

11주차 정리노트

note#6

2023. 11. 13(월)

16조 이수영, 원준서

목차

Pair Programming ppt p.3~p.39

- [1차시] virtual 키워드 유무(ppt 6p, 9-2)
- [2차시] 손자 클래스 함수 호출(예제 9-3)
- [2차시] 소멸 순서 예측(예제 9-6)
- [2차시] Shape(C/R/L/Main)(20p, 22p)
- [3차시] 예제9-7 안 보고 실습(Calculator)
Adder, Subtractor, Calculator

정리노트

ppt p.40~p.50

- 가상 함수와 오버라이딩
- 함수 재정의 vs 오버라이딩
- 동적 바인딩
- C++ 오버라이딩
- 오버라이딩 범위 지정 연산자(:)
- 가상 소멸자
- 오버로딩 vs 함수 재정의 vs 오버라이딩 전격 비교
- 가상 함수 오버라이딩
- 기본 클래스의 포인터 활용
- 추상 클래스

[1차시 이수영] 6p, 예제 9-2(main) 참조해서 왼쪽/오른쪽 프로그래밍해보기

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the code editor displays `Chap9sy.cpp` with the following content:

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

class Base {
public:
    void f() {
        cout << "Base::f() called" << endl;
    }
};

class Derived : public Base {
public:
    void f() {
        cout << "Derived::f() called" << endl;
    }
};

int main() {
    Derived d, * pDer;
    pDer = &d;
    pDer = &d;
    pDer->f();

    Base* pBase;
    pBase = pDer;
    pBase->f();
}
```

A yellow callout box contains the Korean text: "기본 클래스의 f()", "파생 클래스의 f()", and "다 1번 호출됩니다." (The base class's f() and derived class's f() are called once each).

The bottom center shows the Microsoft Visual Studio Debug Console output:

```
Microsoft Visual Studio 디버그 콘솔
Derived::f() called
Base::f() called
C:\Users\user\source\repos\Project11sy\x64\Debug\Project11sy.exe(프로세스 11860개)이(가) 종료되었습니다.(코드: 0개).
이 상을 닫으려면 아무 키나 누르세요...
```

The Solution Explorer on the right shows the project structure: `Project11sy` containing `Chap9sy.cpp`.

[1차시 이수영] 6p, 예제 9-2(main) 참조해서 왼쪽/오른쪽 프로그래밍해보기

파생 클래스의 f()가 2번 실행됩니다.
기본 클래스의 f()가 호출되지 않습니다.

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

class Base {
public:
    virtual void f() {
        cout << "Base::f() called" << endl;
    }
};

class Derived : public Base {
public:
    virtual void f() {
        cout << "Derived::f() called" << endl;
    }
};

int main() {
    Derived d, * pDer;
    pDer = &d;
    pDer = &d;
    pDer->f();

    Base* pBase;
    pBase = pDer;
    pBase->f();
}
```

Microsoft Visual Studio 디버그 콘솔

```
Derived::f() called
Derived::f() called
C:\Users\user\source\repos\Project11sy\x64\Debug\Project11sy.exe(프로세스 13540개)이(가) 종료되었습니다(코드: 0x0).
```

144 % 문 제가 검색되지 않음 오류 목록 출력

빌드 성공 찾기

9°C 맑음 오후 12:17 2023-11-13

[1차시 이] 예제 9-2 virtual 키워드 있을 경우 즉, 오버라이딩의 경우 실행결과

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the code editor displays `Chap9sy.cpp` with the following content:

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

class Base {
public:
    // virtual 키워드 있을 경우
    virtual void f() { cout << "Base::f() called" << endl; }
};

class Derived : public Base {
public:
    virtual void f() { cout << "Derived::f() called" << endl; }
};

int main() {
    Derived d, * pDer;
    pDer = &d;
    pDer->f(); // Derived::f() 호출
    Base* pBase;
    pBase = pDer; // 업 캐스팅
    pBase->f(); // 동적 바인딩 발생!! Derived::f() 실행
}
```

A yellow callout box highlights the note: "기본 클래스에 virtual 키워드 있으면
파생 클래스의 f()만 2번 호출됩니다.
기본 클래스 f() 호출되지 않습니다."

The bottom window shows the output of the program in the Microsoft Visual Studio Debug Console:

```
Derived::f() called
Derived::f() called
C:\Users\user\source\repos\Project11sy\x64\Debug\Project11sy.exe(프로세스 16380개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

The status bar at the bottom right shows the date and time: "오늘 12:27 2023-11-13".

[1차시 이] 예제 9-2 virtual 키워드 없을 경우 즉, 함수 재정의의 경우 실행결과

The screenshot shows the Microsoft Visual Studio interface. In the center, there is a yellow callout box containing the following text:

기본 클래스에 virtual 키워드 없으면
파생 클래스의 f()가 1번 호출되고,
기본 클래스 f()도 호출됩니다.

The code editor window displays the following C++ code:

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

class Base {
public:
    // virtual 키워드 없을 경우
    void f() { cout << "Base::f() called" << endl; }
};

class Derived : public Base {
public:
    void f() { cout << "Derived::f() called" << endl; }
};

int main() {
    Derived d, * pDer;
    pDer = &d;
    pDer->f(); // Derived::f() 호출
    Base* pBase;
    pBase = pDer; // 업 캐스팅
    pBase->f(); // 동적 바인딩 발생!! Derived::f() 실행
}
```

The Task List window at the bottom shows the output of the program:

```
Microsoft Visual Studio 디버그 콘솔
Derived::f() called
Base::f() called

C:\Users\user\source\repos\Project11sy\x64\Debug\Project11sy.exe(프로세스 12916개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

The status bar at the bottom right indicates the date and time: 2023-11-13 오후 12:28.

[1차시 원준서] 6p, 예제 9-2(main) 참조해서 왼쪽/오른쪽 프로그래밍해보기

The screenshot shows the Microsoft Visual Studio interface. The top menu bar includes: 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(T), 확장(X), 창(W), 도움말(H). The title bar says "11주차 실습". The toolbar includes icons for file operations like Open, Save, and Print, along with build and debug buttons. The status bar at the bottom shows "132 %", "문제가 검색되지 않음", and "줄: 26 문자: 2 CRLF".

The code editor displays the following C++ code:

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

class Base {
public:
    void f() {
        cout << "Base::f() called" << endl;
    }
};

class Derived : public Base {
public:
    void f() {
        cout << "Derived::f() called" << endl;
    }
};

int main() {
    Derived d, * pDer;
    pDer = &d;
    pDer->f();

    Base* pBase;
    pBase = pDer;
    pBase->f();
}
```

The code defines a base class `Base` and a derived class `Derived`, both containing a `f()` method. In the `main` function, an object `d` of type `Derived` is created, and its `f()` method is called via a pointer `pDer`. Then, the pointer is cast to a `Base*` and its `f()` method is called again.

The right side of the interface shows the "Microsoft Visual Studio 디버그 콘솔" (Debug Console) window. It displays the following output:

```
Derived::f() called
Base::f() called
```

Below the output, there is some explanatory text in Korean:

C:\Users\won2st\Desktop\객체지향프로그래밍\11주차 실습\11주차 실습\11주차 실습.exe(프로세스 18292개)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 원준서] 6p, 예제 9-2(main) 참조해서 왼쪽/오른쪽 프로그래밍해보기

The screenshot shows the Microsoft Visual Studio interface. The top menu bar includes '파일(F)', '편집(E)', '보기(V)', 'Git(G)', '프로젝트(P)', '빌드(B)', '디버그(D)', '테스트(S)', '분석(N)', '도구(T)', '확장(X)', '창(W)', '도움말(H)', '검색 -' (Search), and '11주차 실습' (Week 11 Practice). The toolbar below has icons for file operations like Open, Save, and Print, along with build and run buttons. The status bar indicates 'Debug x64'.

The code editor displays the following C++ code:

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

class Base {
public:
    virtual void f() {
        cout << "Base::f() called" << endl;
    }
};

class Derived : public Base {
public:
    virtual void f() {
        cout << "Derived::f() called" << endl;
    }
};

int main() {
    Derived d, *pDer;
    pDer = &d;
    pDer->f();

    Base* pBase;
    pBase = pDer;
    pBase->f();
}
```

The code defines a base class `Base` with a virtual member function `f()`. It also defines a derived class `Derived` that inherits from `Base` and overrides the `f()` function. In the `main` function, it creates an object `d` of type `Derived` and a pointer `pDer` pointing to `d`. It then calls `pDer->f()`, which prints "Base::f() called". Next, it declares a `Base*` pointer `pBase` and sets it to `pDer`. Finally, it calls `pBase->f()`, which prints "Derived::f() called".

The bottom right window is the 'Microsoft Visual Studio 디버그 콘솔' (Debug Console), showing the output of the program:

```
Derived::f() called
Derived::f() called
```

The console also contains some informational text about the debugger's behavior when it is terminated.

[2차시 이수영] 예제 9-3 손자 클래스 동적바인딩

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the code editor displays `Chap9sy.cpp` with the following content:

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

class Base {
public:
    virtual void f() { cout << "Base::f() called" << endl; }
};

class Derived : public Base {
public:
    void f() { cout << "Derived::f() called" << endl; }
};

class GrandDerived : public Derived {
public:
    void f() { cout << "GrandDerived::f() called" << endl; }
};

int main() {
    GrandDerived g;
    Base* bp;
    Derived* dp;
    GrandDerived* gp;
    bp = dp = gp = &g;
    bp->f();
    dp->f();
    gp->f();
}
```

The code defines three classes: `Base`, `Derived`, and `GrandDerived`. Each class has a `f()` method. In `main()`, pointers `bp`, `dp`, and `gp` are all set to point to the same `GrandDerived` object `g`. When each pointer is dereferenced and its `f()` method is called, only the `GrandDerived::f()` method is executed three times, as indicated by the output in the Immediate Window.

**기본 클래스의 `f()`, 자식 클래스의 `f()`
모두 호출되지 않습니다.
기본/자식 클래스를 상속받은
손자 클래스의 `f()`만 3번 호출됩니다.**

Microsoft Visual Studio 디버그 콘솔
GrandDerived::f() called
GrandDerived::f() called
GrandDerived::f() called
C:\Users\user\source\repos\Project11sy\x64\Debug\Project11sy.exe(프로세스 12300개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...

[2차시 원준서] 예제 9-3 손자 클래스 동적바인딩

The screenshot shows the Microsoft Visual Studio IDE interface. The menu bar includes '파일(F)', '편집(E)', '보기(V)', 'Git(G)', '프로젝트(P)', '빌드(B)', '디버그(D)', '테스트(S)', '분석(N)', '도구(T)', '확장(X)', '창(W)', '도움말(H)'. The toolbar includes icons for file operations, search, and debugging. The solution explorer shows files: Subtractor.h, Adder.h, 소스3.cpp, 소스2.cpp (selected), 소스1.cpp, and 소스.cpp. The properties window is visible on the right. The main code editor contains the following C++ code:

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Base{
6 public:
7     virtual void f() { cout << "Base::f() called" << endl; }
8 };
9
10 class Derived : public Base {
11 public:
12     void f() { cout << "Derived::f() called" << endl; }
13 };
14
15 class GrandDerived : public Derived {
16 public:
17     void f() { cout << "GrandDerived::f() called" << endl; }
18 };
19
20 int main() {
21     GrandDerived g;
22     Base* bp;
23     Derived* dp;
24     GrandDerived* gp;
25     bp = dp = gp = &g;
26     bp->f();
27     dp->f();
28     gp->f();
29 }
```

The output window (Microsoft Visual Studio 디버그 콘솔) displays the following text:

```
GrandDerived::f() called
GrandDerived::f() called
GrandDerived::f() called
```

A tooltip message is visible in the bottom right corner of the output window:

C:\Users\won2s\Desktop\객체지향프로그래밍\11주차 실습\11주차 실습\x64\Debug\11주차 실습.exe(프로세스 3448개)이(가
되었습니다.(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...)

