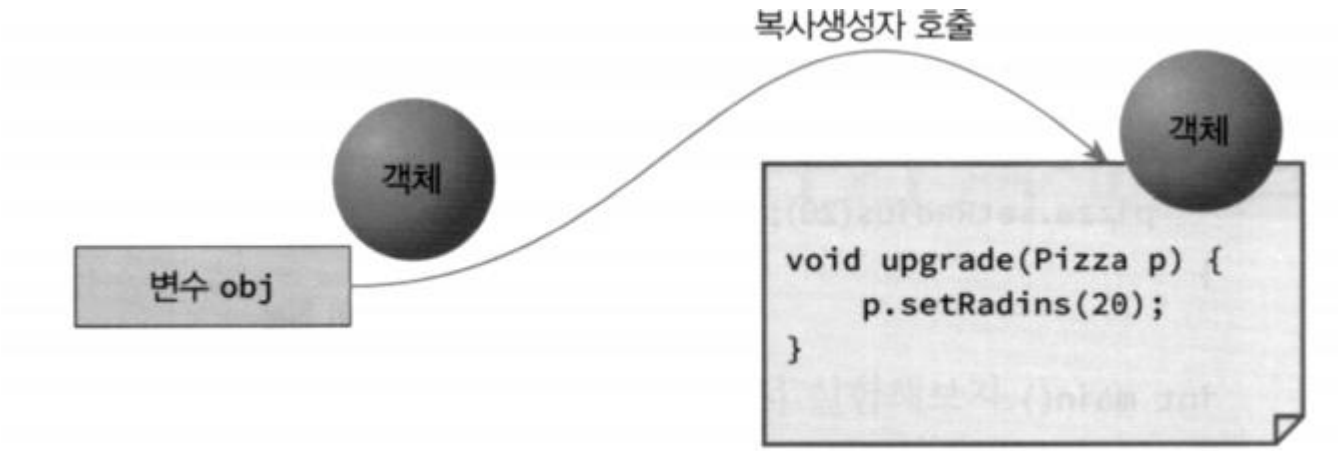
복사생성자와 정적멤버

함수로 객체 전달하기

■ 객체를 함수로 전달하기

○ 함수 호출시 객체를 매개변수로 전달하면 객체의 복사가 일어남



함수로 객체 전달하기

■ 객체의 주소를 함수로 전달하기

○ 함수 호출시 객체의 주소를 전달하면 객체의 복사가 일어나지 않음

```
void upgrade(Pizza *p) {
    p->setRadius(20);
}

int main()
{
    Pizza obj(10);
    upgrade(&obj);

    obj.print();
    return 0;
}
```

함수로 객체 전달하기

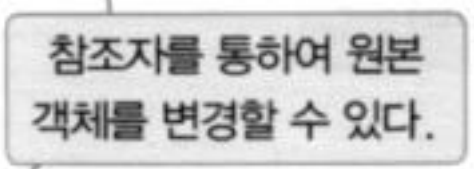
■ 참조자 매개변수 사용하기

○ 효과는 포인터를 넘기는 것과 동일하나 포인터 보다 가독성이 좋아짐

```
void upgrade(Pizza& pizza) {
    pizza.setRadius(20);
}

int main()
{
    Pizza obj(10);
    upgrade(obj);

    obj.print();
    return 0;
}
```



