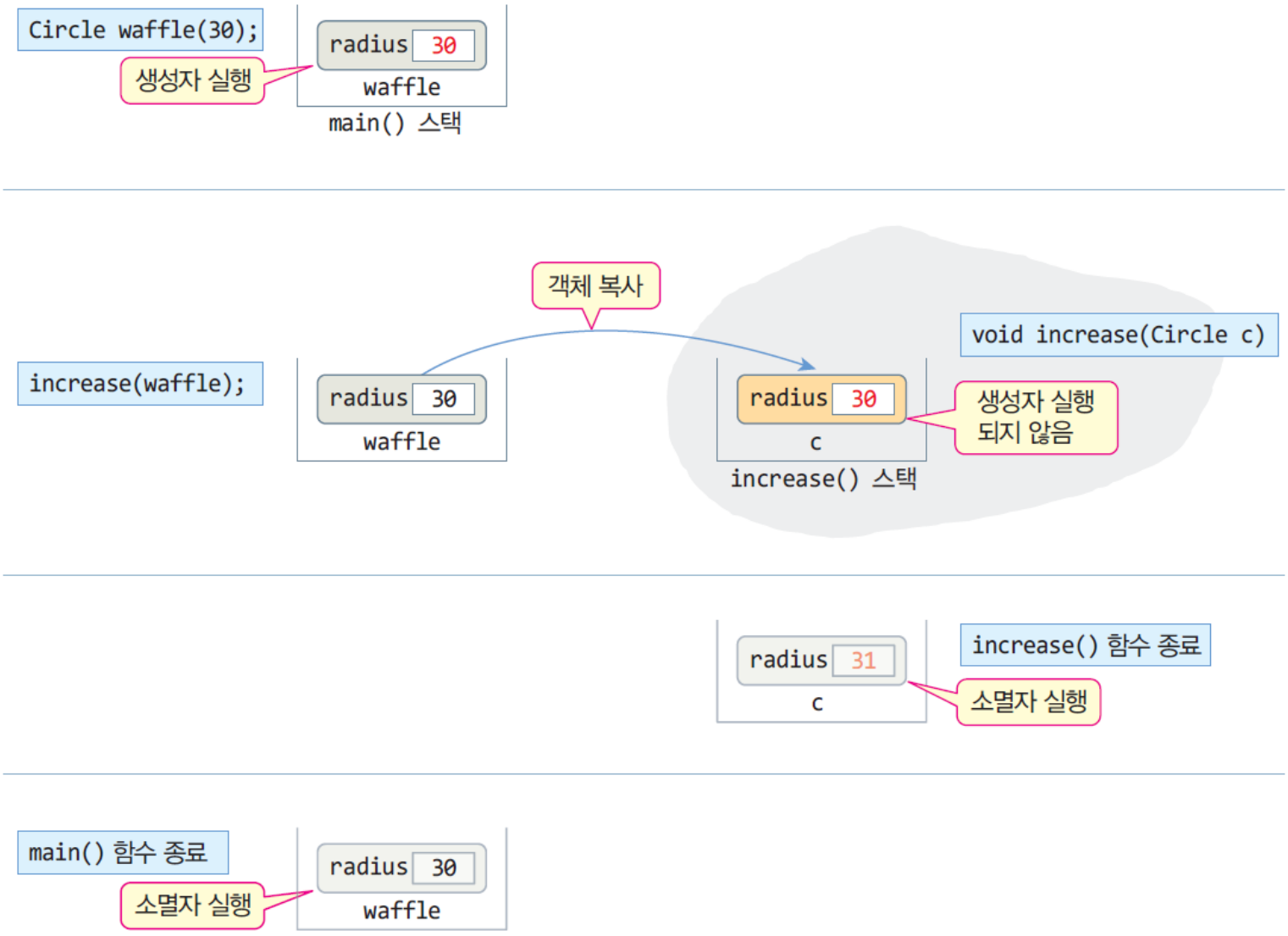
waffle 생성

c의 생성자 실행되지 않았음

c 소멸

waffle 소멸

'값에 의한 호출'시에 생성자와 소멸자의 비대칭 실행

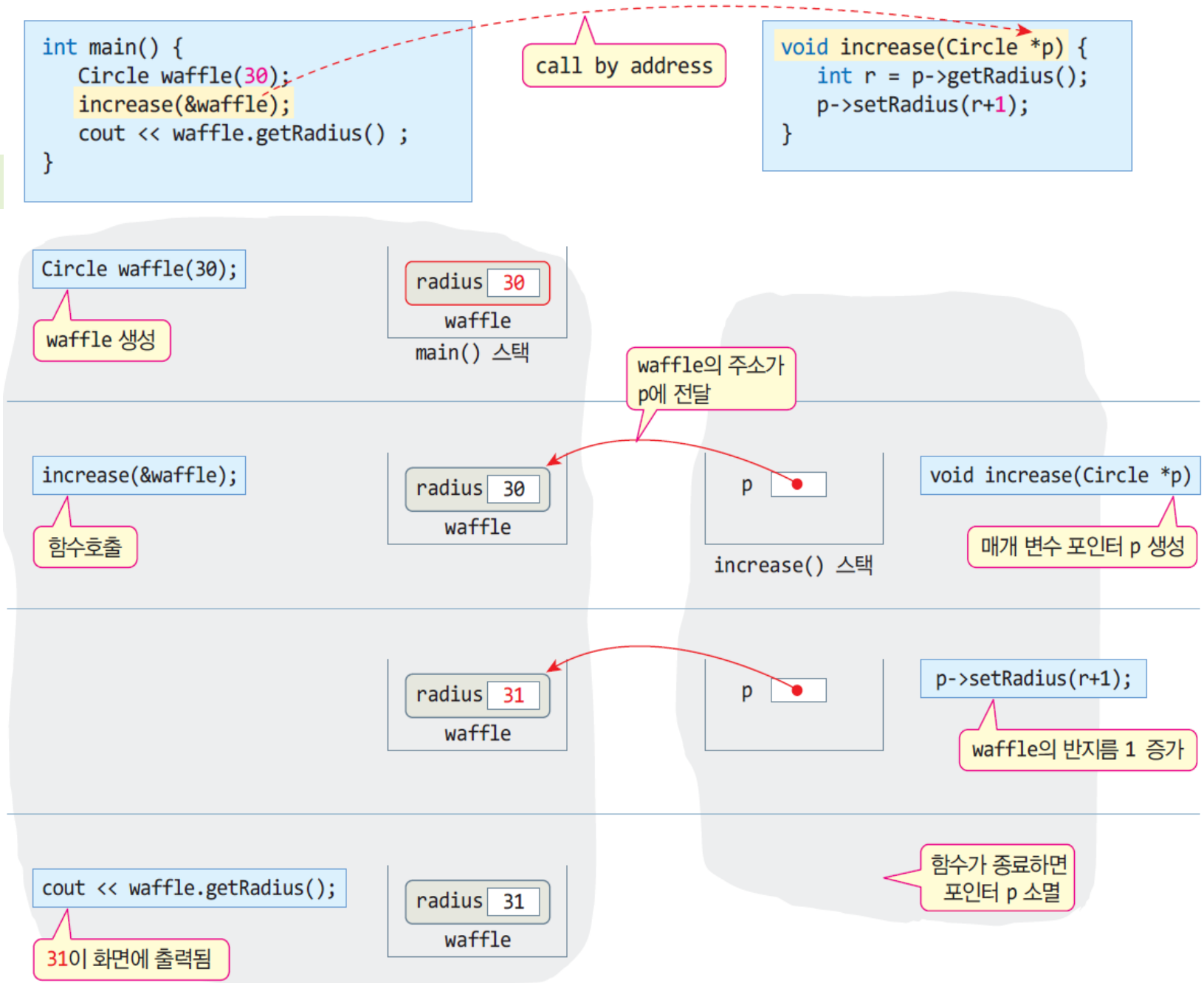


함수에 객체 전달 – '주소에 의한 호출'로

- 함수 호출시 객체의 주소만 전달
  - 함수의 매개 변수는 객체에 대한 포인터 변수로 선언
  - 함수 호출 시 생성자 소멸자가 실행되지 않는 구조

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

31



객체 치환 및 객체 리턴

- 객체 치환
  - 동일한 클래스 타입의 객체끼리 치환 가능
  - 객체의 모든 데이터가 비트 단위로 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
```

예제 5-3 객체 리턴


```
#include <iostream>
using namespace std;

class Circle {
    int radius;
public:
    Circle() { radius = 1; }
    Circle(int radius) { this->radius = radius; }
    void setRadius(int radius) { this->radius = radius; }
    double getArea() { return 3.14*radius*radius; }
};

Circle getCircle() {
    Circle tmp(30);
    return tmp; // 객체 tmp을 리턴한다.
}

int main() {
    Circle c; // 객체가 생성된다. radius=1로 초기화된다.
    cout << c.getArea() << endl;

    c = getCircle();
    cout << c.getArea() << endl;
}
```

tmp 객체의 복사본이 리턴된다.

tmp 객체가 c에 복사된다. c의 radius는 30이 된다.

```
3.14
2826
```