

# 7주차note #4

## 정리노트&PairProgramming

2023. 10. 16(월)  
16조 이수영, 원준서

# 목차

- **슬라이드 03-17: 정리노트**

- 객체에 대한 포인터

- 객체 배열 생성 및 소멸

- 객체 배열 초기화

- 동적 메모리 할당 및 반환

- 배열의 동적 할당 및 반환

- 객체의 동적 생성 및 반환

- 객체 배열의 동적 생성 및 반환

- this 포인터

- **슬라이드 18-26: Pair Programming**

# 정리노트

- 강의 때 다룬 4장 '객체 포인터와 객체 배열, 객체 동적 생성'의 학습목표를 기반으로 핵심 개념을 논의하고 정리했습니다.

# 객체에 대한 포인터

- 원준서: C언어의 포인터와 동일하며, 객체의 주소 값을 가지는 변수입니다. 포인터로 멤버를 접근할 때는 객체포인터->멤버와 같이 사용합니다. 객체에 대한 포인터를 선언한 후, 포인터에 객체 주소를 저장하고, 멤버 함수를 호출할 수 있습니다.

# 객체에 대한 포인터

- 이수영: 포인터 개념을 막연하게 알고 있다고 생각했는데, 막상 설명하려고 하니 제대로 아는 게 아니라는 걸 깨닫게 되었습니다. 멤버(local) 함수 또는 멤버 변수에 접근할 때 객체+dot(.)+멤버 함수 이런 식으로 접근한다는 점이 기억에 남습니다. 예제 4-1에서 다뤘듯, \*p=20;은 p의 주소가 20인 게 아니라, p가 가리키는 곳에 값 20을 저장하는 것이라는 점을 여러 번 반복해주셔서 확실히 알 수 있었습니다. 그리고 손이 매운 사람을 구인한다는 내용이 기억에 남습니다. 안 보고도 코드가 바로 나올 수 있도록 반복하는 과정이 중요하다는 점을 생각해보았습니다.

# 객체 배열 생성 및 소멸

- 원준서: 객체 배열 선언은 기본 타입의 배열 선언과 형식이 동일합니다. 객체 배열 선언을 하기 위해서는 객체 배열을 위한 공간을 할당하고, 배열의 각 원소 객체마다 생성자를 실행합니다. 생성자는 매개 변수가 없는 생성자가 호출되는데, 매개 변수가 있는 생성자를 호출할 수는 없습니다. 매개 변수가 있는 생성자를 호출한다면 오류가 발생합니다. 배열 소멸은 배열의 각 객체마다 소멸자를 호출하고, 생성의 반대순으로 소멸합니다.  
ex)c[2]의 소멸자, c[1]의 소멸자, c[0]의 소멸자 순으로 실행.

# 객체 배열 생성 및 소멸

- 이수영: 예제 4-2를 안 보고 코딩하는 것이 처음부터 막히는 걸 보면서 강의 내용을 모두 이해하더라도 직접 프로그래밍하는 어려움을 느꼈습니다. `main()`에서 객체 배열을 먼저 생성한 후, `배열[인덱스].멤버함수(설정 값);` 을 반복하는 것을 여러 번 반복해야 할 것 같습니다. 그리고 배열의 경우에는 매개변수가 있는 생성자는 호출할 수 없다는 점을 예제 4-2 이후에 다른 예제에서도 계속 다루면서 확실히 알 수 있었습니다. 그리고 생성자를 명시적으로 만들지 않으면 컴파일 에러가 안 생긴다는 부분을 공부하면서, 지난 시간에 배운 내용을 복습할 수 있었습니다.

# 객체 배열 초기화

- 원준서: 객체 배열 초기화 방법은 배열의 각 원소 객체당 생성자를 지정하는 방법이 존재합니다. 예를 들어 `Circle circleArray[3] = { Circle(10), Circle(20), Circle() };` 와 같이 작성한다면, `circleArray[0]` 객체가 생성될 때 생성자 `Circle(10)`을 호출하고, 마찬가지로 `CircleArray[1]`, `CircleArray[2]` 객체가 생성될 때 생성자 `Circle(20)`, `Circle()`을 호출합니다.



# 객체 배열 초기화

- 이수영: 2차원 배열 초기화가 가장 기억에 남습니다. 특히 기본 생성자는 1로 초기화되어 있지만, 직접 값을 지정하는 부분이 기억납니다. 강의자료 11페이지를 예로 들면, `Circle(1)`은 `[0,0]`에 1이라는 값으로 초기화하는 것이고, `[1,2]`에 해당하는 `Circle()`은 디폴트로 호출한다는 의미가 기억에 납니다. 이 부분을 다루기 전에 예제를 실습할 때는 디폴트가 1이면 다른 값을 설정하고 싶을 때는 어떻게 하는 거지?라는 막연한 궁금증을 가졌는데, 이 부분을 배우면서 그 궁금증이 해소돼서 더 기억에 남습니다.