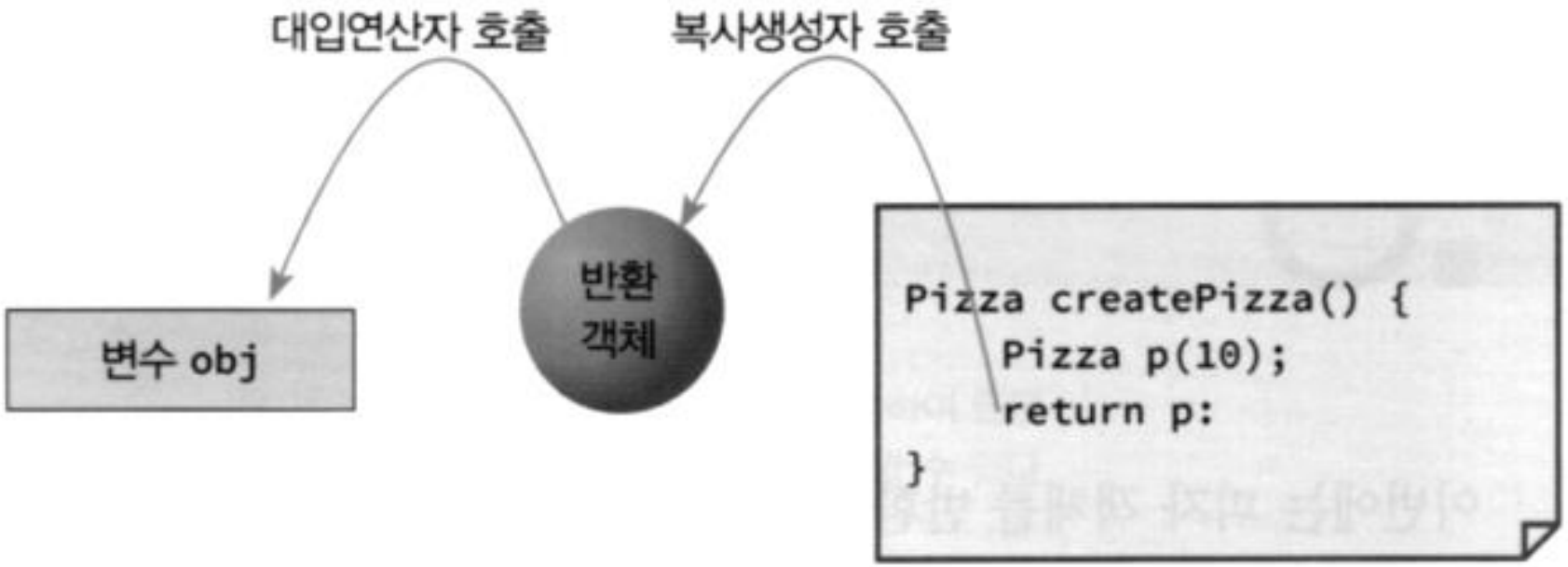
함수가 객체를 반환하기

■ 함수가 객체를 반환

○ 객체를 함수의 매개변수로 전달할 때와 같이 객체의 복사가 일어남

○

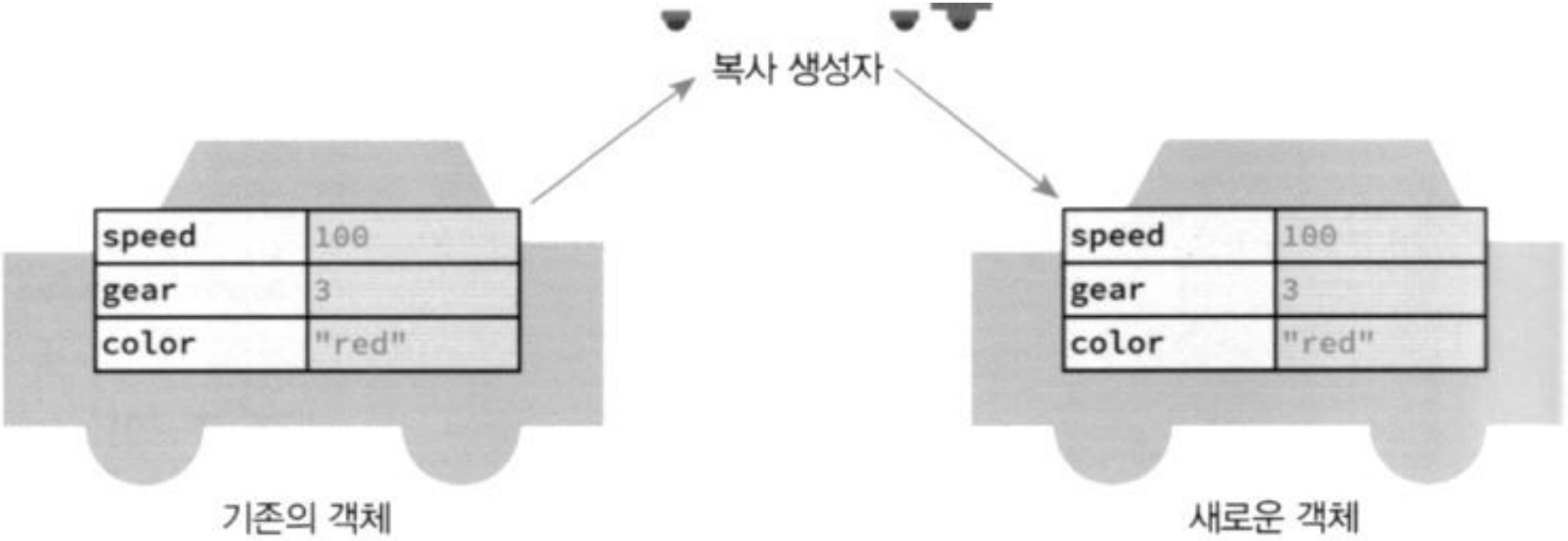
```
Pizza pizza = createPizza();
```



복사 생성자

■ 복사 생성자가 사용되는 시점

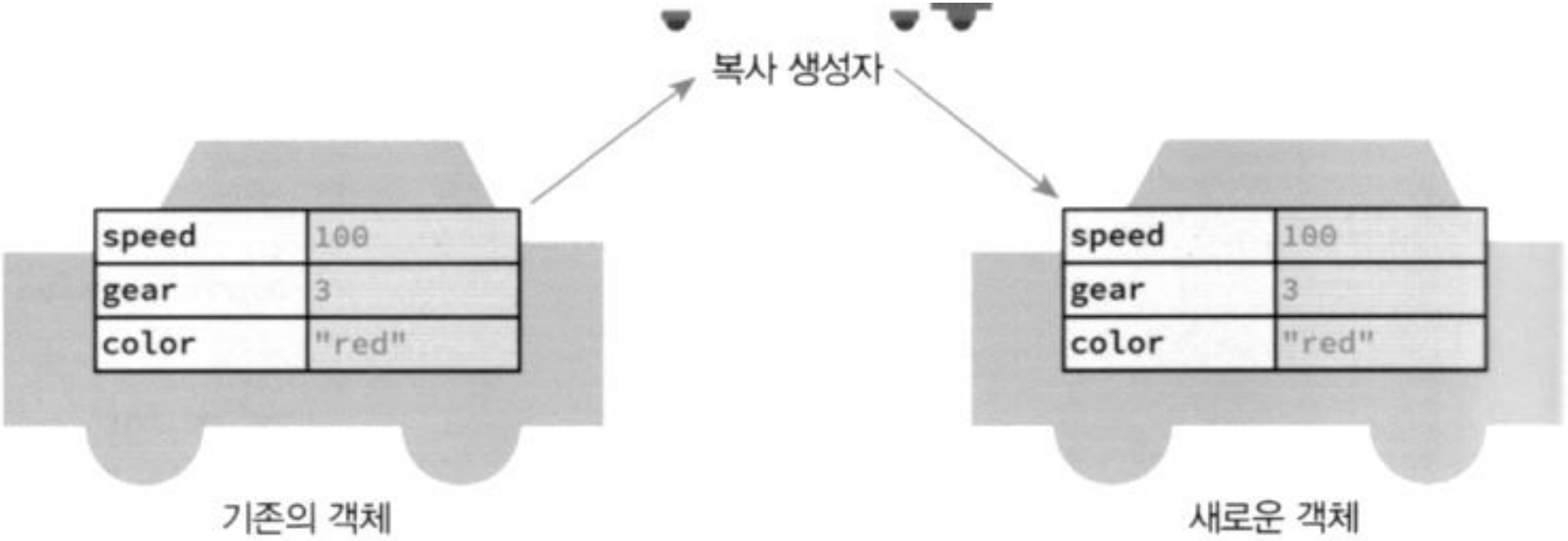
- ○ 함수의 매개변수로 객체를 전달할 때
- ○ 객체를 대입할 때



