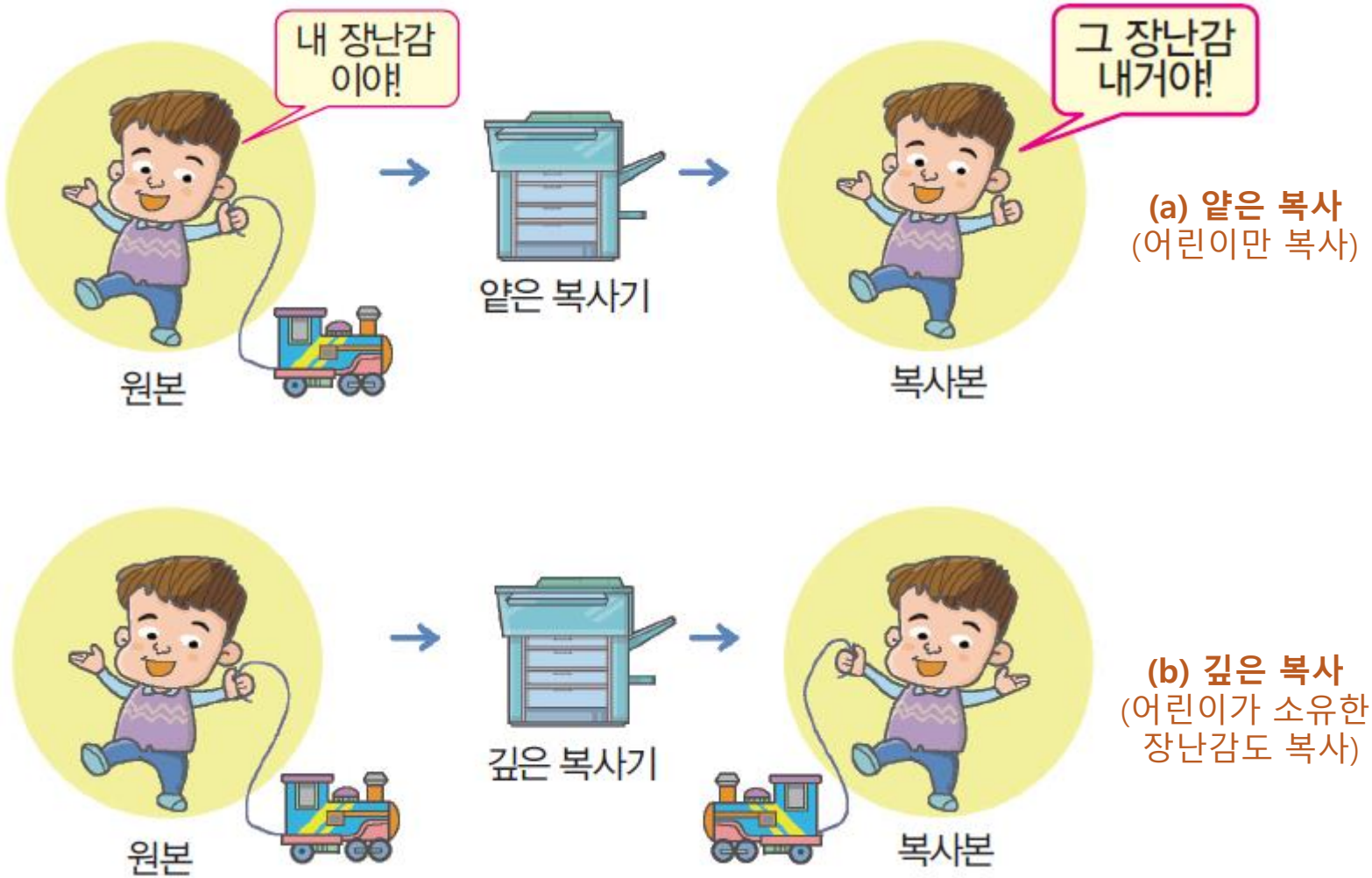
얕은 복사와 깊은 복사

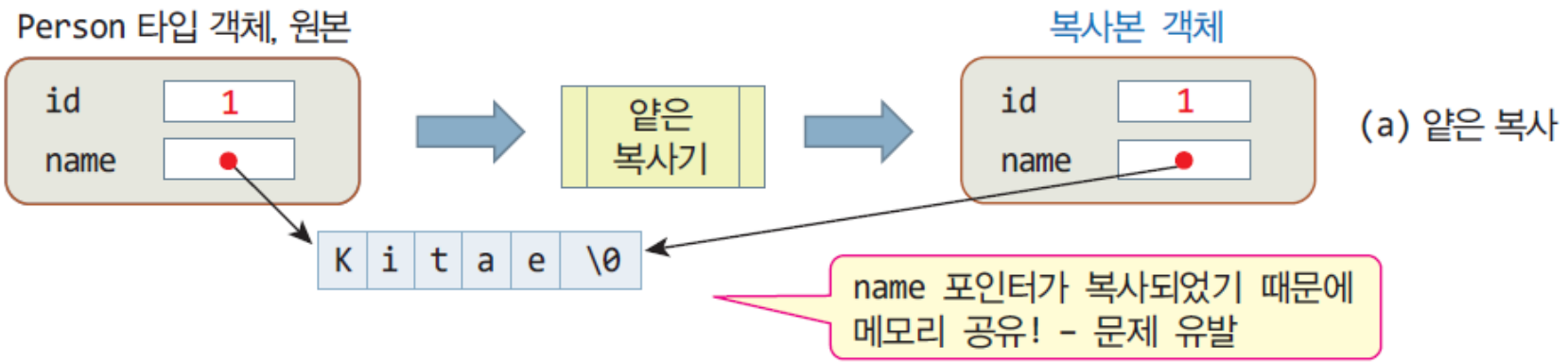

내 장난감 이야!
얕은 복사기
그 장난감 내거야!
원본
복사본
(a) 얕은 복사
(어린이만 복사)
깊은 복사기
원본
복사본
(b) 깊은 복사
(어린이가 소유한 장난감도 복사)

