

13주차 정리노트

note#8

2023. 11. 27(월)
16조 이수영, 원준서

목차

실습(ppt p3-p19)

- [1차시] cplusplus IOS
- [1차시] 11-1
- [1차시] 11-2
- [1차시] 11-4
- [2차시] 11-5
- [2차시] 11-7
- [2차시] 11-9
- [2차시] 11-11

정리노트(ppt p20-p40)

- C++ 입출력 기초
- C++ 입출력 스트림 버퍼
- C++ 표준은 스트림 입출력만 지원
- 2003년 이전의 C++ 라이브러리 약점
- 입출력 클래스
- C++ 표준 입출력 스트림 객체
- 문자열 입력
- 포맷 입출력
- 조작자
- 삽입 연산자(<<)와 추출 연산자(>>)

[1차시 이수영]

cplusplus Input/Output 계층구조 확인

Reference : Input/Output

library

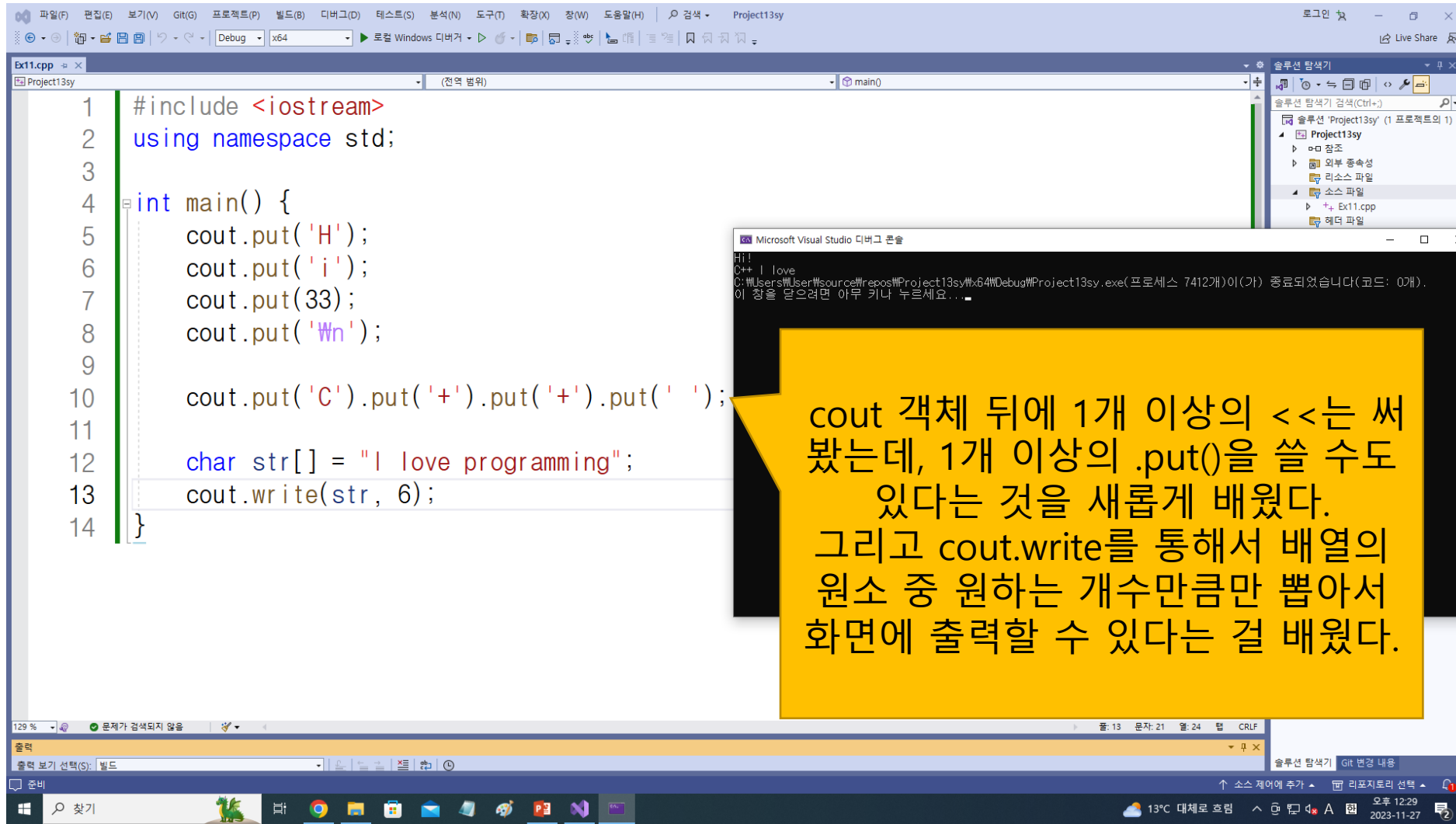
Input/Output

Input/Output library

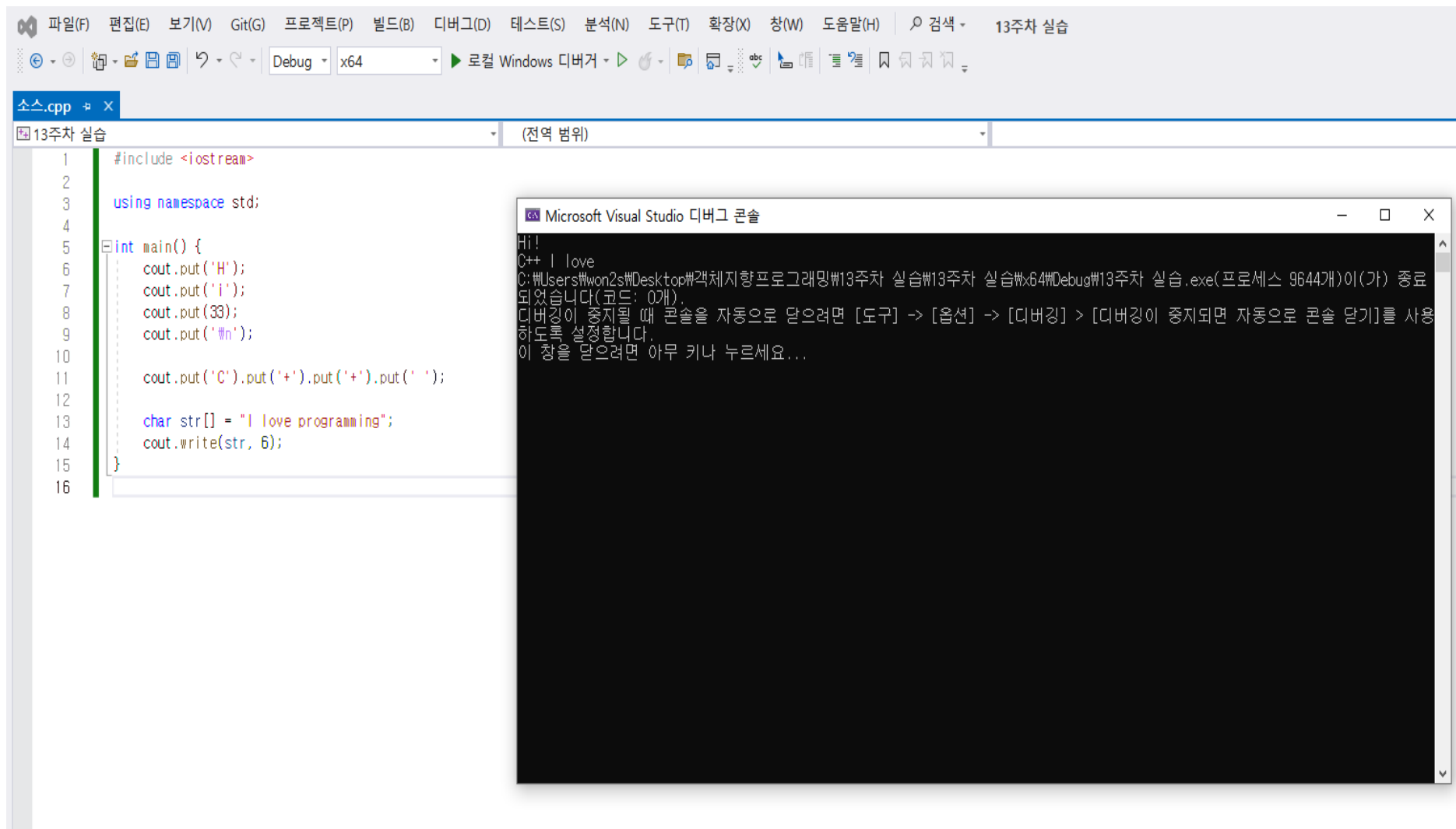
The iostream library is an object-oriented library that provides input and output functionality using streams.

A stream is an abstraction that represents a device on which input and output operations are performed. A stream can basically be represented as a sequence of characters of indefinite length.