복사 생성자

■ 복사 생성자가 사용되는 시점

- ○ 함수의 매개변수로 객체를 전달할 때
- ○ 객체를 대입할 때



복사 생성자

■ 복사 생성자

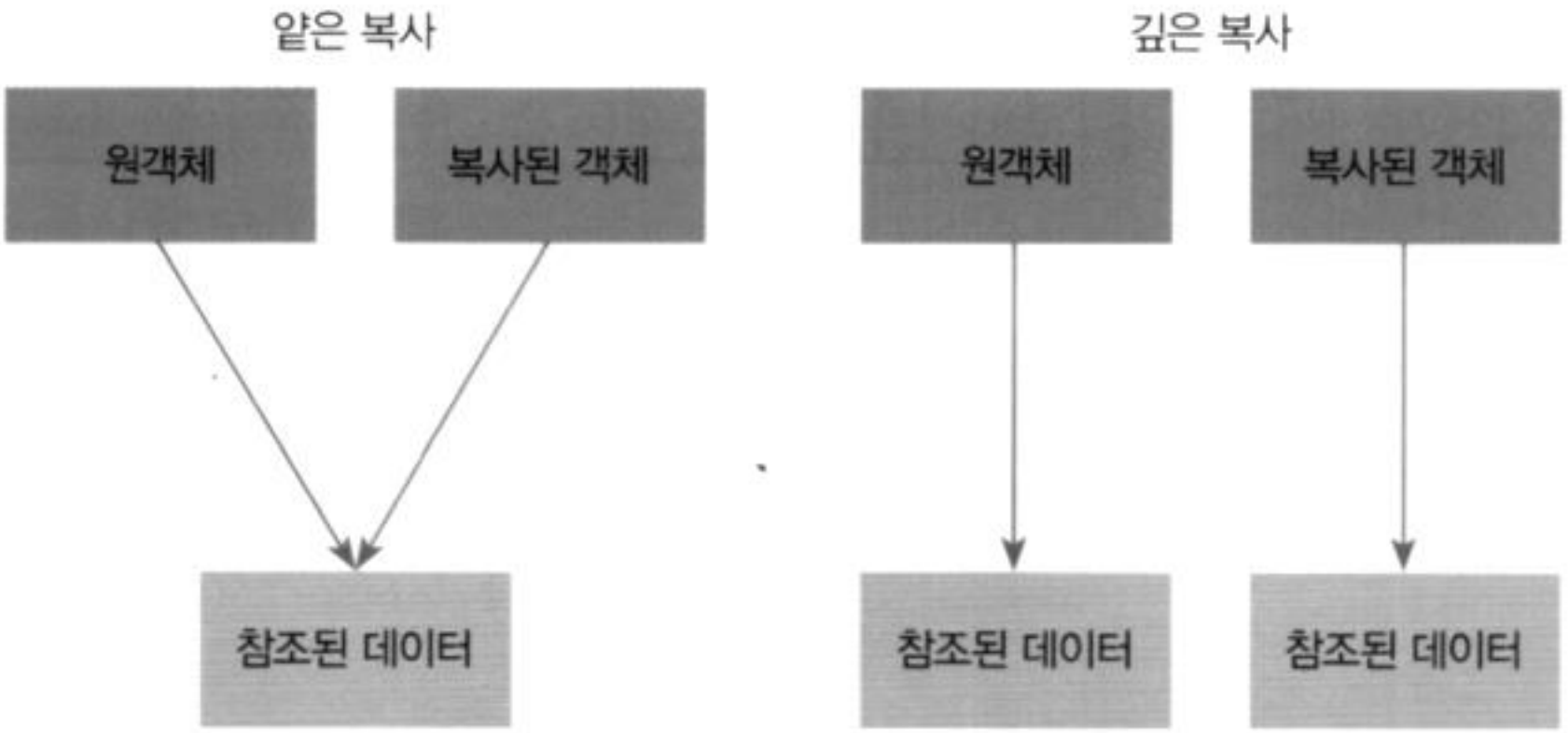
```
MyClass(const MyClass & other) {
  // 객체 초기화
}
```

- MyClass(MyClass other)로 하면 무한 루프 발생 하므로 주의!
- 복사 생성자를 정의하지 않으면 자동으로 추가
  - 멤버 변수에 대한 얕은 복사 진행
- 깊은 복사가 필요한 경우 복사 생성자 정의 필요

복사 생성자

■ 복사 생성자

○ 얕은 복사와 깊은 복사



복사 생성자

■ chapter07/ex01_copy.cpp] 복사 생성자 - 디폴트 복사 생성자

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

class Person {
public :
  int age;
  Person(int a): age(a) {}
};
```

복사 생성자

■ chapter07/ex01_copy.cpp] 복사 생성자 - 디폴트 복사 생성자