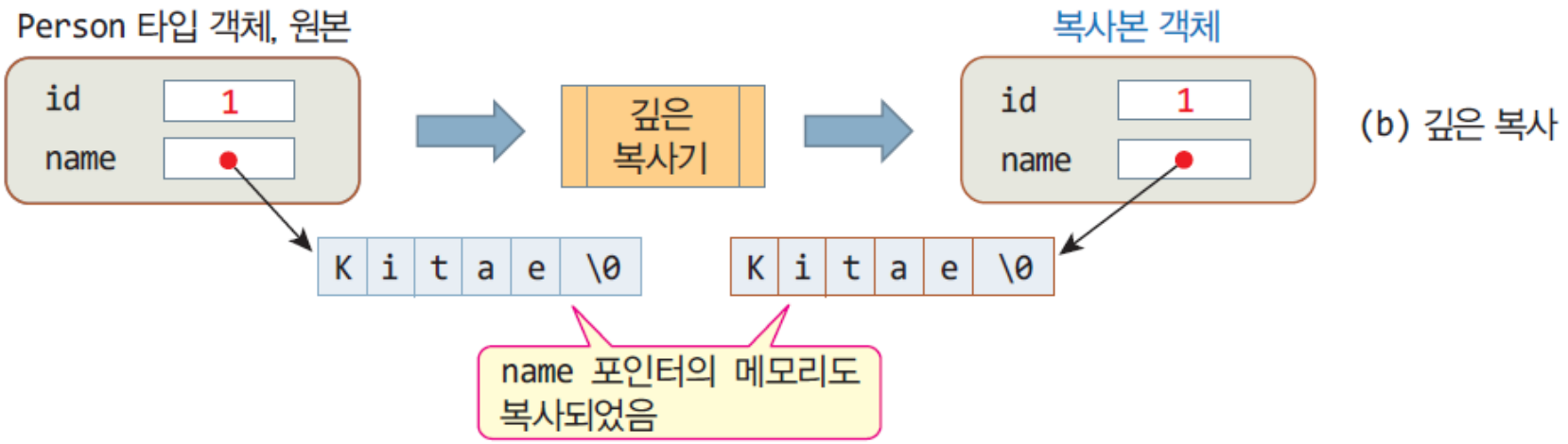
C++에서 얕은 복사와 깊은 복사

- 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;
   ..................
};
```





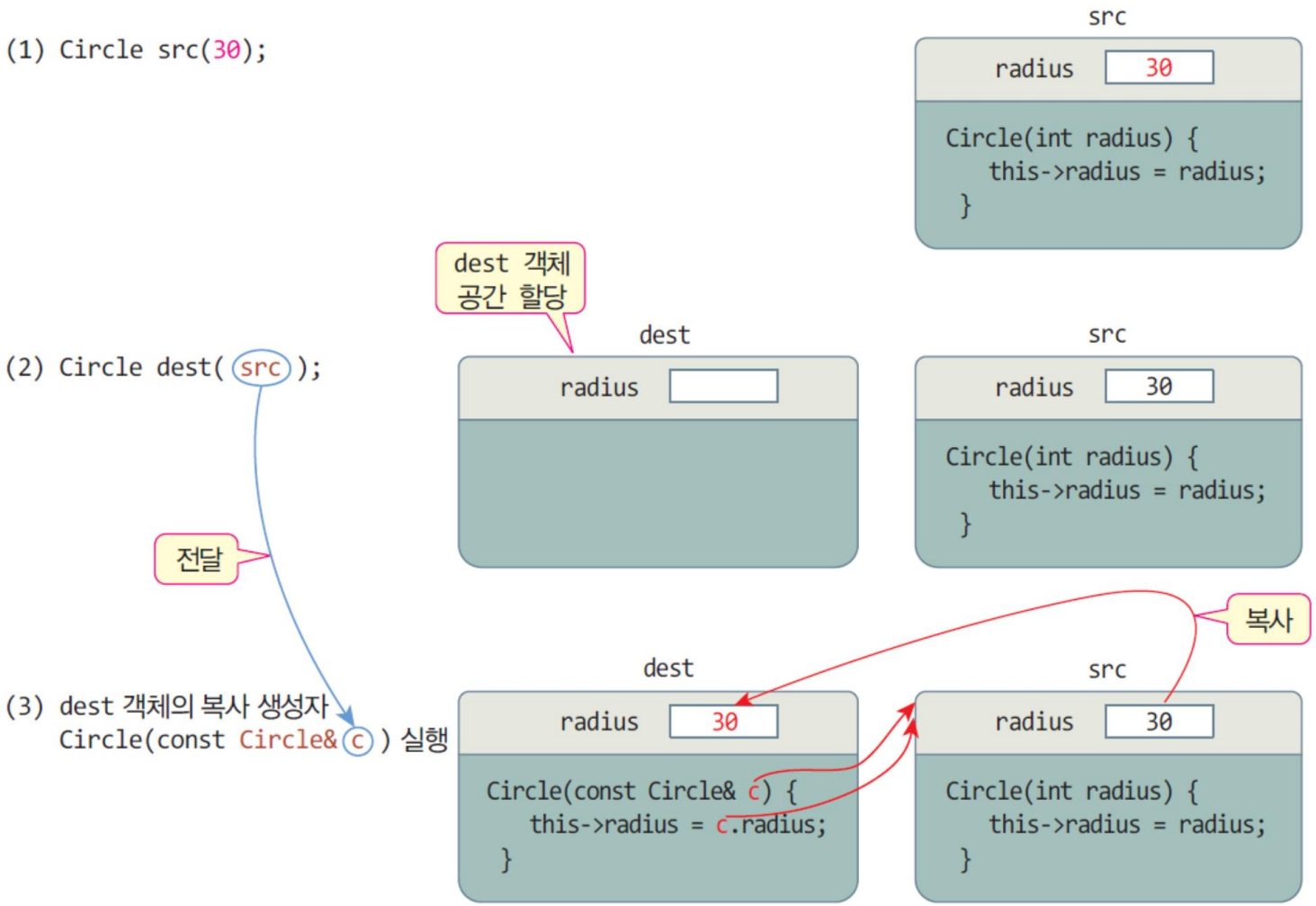
복사 생성자

- 복사 생성자(copy constructor)란?
  - 객체의 복사 생성시 호출되는 특별한 생성자
- 특징
  - 한 클래스에 오직 한 개만 선언 가능
  - 복사 생성자는 보통 생성자와 클래스 내에 중복 선언 가능
  - 모양
    - 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

복사 생성 과정


(1) Circle src(30);
src
radius 30
Circle(int radius) {
this->radius = radius;
}
dest 객체
공간 할당
(2) Circle dest(src);
dest
radius
src
radius 30
Circle(int radius) {
this->radius = radius;
}
전달
복사
(3) dest 객체의 복사 생성자
Circle(const Circle& c) 실행
dest
radius 30
Circle(const Circle& c) {
this->radius = c.radius;
}
src
radius 30
Circle(int radius) {
this->radius = radius;
}

예제 5-9 Circle의 복사 생성자와 객체 복사


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

class Circle {
private:
    int radius;
public:
    Circle(const Circle& c); // 복사 생성자 선언
    Circle() { radius = 1; }
    Circle(int radius) { this->radius = radius; }
    double getArea() { return 3.14*radius*radius; }
};

Circle::Circle(const Circle& c) { // 복사 생성자 구현
    this->radius = c.radius;
    cout << "복사 생성자 실행 radius = " << radius << endl;
}

int main() {
    Circle src(30); // src 객체의  보통 생성자 호출
    Circle dest(src); // dest 객체의 복사 생성자 호출

    cout << "원본의 면적 = " << src.getArea() << endl;
    cout << "사본의 면적 = " << dest.getArea() << endl;
}
```