[1차시 이수영] 11-1 실습



[1차시 원준서] 11-1 실습

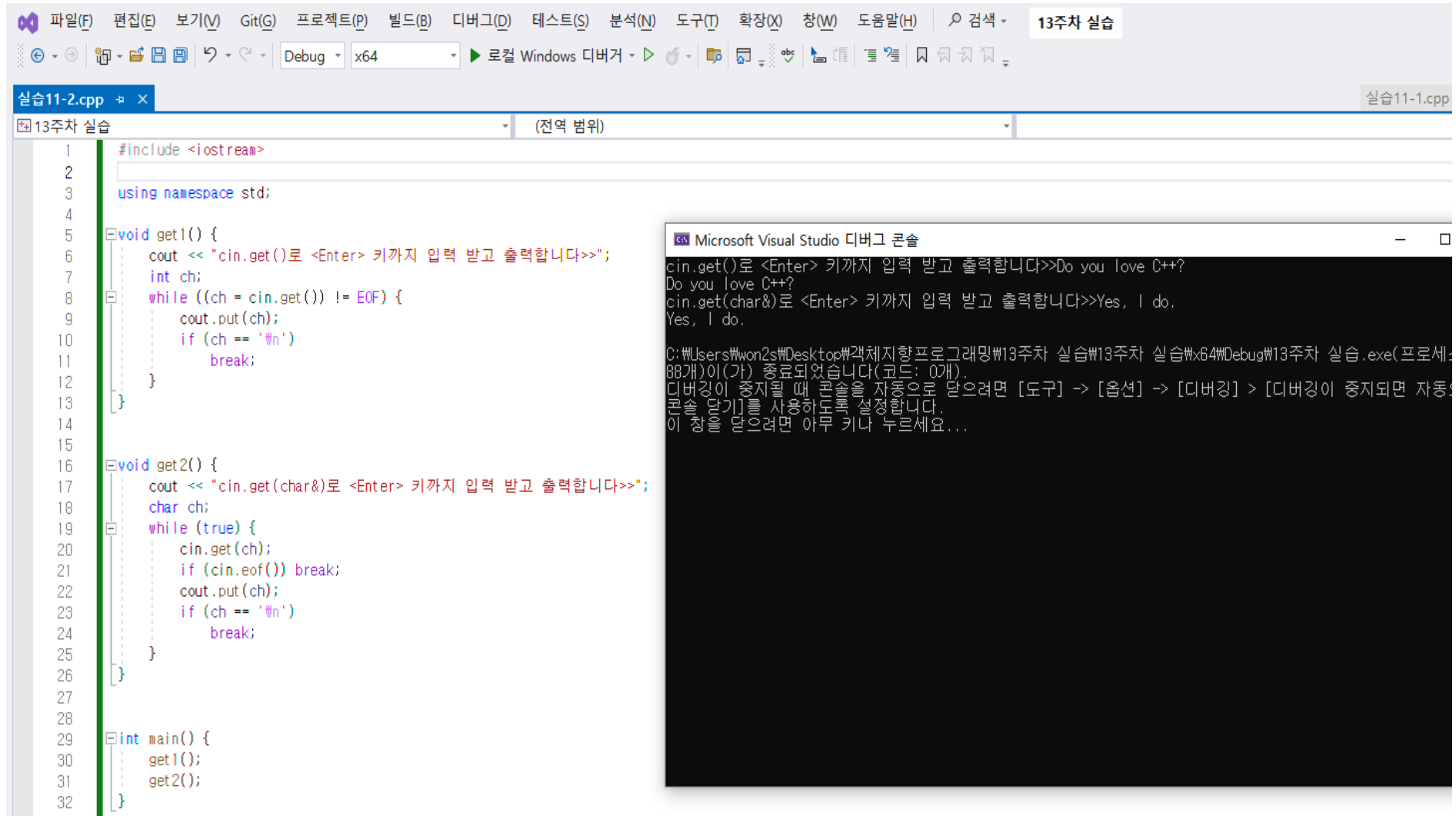


[1차시 이수영] 11-2 안 보고 실습

```
1  #include <iostream>
2  using namespace std;
3
4  void get1() {
5      char ch;
6      while (ch != EOF)
7          cin.get(ch);
8  }
9
10 void get2() {
11     char &char;
12     while (&char != EOF)
13         cin.get(&char);
14 }
15
16 // main()만 copy, get1(), get2() 안 보고 구현
17 int main() {
18     get1(); // cin.get()을 이용하는 사례
19     get2(); // cin.get(char&)을 이용하는 사례
20 }
```

[5] EOF와 비교하려면 ch를 char가 아니라 int 타입으로 선언해야 했다.
[6] while () 안에 ch=cin.get()를 모두 써줘야했는데, ch가 EOF가 아니면 ch를 cin.get(ch)한다고 입력했다. 코드를 작성하기 전에 어떤 흐름으로 진행될 지 충분히 고려해야 할 것. 시간에 쫓기지 말고 충분히 고민하자.

[1차시 원준서] 11-2 안 보고 실습



The image shows a Visual Studio IDE with a C++ file named '실습11-2.cpp'. The code defines two functions, `get1()` and `get2()`, and a `main()` function. `get1()` uses `cin.get()` to read characters until an Enter key is pressed. `get2()` uses `cin.get(char&)` to read characters until an Enter key is pressed. The `main()` function calls both `get1()` and `get2()`.

```
1 #include <iostream>
2
3 using namespace std;
4
5 void get1() {
6     cout << "cin.get()로 <Enter> 키까지 입력 받고 출력합니다>>";
7     int ch;
8     while ((ch = cin.get()) != EOF) {
9         cout.put(ch);
10        if (ch == '\n')
11            break;
12    }
13 }
14
15
16 void get2() {
17     cout << "cin.get(char&)로 <Enter> 키까지 입력 받고 출력합니다>>";
18     char ch;
19     while (true) {
20         cin.get(ch);
21         if (cin.eof()) break;
22         cout.put(ch);
23         if (ch == '\n')
24             break;
25     }
26 }
27
28
29 int main() {
30     get1();
31     get2();
32 }
```

The debug console window shows the output of the program. It displays the prompts from `get1()` and `get2()`, followed by the user input "Do you love C++?" and "Yes, I do.".

```
Microsoft Visual Studio 디버그 콘솔
cin.get()로 <Enter> 키까지 입력 받고 출력합니다>>Do you love C++?
Do you love C++?
cin.get(char&)로 <Enter> 키까지 입력 받고 출력합니다>>Yes, I do.
Yes, I do.
C:\Users\won2s\Desktop\객체지향프로그래밍\13주차 실습\13주차 실습\Debug\13주차 실습.exe(프로세
88개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동
콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

[1차시 이수영] 11-4 get() vs getline() 비교

(1) get()

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      char line[80];
6      cout << "cin.get() 함수로 라인을 읽습니다." << endl;
7      cout << "exit를 입력하면 루프가 끝납니다." << endl;
8
9      int no = 1; // 라인 번호
10     while (true) {
11         cout << "라인 " << no << " >> ";
12         cin.get(line, 80); // 79개까지의 문자 읽음
13         if (strcmp(line, "exit") == 0)
14             break;
15         cout << "echo --> ";
16         cout << line << endl; // 읽은 라인을 화면에 출력
17         no++; // 라인 번호 증가
18     }
19 }
```

출력

```
빌드 시작...
1>----- 빌드 시작: 프로젝트: Project13sy, 구성: Debug x64 -----
1>Project13sy.vcxproj -> C:\Users\User\source\repos\Project13sy\Debug\Project13sy.exe
===== 빌드: 1개 성공, 0개 실패, 0개 최신 상태, 0개 건너뛴 =====
===== 빌드(가) 오후 1:00에 시작되었고 00.147 초(가) 소요됨 =====
```