# 동적 메모리 할당 및 반환

- 원준서: 정적 할당은 변수 선언을 통해 필요한 메모리를 할당합니다. 많은 양의 메모리는 배열 선언을 통해 할당됩니다. 동적 할당은 필요한 양이 예측되지 않는 경우. 프로그램 작성 시 할당 받을 수 없습니다. 실행 중에 힙 메모리에서 할당합니다.

※힙은 운영체제가 프로세스(프로그램)의 실행을 시작시킬 때 동적 할당 공간으로 준 메모리 공간입니다. C언어의 동적 메모리 할당은 malloc()/free() 라이브러리 함수를 사용하지만, C++언어의 동적 메모리 할당/반환은 new연산자와 delete 연산자를 사용합니다. C언어는 동적 메모리 할당을 위해 malloc()/free() 라이브러리 함수를 사용하지만, C언어와는 다르게 C++언어는 동적 메모리 할당/반환을 위해 new연산자와 delete연산자를 사용한다는 것을 새롭게 알게 되었습니다. new연산자는 기본 타입 메모리, 배열, 객체, 객체 배열을 할당합니다. 객체의 동적 생성은 힙 메모리로부터 객체를 위한 메모리 할당을 요청하고, 객체 할당 시 생성자를 호출합니다. delete연산자는 new로 할당 받은 메모리를 반환하고, 객체의 동적 소멸은 소멸자를 호출 뒤 객체를 힙에 반환합니다. new/delete의 연산자의 사용 형식은 다음과 같습니다. 데이터타입 \*포인터변수 = new 데이터타입; delete 포인터변수; delete 사용 시 주의 사항으로는 적절치 못한 포인터로 delete하면 실행 시간 오류가 발생합니다. 동적으로 할당 받지 않는 메모리 반환은 오류가 발생하며, 동일한 메모리를 두 번 반환해도 오류가 발생합니다.

# 동적 메모리 할당 및 반환

- 이수영: 7주차 수업에서 가장 기억에 남는 부분은 동적 메모리, this 연산자입니다. 일반적으로 교재 예제는 개념을 잘 익히기 위한 개론적인 내용에서 끝날 때가 있습니다. 그런데 실제 프로그램은 사용자에 따라 메모리에 값을 할당하는 경우가 많은데, 이처럼 메모리를 얼마나 할당해줘야 하는지 예측되지 않을 때 힙(heap)메모리 영역이 따로 있다는 점을 칠판에 그림으로 말씀 해주셔서 그 그림이 기억에 남습니다. 코드 영역, 데이터 영역이 있고, 런타임 때 사용자가 사용할 수 있는 영역인 힙(heap) 영역, local 변수, local 함수가 임시로 저장되는 스택(stack) 영역들을 그림으로 보여 주셔서, 메모리를 쉽게 이해할 수 있었습니다. 동적인 경우에 할당은 new, 반환은 delete를 사용합니다.

# 배열의 동적 할당 및 변환

- 원준서: 배열의 동적 할당 및 반환일 때, new/delete 연산자의 사용 형식은 다음과 같습니다.  
데이터타입 \*포인터변수 = new 데이터타입 [배열의 크기] //동적 배열 할당  
delete [] 포인터변수; //배열 반환 동적 할당 메모리 초기화는 동적 할당 시 초기화됩니다.  
데이터타입 \*포인터변수 = new 데이터타입(초깃값); 배열은 동적 할당 시 초기화가 불가능합니다. delete시 []를 생략한다면, 컴파일 오류는 아니지만 비정상적인 반환이 발생합니다.

# 배열의 동적 할당 및 변환

- 이수영: 앞의 '동적 메모리 할당 및 반환'에서 배열은 [배열의 크기], 그리고 delete 뒤에는 대괄호 '[]'를 사용한다는 점만 추가돼 앞선 내용과 연관지어 쉽게 익힐 수 있었습니다. 또 예제 4-6의 문제를 5가지로 나누면서 여러 번 반복해야겠다고 느꼈습니다. cin으로 사용자에게 값을 입력받고, 그 값을 배열의 크기로 쓰면 된다는 점, n번째 정수와 같이 출력되기 위해서는 for 반복문을 쓰면 된다는 점이 가장 기억에 남습니다. 1학년 때 코드를 짤 때 n번째는 00입니다. 에서 n번째를 어떻게 코드를 작성해야 할 지 오랫동안 고민한 적이 있는데, for문을 쓰면 쉽게 구현할 수 있다는 걸 알고 for문을 이해했는데, 예제 4-6을 하며 그 때가 생각났습니다. 확실히 오래 고민할수록 오래 기억되는 것 같아요.

# 객체의 동적 생성 및 반환

- 원준서: 객체의 동적 생성 및 반환은 다음과 같이 작성할 수 있습니다. 클래스이름 \*포인터변수 = new 클래스이름;  
클래스이름 \*포인터변수 = new 클래스이름(생성자매개변수리스트);  
delete 포인터변수;
- 이수영: 앞선 '배열'의 동적 할당이 데이터타입 \*포인터변수 = new 데이터타입 [배열의 크기]였다면, '객체'의 동적 할당은 [배열의 크기]가 아니라 (초기값)만 달라진다. 그리고 '배열' 반환은 delete뒤에 대괄호가 반드시 있어야 했다면, '객체'의 반환은 delete 뒤에 []를 생략해야 원하는 결과로 반환이 잘 이뤄진다는 걸 비교하며 배웠습니다.