[2차시 이수영]

예제 9-6 결과(소멸순서) 예측

❖ 예측

- 파생 클래스 소멸자인 dp에 먼저 소멸되고,
기본 클래스 소멸자인 bp가 나중에 소멸된다.
- 즉, ~Derived(), ~Base() 순으로 실행결과가 출력될 것이다.

❖ 놓친 부분

- delete dp; 일 때 ~Derived()가, delete bp; 일 때 ~Base()가 출력될 줄 알았는데 아니었다. delete dp; 일 때도 delete bp; 일 때도 다 각각 ~Derived()와 ~Base()가 출력된다. ~Derived() ~Base()가 1번만 출력 되는 게 아니라 ~Derived() ~Base() ~Derived() ~Base() 이렇게 된다.

[2차시 이수영] ppt 20p, 22p 실습해보기 (1/9) Shape.h

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the content of the `Shape.h` header file. The code defines a `Shape` class with a `next` pointer to the next shape in a linked list. It includes protected methods for drawing and painting, and public methods for adding shapes to the list and getting the next shape. The `Shape.h` file is highlighted in the tabs at the top. The Solution Explorer on the right shows the project structure for 'Project11sy'.

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

class Shape {
    Shape* next;
protected:
    virtual void draw();
public:
    Shape() { next = NULL; }
    virtual ~Shape() { }
    void paint();
    Shape* add(Shape* p);
    Shape* getNext() { return next; }
};
```

[2차시 이수영] ppt 20p, 22p 실습해보기 (2/9) Shape.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the `Shape.cpp` file, which contains the following code:

```
#include <iostream>
#include "Shape.h"
using namespace std;

void Shape::paint() {
    draw();
}

void Shape::draw() {
    cout << "--Shape--" << endl;
}

Shape* Shape::add(Shape* p) {
    this->next = p;
    return p;
}
```

The Solution Explorer on the right side of the interface shows the project structure for `Project11sy`:

- Project11sy
- 정조
- 외부 종속성
- 리소스 파일
- 소스 파일
 - Chap9sy.cpp
 - Circle.cpp
 - Circle.h
 - Line.cpp
 - Line.h
 - Main.cpp
 - Rect.cpp
 - Rect.h
 - Shape.cpp
 - Shape.h
- 헤더 파일

[2차시 이수영] ppt 20p, 22p 실습해보기 (3/9) Circle.h

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the content of the Circle.h file, which defines a class Circle that inherits from Shape. The code includes an include directive for iostream and a protected member function draw(). The Solution Explorer on the right shows the project structure for 'Project11sy'.

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

class Circle : public Shape {
protected:
    virtual void draw();
};
```

Project11sy

- Project11sy
- 정조
- 외부 종속성
- 리소스 파일
- 소스 파일
 - Chap9sy.cpp
 - Circle.cpp
 - Line.cpp
 - Line.h
 - Main.cpp
 - Rect.cpp
 - Rect.h
 - Shape.cpp
 - Shape.h
- 헤더 파일

[2차시 이수영] ppt 20p, 22p 실습해보기 (4/9) Circle.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Circle.cpp` within the `Project11sy` project. The code includes includes for `<iostream>`, `"Shape.h"`, and `"Circle.h"`, and defines a `draw()` method that outputs "Circle". The Solution Explorer on the right shows the project structure with files like `Chap9sy.cpp`, `Circle.cpp`, `Circle.h`, `Line.cpp`, `Line.h`, `Main.cpp`, `Rect.cpp`, `Rect.h`, `Shape.cpp`, and `Shape.h`. The status bar at the bottom indicates the system is connected to KOSDAQ 100.

```
#include <iostream>
#include "Shape.h"
#include "Circle.h"
using namespace std;

void Circle::draw() {
    cout << "Circle" << endl;
}
```

[2차시 이수영] ppt 20p, 22p 실습해보기 (5/9) Rect.h

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "Project11sy". The menu bar includes "파일(F)", "편집(E)", "보기(V)", "Git(G)", "프로젝트(P)", "빌드(B)", "디버그(D)", "테스트(S)", "분석(N)", "도구(T)", "확장(X)", "창(W)", "도움말(H)", and "검색 -". The toolbar has icons for file operations like Open, Save, and Build. The status bar at the bottom shows "144 %", "문제가 검색되지 않음", "줄: 7 문자: 3 탭 CRLF", "저장되었습니다.", "KOSDAQ 100 -1.69%", "오늘 1:48", and "2023-11-13".

The code editor window displays the content of the Rect.h file:

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

class Rect : public Shape {
protected:
    virtual void draw();
};
```

The Solution Explorer window on the right shows the project structure:

- 솔루션 탐색기
- 솔루션 'Project11sy' (1 프로젝트의 1)
- Project11sy
 - 정조
 - 외부 종속성
 - 리소스 파일
 - 소스 파일
 - Chap9sy.cpp
 - Circle.cpp
 - Circle.h
 - Line.cpp
 - Line.h
 - Main.cpp
 - Rect.cpp
 - Rect.h
 - Shape.cpp
 - Shape.h

[2차시 이수영] ppt 20p, 22p 실습해보기 (6/9) Rect.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Rect.cpp` within the `Project11sy` project. The code includes includes for `<iostream>`, `Shape.h`, and `Rect.h`, and defines a `draw()` method that outputs "Rectangle". The Solution Explorer on the right shows the project structure with files like `Line.h`, `Line.cpp`, `Rect.h`, `Rect.cpp`, `Circle.h`, `Circle.cpp`, `Shape.h`, and `Chap9sy.cpp`.

```
#include <iostream>
#include "Shape.h"
#include "Rect.h"
using namespace std;

void Rect::draw() {
    cout << "Rectangle" << endl;
}
```

[2차시 이수영] ppt 20p, 22p 실습해보기 (7/9) Line.h

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the content of the Line.h file, which defines a class Line that inherits from Shape. The code includes an include directive for <iostream> and a protected member function draw(). The Solution Explorer on the right shows the project structure for 'Project11sy'.

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

class Line : public Shape {
protected:
    virtual void draw();
};
```

Project11sy

- Project11sy
- 정조
- 외부 종속성
- 리소스 파일
- 소스 파일
 - Chap9sy.cpp
 - Circle.cpp
 - Circle.h
 - Line.cpp
 - Line.h
 - Main.cpp
 - Rect.cpp
 - Rect.h
 - Shape.cpp
 - Shape.h
- 헤더 파일

[2차시 이수영] ppt 20p, 22p 실습해보기 (8/9) Line.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for the `Line.cpp` file, which includes includes for `<iostream>`, `"Shape.h"`, and `"Line.h"`, and a `draw()` function that outputs "Line". The Solution Explorer on the right shows the project structure for "Project11sy" with files like `Chap9sy.cpp`, `Circle.cpp`, `Circle.h`, `Line.cpp`, `Line.h`, `Main.cpp`, `Rect.cpp`, `Rect.h`, and `Shape.cpp`. The status bar at the bottom indicates the file is saved, the current line is 8, and the character count is 2.

```
#include <iostream>
#include "Shape.h"
#include "Line.h"
using namespace std;

void Line::draw() {
    cout << "Line" << endl;
}
```

[2차시 이수영] ppt 20p, 22p 실습해보기 (9/9) Main.cpp

The screenshot shows the Microsoft Visual Studio interface with the following components:

- Editor Area:** Displays the `Main.cpp` file content. The code implements a linked list of shapes (Circle, Rectangle, Line) using pointer arithmetic and dynamic memory allocation.
- Solution Explorer:** Shows the project structure for "Project11sy". It includes files like `Shape.h`, `Circle.h`, `Line.h`, `Rect.h`, `Shape.cpp`, `Circle.cpp`, `Line.cpp`, `Rect.cpp`, and `Main.cpp`.
- Debug Console:** Shows the output of the program. It prints the names of the shapes added to the list: Circle, Rectangle, Circle, Line, Rectangle. It also indicates that the process has ended with a status of 0.
- Taskbar:** At the bottom, the taskbar shows various open applications including Microsoft Word, Microsoft Excel, and Microsoft Powerpoint.