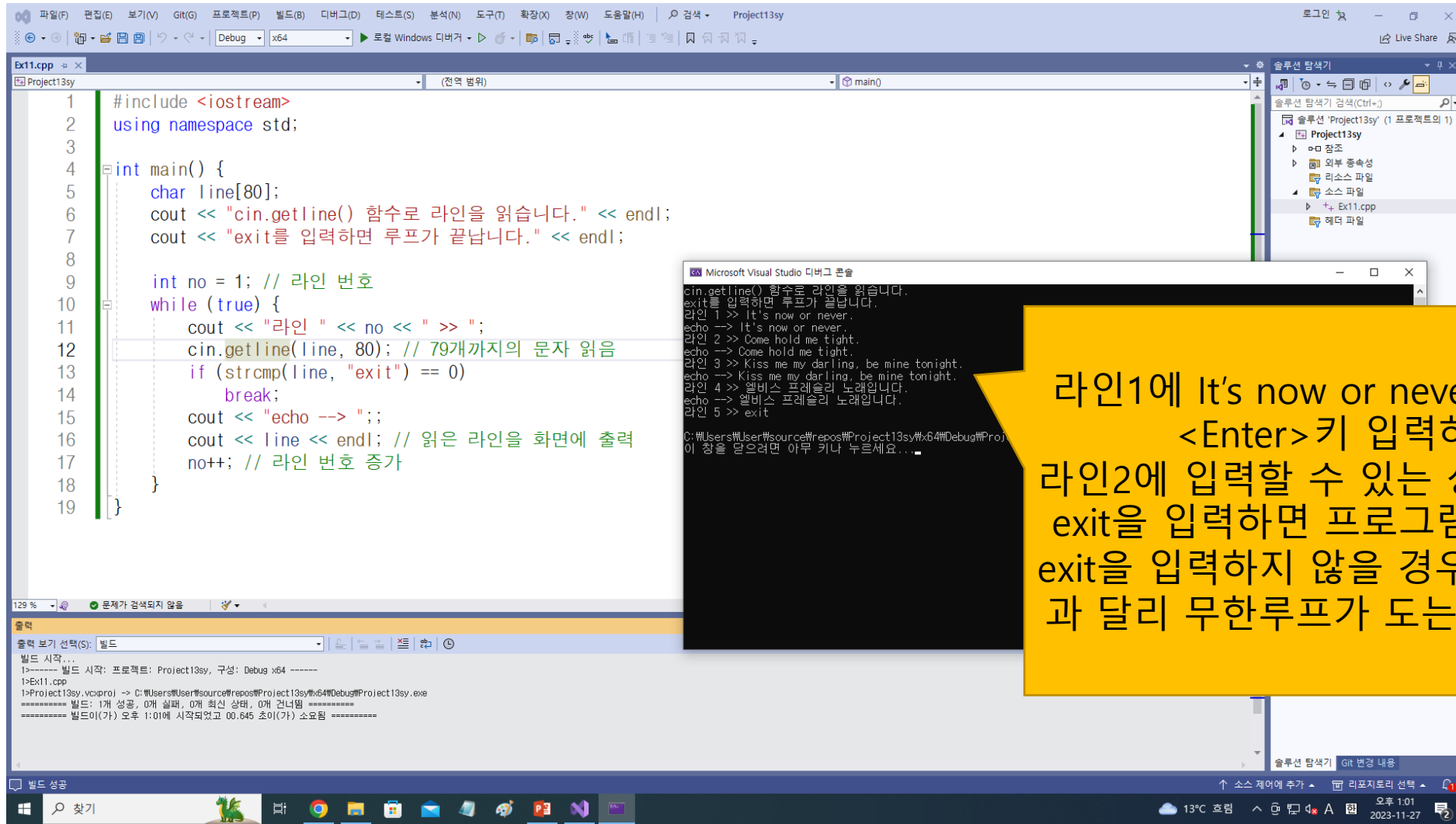
출력

```
37008 >> echo -->
37009 >> echo -->
37010 >> echo -->
37011 >> echo -->
37012 >> echo -->
37013 >> echo -->
37014 >> echo -->
37015 >> echo -->
37016 >> echo -->
37017 >> echo -->
37018 >> echo -->
37019 >> echo -->
37020 >> echo -->
37021 >> echo -->
37022 >> echo -->
37023 >> echo -->
37024 >> echo -->
37025 >> echo -->
37026 >> echo -->
37027 >> echo -->
37028 >> echo -->
37029 >> echo -->
37030 >> echo -->
37031 >> echo -->
37032 >> echo -->
37033 >> echo -->
37034 >> echo -->
37035 >> echo -->
37036 >> echo -->
37037 >> echo -->
```

라인1에 It's now or never. 입력하고 <Enter> 키 입력하자마자 라인 2부터 무한루프가 돌아 더 입력 불가능한 상태가 되어버린다.

[1차시 이수영] 11-4 get() vs getline() 비교

(2) getline()



```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     char line[80];
6     cout << "cin.getline() 함수로 라인을 읽습니다." << endl;
7     cout << "exit를 입력하면 루프가 끝납니다." << endl;
8
9     int no = 1; // 라인 번호
10    while (true) {
11        cout << "라인 " << no << " >> ";
12        cin.getline(line, 80); // 79개까지의 문자 읽음
13        if (strcmp(line, "exit") == 0)
14            break;
15        cout << "echo --> ";
16        cout << line << endl; // 읽은 라인을 화면에 출력
17        no++; // 라인 번호 증가
18    }
19 }
```

Microsoft Visual Studio 디버그 콘솔

```
cin.getline() 함수로 라인을 읽습니다.
exit를 입력하면 루프가 끝납니다.
라인 1 >> It's now or never.
echo --> It's now or never.
라인 2 >> Come hold me tight.
echo --> Come hold me tight.
라인 3 >> Kiss me my darling, be mine tonight.
echo --> Kiss me my darling, be mine tonight.
라인 4 >> 엘비스 프레슬리 노래입니다.
echo --> 엘비스 프레슬리 노래입니다.
라인 5 >> exit
C:\Users\User\source\repos\Project13sy\Debug\Project13sy.exe
이 창을 알으려면 아무 키나 누르세요...
```

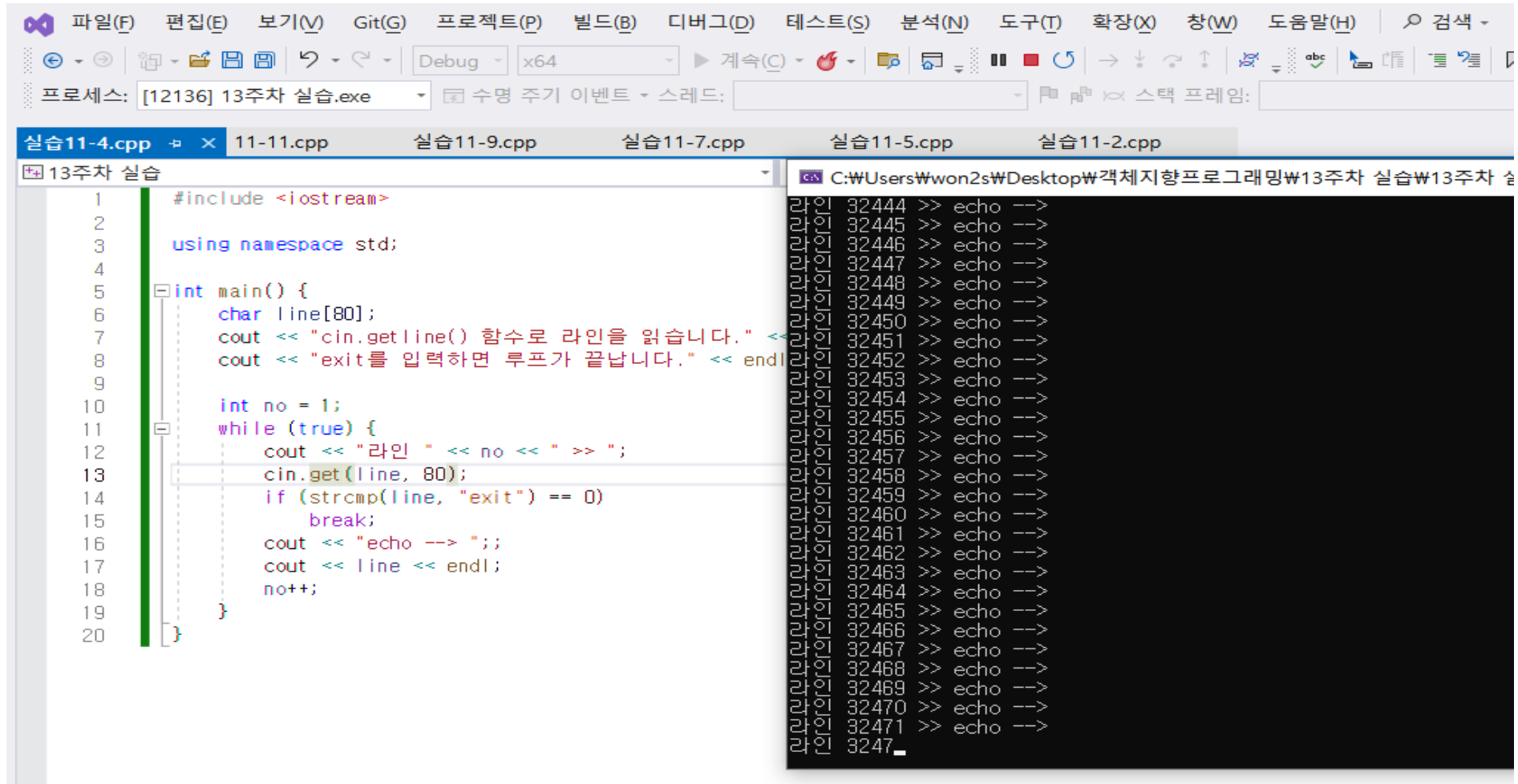
출력

```
빌드 시작...
1>----- 빌드 시작: 프로젝트: Project13sy, 구성: Debug x64 -----
1>Ex11.cpp
1>Project13sy.vcxproj -> C:\Users\User\source\repos\Project13sy\Debug\Project13sy.exe
===== 빌드: 1개 성공, 0개 실패, 0개 최신 상태, 0개 건너뛴 =====
===== 빌드이(가) 오후 1:01에 시작되었고 00.645 초이(가) 소요됨 =====
```

라인1에 It's now or never. 입력하고
<Enter>키 입력하면
라인2에 입력할 수 있는 상태가 된다.
exit을 입력하면 프로그램이 끝나며
exit을 입력하지 않을 경우 앞선 get()
과 달리 무한루프가 도는 일은 없다.