dest 객체가 생성될 때 Circle(const Circle& c)

```
복사 생성자 실행 radius = 30
원본의 면적 = 2826
사본의 면적 = 2826
```

디폴트 복사 생성자

- 복사 생성자가 선언되어 있지 않는 클래스
  - 컴파일러는 자동으로 디폴트 복사 생성자 삽입

```
class Circle {
    int radius;
pubic:
    Circle(int r);
    double getArea();
};
```

복사 생성자 없음

복사 생성자 없는데 컴파일 오류?

```
Circle dest(src); // 복사 생성. Circle(const Circle&) 호출
```

```
Circle::Circle(const Circle& c) {
    this->radius = c.radius;
    // 원본 객체 c의 각 멤버를 사본(this)에 복사한다.
}
```

디폴트 복사 생성자

디폴트 복사 생성자 사례


```
class Book {
   double price;    // 가격
   int pages;       // 페이지수
   char *title;     // 제목
   char *author;    // 저자이름
public:
   Book(double pr, int pa, char* t, char* a);
   ~Book()
};
```

복사 생성자가 없는 Book 클래스

컴파일러가 삽입하는
디폴트 복사 생성자

```
Book(const Book& book) {
   this->price = book.price;
   this->pages = book.pages;
   this->title = book.title;
   this->author = book.author;
}
```

예제 5–10 얕은 복사 생성자를 사용하여 프로그램이 비정상 종료되는 경우


```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstring>
using namespace std;

class Person { // Person 클래스 선언
    char* name;
    int id;
public:
    Person(int id, const char* name); // 생성자
    ~Person(); // 소멸자
    void changeName(const char *name);
    void show() { cout << id << ',' << name << endl; }
};

Person::Person(int id, const char* name) { // 생성자
    this->id = id;
    int len = strlen(name); // name의 문자 개수
    this->name = new char [len+1]; // name 문자열 공간 할당
    strcpy(this->name, name); // name에 문자열 복사
}

Person::~Person() {// 소멸자
    if(name) // 만일 name에 동적 할당된 배열이 있으면
        delete [] name; // 동적 할당 메모리 소멸
}

void Person::changeName(const char* name) { // 이름 변경
    if(strlen(name) > strlen(this->name))
        return;
    strcpy(this->name, name);
}
```

컴파일러에 의해
디폴트 복사 생성자 삽입

```
Person::Person(const Person& p) {
    this->id = p.id;
    this->name = p.name;
}
```