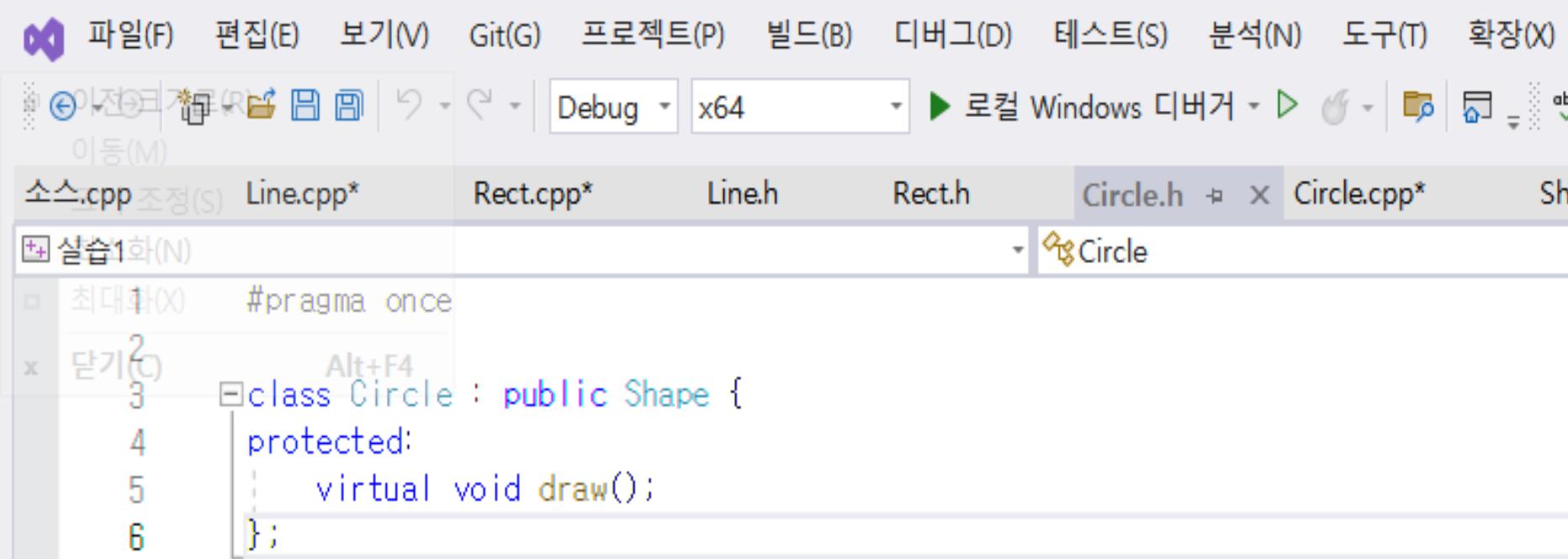
```
#include <iostream>
#include "Shape.h"
#include "Circle.h"
#include "Rect.h"
#include "Line.h"
using namespace std;

int main() {
    Shape* pStart = NULL;
    Shape* pLast;
    pStart = new Circle(); // 처음에 원 도형을 생성한다.
    pLast = pStart;
    pLast = pLast->add(new Rect()); // 사각형 객체 생성
    pLast = pLast->add(new Circle()); // 원 객체 생성
    pLast = pLast->add(new Line()); // 선 객체 생성
    pLast = pLast->add(new Rect()); // 사각형 객체 생성

    // 현재 연결된 모든 도형을 화면에 그린다.
    Shape* p = pStart;
    while (p != NULL) {
        p->paint();
        p = p->getNext();
    }

    // 현재 연결된 모든 도형을 삭제한다.
    p = pStart;
    while (p != NULL) {
        Shape* q = p->getNext(); // 다음 도형 주소 기억
        delete p; // 기본 클래스의 가상 소멸자 호출
        p = q; // 다음 도형 주소를 p에 저장
    }
}
```

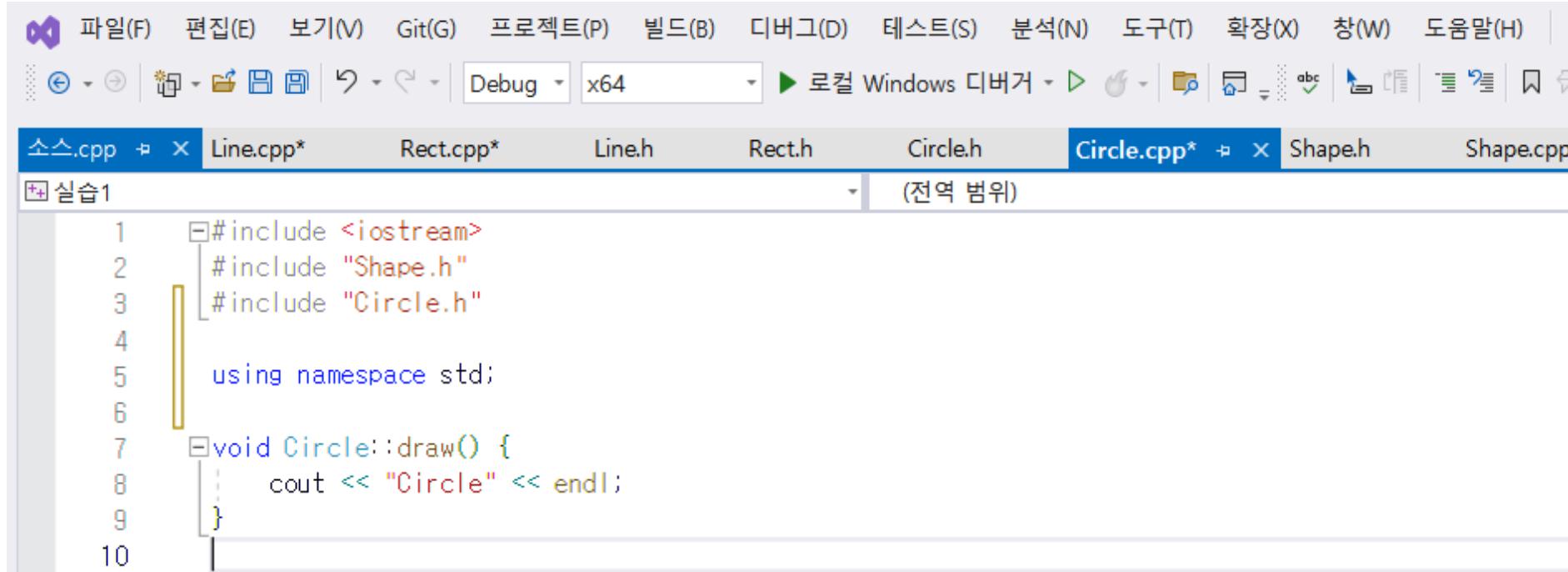
[2차시 원준서] ppt 20p, 22p 실습해보기 (1/9) Circle.h



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar includes the VS logo and Korean menu items: 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(T), 확장(X). The toolbar has icons for file operations like Open, Save, and Build. The status bar shows Debug, x64, and 로컬 Windows 디버거. The solution explorer shows files: 소스.cpp, Line.cpp*, Rect.cpp*, Line.h, Rect.h, Circle.h (selected), Circle.cpp*, and Shape.h. The code editor displays the following C++ code:

```
#pragma once
class Circle : public Shape {
protected:
    virtual void draw();
};
```

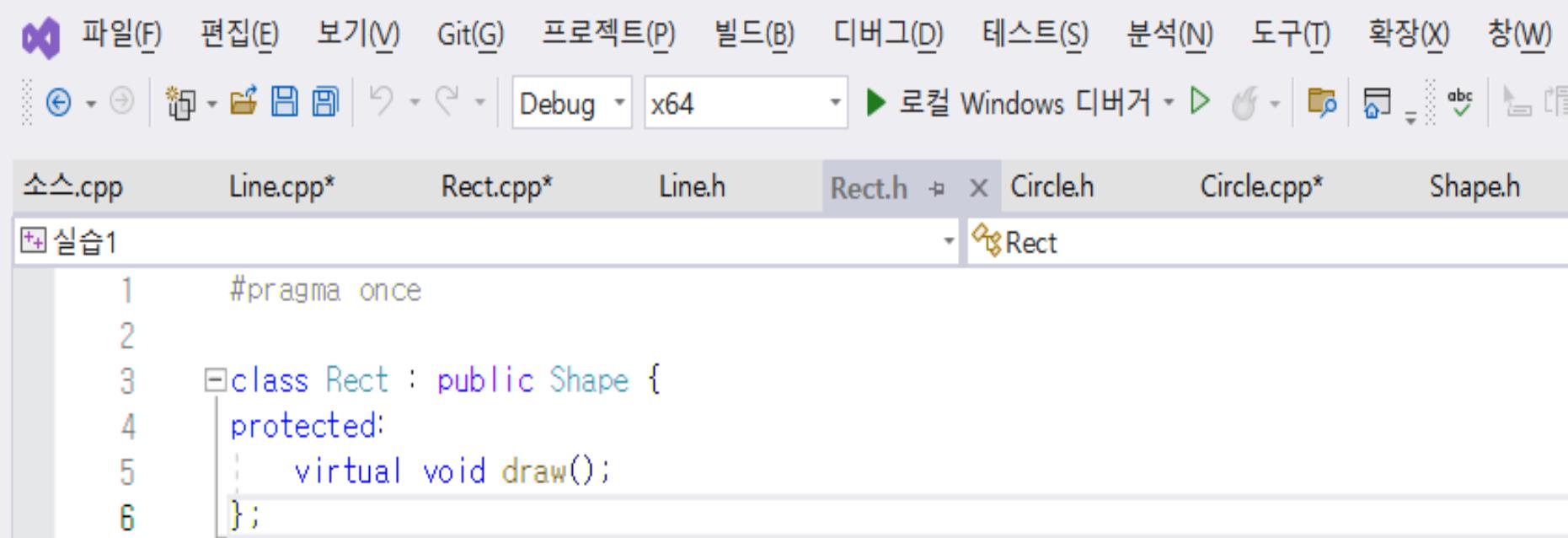
[2차시 원준서] ppt 20p, 22p 실습해보기 (2/9) Circle.cpp



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar includes standard menu items like 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(T), 확장(X), 창(W), and 도움말(H). Below the title bar is a toolbar with various icons. The tabs at the top of the code editor show several files: 소스.cpp, Line.cpp*, Rect.cpp*, Line.h, Rect.h, Circle.h, Circle.cpp*, Shape.h, and Shape.cpp. The Circle.cpp* tab is currently selected. The code editor displays the following C++ code:

```
1 #include <iostream>
2 #include "Shape.h"
3 #include "Circle.h"
4
5 using namespace std;
6
7 void Circle::draw() {
8     cout << "Circle" << endl;
9 }
10
```