[1차시 원준서] 11-4 get() vs getline() 비교

(1) get()



The screenshot displays the Visual Studio IDE with a C++ project. The code in `11-11.cpp` uses `cin.getline()` to read input lines. The console window shows the program's execution, including prompts and user input.

```
#include <iostream>

using namespace std;

int main() {
    char line[80];
    cout << "cin.getline() 함수로 라인을 읽습니다." << endl;
    cout << "exit를 입력하면 루프가 끝납니다." << endl;

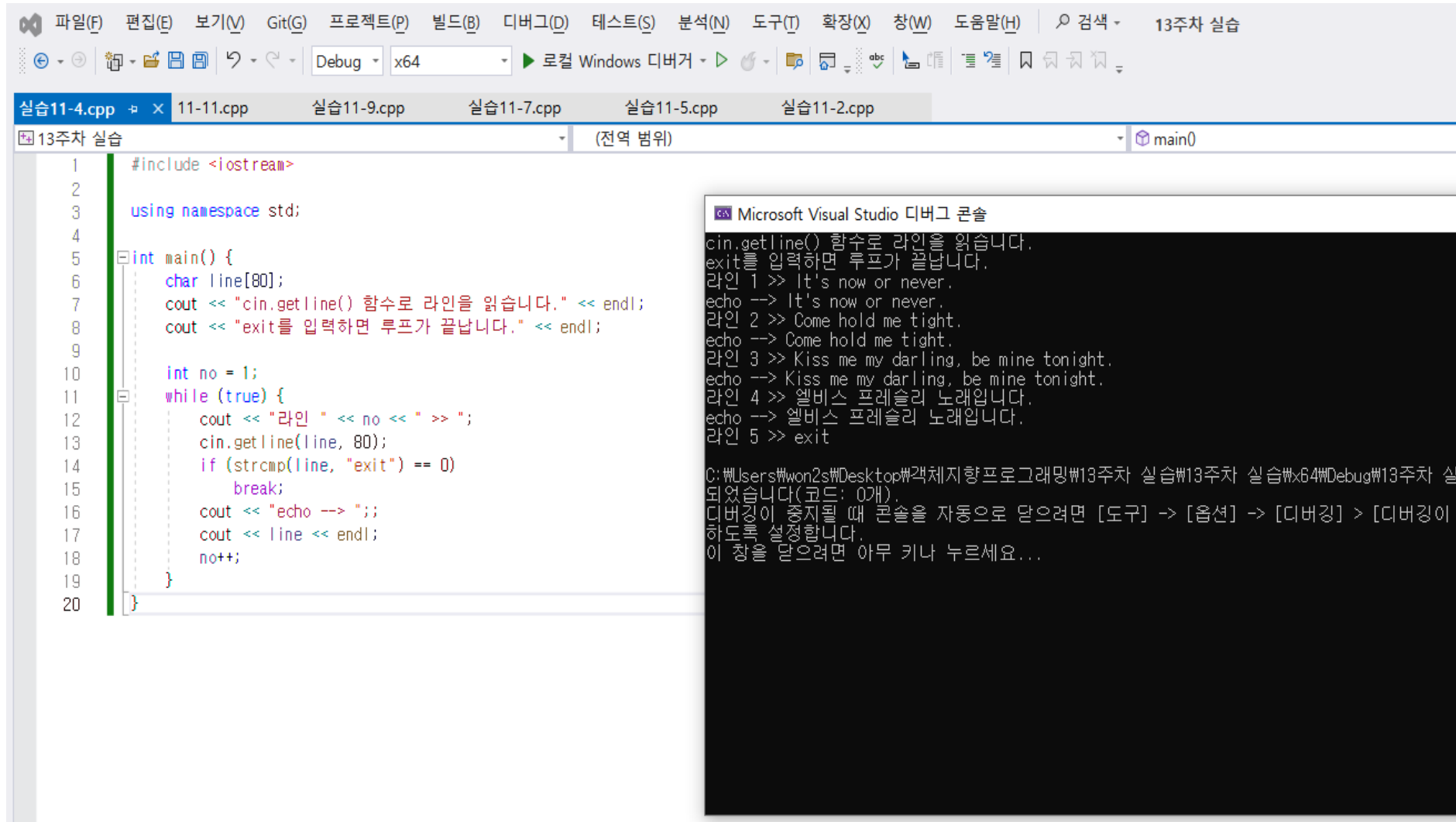
    int no = 1;
    while (true) {
        cout << "라인 " << no << " >> ";
        cin.getline(line, 80);
        if (strcmp(line, "exit") == 0)
            break;
        cout << "echo --> ";
        cout << line << endl;
        no++;
    }
}
```

Console Output:

```
라인 32444 >> echo -->
라인 32445 >> echo -->
라인 32446 >> echo -->
라인 32447 >> echo -->
라인 32448 >> echo -->
라인 32449 >> echo -->
라인 32450 >> echo -->
라인 32451 >> echo -->
라인 32452 >> echo -->
라인 32453 >> echo -->
라인 32454 >> echo -->
라인 32455 >> echo -->
라인 32456 >> echo -->
라인 32457 >> echo -->
라인 32458 >> echo -->
라인 32459 >> echo -->
라인 32460 >> echo -->
라인 32461 >> echo -->
라인 32462 >> echo -->
라인 32463 >> echo -->
라인 32464 >> echo -->
라인 32465 >> echo -->
라인 32466 >> echo -->
라인 32467 >> echo -->
라인 32468 >> echo -->
라인 32469 >> echo -->
라인 32470 >> echo -->
라인 32471 >> echo -->
라인 3247_
```

[1차시 원준서] 11-4 get() vs getline() 비교

(2) getline()



The image shows a screenshot of the Microsoft Visual Studio IDE. The main window displays a C++ source file named '실습11-4.cpp'. The code uses `cin.getline()` to read input lines. The debug console on the right shows the program's execution, including prompts and user input.

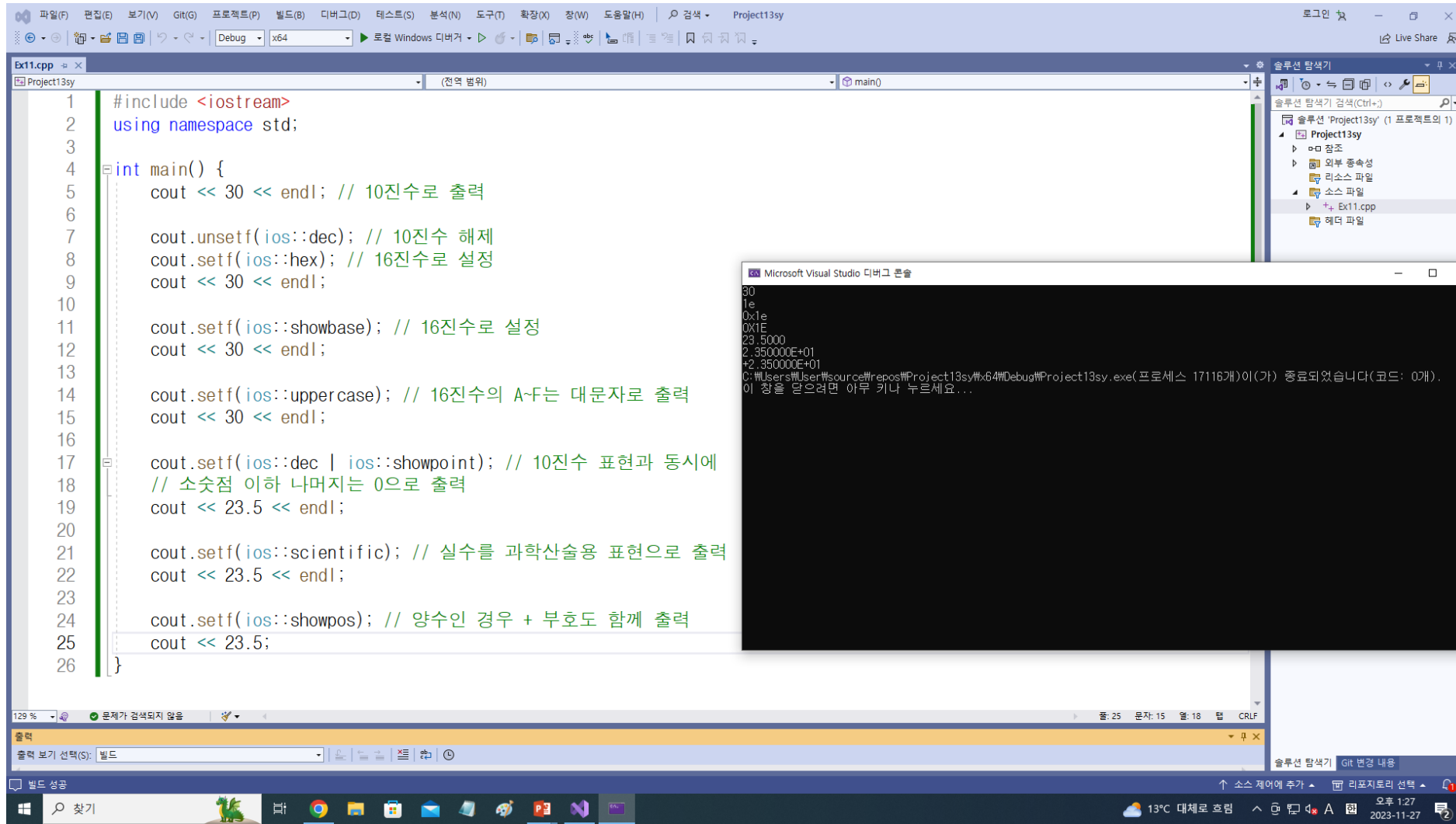
```
1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     char line[80];
7     cout << "cin.getline() 함수로 라인을 읽습니다." << endl;
8     cout << "exit를 입력하면 루프가 끝납니다." << endl;
9
10    int no = 1;
11    while (true) {
12        cout << "라인 " << no << " >> ";
13        cin.getline(line, 80);
14        if (strcmp(line, "exit") == 0)
15            break;
16        cout << "echo --> ";
17        cout << line << endl;
18        no++;
19    }
20 }
```