```
int main(int argc, char const *argv[]) {
  Person kim{21};
  Person clone{kim};  // 복사 생성자 호출

  cout << "kim의 나이: " << kim.age << " clone의 나이: " << clone.age << endl;
  kim.age = 23;
  cout << "kim의 나이: " << kim.age << " clone의 나이: " << clone.age << endl;

  return 0;
}
```

```
kim의 나이: 21 clone의 나이: 21
kim의 나이: 23 clone의 나이: 21
```

복사 생성자

■ chapter07/ex02_copy.cpp] 복사 생성자 – 얕은 복사의 문제점

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

class MyArray {
public :
  int size;
  int *data;
  MyArray(int size) {
    this->size = size;
    data = new int[size];
  }
  ~MyArray() {
    if(data != NULL) {
      delete []data;
    }
  }
};
```

복사 생성자

■ chapter07/ex02_copy.cpp] 복사 생성자 – 얕은 복사의 문제점

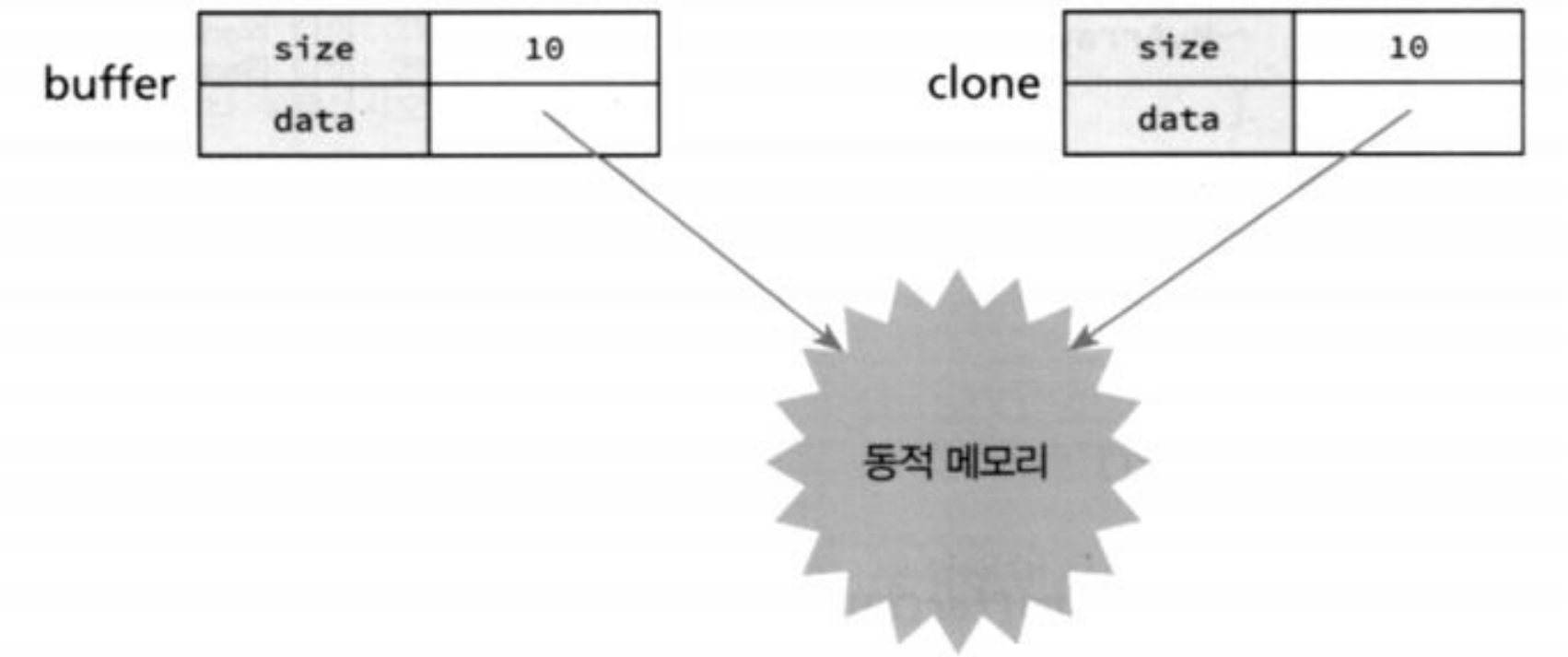
```
int main(int argc, char const *argv[])
{
  MyArray buffer(10);
  buffer.data[0] = 1;
  {
    MyArray clone = buffer; // 복사 생성자 호출
  }  // clone 삭제

  buffer.data[0] = 2;
  return 0;
}
```