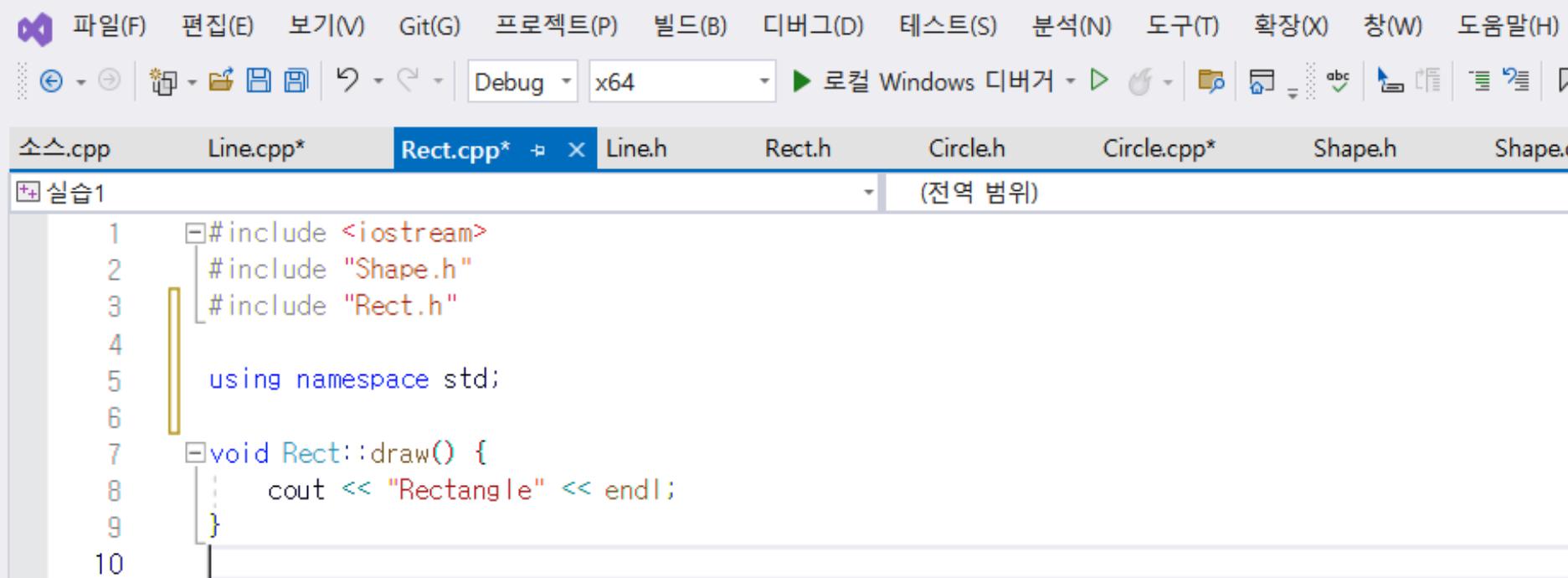
[2차시 원준서] ppt 20p, 22p 실습해보기 (3/9) Rect.h



The screenshot shows the Microsoft Visual Studio IDE interface. The menu bar includes '파일(F)', '편집(E)', '보기(V)', 'Git(G)', '프로젝트(P)', '빌드(B)', '디버그(D)', '테스트(S)', '분석(N)', '도구(T)', '확장(X)', and '창(W)'. The toolbar below the menu has icons for file operations like Open, Save, and Build. The status bar indicates 'Debug x64' and '로컬 Windows 디버거'. The tabs at the top show files: '소스.cpp', 'Line.cpp*', 'Rect.cpp*', 'Line.h', 'Rect.h', 'Circle.h', 'Circle.cpp*', and 'Shape.h'. The 'Rect.h' tab is selected. In the code editor, the 'Rect' class definition is visible:

```
1 #pragma once
2
3 class Rect : public Shape {
4 protected:
5     virtual void draw();
6 }
```

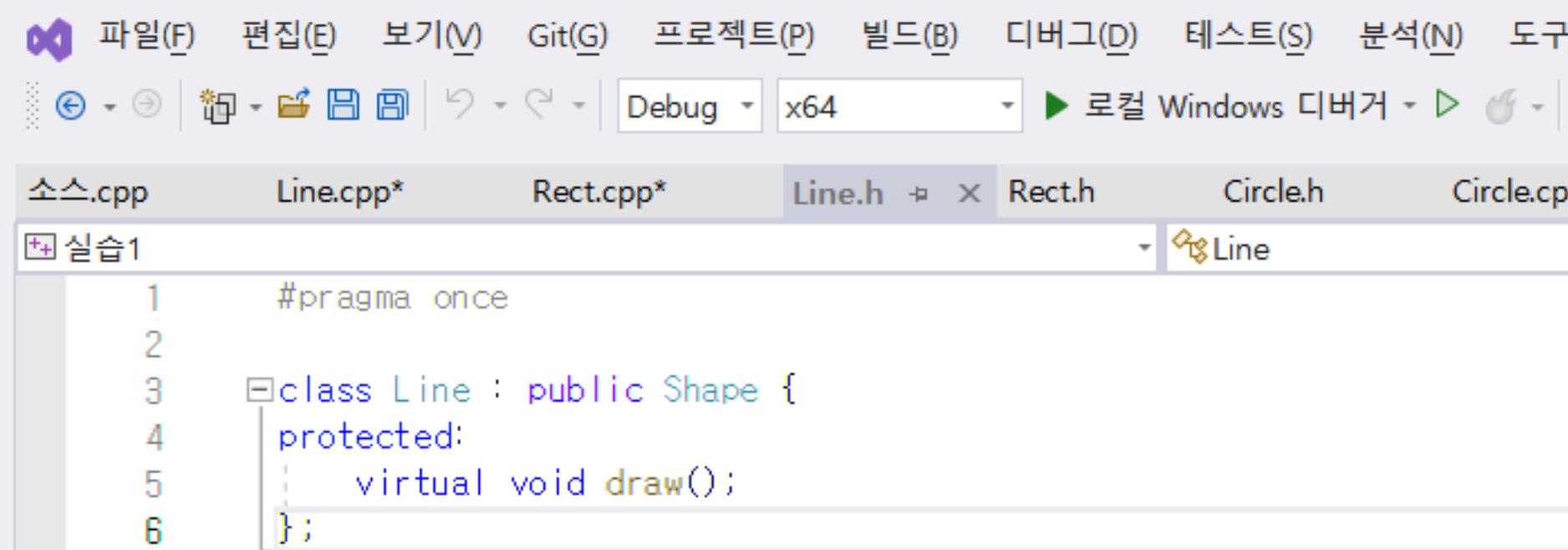
[2차시 원준서] ppt 20p, 22p 실습해보기 (4/9) Rect.cpp



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar includes the standard menu items: 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(T), 확장(X), 창(W), 도움말(H). Below the title bar is the toolbar with various icons. The solution explorer shows files: 소스.cpp, Line.cpp*, Rect.cpp*, Line.h, Rect.h, Circle.h, Circle.cpp*, Shape.h, Shape.c. The Rect.cpp tab is selected. The code editor displays the following C++ code:

```
1 #include <iostream>
2 #include "Shape.h"
3 #include "Rect.h"
4
5 using namespace std;
6
7 void Rect::draw() {
8     cout << "Rectangle" << endl;
9 }
```

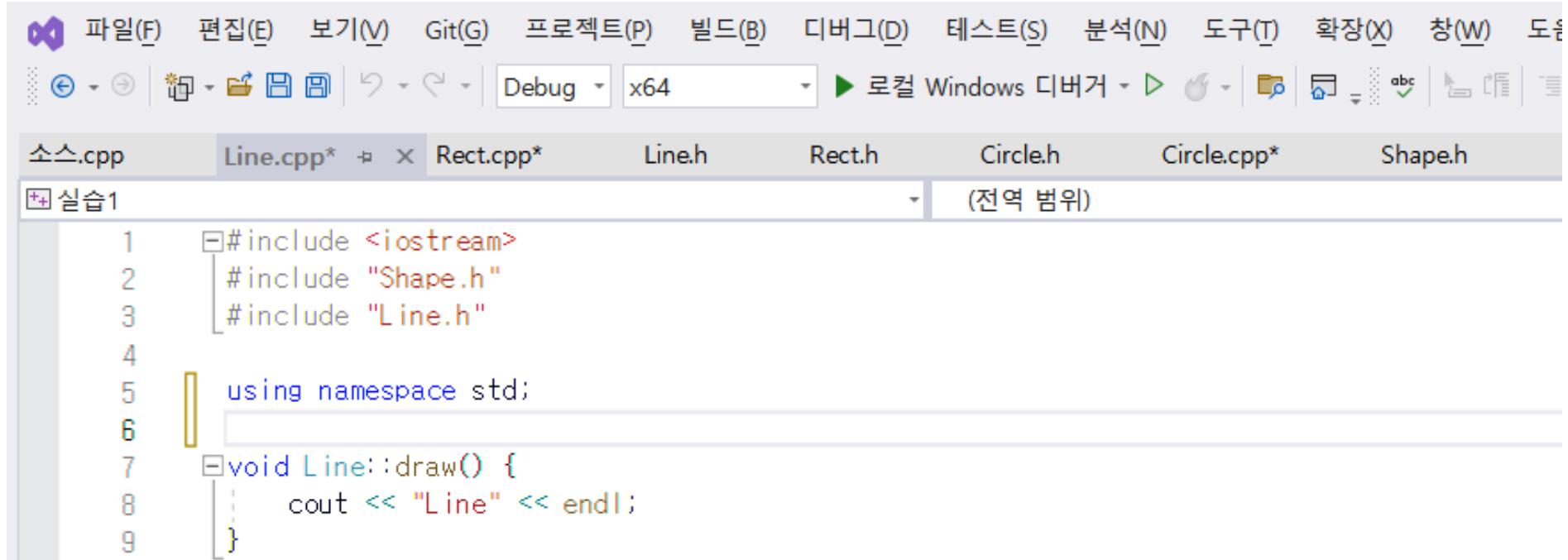
[2차시 원준서] ppt 20p, 22p 실습해보기 (5/9) Line.h



The screenshot shows the Microsoft Visual Studio IDE interface. The menu bar includes '파일(F)', '편집(E)', '보기(V)', 'Git(G)', '프로젝트(P)', '빌드(B)', '디버그(D)', '테스트(S)', '분석(N)', and '도구'. The toolbar below the menu has icons for back, forward, search, and other development tools. The status bar indicates 'Debug x64' and '로컬 Windows 디버거'. The tabs at the top show files: '소스.cpp', 'Line.cpp*', 'Rect.cpp*', 'Line.h', 'Rect.h', 'Circle.h', and 'Circle.cp'. The 'Line.h' tab is selected. The code editor displays the following C++ code:

```
1 #pragma once
2
3 class Line : public Shape {
4 protected:
5     virtual void draw();
6 }
```