Microsoft Visual Studio 디버그 콘솔

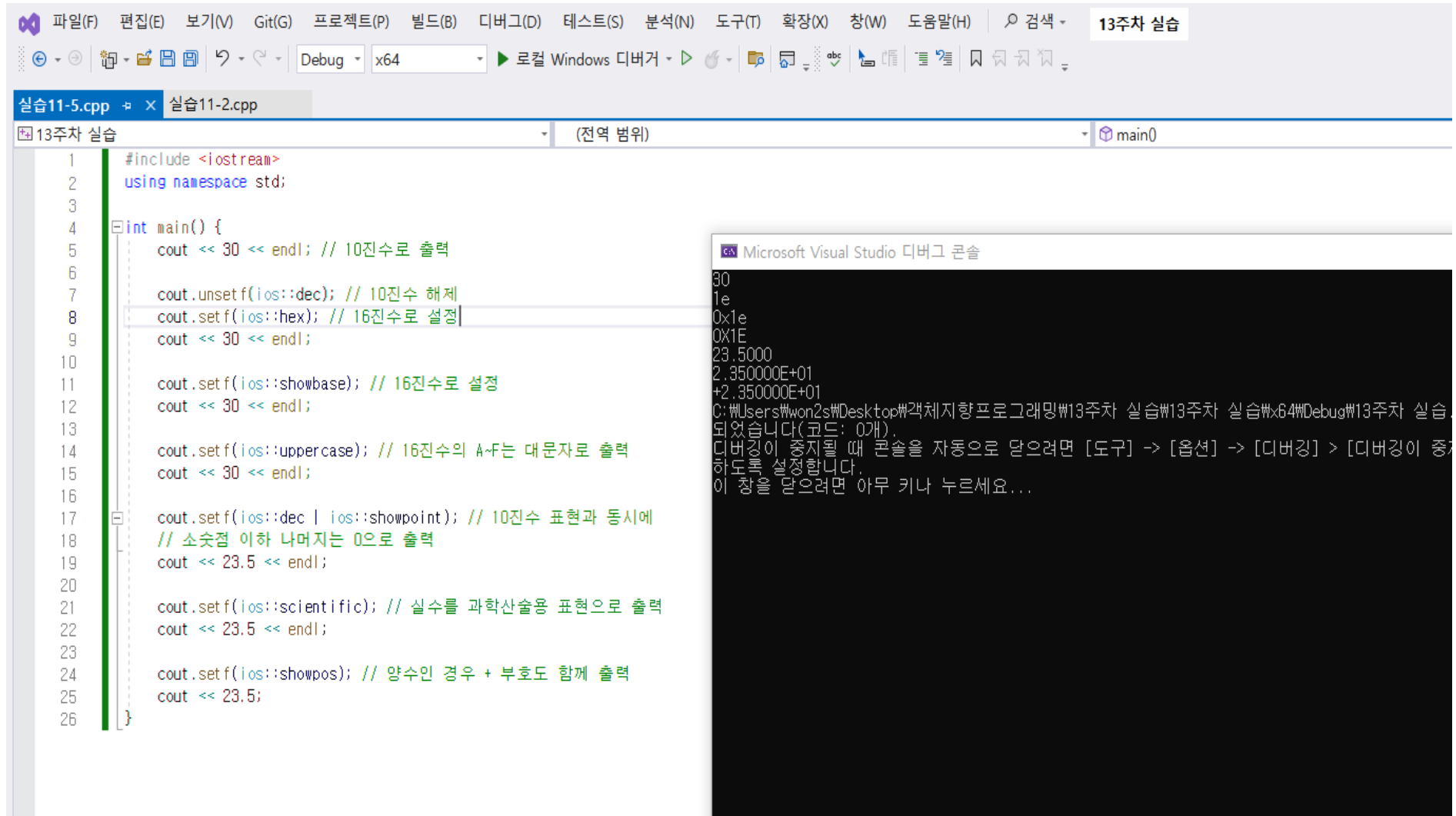
cin.getline() 함수로 라인을 읽습니다.
exit를 입력하면 루프가 끝납니다.
라인 1 >> It's now or never.
echo --> It's now or never.
라인 2 >> Come hold me tight.
echo --> Come hold me tight.
라인 3 >> Kiss me my darling, be mine tonight.
echo --> Kiss me my darling, be mine tonight.
라인 4 >> 엘비스 프레슬리 노래입니다.
echo --> 엘비스 프레슬리 노래입니다.
라인 5 >> exit

C:\Users#won2s\Desktop#객체지향프로그래밍#13주차 실습#13주차 실습#x64#Debug#13주차 실
되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이
하도록 설정합니다].
이 창을 닫으려면 아무 키나 누르세요...