# 객체 배열의 동적 생성 및 반환

- 원준서: 객체 배열의 동적 생성 및 반환은 다음과 같이 작성할 수 있습니다. 클래스이름 \*포인터변수 = new 클래스이름 [배열 크기]; delete [] 포인터변수; // 포인터변수가 가리키는 객체 배열을 반환. 동적으로 생성된 배열도, 보통 배열처럼 사용합니다.
- 이수영: 이 즈음 되니 여러 개념이 섞이기 시작했습니다. 생성[할당]/소멸[반환]을 배우는데, 배열과 객체의 차이를 명확하게 잘 구분하지 못하면 기억이 안 날때마다 책을 들여다볼 것 같아서 계속 반복할 필요성을 느꼈습니다. 제가 떠올린 건 배열은 대괄호가 붙고, 객체는 대괄호가 붙으면 안 된다고 기억하려 합니다.

# this 포인터

- 원준서: this 포인터는 객체 자신 포인터로, 클래스의 멤버 함수 내에서만 사용합니다. 개발자가 선언하는 변수가 아니고, 컴파일러가 선언한 변수로, 멤버 함수의 컴파일러에 의해 묵시적으로 삽입 선언되는 매개 변수입니다. 각 객체 속의 this는, 다른 객체의 this와는 다릅니다. this가 필요한 경우에는 매개변수의 이름과 멤버 변수의 이름이 같거나, 멤버 함수가 객체 자신의 주소를 리턴할 때, 연산자 중복 시에 필요합니다. this의 제약 사항으로는 멤버 함수가 아닌 함수에서 this를 사용할 수 없는데, 이는 객체와의 관련성이 없기 때문이고, static 멤버 함수에서 this를 사용할 수 없는 이유는 객체가 생기기 전에 static 함수 호출이 있을 수 있기 때문입니다. 매개변수의 이름과 멤버 변수의 이름이 같을 경우의 해결법을 잘 몰랐는데, this 포인터를 사용하면 해결할 수 있다는 것을 새롭게 알게 되었습니다.



# this 포인터

- 이수영: this는 자바에서도 배우는 키워드인데, this 포인터라는 개념을 C++에서 또 접하게 됐습니다. 키워드 자체는 익숙해도 개념을 설명하기는 어색했습니다. 자기 자신을 가리킵니다.라고 했지만, 그래서 어떨 때 쓰이는데?에 대한 구체적인 예시들을 답하긴 어려웠습니다. 우선 오늘 배운 내용에 적용한다면, static 멤버는 this 포인터로 호출할 수 없는데 시점이 다르기 때문입니다. 컴파일 때 static 멤버가 호출되는데, 객체는 그 이후에 호출되기 때문입니다. 그리고 객체 1, 2, 3 각각에 this가 있어도, 이 this는 각각 어떤 객체에 소속되는지 여부에 따라 1, 2, 3을 가리킵니다. 즉, 객체 2의 this는 1을 가리키지 않습니다. 이것 역시 교안 31p 그림을 통해 확실히 알 수 있었습니다.