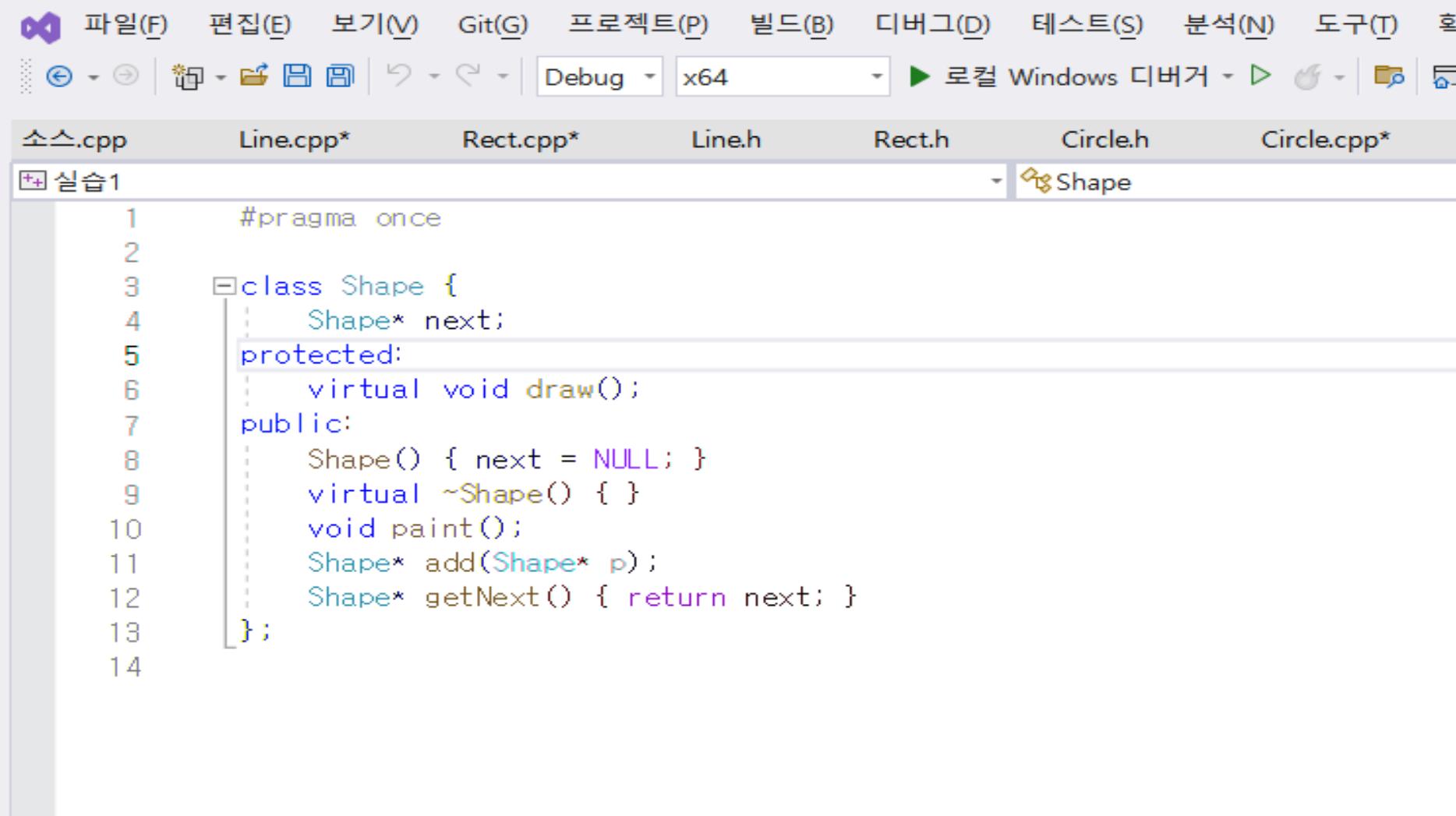
[2차시 원준서] ppt 20p, 22p 실습해보기 (6/9) Line.cpp



The screenshot shows the Microsoft Visual Studio IDE interface. The menu bar includes '파일(F)', '편집(E)', '보기(V)', 'Git(G)', '프로젝트(P)', '빌드(B)', '디버그(D)', '테스트(S)', '분석(N)', '도구(T)', '확장(X)', '창(W)', and '도움'. The toolbar has icons for file operations like Open, Save, and Build. The status bar shows 'Debug x64' and '로컬 Windows 디버거'. The solution explorer shows files: 소스.cpp, Line.cpp*, Rect.cpp*, Line.h, Rect.h, Circle.h, Circle.cpp*, and Shape.h. The current file is Line.cpp*. The code editor displays the following C++ code:

```
1 #include <iostream>
2 #include "Shape.h"
3 #include "Line.h"
4
5 using namespace std;
6
7 void Line::draw() {
8     cout << "Line" << endl;
9 }
```

[2차시 원준서] ppt 20p, 22p 실습해보기 (7/9) Shape.h

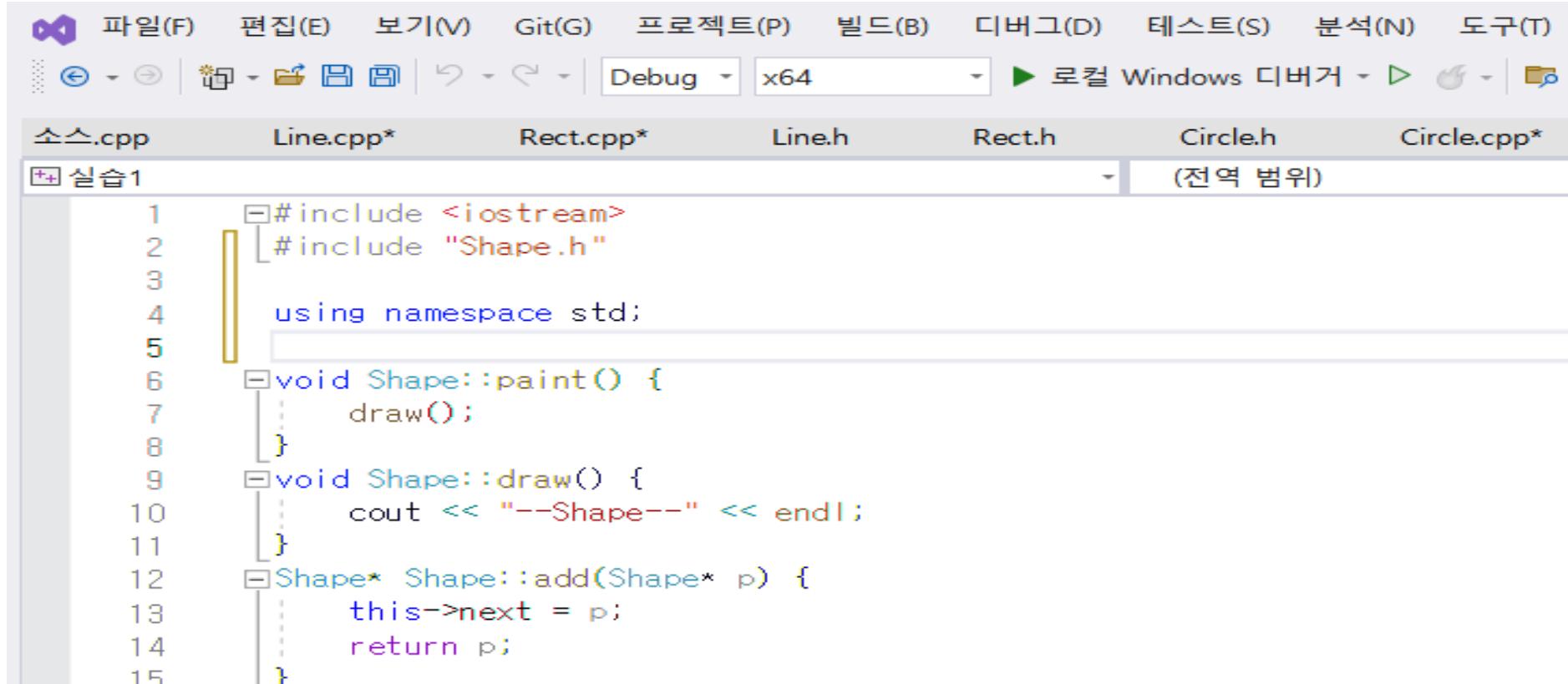


The screenshot shows the Microsoft Visual Studio IDE interface. The menu bar includes '파일(F)', '편집(E)', '보기(V)', 'Git(G)', '프로젝트(P)', '빌드(B)', '디버그(D)', '테스트(S)', '분석(N)', '도구(T)', and '확장(E)'. The toolbar contains icons for file operations like Open, Save, and Build. The status bar shows 'Debug' mode, 'x64' architecture, and '로컬 Windows 디버거' (Local Windows Debugger). The code editor displays the 'Shape.h' header file:

```
#pragma once

class Shape {
    Shape* next;
protected:
    virtual void draw();
public:
    Shape() { next = NULL; }
    virtual ~Shape() { }
    void paint();
    Shape* add(Shape* p);
    Shape* getNext() { return next; }
};
```

[2차시 원준서] ppt 20p, 22p 실습해보기 (8/9) Shape.cpp



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar includes the VS logo, 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(T), and Debug/x64 buttons. Below the title bar is a toolbar with icons for back, forward, search, and other functions. The main window displays the Shape.cpp file content. The code is as follows:

```
1 #include <iostream>
2 #include "Shape.h"
3
4 using namespace std;
5
6 void Shape::paint() {
7     draw();
8 }
9 void Shape::draw() {
10    cout << "--Shape--" << endl;
11 }
12 Shape* Shape::add(Shape* p) {
13     this->next = p;
14     return p;
15 }
```

[2차시 원준서] ppt 20p, 22p 실습해보기 (9/9) Main.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Main.cpp`. The code implements a linked list of shapes (Circle, Rectangle, Line) using a pointer-to-pointer (`Shape**`) approach. It starts by creating a `Circle` object and setting it as both `pStart` and `pLast`. Then, it adds three more shapes to the list: a `Rect`, another `Circle`, and a `Line`. After printing each shape's details in the Immediate Window, it iterates through the list and deletes each shape individually, updating the pointers to maintain the list structure.

```
#include <iostream>
#include "Shape.h"
#include "Circle.h"
#include "Rect.h"
#include "Line.h"

using namespace std;

int main() {
    Shape* pStart = NULL;
    Shape* pLast;

    pStart = new Circle(); // 처음에 원 도형을 생성한다.
    pLast = pStart;

    pLast = pLast->add(new Rect()); // 사각형 객체 생성
    pLast = pLast->add(new Circle()); // 원 객체 생성
    pLast = pLast->add(new Line()); // 선 객체 생성
    pLast = pLast->add(new Rect()); // 사각형 객체 생성
    // 현재 연결된 모든 도형을 화면에 그린다.

    Shape* p = pStart;
    while (p != NULL) {
        p->paint();
        p = p->getNext();
    }

    // 현재 연결된 모든 도형을 삭제한다.
    p = pStart;
    while (p != NULL) {
        Shape* q = p->getNext(); // 다음 도형 주소 기억
        delete p; // 기본 클래스의 가상 소멸자 호출
        p = q; // 다음 도형 주소를 p에 저장
    }
}
```

Microsoft Visual Studio 디버그 콘솔

```
Circle
Rectangle
Circle
Line
Rectangle
```

C:\Users\won2s\Desktop\액체지향프로그래밍\1주차 실습\실습1\x64\Debug\실습1.exe(프로세스 20640개)이(가) 종료
코드: 0x00000000.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 하도록 설정합니다].
이 창을 닫으려면 아무 키나 누르세요...