[2차시 이수영] 11-5 확인만 하고 넘어감



[2차시 원준서] 11-5 확인만 하고 넘어감



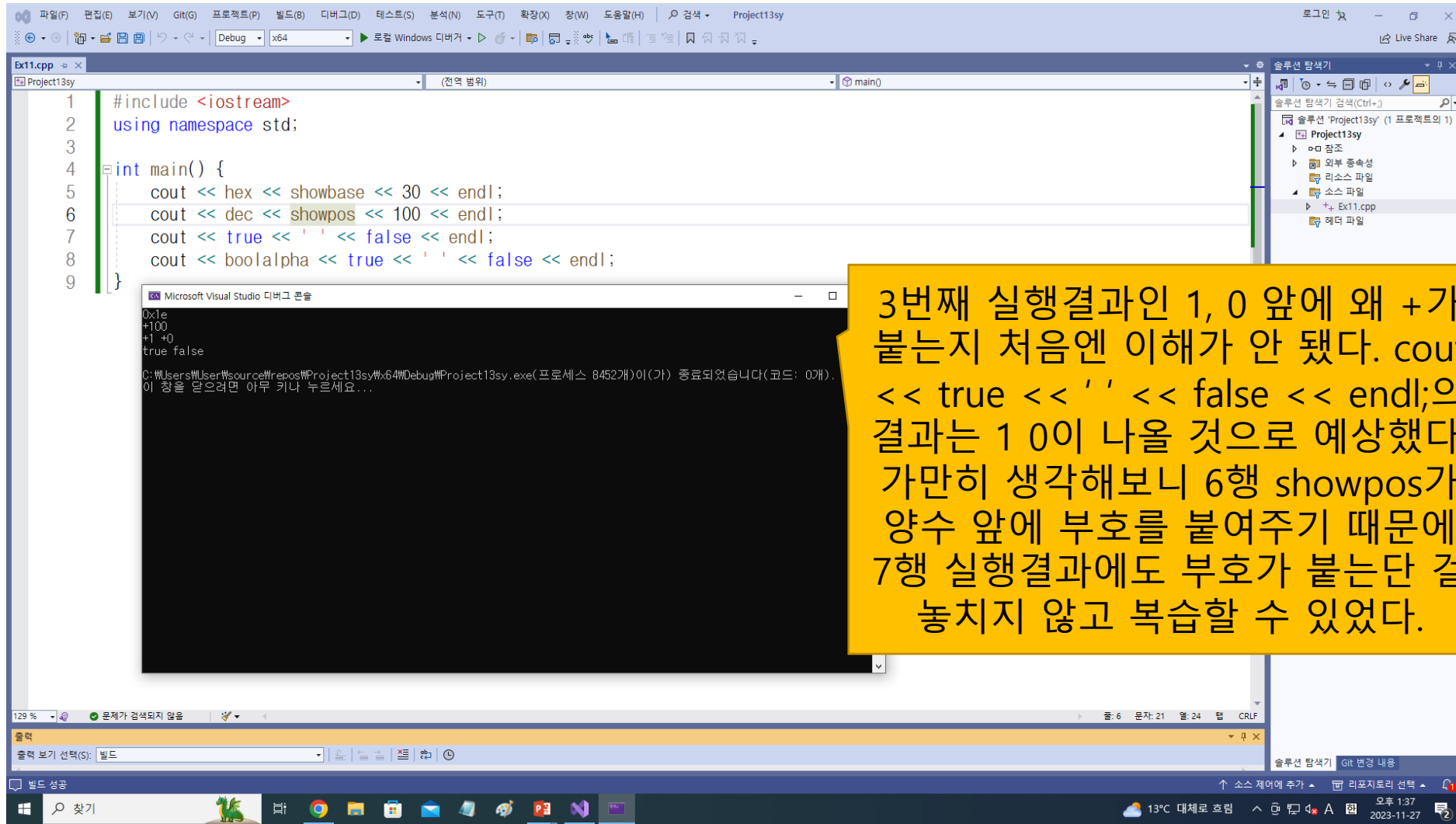
The image shows a screenshot of the Microsoft Visual Studio IDE. The main editor window displays a C++ file named '실습11-5.cpp'. The code is as follows:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << 30 << endl; // 10진수로 출력
6
7     cout.unsetf(ios::dec); // 10진수 해제
8     cout.setf(ios::hex); // 16진수로 설정
9     cout << 30 << endl;
10
11     cout.setf(ios::showbase); // 16진수로 설정
12     cout << 30 << endl;
13
14     cout.setf(ios::uppercase); // 16진수의 A-F는 대문자로 출력
15     cout << 30 << endl;
16
17     cout.setf(ios::dec | ios::showpoint); // 10진수 표현과 동시에
18     // 소숫점 이하 나머지는 0으로 출력
19     cout << 23.5 << endl;
20
21     cout.setf(ios::scientific); // 실수를 과학산술용 표현으로 출력
22     cout << 23.5 << endl;
23
24     cout.setf(ios::showpos); // 양수인 경우 + 부호도 함께 출력
25     cout << 23.5;
26 }
```

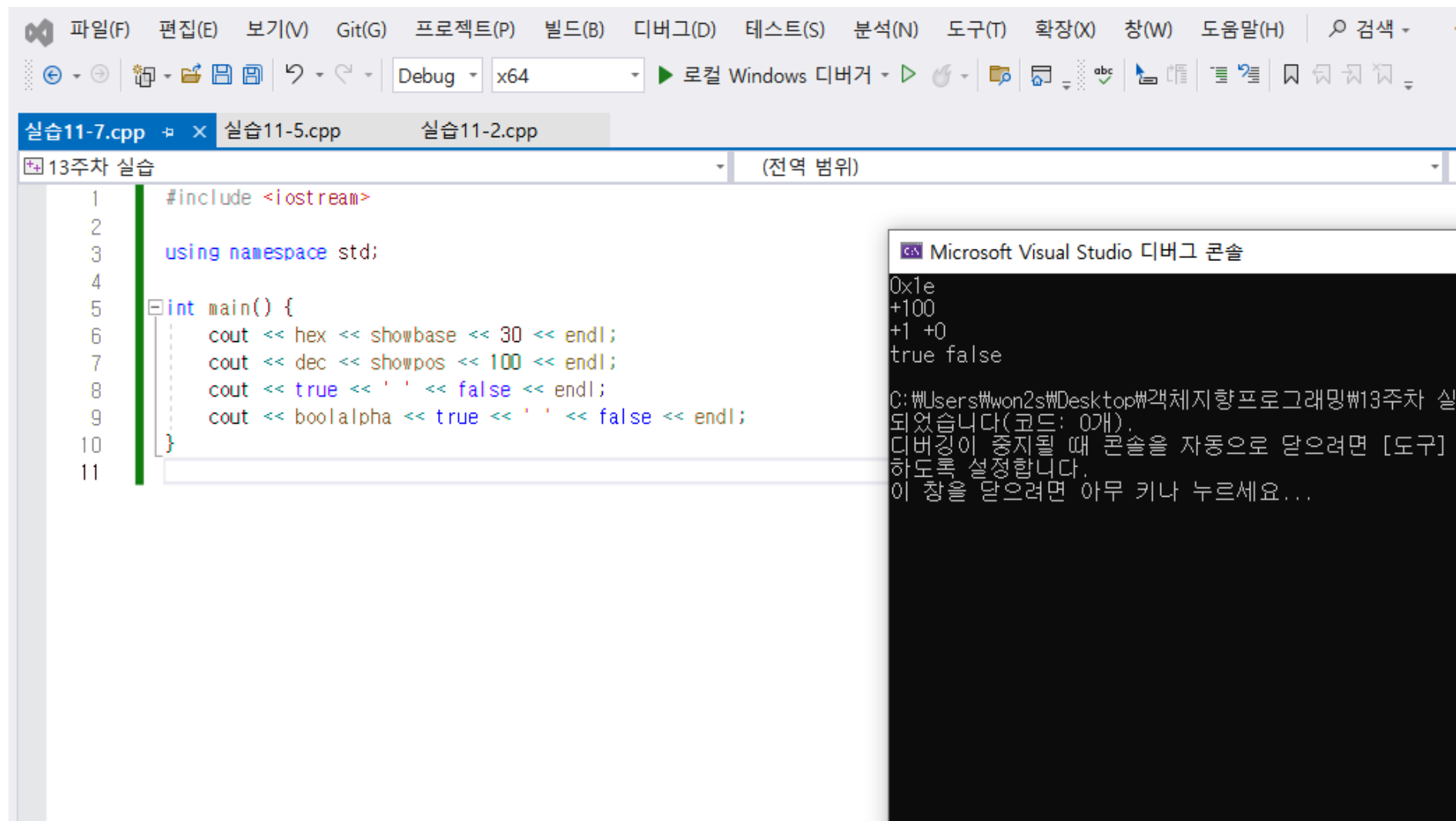
The 'Debug Console' window is open, showing the output of the program:

```
30
1e
0x1e
0X1E
23.5000
2.350000E+01
+2.350000E+01
C:\Users\won2s\Desktop\객체지향프로그래밍\13주차 실습\13주차 실습\64\Debug\13주차 실습
되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

[1차시 이수영] 11-7



[1차시 원준서] 11-7



The screenshot displays the Microsoft Visual Studio IDE. The main editor window shows a C++ file named '실습11-7.cpp' with the following code:

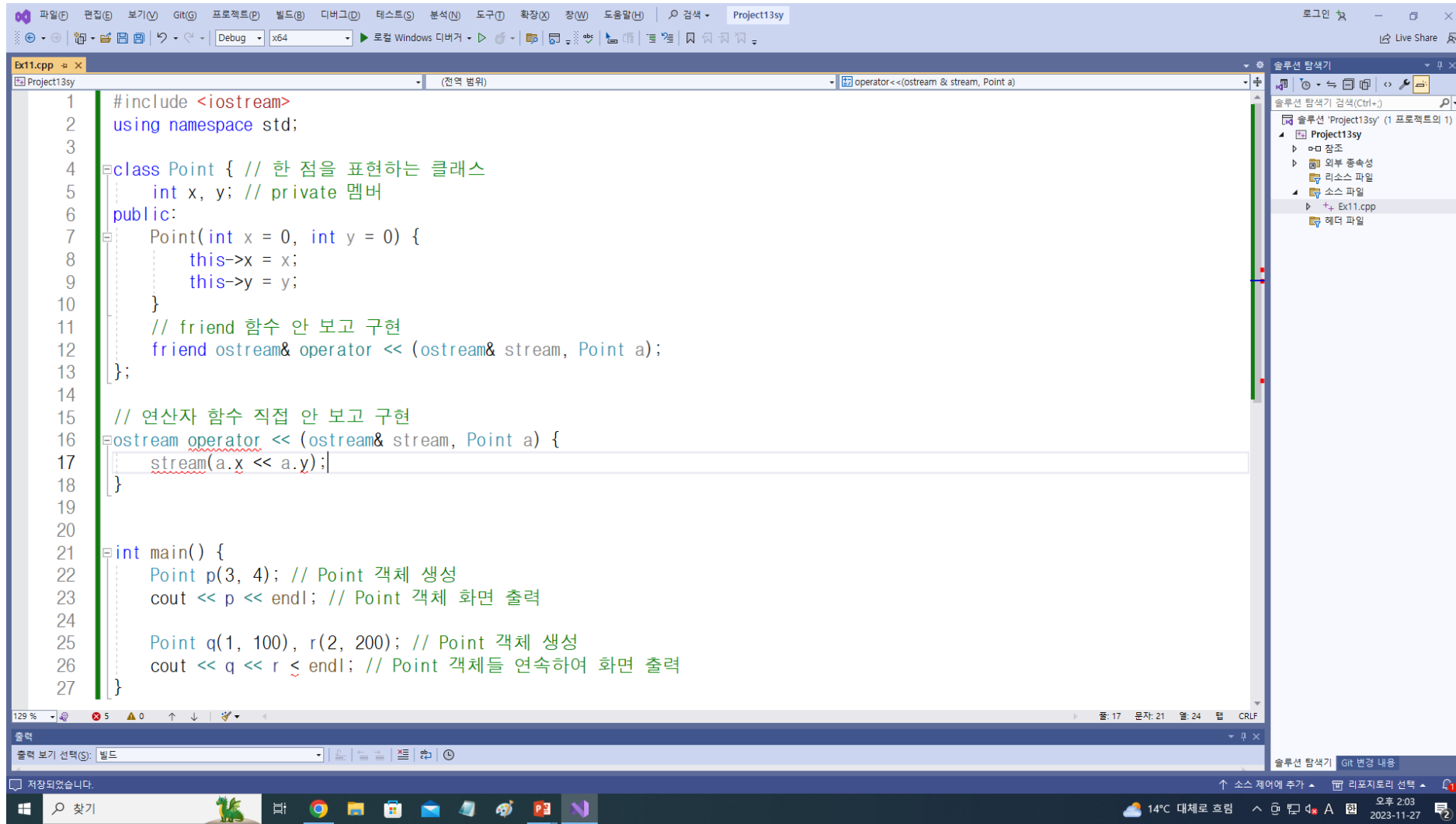
```
1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     cout << hex << showbase << 30 << endl;
7     cout << dec << showpos << 100 << endl;
8     cout << true << ' ' << false << endl;
9     cout << boolalpha << true << ' ' << false << endl;
10 }
11
```