# Pair Programming

# [1차시] 포인터 개념 논의

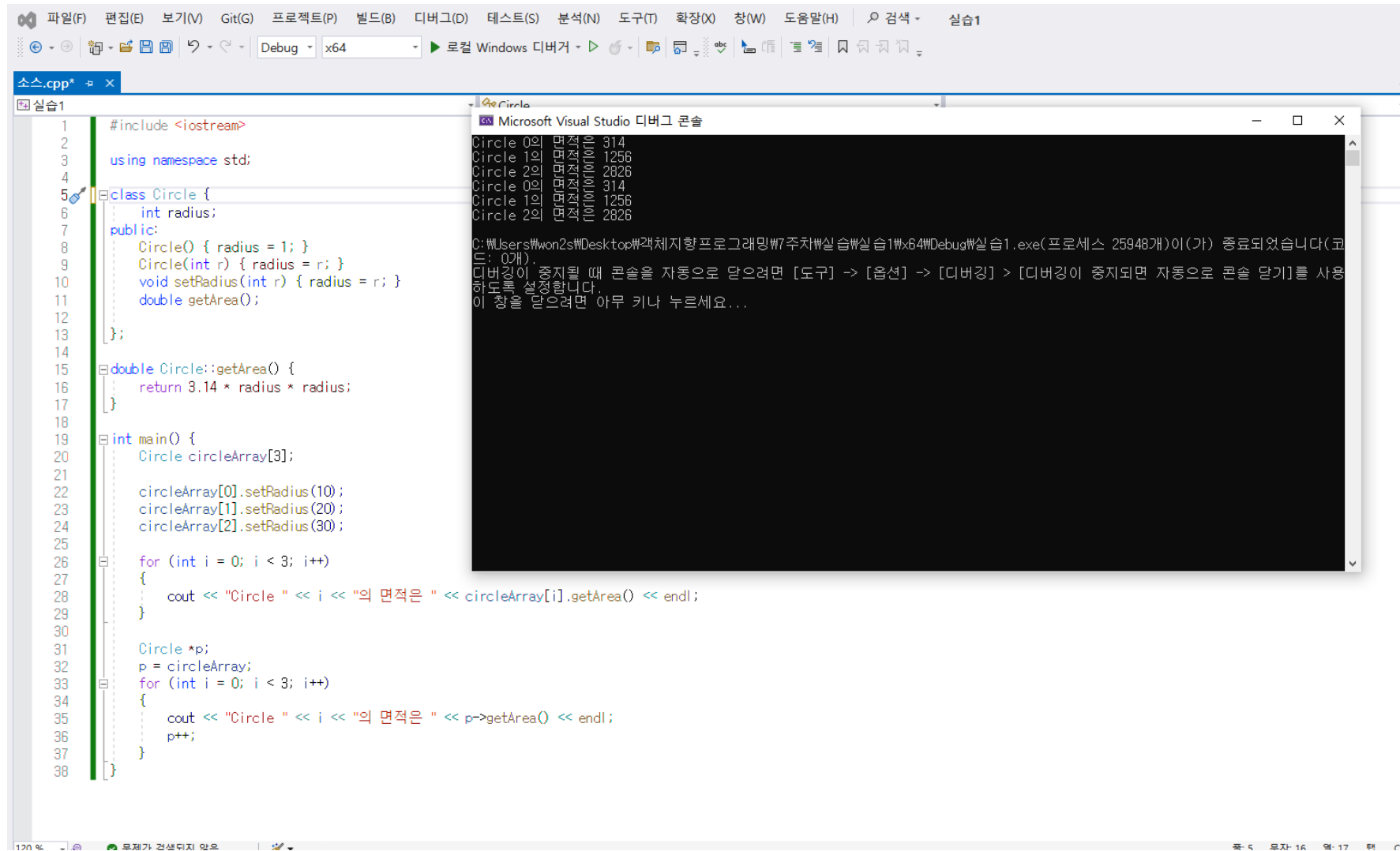
- 포인터는 '주소' 개념입니다. 집주소가 101호라고 할 때, 101호는 포인터에 해당합니다.
- string 타입의 변수 a가 있다고 할 때,  
a = "원준서", 이후에 a = "이수영" 이런 식으로 입주민이 변하듯,  
변수의 값은 변할 수 있습니다. 그런데 변수 a 자체의 주소(101호)  
는 변하지 않습니다.
- 어려웠던 점: 포인터에 대한 개념을 어렵듯이 주소로 알고 있는데, 한문장으로 일목요연하게 정리하기가 어려웠습니다.

# [1차시] 예제 4-2 코드 안 보고 실습하기

- 선언부에서 디폴트 생성자, int 매개변수를 가진 생성자를 나눠서 구현하는 부분이  
4-2 코드를 안 보고 입력해야 할 때는 바로 생각나지 않았습니니다.  
그래도 서로 논의하는 과정에서 오랜 시간 고민하는 과정을 보낸 후에 떠올라서, 앞으로 오랫동안 기억할 수 있을 것 같습니다.

# [1차시] 예제 4-2 코드 안 보고 실습하기

## 코드 (코딩: 원준서, 도움: 이수영)



The image shows a screenshot of the Microsoft Visual Studio IDE. The main window displays a C++ source file named '소스.cpp'. The code defines a 'Circle' class with a 'radius' attribute and methods 'Circle()', 'Circle(int r)', 'setRadius(int r)', and 'getArea()'. The 'main' function creates an array of three 'Circle' objects, sets their radii to 10, 20, and 30, and then prints their areas using both direct method calls and pointer access.

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

class Circle {
public:
    Circle() { radius = 1; }
    Circle(int r) { radius = r; }
    void setRadius(int r) { radius = r; }
    double getArea();
};

double Circle::getArea() {
    return 3.14 * radius * radius;
}

int main() {
    Circle circleArray[3];

    circleArray[0].setRadius(10);
    circleArray[1].setRadius(20);
    circleArray[2].setRadius(30);

    for (int i = 0; i < 3; i++)
    {
        cout << "Circle " << i << "의 면적은 " << circleArray[i].getArea() << endl;
    }