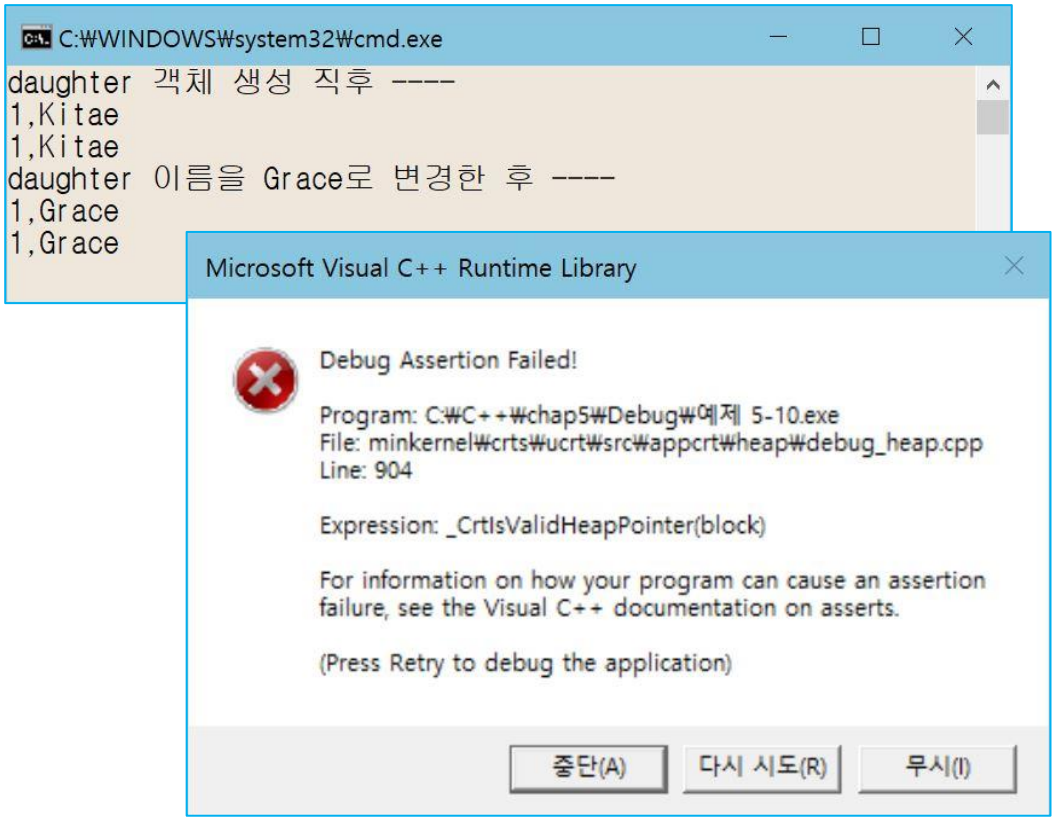
name 메모리 반환

```
int main() {
    Person father(1, "Kitae");          // (1) father 객체 생성
    Person daughter(father);            // (2) daughter 객체 복사 생성. 복사생성자호출

    cout << "daughter 객체 생성 직후 ----" << endl;
    father.show();                      // (3) father 객체 출력
    daughter.show();                    // (3) daughter 객체 출력

    daughter.changeName("Grace"); // (4) daughter의 이름을 "Grace"로 변경
    cout << "daughter 이름을 Grace로 변경한 후 ----" << endl;
    father.show();                      // (5) father 객체 출력
    daughter.show();                    // (5) daughter 객체 출력

    return 0;                           // (6), (7) daughter, father 객체 소멸
}
```

컴파일러가 삽입한
디폴트 복사 생성자 호출

daughter, father 순으로 소멸.
father가 소멸할 때, 프로그램
비정상 종료됨

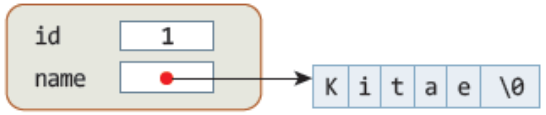
예제 5-10의 실행 결과


C:₩WINDOWS₩system32₩cmd.exe
daughter 객체 생성 직후 ----
1,Kitae
1,Kitae
daughter 이름을 Grace로 변경한 후 ----
1,Grace
1,Grace
Microsoft Visual C++ Runtime Library
Debug Assertion Failed!
Program: C:₩C++₩chap5₩Debug₩예제 5-10.exe
File: minkernel₩crts₩ucrt₩src₩appcrt₩heap₩debug_heap.cpp
Line: 904
Expression: _CrtIsValidHeapPointer(block)
For information on how your program can cause an assertion failure, see the Visual C++ documentation on asserts.
(Press Retry to debug the application)
중단(A)
다시 시도(R)
무시(I)