The 'Debug' window on the right, titled 'Microsoft Visual Studio 디버그 콘솔', shows the output of the program:

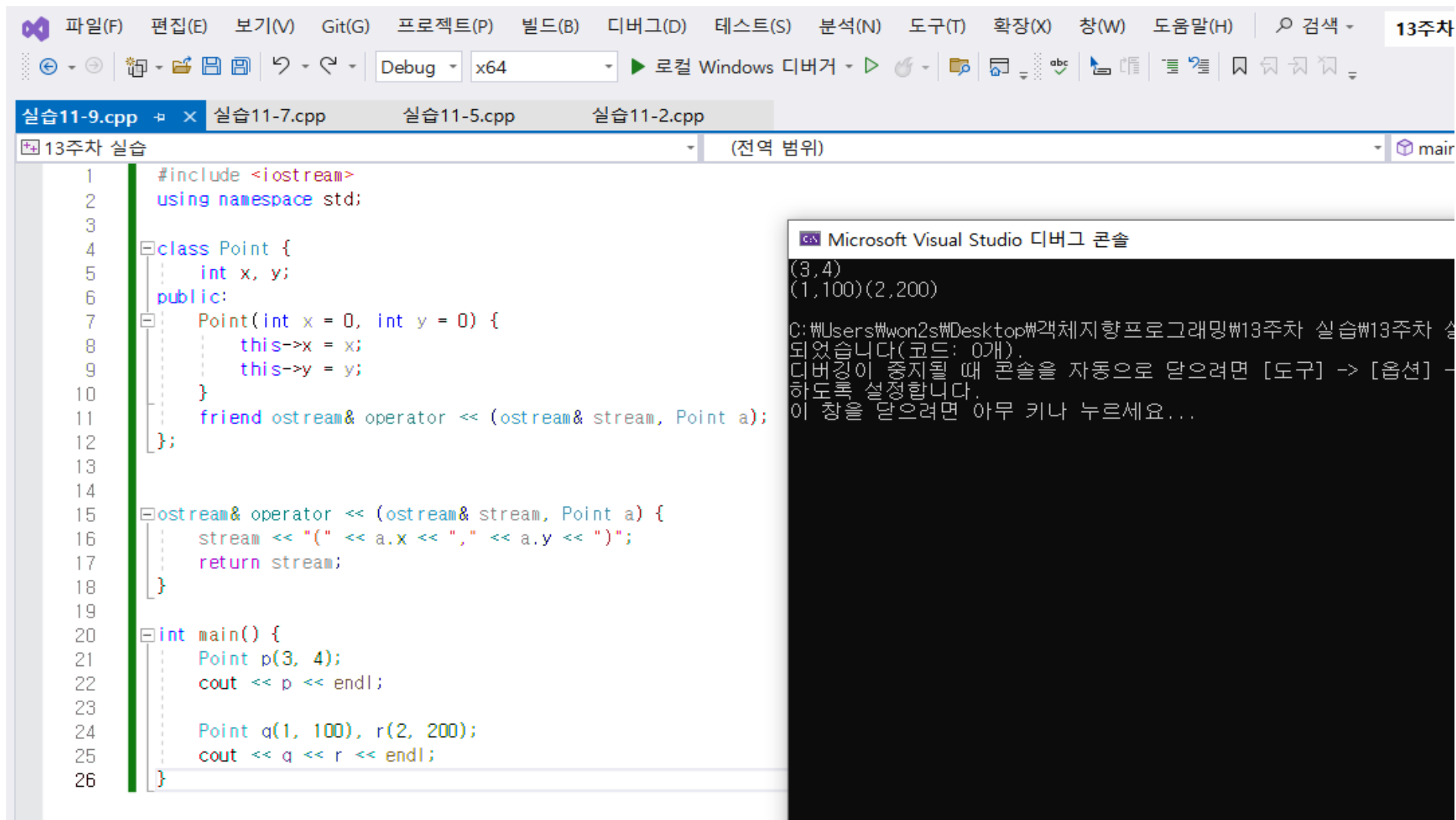
```
0x1e
+100
+1 +0
true false
```

Below the output, a message states: 'C:\Users\won2s\Desktop\객체지향프로그래밍\13주차 실... 되었습니다(코드: 0개). 디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] 하도록 설정합니다. 이 창을 닫으려면 아무 키나 누르세요...'.

[2차시 이수영] 11-9



[2차시 원준서] 11-9



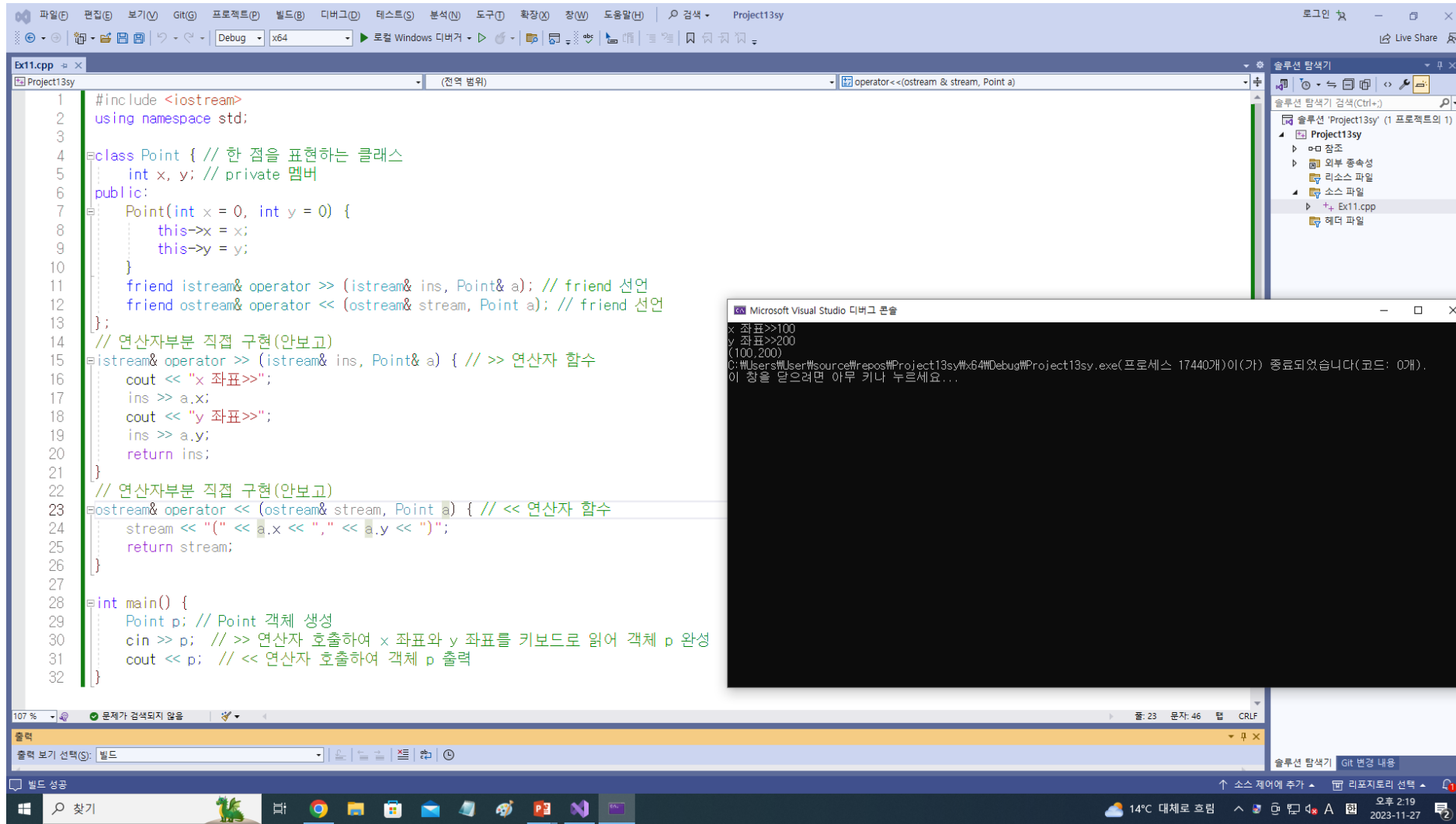
The image shows a screenshot of the Microsoft Visual Studio IDE. The main editor window displays a C++ file named '실습11-9.cpp'. The code defines a 'Point' class with two integer members 'x' and 'y', a constructor, and an overloaded stream insertion operator. The 'main' function creates two 'Point' objects, 'p' and 'q', and prints them using 'cout'.

```
1 #include <iostream>
2 using namespace std;
3
4 class Point {
5     int x, y;
6 public:
7     Point(int x = 0, int y = 0) {
8         this->x = x;
9         this->y = y;
10    }
11    friend ostream& operator << (ostream& stream, Point a);
12 };
13
14
15 ostream& operator << (ostream& stream, Point a) {
16     stream << "(" << a.x << "," << a.y << ")";
17     return stream;
18 }
19
20 int main() {
21     Point p(3, 4);
22     cout << p << endl;
23
24     Point q(1, 100), r(2, 200);
25     cout << q << r << endl;
26 }
```