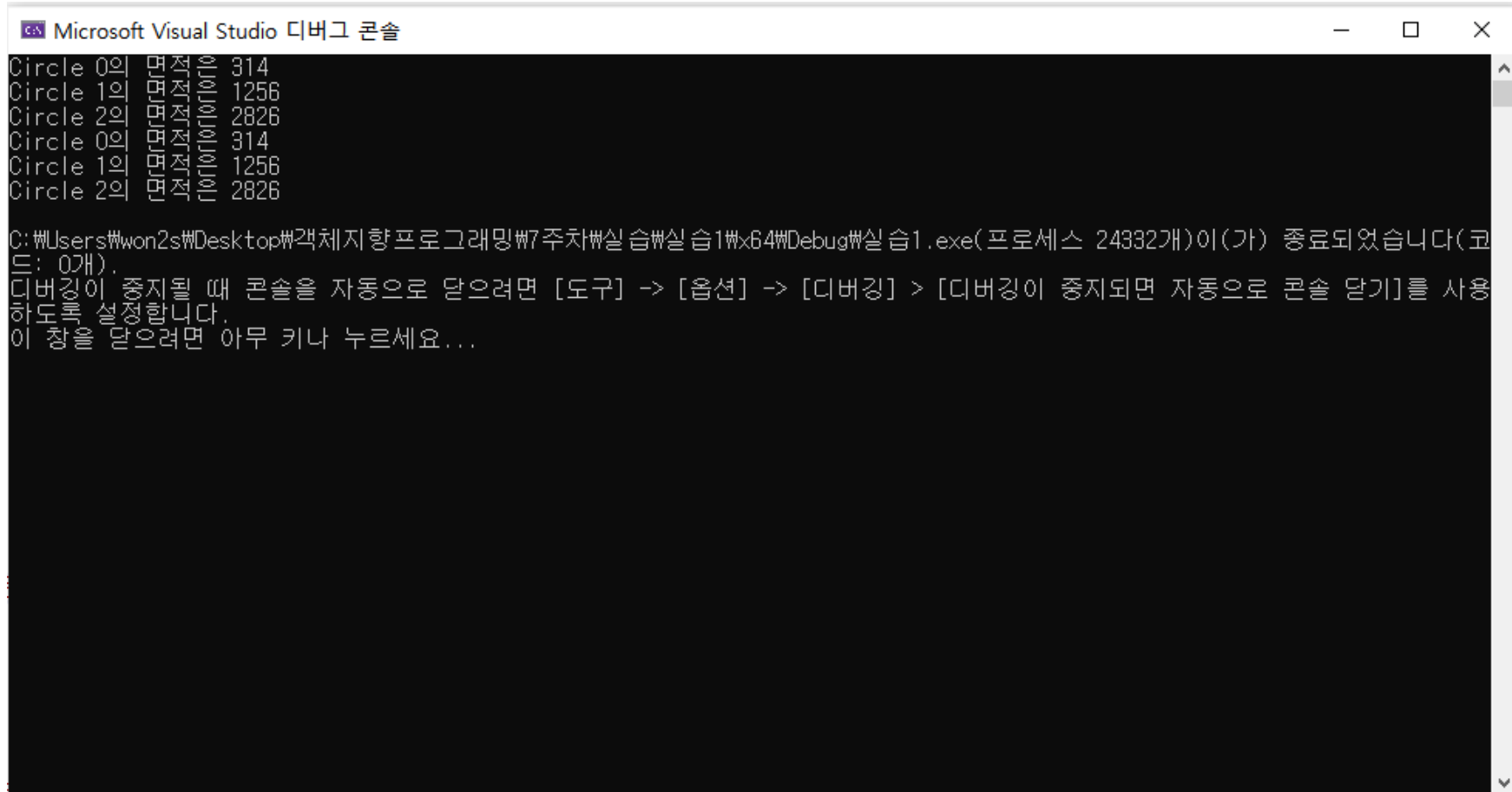
    Circle *p;
    p = circleArray;
    for (int i = 0; i < 3; i++)
    {
        cout << "Circle " << i << "의 면적은 " << p->getArea() << endl;
        p++;
    }
}
```

The 'Microsoft Visual Studio 디버그 콘솔' (Debug Console) window is open, showing the output of the program. It lists the area for each circle (0, 1, 2) using both direct and pointer access, with values 314, 1256, and 2826 respectively. Below the output, a message indicates that the process has finished successfully.

```
Circle 0의 면적은 314
Circle 1의 면적은 1256
Circle 2의 면적은 2826
Circle 0의 면적은 314
Circle 1의 면적은 1256
Circle 2의 면적은 2826

C:\Users\won2s\Desktop\책체지향프로그래밍7주차\실습\실습1\Debug\실습1.exe (프로세스 25948개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

