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

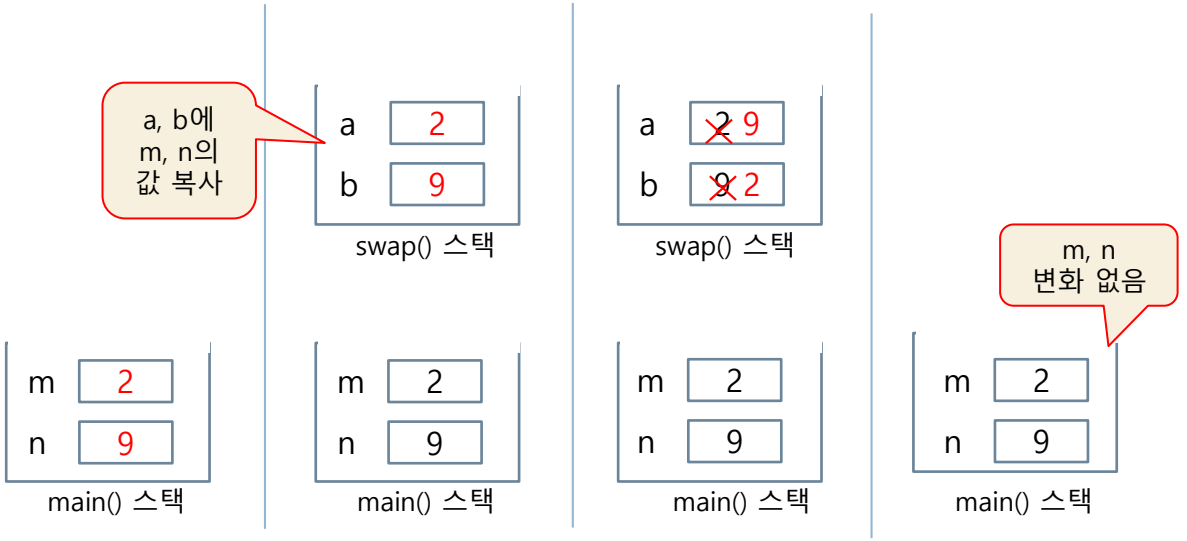
값에 의한 호출과 주소에 의한 호출

(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

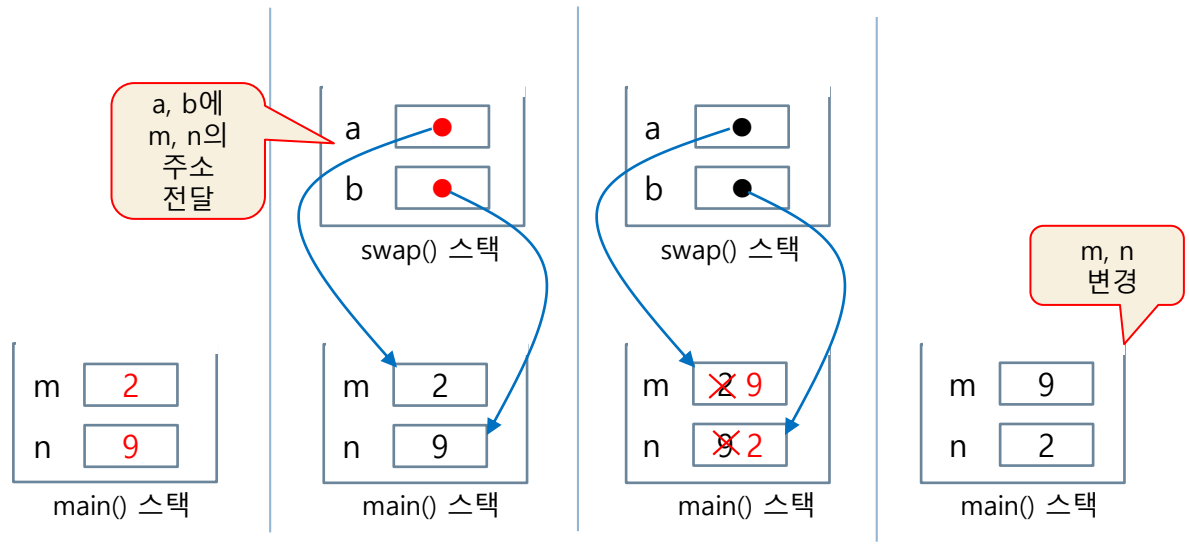