[3차시 이수영] 예제 9-7 안 보고 실습

(1/5) Adder.h

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Adder.h`:

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

class Adder : public Calculator {
protected:
    virtual void run();
};
```

The Solution Explorer on the right shows the project structure for `Project11sy`:

- Project11sy
- > 참조
- > 외부 종속성
- > 리소스 파일
- > 소스 파일
 - + Adder.cpp
 - + Adder.h
 - + Calculator.cpp
 - + Chap9sy.cpp
 - + Circle.cpp
 - + Circle.h
 - + Line.cpp
 - + Line.h
 - + Main.cpp
 - + Rect.cpp
 - + Rect.h
 - + Shape.cpp
 - + Shape.h
 - + Subtractor.cpp
 - + Subtractor.h
- > 헤더 파일

The status bar at the bottom shows the date and time as "2023-11-13 오후 2:22".

[3차시 이수영] 예제 9-7 안 보고 실습

(2/5) Adder.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for the `Adder.cpp` file, which contains the following code:

```
#include <iostream>
#include "Adder.h"
using namespace std;

void Adder::run() {
    cout << "Adder" << endl;
}
```

The Solution Explorer on the right side shows the project structure for `Project11sy`:

- Project11sy
- > 참조
- > 외부 종속성
- > 리소스 파일
- > 소스 파일
 - + Adder.cpp
 - + Adder.h
 - + Calculator.cpp
 - + Chap9sy.cpp
 - + Circle.cpp
 - + Circle.h
 - + Line.cpp
 - + Line.h
 - + Main.cpp
 - + Rect.cpp
 - + Rect.h
 - + Shape.cpp
 - + Shape.h
 - + Subtractor.cpp
 - + Subtractor.h
- > 헤더 파일

The status bar at the bottom indicates the following information: 144%, 9 오류, 0 경고, 0 메시지, 빌드 + IntelliSense, 9°C 맑음, 오후 2:23, 2023-11-13.

[3차시 이수영] 예제 9-7 안 보고 실습

(3/5) Subtractor.h

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "Project11sy". The menu bar includes "파일(F)", "편집(E)", "보기(V)", "Git(G)", "프로젝트(P)", "빌드(B)", "디버그(D)", "테스트(S)", "분석(N)", "도구(T)", "확장(X)", "창(W)", "도움말(H)", and "검색 -". The toolbar has icons for file operations like Open, Save, and Build. The status bar at the bottom shows "144 %", "문제가 검색되지 않음", "풀: 4 문자: 17", "CRLF", "오류 목록", "전체 솔루션", "12 오류", "0 경고", "0 메시지", "빌드 + IntelliSense", "검색 오류 목록", "솔루션 탐색기", "Git 변경 내용", "저장되었습니다.", "9°C 맑음", "오후 2:22", "2023-11-13", and a battery icon.

The code editor displays the content of Subtractor.h:

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

class Subtractor : public Calculator {
protected:
    virtual void run();
};
```

The Solution Explorer on the right shows the project structure:

- 솔루션 탐색기
- 솔루션 'Project11sy' (1 프로젝트의 1)
- Project11sy
 - 외부 종속성
 - 리소스 파일
 - 소스 파일
 - Adder.cpp
 - Adder.h
 - Calculator.cpp
 - Calculator.h
 - Chap9sy.cpp
 - Circle.cpp
 - Circle.h
 - Line.cpp
 - Line.h
 - Main.cpp
 - Rect.cpp
 - Rect.h
 - Shape.cpp
 - Shape.h
 - Subtractor.cpp
 - Subtractor.h
 - 헤더 파일

[3차시 이수영] 예제 9-7 안 보고 실습 (4/5) Subtractor.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Subtractor.cpp` within the `Project11sy` project. The code includes the `<iostream>` header, the header `"Subtractor.h"`, and a `using namespace std;` directive. It contains a single function `void Subtractor::run()` that outputs the string "Subtractor" followed by a new line character. The Solution Explorer on the right side shows the project structure with files like `Adder.cpp`, `Calculator.cpp`, `Main.cpp`, `Line.h`, `Rect.cpp`, `Circle.h`, `Shape.cpp`, `Shape.h`, and `Chap9sy.cpp`. The status bar at the bottom provides information such as the current file, line number, and build status.

```
#include <iostream>
#include "Subtractor.h"
using namespace std;

void Subtractor::run() {
    cout << "Subtractor" << endl;
}
```

[3차시 이수영] 예제 9-7 안 보고 실습 (5/5) Calculator.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Calculator.cpp`. The code defines a `Calculator` class with an `input()` method, a protected section containing variables `a` and `b`, and a `calc(int a, int b)` virtual function. It also includes a `run()` method that calls `input()` and then prints the result of `calc(a, b)`. The `main()` function creates instances of `Adder` and `Subtractor`, and calls their `run()` methods. The Solution Explorer on the right shows the project structure for `Project11sy`, including files like `Adder.h`, `Calculator.h`, and `Shape.h`.

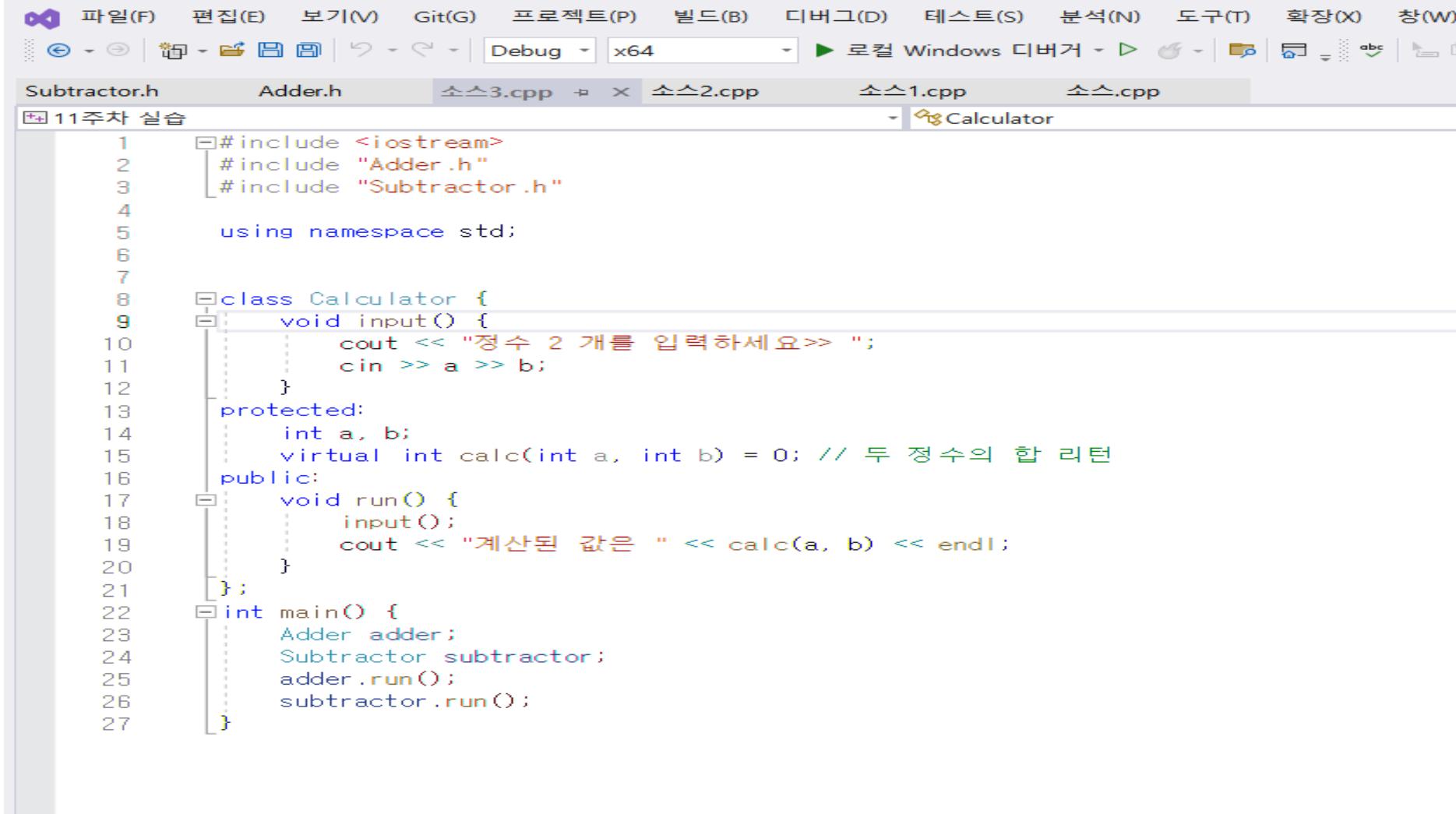
```
#include <iostream>
#include "Adder.h"
#include "Subtractor.h"
using namespace std;

class Calculator {
public:
    void input() {
        cout << "정수 2 개를 입력하세요>> ";
        cin >> a >> b;
    }
protected:
    int a, b;
    virtual int calc(int a, int b) = 0; // 두 정수의 합 리턴
public:
    void run() {
        input();
        cout << "계산된 값은 " << calc(a, b) << endl;
    }
};

int main() {
    Adder adder;
    Subtractor subtractor;
    adder.run();
    subtractor.run();
}
```

[3차시 원준서] 예제 9-7 안 보고 실습

(1/5) Calculator.cpp



The screenshot shows the Microsoft Visual Studio IDE interface with the following details:

- Menu Bar:** 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(T), 확장(X), 창(W)
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Run.
- Toolbox:** Includes icons for Solution Explorer, Properties, Task List, etc.
- Status Bar:** Debug, x64, 로컬 Windows 디버거.
- Code Editor:** The active tab is "소스3.cpp". Other tabs include "소스2.cpp", "소스1.cpp", and "소스.cpp".
The code itself is as follows:

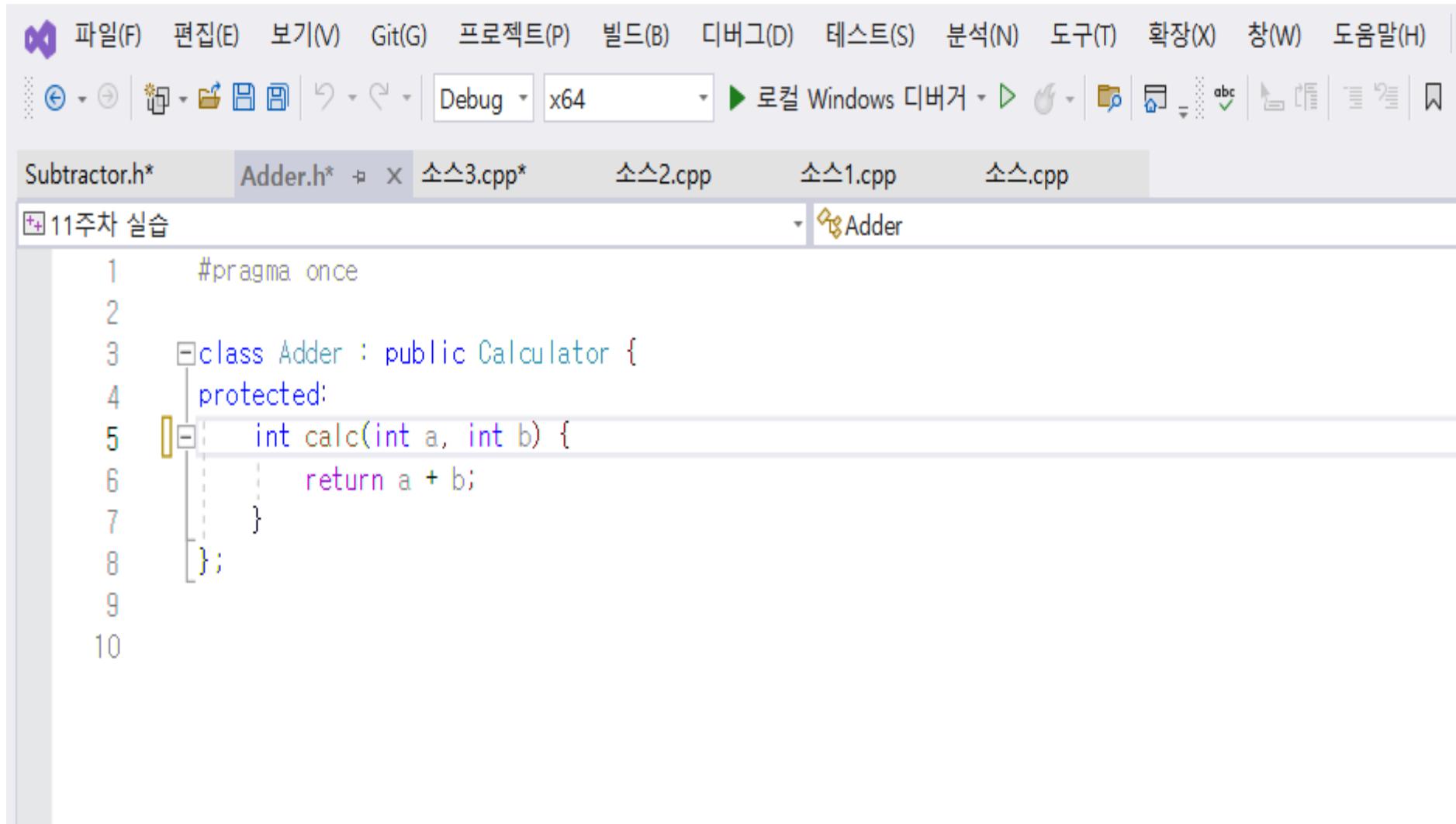
```
#include <iostream>
#include "Adder.h"
#include "Subtractor.h"

using namespace std;

class Calculator {
public:
    void input() {
        cout << "정수 2 개를 입력하세요>> ";
        cin >> a >> b;
    }
protected:
    int a, b;
    virtual int calc(int a, int b) = 0; // 두 정수의 합 리턴
public:
    void run() {
        input();
        cout << "계산된 값은 " << calc(a, b) << endl;
    }
};

int main() {
    Adder adder;
    Subtractor subtractor;
    adder.run();
    subtractor.run();
}
```

[3차시 원준서] 예제 9-7 안 보고 실습 (2/5) Adder.h



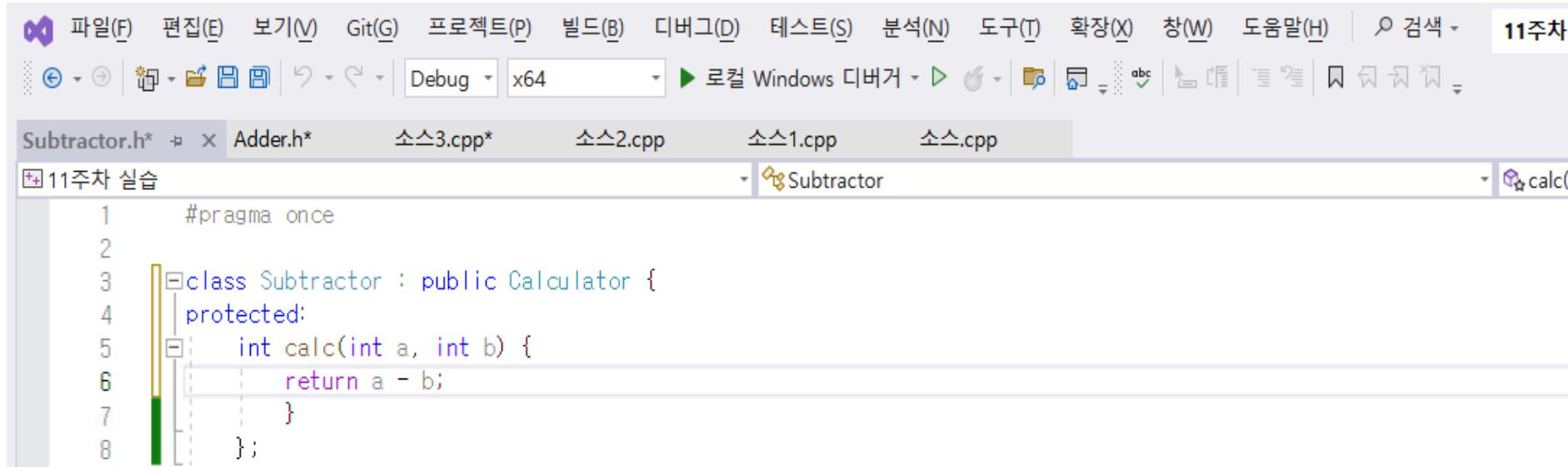
The screenshot shows the Microsoft Visual Studio IDE interface. The title bar includes the standard menu items: 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(T), 확장(X), 창(W), 도움말(H). Below the title bar is the toolbar with various icons. The solution explorer shows files: Subtractor.h*, Adder.h*, 소스3.cpp*, 소스2.cpp, 소스1.cpp, and 소스.cpp. The current file being edited is Adder.h*. The code editor displays the following C++ code:

```
1 #pragma once
2
3 class Adder : public Calculator {
4 protected:
5     int calc(int a, int b) {
6         return a + b;
7     }
8 }
9
10
```

The code editor has syntax highlighting and code folding. The line numbers 1 through 10 are visible on the left. The class definition for Adder is expanded, showing its inheritance from Calculator and its protected member function calc which returns the sum of two integers.

[3차시 원준서] 예제 9-7 안 보고 실습

(3/5) Subtractor.h



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "11주차". The toolbar includes standard icons for file operations, Git, and debugging. The menu bar has Korean labels: 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(T), 확장(X), 창(W), 도움말(H). The solution explorer shows files: Subtractor.h*, Adder.h*, 소스3.cpp*, 소스2.cpp, 소스1.cpp, and 소스.cpp. The current file is Subtractor.h*. The code editor displays the following C++ code:

```
1 #pragma once
2
3 class Subtractor : public Calculator {
4 protected:
5     int calc(int a, int b) {
6         return a - b;
7     }
8 }
```

[3차시 원준서] 예제 9-7 안 보고 실습 (4/5) Subtractor.h

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Subtractor.h`:

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

class Subtractor : public Calculator {
protected:
    virtual void run();
};
```

The Solution Explorer on the right side shows the project structure:

- Project11sy (1 프로젝트의 1)
 - Project11sy
 - 소스 파일
 - Adder.cpp
 - Adder.h
 - Calculator.cpp
 - Chap9sy.cpp
 - Circle.cpp
 - Circle.h
 - Line.cpp
 - Line.h
 - Main.cpp
 - Rect.cpp
 - Rect.h
 - Shape.cpp
 - Shape.h
 - Subtractor.cpp
 - 헤더 파일

The status bar at the bottom provides system information: 9°C 맑음, 오후 2:22, 2023-11-13.

[3차시 원준서] 예제 9-7 안 보고 실습 (5/5) Subtractor.cpp

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `Subtractor.cpp` within the `Project11sy` project. The code contains a single function `run()` that outputs "Subtractor" to the console.