예제 5-10의 실행 과정

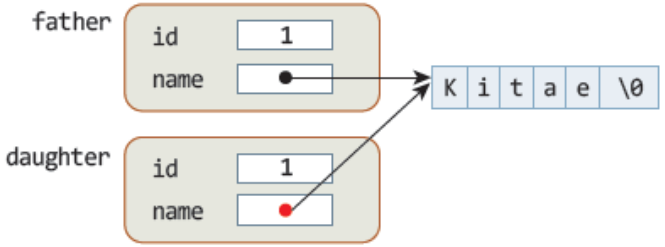
(1) `Person father(1, "Kitae");`

father 객체 생성



(2) `Person daughter(father);`

father를 복사한 daughter 객체 생성



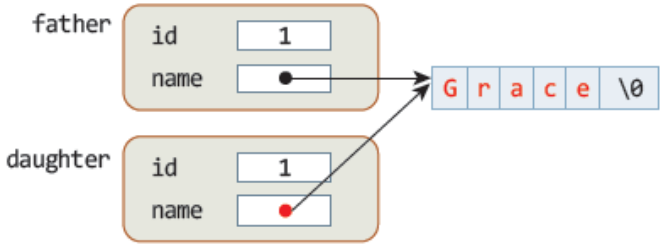
(3)
```
father.show();
daughter.show();
```

실행 결과

```
1,Kitae
1,Kitae
```

(4) `daughter.changeName("Grace");`

daughter의 이름 변경

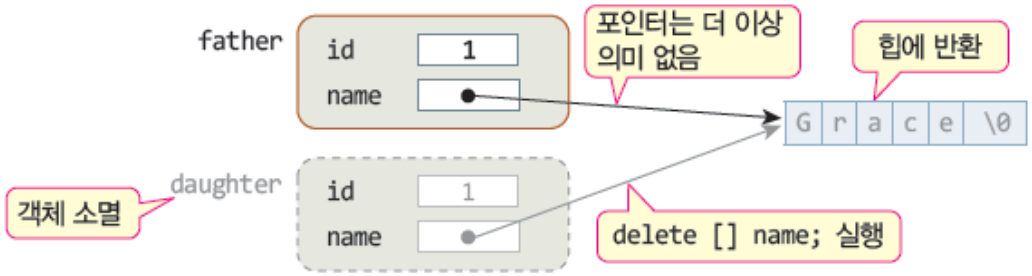


(5)
```
father.show();
daughter.show();
```

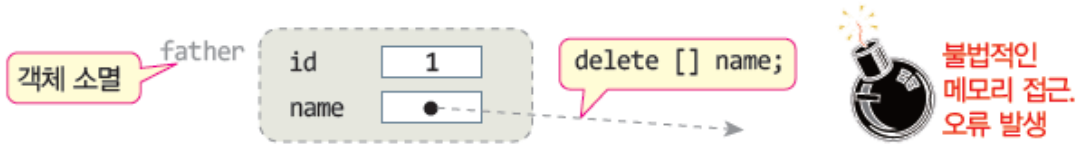
실행 결과

```
1,Grace
1,Grace
```

(6) daughter 객체 소멸



(7) father 객체 소멸



예제 5–11 깊은 복사 생성자를 가진 정상적인 Person 클래스

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstring>
using namespace std;

class Person { // Person 클래스 선언
    char* name;
    int id;
public:
    Person(int id, const char* name); // 생성자
    Person(const Person& person); // 복사 생성자
    ~Person(); // 소멸자
    void changeName(const char *name);
    void show() { cout << id << ',' << name << endl; }
};

Person::Person(int id, const char* name) { // 생성자
    this->id = id;
    int len = strlen(name); // name의 문자 개수
    this->name = new char [len+1]; // name 문자열 공간 할당
    strcpy(this->name, name); // name에 문자열 복사
}