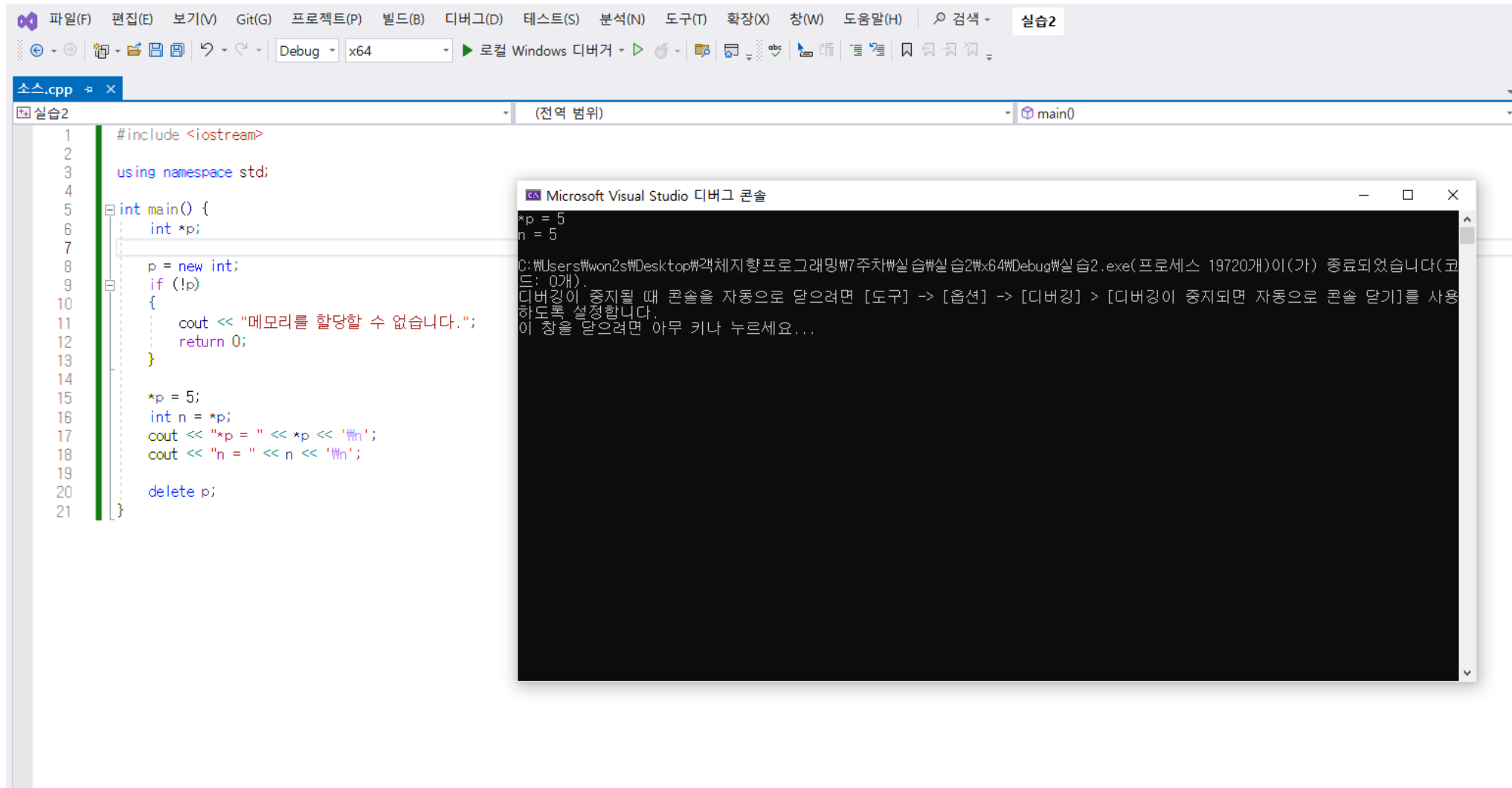
# [1차시] 예제 4-2 코드 안 보고 실습하기 실행결과



```
Microsoft Visual Studio 디버그 콘솔
Circle 0의 면적은 314
Circle 1의 면적은 1256
Circle 2의 면적은 2826
Circle 0의 면적은 314
Circle 1의 면적은 1256
Circle 2의 면적은 2826
C:\Users\won2s\Desktop\객체지향프로그래밍\7주차\실습\실습1\64\Debug\실습1.exe(프로세스 24332개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

# [2차시] 예제 4-5 코드 안 보고 실습하기

## 코드 + 결과 (코딩: 원준서, 도움: 이수영)



The image shows a screenshot of the Microsoft Visual Studio IDE. The main window displays a C++ source file named `소스.cpp` with the following code:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int *p;
7
8      p = new int;
9      if (!p)
10     {
11         cout << "메모리를 할당할 수 없습니다.";
12         return 0;
13     }
14
15     *p = 5;
16     int n = *p;
17     cout << "*p = " << *p << '\n';
18     cout << "n = " << n << '\n';
19
20     delete p;
21 }
```

The code is being debugged, and the output window (Microsoft Visual Studio 디버그 콘솔) shows the following output:

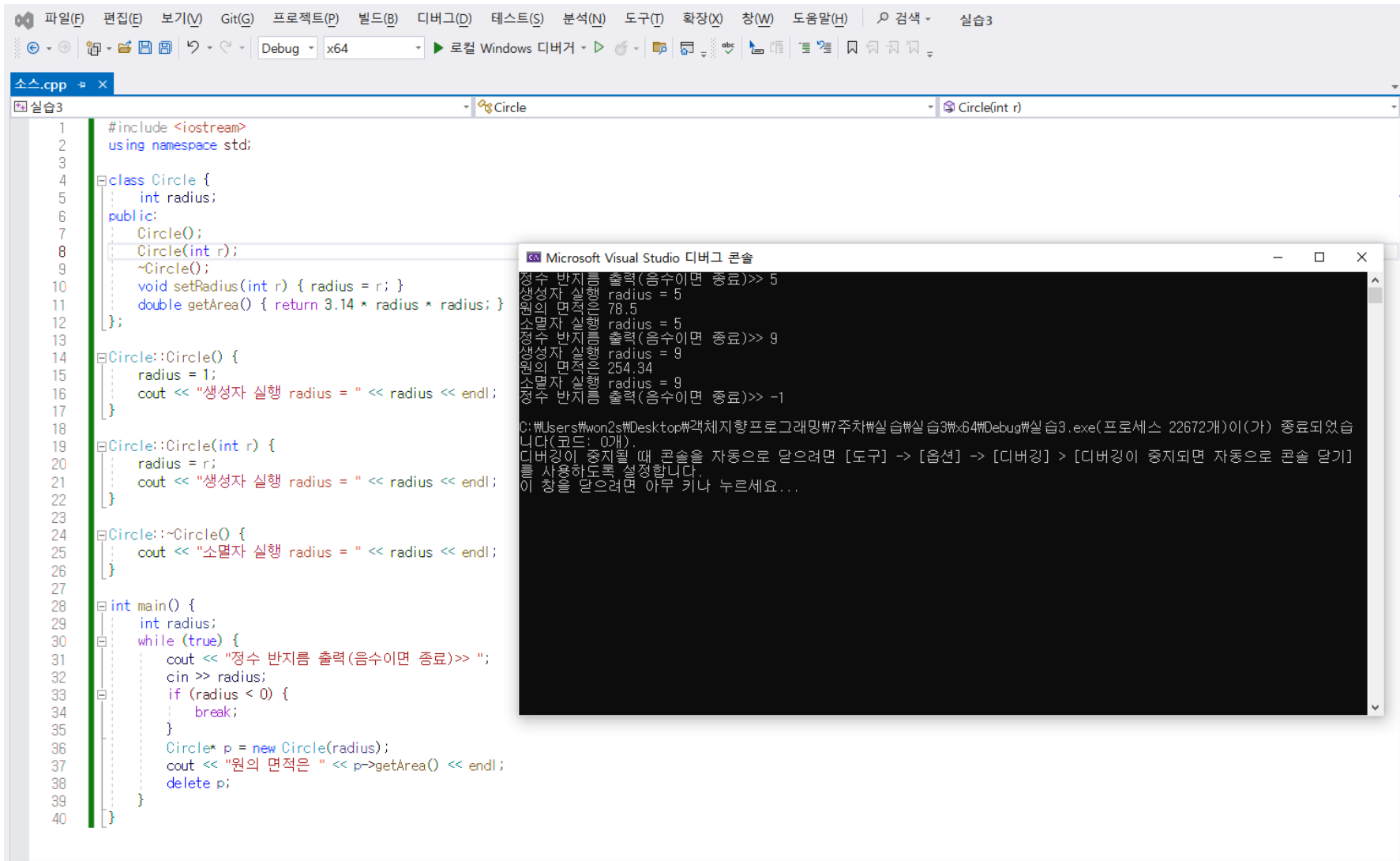
```
*p = 5
n = 5
C:\Users\won2s\Desktop\객체지향프로그래밍\7주차\실습\실습2\Debug\실습2.exe (프로세스 19720개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

## [2차시] 배열의 동적 할당 및 반환

- 앞서 본 new, delete와 같지만  
배열의 크기가 나타나는 부분과  
delete할 때, 배열 반환이 대괄호[ ]를 써서  
배열 반환을 한다는 점만 다르다.



# [3차시] 예제 4-8 선언부/구현부는 copy, main() 직접 안 보고 개인실습(원준서)



The image shows a Visual Studio IDE with a C++ source file named '소스.cpp' and a debug console window open. The source code defines a 'Circle' class with a 'radius' attribute and methods for setting the radius, getting the area, and a destructor. The 'main' function uses a loop to create and delete 'Circle' objects with different radii, outputting the results to the console.

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

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

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

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

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

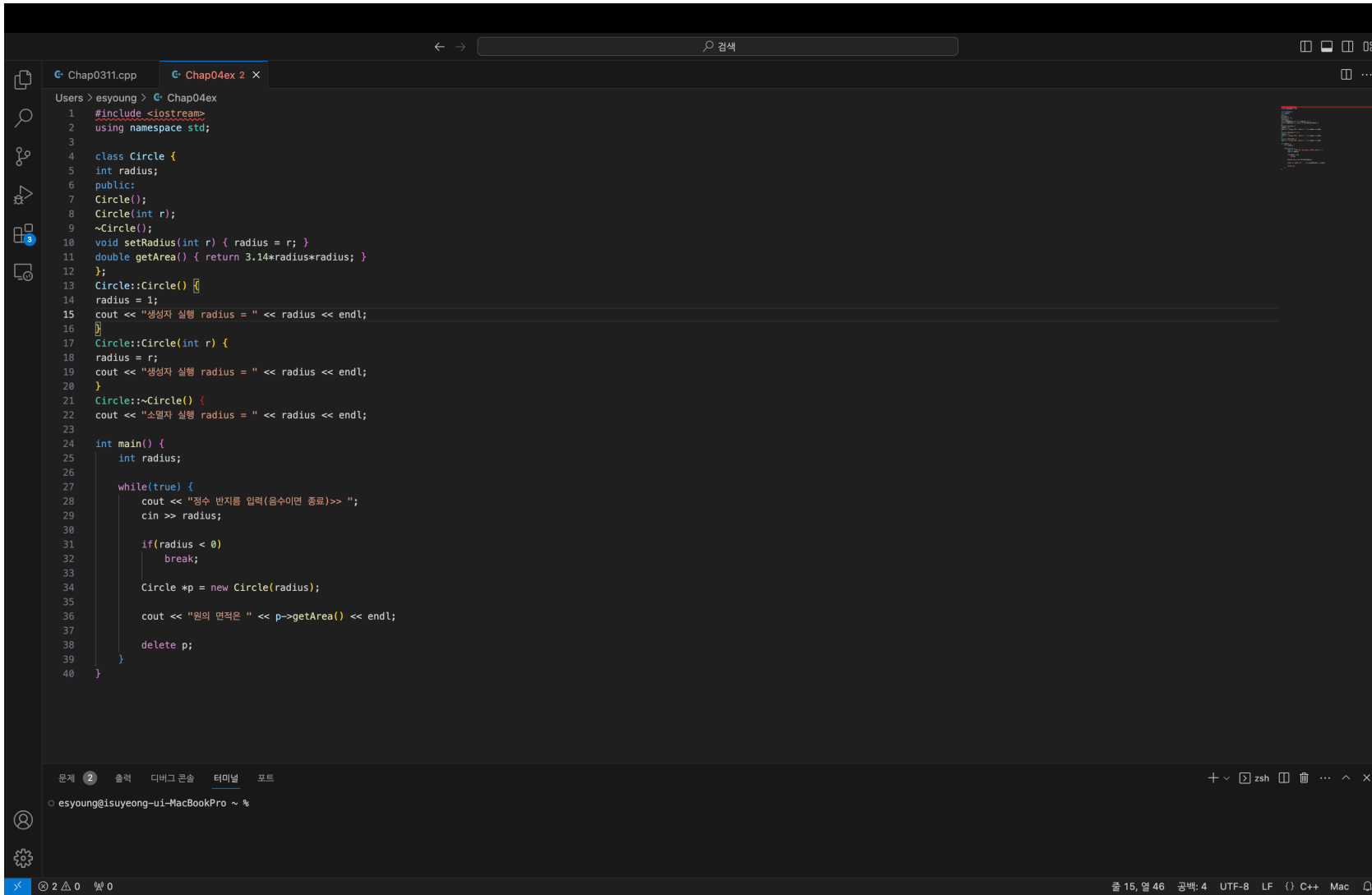
int main() {
    int radius;
    while (true) {
        cout << "정수 반지름 출력(음수이면 종료)>> ";
        cin >> radius;
        if (radius < 0) {
            break;
        }
        Circle* p = new Circle(radius);
        cout << "원의 면적은 " << p->getArea() << endl;
        delete p;
    }
}
```