복사 생성자

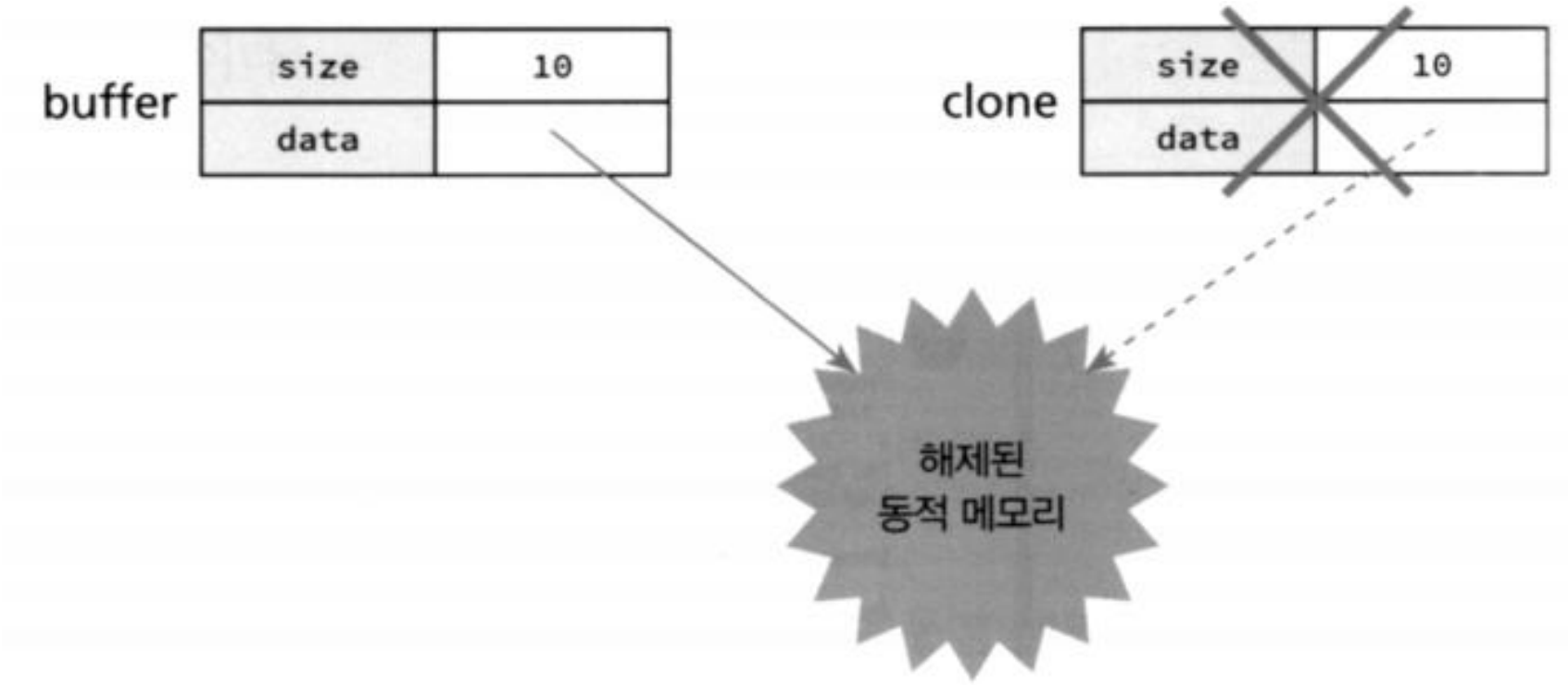
■ 복사 생성자 – 얕은 복사의 문제점

○ `MyArray clone = buffer; // 복사 생성자 호출`



복사 생성자