Person::Person(const Person& person) { // 복사 생성자
    this->id = person.id; // id 값 복사
    int len = strlen(person.name);// name의 문자 개수
    this->name = new char [len+1]; // name을 위한 공간 할당
    strcpy(this->name, person.name); // name의 문자열 복사
    cout << "복사 생성자 실행. 원본 객체의 이름 " << this->name << endl;
}

Person::~Person() {// 소멸자
    if(name) // 만일 name에 동적 할당된 배열이 있으면
        delete [] name; // 동적 할당 메모리 소멸
}

void Person::changeName(const char* name) { // 이름 변경
    if(strlen(name) > strlen(this->name))
        return; // 현재 name에 할당된 메모리보다 긴 이름으로 바꿀 수 없다.
    strcpy(this->name, name);
}
```

id 복사

name 복사

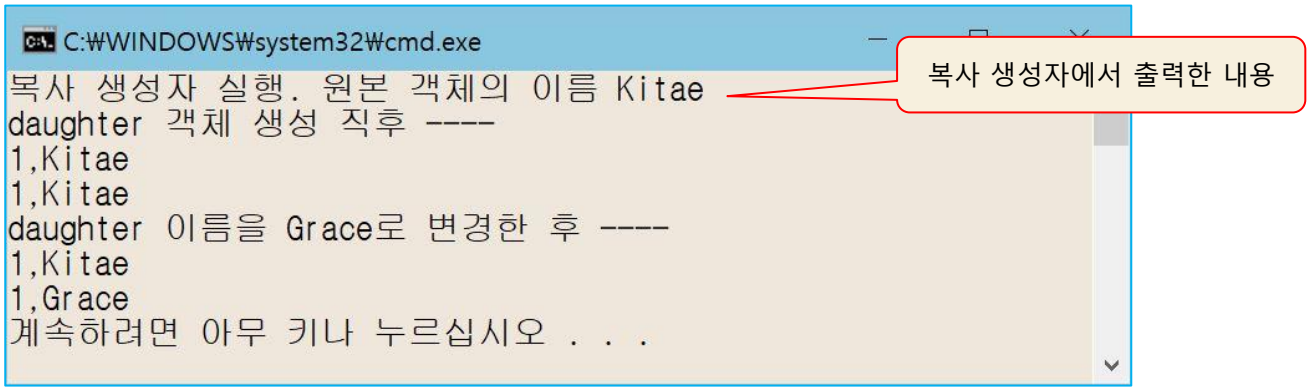
name 메모리 반환

```
int main() {
    Person father(1, "Kitae");          // (1) father 객체 생성
    Person daughter(father);            // (2) daughter 객체 복사 생성. 복사생성자호출

    cout << "daughter 객체 생성 직후 ----" << endl;
    father.show();                      // (3) father 객체 출력
    daughter.show();                    // (3) daughter 객체 출력

    daughter.changeName("Grace"); // (4) daughter의 이름을 "Grace"로 변경
    cout << "daughter 이름을 Grace로 변경한 후 ----" << endl;
    father.show();                      // (5) father 객체 출력
    daughter.show();                    // (5) daughter 객체 출력

    return 0;                           // (6), (7) daughter, father 객체 소멸
}
```

Person에 작성된 깊은 복사 생성자 호출

daughter, father 순으로 소멸

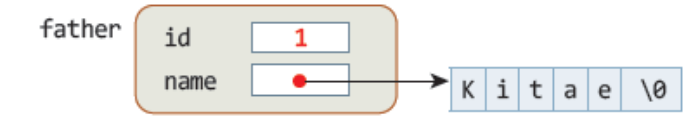
예제 5-11의 실행 결과


C:₩WINDOWS₩system32₩cmd.exe
복사 생성자 실행. 원본 객체의 이름 Kitae
daughter 객체 생성 직후 ----
1,Kitae
1,Kitae
daughter 이름을 Grace로 변경한 후 ----
1,Kitae
1,Grace
계속하려면 아무 키나 누르십시오 . . .
복사 생성자에서 출력한 내용

예제 5-11의 실행 과정

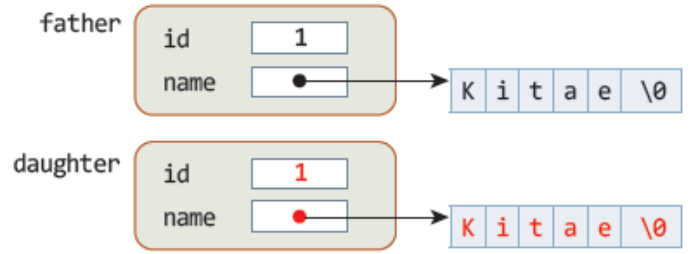
(1) Person father(1, "Kitae");

father 객체 생성



(2) Person daughter(father);

father를 복사한 daughter 객체 생성



실행 결과