(1) swap() 호출 전 (2) swap() 호출 직후 (3) swap() 실행 (4) swap() 리턴 후

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

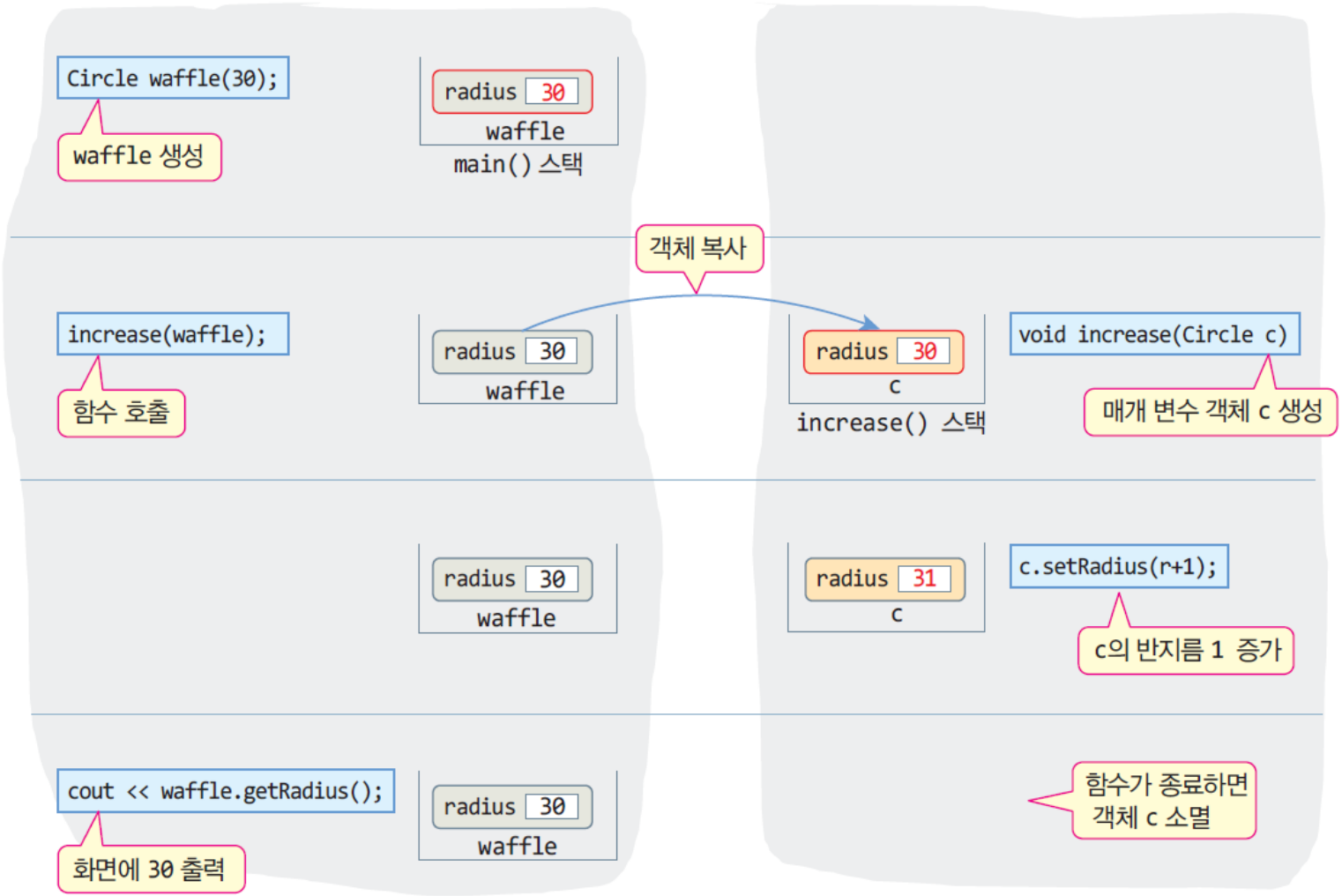
실행 결과

```
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
```