■ 복사 생성자 – 얕은 복사의 문제점



복사 생성자

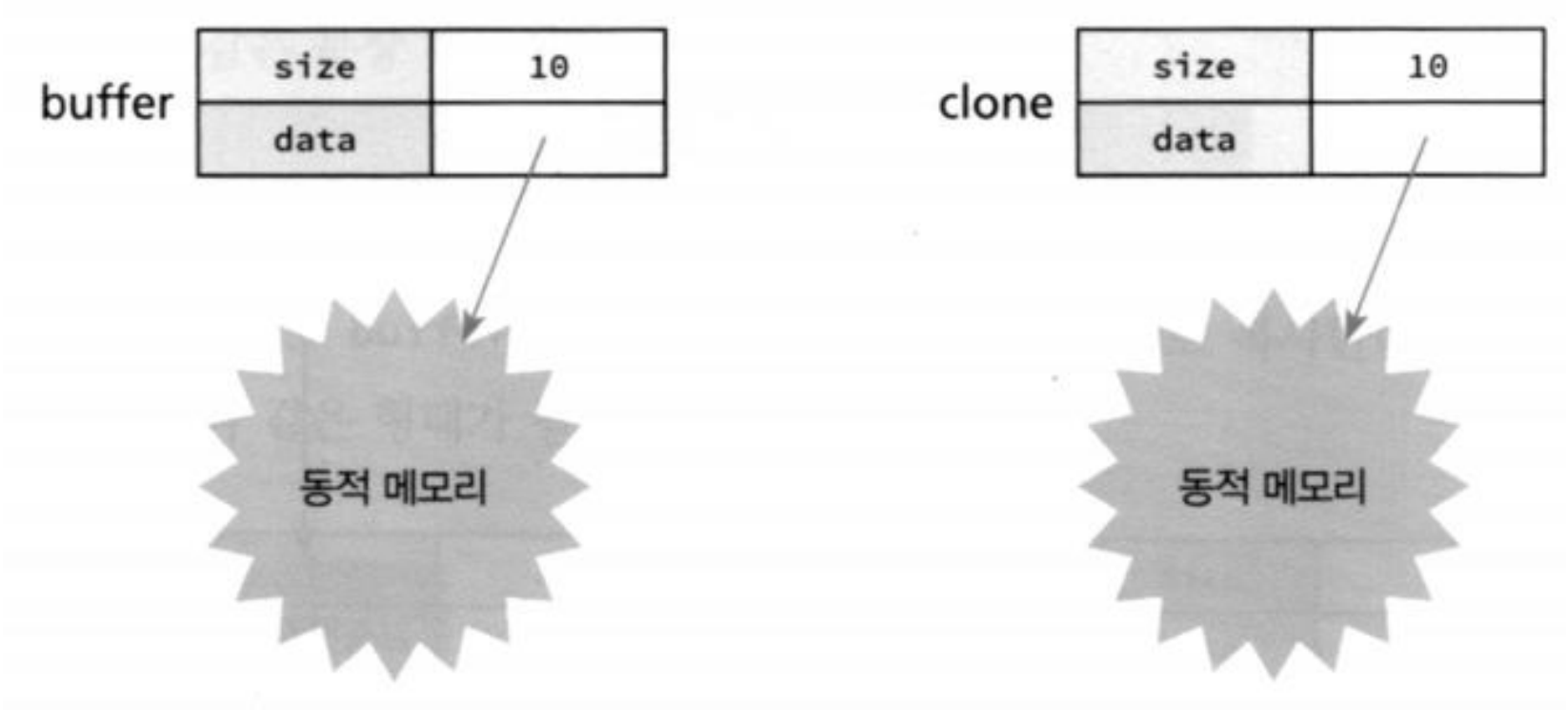
■ chapter07/ex03_deepcopy.cpp] 복사 생성자 – 깊은 복사