```
#include <iostream>
#include "Subtractor.h"
using namespace std;

void Subtractor::run() {
    cout << "Subtractor" << endl;
}
```

The Solution Explorer on the right side of the interface shows the project structure, including files like `Adder.h`, `Calculator.cpp`, `Main.cpp`, and several `Shape` and `Rect` files.

[정리노트 원준서&이수영] 가상 함수와 오버라이딩

- 원: 가상 함수는 `virtual` 키워드로 선언된 멤버 함수로, 동적 바인딩 지시어이며, 컴파일러에게 함수에 대한 호출 바인딩을 실행 시간까지 미루도록 지시합니다. 함수 오버라이딩은 파생 클래스에서 기본 클래스의 가상 함수와 동일한 이름의 함수를 선언합니다. 기본 클래스의 가상 함수의 존재감을 상실시키고, 파생 클래스에서 오버라이딩한 함수가 호출되도록 동적 바인딩합니다. 함수 재정의라고도 부르며, 다형성의 한 종류입니다.
- 이: `virtual` 키워드가 있는 멤버 함수는 런타임 시에 바인딩되는 반면, `virtual` 키워드가 없는 멤버 함수는 컴파일 시 바인딩된다. 자바는 다 런타임 시에 동적 바인딩되는 점이 C++과 다르다.

[정리노트 원준서&이수영] 함수 재정의와 오버라이딩 용어의 혼란 정리

- 원: 함수 재정의라는 용어를 사용할 때는 신중을 가해야 합니다. 가상 함수를 재정의하는 경우와 아닌 경우에 따라 프로그램의 실행이 완전히 달라지기 때문입니다. 가상 함수를 재정의하는 오버라이딩의 경우 함수가 호출되는 실행 시간에 동적 바인딩이 일어나지만, 그렇지 않은 경우 컴파일 시간에 결정된 함수가 단순히 호출됩니다.(정적 바인딩)
- 이: 함수 재정의는 기본 클래스 멤버, 파생 클래스 멤버가 각각 컴파일 시에 호출되므로 Base클래스도 호출될 수 있는 반면에, 오버 라이딩은 기본 클래스 멤버가 존재감을 잃어 파생 클래스 멤버만 호출되므로 Base클래스는 호출될 수 없는 점이 다르다. 함수 재정의와 오버라이딩 모두 함수 이름, 매개변수 개수/타입이 같지만, virtual 키워드 유무로 구분된다. 오버라이딩은 virtual 키워드가 있다. 반면 함수 재정의는 virtual 키워드가 없다.

[정리노트 원준서&이수영]

동적 바인딩

- 원: 동적 바인딩은 파생 클래스에 대해 기본 클래스에 대한 포인터로 가상 함수를 호출하는 경우 객체 내에 오버라이딩한 파생 클래스의 함수를 찾아 실행합니다. 실행 중에 이루어지며, 실행시간 바인딩, 런타임 바인딩, 늦은 바인딩으로 불립니다.
- 이: 기본 클래스는 인터페이스 역할을 하고, 파생 클래스에 대해 동적 바인딩되면, 파생 클래스 멤버 함수가 런타임 시 실행된다.
'Shape *pShape =' 부분은 동일하고, new 뒤에 어떤 파생 클래스명이 나오느냐에 따라 그 파생 클래스의 멤버 함수가 동적 바인딩에 의해 호출될 것입니다. 예를 들어 Shape 클래스를 상속받은 Circle 클래스의 함수를 실행하고 싶으면 new Circle();로 업캐스팅하면 된다. 반면 Rect 클래스 함수를 실행하고 싶다면 new Rect(); 하면 된다.

[정리노트 원준서&이수영] C++ 오버라이딩의 특징

- 원: 오버라이딩의 성공 조건은 가상 함수 이름, 매개 변수 타입과 개수, 리턴 타입이 모두 일치해야 합니다. 오버라이딩 시 virtual 지시어 생략이 가능합니다. 가상 함수의 virtual 지시어는 상속되며, 파생 클래스에서 virtual 생략 가능합니다. 가상 함수의 접근 지정으로는 private, protected, public 중 자유롭게 지정이 가능합니다.
- 이: 오버라이딩은 함수명, 매개변수 타입/개수, 리턴타입이 같고, 기본 클래스 멤버함수가 아닌 파생 클래스 멤버함수 앞에 virtual이 생략 가능하다는 특징이 있다. 오버라이딩과 달리 오버로딩은 함수명, 리턴 타입이 같아야 되지만 매개변수 타입/개수가 다를 수 있습니다.

[정리노트 원준서&이수영] 오버라이딩과 범위 지정 연산자(:)

- 원: 범위 지정 연산자(:)는 정적 바인딩을 지시하며, 기본클래스::가상함수() 형태로 기본 클래스의 가상 함수를 정적 바인딩으로 호출합니다.
- 이: 원래 파생 클래스의 멤버 함수만 실행되는데, 만약 기본 클래스의 멤버 함수가 실행되길 원한다면, 기본클래스::가상함수();를 작성하면 된다. 파생 클래스 Circle의 draw()뿐만 아니라 기본 클래스 Shape의 draw()도 실행되도록 조작할 수 있는 것이다. 런타임 시의 동적 바인딩이 아니라 컴파일 시에 정적 바인딩이 가능하도록 하는 것이 바로 범위 지정 연산자의 역할이 된다.

[정리노트 원준서&이수영] 가상 소멸자

- 원: 가상 소멸자는 virtual 키워드로 선언하며, 소멸자 호출 시 동적 바인딩이 발생합니다.
- 이: 파생 클래스의 소멸자를 선언하지 않으면, 파생 클래스는 곧 좀비가 되어 메모리만 차지하고 만다. 특히 '가상' 소멸자이므로 virtual 키워드를 앞에 붙인다. 즉, virtual ~기본 클래스명(); virtual ~파생 클래스명();의 형태다. 특히 virtual ~파생 클래스명();은 자신의 코드를 실행(파생 클래스 소멸자 실행)된 이후에 기본 클래스의 소멸자를 호출하는 기능도 포함한다. 즉, 파생 클래스 소멸자가 실행되면 자연스럽게 기본 소멸자도 실행된다.

[정리노트 원준서&이수영] 오버로딩, 함수 재정의, 오버라이딩 비교

- 원: **오버로딩**의 정의는 매개 변수 타입이나 개수가 다르지만, 이름이 같은 함수들이 중복 작성되는 것입니다. 클래스의 멤버들 사이, 외부 함수들 사이, 그리고 기본 클래스와 파생 클래스 사이에 존재 가능합니다. 목적으로는 이름이 같은 여러 개의 함수를 중복 작성하여, 사용 편의성을 향상시킵니다. 정적 바인딩이며, 컴파일 시에 중복된 함수들의 호출을 구분합니다.

함수 재정의는 기본 클래스의 멤버 함수를 파생 클래스에서 이름, 매개 변수 타입과 개수, 리턴 타입까지 완벽히 같은 원형으로 재작성하는 것입니다. 상속 관계이며, 기본 클래스의 멤버 함수와 별도로 파생 클래스에서 필요하여 재작성합니다. 정적 바인딩이며, 컴파일 시에 함수의 호출을 구분합니다.

오버라이딩은 기본 클래스의 가상 함수를 파생 클래스에서 이름, 매개 변수 타입과 개수, 리턴 타입까지 완벽히 같은 원형으로 재작성하는 것입니다. 상속 관계이며, 기본 클래스에 구현된 가상 함수를 무시하고, 파생 클래스에서 새로운 기능으로 재작성하고자 합니다. 동적 바인딩이며, 실행 시간에 오버라이딩된 함수를 찾아 실행합니다.

[정리노트 원준서&이수영] 오버로딩, 함수 재정의, 오버라이딩 비교

- 이: 오버로딩, 함수 재정의, 오버라이딩은 '다형성'을 실제로 구현된다.
오버로딩은 함수명은 같고, **매개변수 타입/개수는 다르고**, 리턴 타입은 상관없이 중복 작성된다. **상속 관계와 무관하다**. 컴파일 시에 호출되는 정적 바인딩이다.
함수 재정의는 함수명도 같고, 매개변수 타입/개수도 같고, 리턴 타입도 같게 해서 중복 작성된다. 상속 관계와 관련된다. 컴파일 시에 호출되는 정적 바인딩이다. 기본 클래스 기능에 +a(알파)가 더해진다.
오버라이딩은 함수명도 같고, 매개변수 타입/개수도 같고, 리턴 타입도 같게 해서 중복 작성된다. 상속 관계와 관련된다. **런타임** 시에 호출되는 **동적 바인딩**이다. 기본 클래스는 인터페이스 역할만 한다.

[정리노트 원준서&이수영] 가상함수 오버라이딩

- 원: 파생 클래스마다 다르게 구현하는 다형성

```
ex) void Circle::draw() { cout << "Circle" << endl; }
```

```
void Rect::draw() { cout << "Rectangle" << endl; }
```

```
void Line::draw() { cout << "Line" << endl; }
```

파생 클래스에서 가상 함수 draw()의 재정의.

동적 바인딩에 의해 어떤 경우에도 자신이 만든 draw()가 호출됨을 보장받습니다.

- 이: 기본 클래스의 가상 함수는 인터페이스 역할을 하고, 파생 클래스의 가상 함수는 재정의된다. C++로 연결리스트를 만들 수 있다.

[정리노트 원준서&이수영] 기본 클래스의 포인터 활용

- 원: 기본 클래스의 포인터로 파생 클래스에 접근합니다.
ex) pStart, pLast, p의 타입이 Shape*
링크드 리스트를 따라 Shape을 상속받은 파생 객체들 접근
p->paint()의 간단한 호출로 파생 객체에 오버라이딩된 draw() 함수 호출
- 이: next로 링크되는 방식으로 파생 클래스 함수가 연달아 호출된다.
기본 클래스 paint()가 호출되면, 동적 바인딩에 의해 파생 클래스의 draw()가 각각의 파생 클래스 특성에 맞게 그려지게 될 것이다.
(원/직사각형/선의 형태로 각기 다르게)

[정리노트 원준서]

추상 클래스

- 원: 기본 클래스의 가상 함수 목적은 파생 클래스에서 재정의할 함수를 알려주는 역할이며, 실행할 코드를 작성할 목적이 아닙니다. 순수 가상 함수는 함수의 코드가 없고 선언만 있는 가상 멤버 함수이며, 선언 방법은 멤버 함수의 원형=0;으로 선언합니다. 추상 클래스는 최소한 하나의 순수 가상 함수를 가진 클래스입니다. 특징으로는 온전한 클래스가 아니므로 객체 생성이 불가능하며, 추상 클래스의 포인터는 선언 가능합니다. 추상 클래스의 목적은 추상 클래스의 인스턴스를 생성할 목적이 아니라, 상속에서 기본 클래스의 역할을 하기 위함입니다. 순수 가상 함수를 통해 파생 클래스에서 구현할 함수의 형태(원형)을 보여주는 인터페이스 역할로, 추상 클래스의 모든 멤버 함수를 순수 가상 함수로 선언할 필요 없습니다. 추상 클래스의 상속은 추상 클래스를 단순 상속하면 자동 추상 클래스이며, 추상 클래스의 구현은 추상 클래스를 상속받아 순수 가상 함수를 오버라이딩합니다. 파생 클래스는 추상 클래스가 아닙니다.
- 이: '순수' 가상 함수는 앞선 '가상 함수'와 달리 코드가 없이 원형=0;으로 선언돼서 정말 인터페이스(API) 역할만 하는 것이다. '순수' 가상 함수가 1개 이상 존재하기만 하면 추상 클래스가 된다. '순수' 가상 함수가 아닌 함수가 있어도 '순수' 가상 함수가 1개라도 존재하면 추상 클래스다. 추상 클래스가 아닌 클래스는 객체가 생성되나, 추상 클래스 객체는 생성할 수 없고 컴파일 오류가 난다. 다만, 순수 가상 함수를 (코드를) '구현'하면, 객체를 생성할 수 있다는 점에 유의하자.