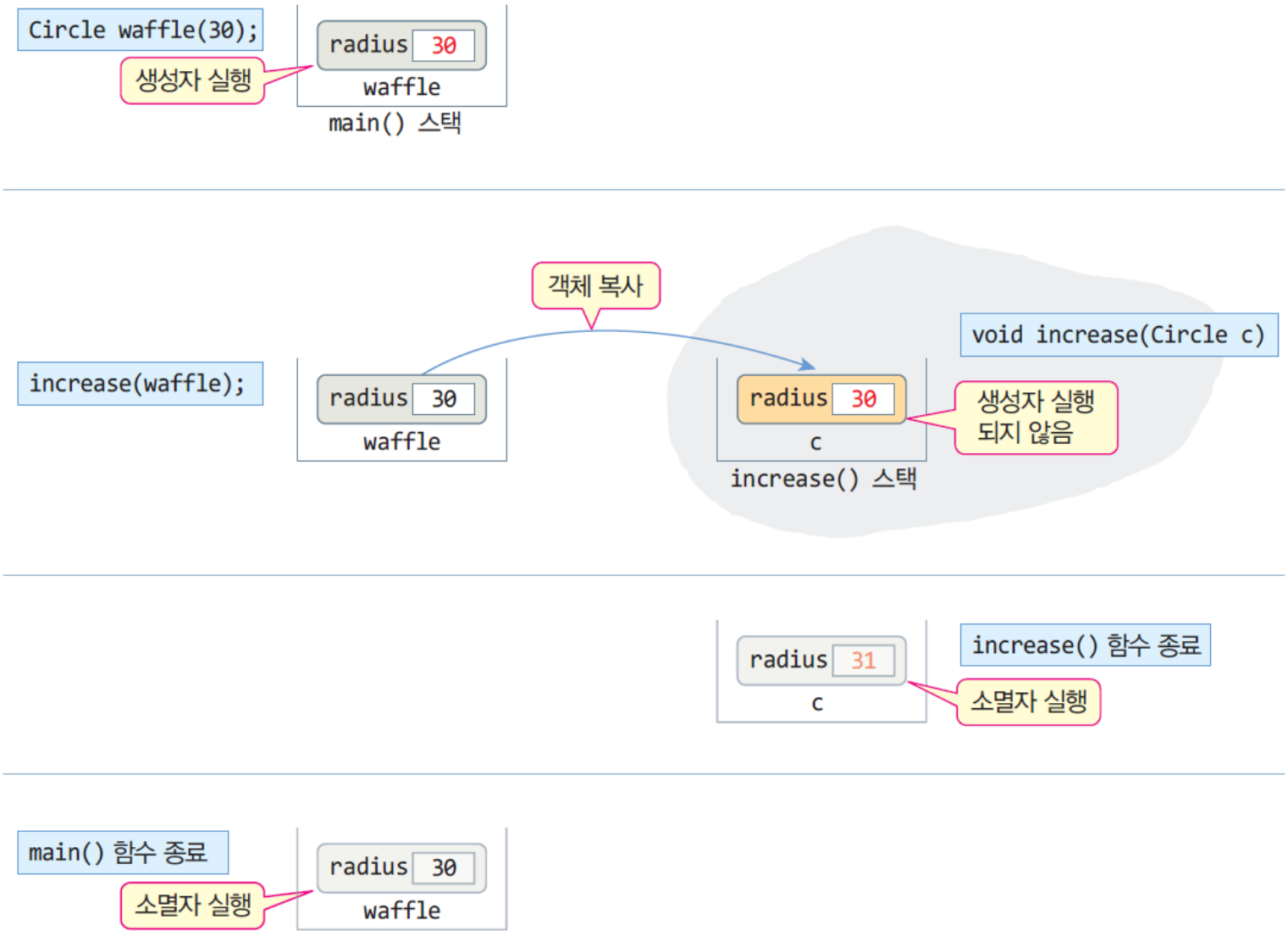
waffle 생성

c의 생성자 실행되지 않았음

c 소멸

waffle 소멸

'값에 의한 호출'시에 생성자와 소멸자의 비대칭 실행

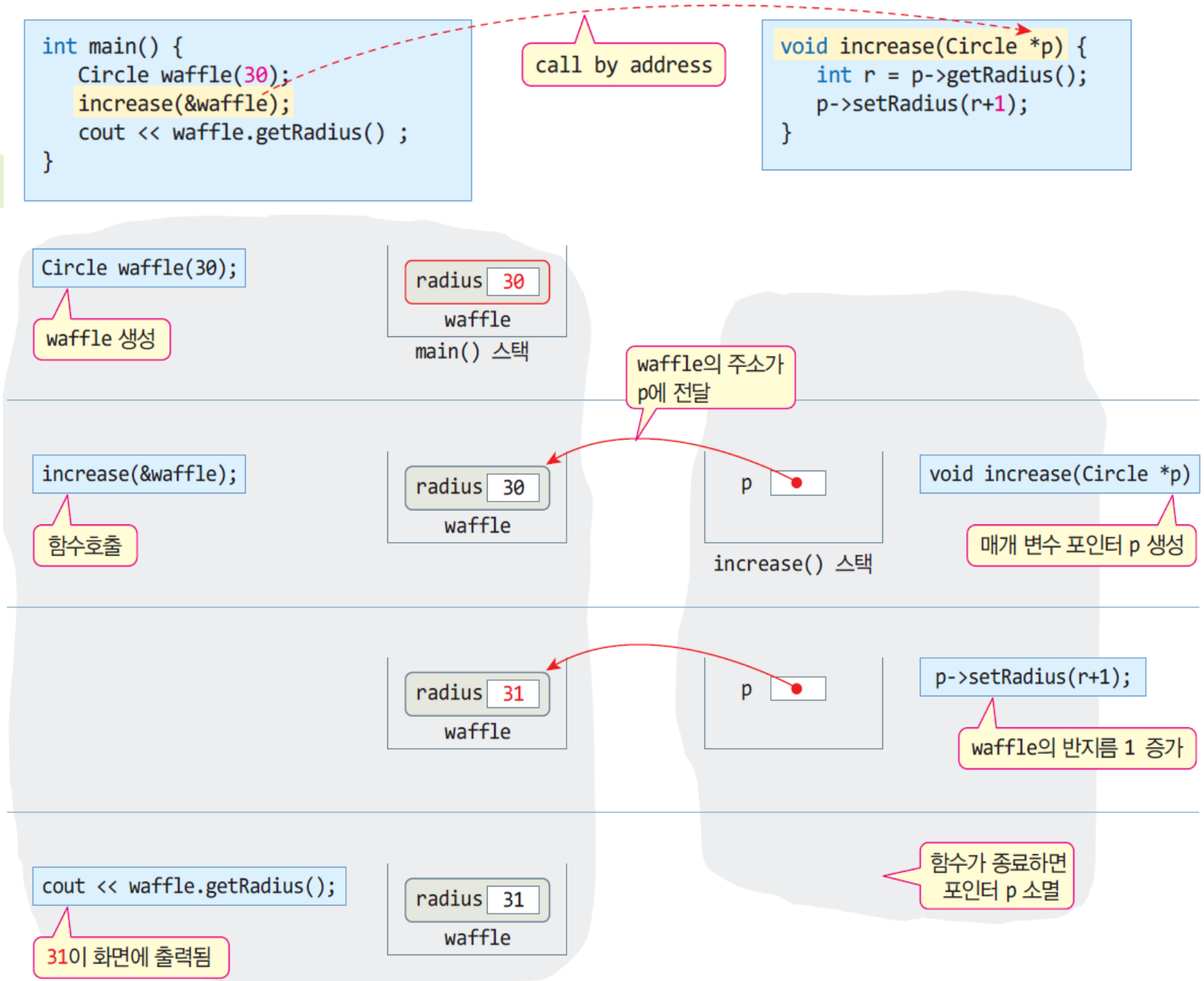


함수에 객체 전달 – '주소에 의한 호출'로

- 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

```
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
```

참조란?