```
복사 생성자 실행 Kitae
```

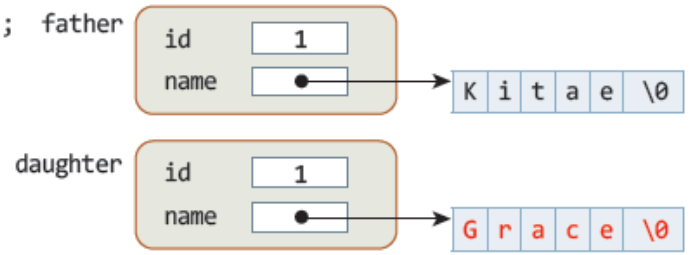
(3) father.show();
daughter.show();

실행 결과

```
1,Kitae
1,Kitae
```

(4) daughter.changeName("Grace");

daughter의 이름 변경

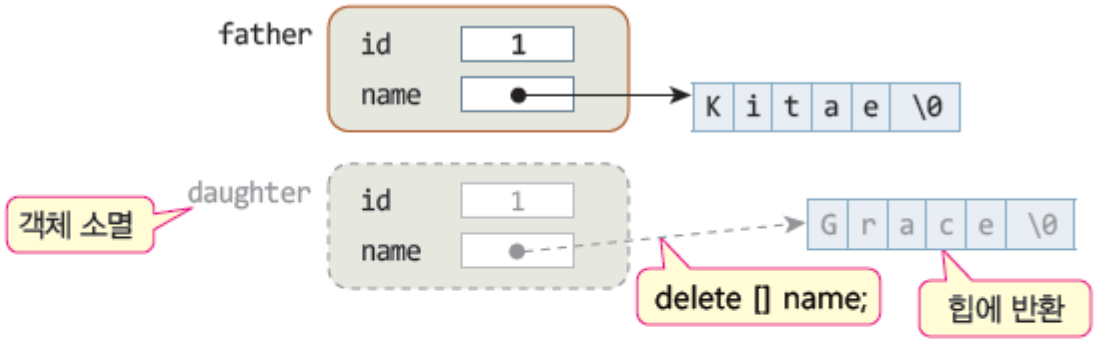


(5) father.show();
daughter.show();

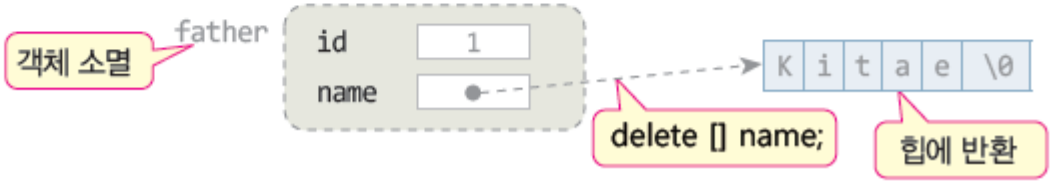
실행 결과

```
1,Kitae
1,Grace
```

(6) daughter 객체 소멸



(7) father 객체 소멸



예제 5-12 묵시적 복사 생성에 의해 복사 생성자가 자동 호출되는 경우


```
void f(Person person) {
    person.changeName("dummy");
}

Person g() {
    Person mother(2, "Jane");
    return mother;
}

int main() {
    Person father(1, "Kitae");
    Person son = father;
    f(father);
    g();
}
```

2. '값에 의한 호출'로 객체가 전달될 때. person 객체의 복사 생성자 호출

3. 함수에서 객체를 리턴할 때.mother 객체의 복사본 생성. 복사본의 복사 생성자 호출

1. 객체로 초기화하여 객체가 생성될 때. son 객체의 복사 생성자 호출