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

class MyArray {
  :
  MyArray(const MyArray& other) {
      size = other.size;
      data = new int[other.size];
      for(int i=0; i<size; i++) {
        data[i] = other.data[i];  // 복사 생성자 호출
      }
  }
  :
};
```

복사 생성자

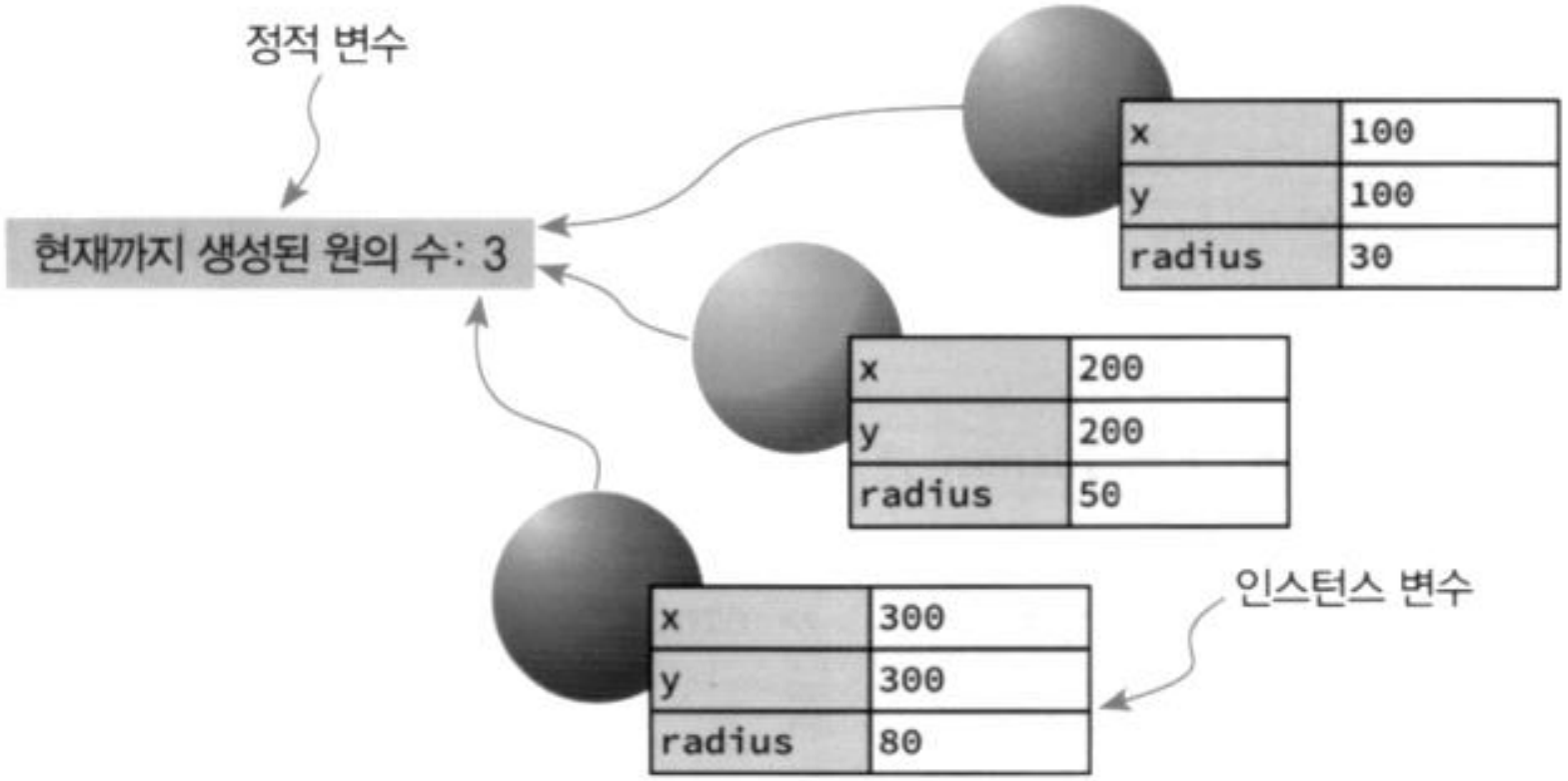
■ 복사 생성자 – 깊은 복사



정적 변수

■ 정적(static) 변수

- ○ 멤버 변수에 static 예약어 지정
- ○ 인스턴스와 무관하게 항상 1개만 존재
- ○ 모든 인스턴스가 공유
- ○



정적 변수

■ chapter07/ex04_static.cpp] 정적(static) 변수

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

class Circle {
  int x, y;
  int radius;
public:
  static int count; // 정적 변수
  Circle()  : x(0), y(0), radius(0) {
    count++;
  }
  Circle(int x, int y, int r)  : x(x), y(y), radius(r) {
    count++;
  }
  ~Circle() {
    count--;
  }
};
```

정적 변수

■ chapter07/ex04_static.cpp] 정적(static) 변수

```
int Circle::count = 0;  // 정적 변수 초기화

int main(int argc, char const *argv[]) {
  Circle c1;
  cout << "원의 개수 : " << Circle::count << endl;

  {
    Circle c2;
    cout << "원의 개수 : " << Circle::count << endl;
  }
  cout << "원의 개수 : " << Circle::count << endl;
  return 0;
}
```

```
원의 개수 : 1
원의 개수 : 2
원의 개수 : 1
```

정적 변수

■ 정적 함수와 정적 상수

```
class Circle {
  int x, y;
  int radius;
  static int count; // 정적 변수

public:
  :
  const static int MAX_COUNT = 100;  // 정적 상수

  static int getCount() {
    return count;
  }
};
```

```
cout << "원의 개수 : " << Circle::getCount() << endl;
```