Microsoft Visual Studio 디버그 콘솔

```
정수 반지름 출력(음수이면 종료)>> 5
생성자 실행 radius = 5
원의 면적은 78.5
소멸자 실행 radius = 5
정수 반지름 출력(음수이면 종료)>> 9
생성자 실행 radius = 9
원의 면적은 254.34
소멸자 실행 radius = 9
정수 반지름 출력(음수이면 종료)>> -1

C:\Users\won2s\Desktop\2학기제2항프로그래밍\7주차\실습\실습3\Debug\실습3.exe (프로세스 22672개)이(가) 종료되었습니다. (코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

# [3차시] 예제 4-8 선언부/구현부는 copy, main() 직접 안 보고 개인실습(이수영)



```
1 #include <iostream>
2 using namespace std;
3
4 class Circle {
5     int radius;
6 public:
7     Circle();
8     Circle(int r);
9     ~Circle();
10    void setRadius(int r) { radius = r; }
11    double getArea() { return 3.14*radius*radius; }
12 };
13 Circle::Circle() {
14     radius = 1;
15     cout << "생성자 실행 radius = " << radius << endl;
16 }
17 Circle::Circle(int r) {
18     radius = r;
19     cout << "생성자 실행 radius = " << radius << endl;
20 }
21 Circle::~Circle() {
22     cout << "소멸자 실행 radius = " << radius << endl;
23 }
24
25 int main() {
26     int radius;
27
28     while(true) {
29         cout << "정수 반지름 입력(음수이면 종료)>> ";
30         cin >> radius;
31
32         if(radius < 0)
33             break;
34
35         Circle *p = new Circle(radius);
36
37         cout << "원의 면적은 " << p->getArea() << endl;
38
39         delete p;
40     }
41 }
```

출 15, 열 46 공백: 4 UTF-8 LF {} C++ Mac