메뚜기는 유재석의 별명

참조(reference)란 가리킨다는 뜻으로,
이미 존재하는 객체나 변수에 대한 별명

참조 활용

- 참조 변수
- 참조에 의한 호출
- 참조 리턴

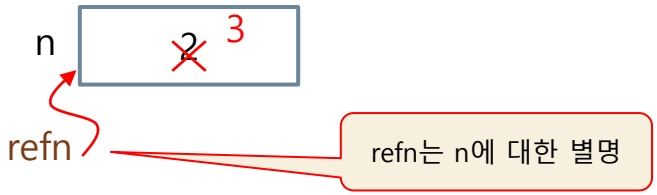
참조 변수

- 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
```

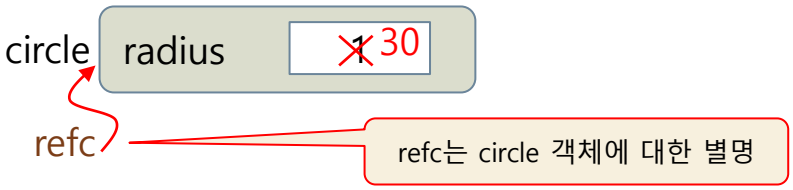
참조 변수 선언 및 사용 사례


```
int n = 2;
int &refn =n;

refn = 3;
```




```
Circle circle;
Circle &refc = circle;

refc.setRadius(30);
```

refc->setRadius(30);으로 하면 안 됨



예제 5–3 기본 타입 변수에 대한 참조

참조 변수 refn 선언

참조에 대한
포인터 변수 선언

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

int main() {
   cout << "i" << '₩t' << "n" << '₩t' << "refn" << endl;
   int i = 1;
   int n = 2;
   int &refn = n; // 참조 변수 refn 선언. refn은 n에 대한 별명
   n = 4;
   refn++; // refn=5, n=5
   cout << i << '₩t' << n << '₩t' << refn << endl;

   refn = i; // refn=1, n=1
   refn++; // refn=2, n=2
   cout << i << '₩t' << n << '₩t' << refn << endl;

   int *p = &refn; // p는 n의 주소를 가짐
   *p = 20; // refn=20, n=20
   cout << i << '₩t' << n << '₩t' << refn << endl;
}
```

```
i       n       refn
1       5       5
1       2       2
1       20      20
```

예제 5–4 객체에 대한 참조


```
#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; }
};

int main() {
   Circle circle;
   Circle &refc = circle;
   refc.setRadius(10);
   cout << refc.getArea() << " " << circle.getArea();
}
```

circle 객체에 대한 참조 변수 refc 선언

```
314 314
```

참조에 의한 호출

- 참조를 가장 많이 활용하는 사례
- call by reference라고 부름
- 함수 형식
  - 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

참조에 의한 호출 사례

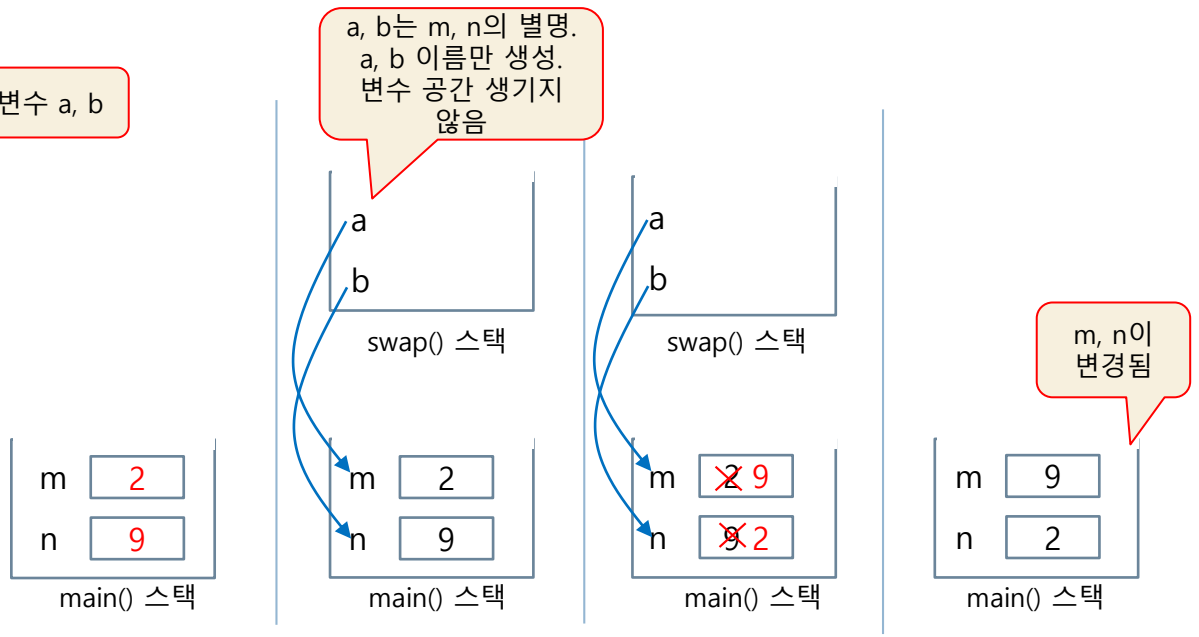

```
#include <iostream>
using namespace std;

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;
}
```

참조 매개 변수 a, b

참조 매개 변수를 보통 변수처럼 사용

함수가 호출되면 m, n에 대한 참조 변수 a, b가 생긴다.

9 2



참조 매개변수가 필요한 사례

- 다음 코드에 어떤 문제가 있을까?
  - average() 함수의 작동
    - 계산에 오류가 있으면 0 리턴, 아니면 평균 리턴
  - 만일 average()가 리턴한 값이 0이라면?
    - 평균이 0인 거야? 아니면 오류가 발생한 거야?

```
int average(int a[], int size) {
    if(size <= 0) return 0;
    int sum = 0;
    for(int i=0; i<size; i++) sum += a[i];
    return sum/size;
}
```

예제 5-5에서 해결

호출

호출

```
int x[ ]={1,2,3,4};
int avg = average(x, 4);

// avg는 2
```

흠, 평균이 2군.
알았어!

```
int x[ ]={1,2,3,4};
int avg = average(x, -1);

// avg는 0
```

평균이 0인 거야,
아니면 오류가 난 거야?

예제 5-5 참조 매개 변수로 평균 리턴하기

참조 매개 변수를 통해 평균을 리턴하고 리턴문을 통해서는 함수의 성공 여부를 리턴하도록 average() 함수를 작성하라

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

bool average(int a[], int size, int& avg) {
   if(size <= 0)
      return false;
   int sum = 0;
   for(int i=0; i<size; i++)
      sum += a[i];
   avg = sum/size;
   return true;
}

int main() {
   int x[] = {0,1,2,3,4,5};
   int avg;
   if(average(x, 6, avg)) cout << "평균은 " << avg << endl;
   else cout << "매개 변수 오류" << endl;

   if(average(x, -2, avg)) cout << "평균은 " << avg << endl;
   else cout << "매개 변수 오류 " << endl;
}
```

참조 매개 변수 avg에 평균 값 전달

avg에 평균이 넘어오고, average()는 true 리턴

avg의 값은 의미없고, average()는 false 리턴

```
평균은 2
매개 변수 오류
```

예제 5–6 참조에 의한 호출로 Circle 객체에 참조 전달

참조 매개 변수 c

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

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

참조에 의한 호출

waffle 객체 생성

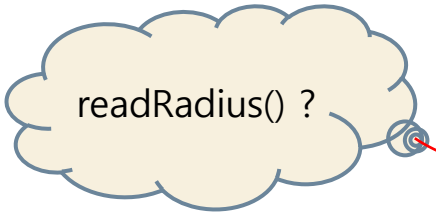
```
생성자 실행 radius = 30
31
소멸자 실행 radius = 31
```

waffle 객체 소멸

```
#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;
}
```

예제 5–7(실습) 참조 매개 변수를 가진 함수 만들기 연습

키보드로부터 반지름 값을 읽어 Circle 객체에 반지름을 설정하는 readRadius() 함수를 작성하라.




```
#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; }
};

int main() {
   Circle donut;
   readRadius(donut);
   cout << "donut의 면적 = " <<donut.getArea() << endl;
}
```

```
정수 값으로 반지름을 입력하세요>>3
donut의 면적 = 28.26
```

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()

예제 5-8 간단한 참조 리턴 사례


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

char& find(char s[], int index) {
    return s[index]; // 참조 리턴
}

int main() {
    char name[] = "Mike";
    cout << name << endl;

    find(name, 0) = 'S'; // name[0]='S'로 변경
    cout << name << endl;

    char& ref = find(name, 2);
    ref = 't'; // name = "Site"
    cout << name << endl;
}
```

s[index] 공간의 참조 리턴

find()가 리턴한 위치에 문자 'm' 저장

ref는 name[2] 참조

```
Mike
Sike
Site
```