The output window on the right, titled 'Microsoft Visual Studio 디버그 콘솔', shows the execution results: the coordinates of point 'p' (3, 4) and then the concatenated coordinates of points 'q' and 'r' (1, 100)(2, 200). Below the output, there is a message in Korean explaining that the program completed successfully (code: 0) and providing instructions on how to close the console window.

(3,4)
(1,100)(2,200)
C:\Users\won2s\Desktop\객체지향프로그래밍\13주차 실습\13주차 실습11-9.cpp
되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] 탭에서
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...

[2차시 이] 11-11 Point, main(), friend copy 연산자(>>/<<)부분만 안 보고 구현



[2차시 원] 11-11 Point, main(), friend copy 연산자(>>/<<)부분만 안 보고 구현

The image shows a Microsoft Visual Studio IDE with a C++ project. The main window displays the source code for `11-11.cpp`. The code defines a `Point` class with `x` and `y` coordinates. It implements stream operators `>>` and `<<` for the `Point` class. The `>>` operator prints the coordinates to the console, and the `<<` operator prints them to a stream. The `main` function creates a `Point` object `p` and uses the stream operators to read input and print the output.

```
1  #include <iostream>
2  using namespace std;
3
4  class Point {
5      int x, y;
6  public:
7      Point(int x = 0, int y = 0) {
8          this->x = x;
9          this->y = y;
10     }
11     friend istream& operator >> (istream& ins, Point& a);
12     friend ostream& operator << (ostream& stream, Point a);
13 };
14
15 istream& operator >> (istream& ins, Point& a) { //
16     cout << "x 좌표>>";
17     ins >> a.x;
18     cout << "y 좌표>>";
19     ins >> a.y;
20     return ins;
21 }
22
23 ostream& operator << (ostream& stream, Point a) {
24     stream << "(" << a.x << "," << a.y << ")";
25     return stream;
26 }
27
28 int main() {
29     Point p;
30     cin >> p;
31     cout << p;
32 }
33
```

The debug console window shows the output of the program:

```
Microsoft Visual Studio 디버그 콘솔
x 좌표>>100
y 좌표>>200
(100,200)
C:\Users\won2s\Desktop\객체지향프로그래밍\13주차 실습\13주차 실
되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] ->
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

[정리노트 원준서]

C++ 입출력 기초

- 스트림은 데이터의 흐름, 혹은 데이터를 전송하는 소프트웨어 모듈입니다. 흐르는 시내와 유사한 개념입니다.
- 스트림의 양 끝에는 프로그램과 장치가 연결되어 있으며, 보낸 순서대로 데이터를 전달하고, 입출력 기본 단위는 바이트입니다.
- C++스트림의 종류는 두 가지로, 입력 스트림과 출력 스트림이 존재합니다.
- 입력 스트림은 입력 장치, 네트워크, 파일로부터 데이터를 프로그램으로 전달하는 스트림입니다.
- 출력 스트림은 프로그램에서 출력되는 데이터를 출력 장치, 네트워크, 파일로 전달하는 스트림입니다.

[정리노트 이수영]

C++ 입출력 기초

- C++에는 입력/출력 스트림이 있다. 입력 스트림은 입력 장치/네트워크/파일의 데이터가 프로그램으로, 반면 출력 스트림은 프로그램 데이터가 출력 장치/네트워크/파일로 전달되게 한다.
- 입력 스트림, 출력 스트림이 그저 하나의 단어라고만 느꼈는데 stream이 흐르는 시냇물이라는 점을 떠올려보니 입력 스트림은 입력 장치에서 데이터가 프로그램으로 물 흐르듯이 흘러가고, 출력 스트림은 프로그램에서 데이터가 출력 장치로 흘러간다는 의미를 이해하기가 더 쉬웠다. 특히 출력 스트림의 경우, 버퍼가 다 차거나 $\backslash n$ 이 나오면 출력되며, 입출력 모두 바이트 단위다.

[정리노트 원준서]

C++ 입출력 스트림 버퍼

- C++ 입출력 스트림은 버퍼를 가집니다.
- 키 입력 스트림의 버퍼의 목적은 입력장치로부터 입력된 데이터를 프로그램으로 전달하기 전에 일시적으로 저장하는 것입니다. 키 입력 도중 수정이 가능하며, Backspace키가 입력되면 이전에 입력된 키를 버퍼에서 지웁니다.
- C++ 응용 프로그램은 사용자의 키 입력이 끝난 시점에서 읽습니다.
- <Enter>키 : 키 입력의 끝을 의미하며, <Enter>키가 입력된 시점부터 키 입력 버퍼에서 프로그램이 읽기 시작합니다.
- 스크린 출력 스트림 버퍼의 목적은 프로그램에서 출력된 데이터를 출력 장치로 보내기 전에 일시적으로 저장하는 것입니다. 출력 장치를 반복적으로 사용하는 비효율성을 개선합니다.
- 버퍼가 꽉 차거나, 강제 출력 명령 시에 출력 장치에 출력합니다.

[정리노트 이수영]

C++ 입출력 스트림 버퍼

- 마치 게임같이 실시간으로 입력이 전달돼야 하는 때가 아니면, 입력장치를 통해 입력된 데이터는 곧장 프로그램에 전달되지 않는다. 즉, 입력 스트림과 출력 스트림 모두 버퍼가 있으며, 다 프로그램에 데이터가 전달되기 전에 데이터를 일시 저장한다는 점은 동일하다.
- 입력 스트림 버퍼는 <Backspace>키로 데이터 수정이 가능하고 <Enter> 키가 입력되면 프로그램이 키 입력 버퍼의 시작점부터 데이터를 읽기 시작한다. <Enter> 키는 C++ 프로그램이 잡은 버퍼로 데이터가 블록별로 한번에 가져올 수 있게 하는데, 이 블록이 스트림이라고 이해하면 좋다.
- 출력 스트림 버퍼는 버퍼가 차거나 강제 출력 명령(flush)가 있을 때 데이터가 출력되기 시작한다. flush라는 명령을 새롭게 배웠다. 사전을 찾아보니 (물로) ~을 씻어 내다/씻어 없애다라는 뜻이다. `cout.flush()` 또는 `'\n' (= 'endl')`일 때 출력이 시작된다.

[정리노트 원준서]

C++ 표준은 스트림 입출력만 지원

- 입출력 방식은 스트림 입출력 방식, 저 수준 입출력 방식으로 두 가지가 존재합니다.
- 스트림 입출력 방식은 스트림 버퍼를 이용한 입출력 방식입니다. 입력된 키는 버퍼에 저장되며, <Enter>가 입력되면 프로그램이 버퍼에서 읽어가는 방식입니다. 출력되는 데이터는 일차적으로 스트림 버퍼에 저장되는데, 버퍼가 꽉 차거나, '\n'을 만나거나, 강제 출력의 경우에만 버퍼가 출력 장치에 출력합니다.
- 저 수준 입출력 방식은 키가 입력되는 즉시 프로그램에 키 값이 전달됩니다. <Backspace>키 그 자체도 프로그램에 바로 전달되며, 게임 등 키 입력이 즉각적으로 필요한 곳에 사용됩니다.
- 프로그램이 출력하는 즉시 출력 장치에 출력되며, 컴파일러마다 다른 라이브러리나 API를 지원합니다.
- C++표준은 스트림 입출력 방식만 지원합니다.
- 스트림 입출력은 모든 표준 C++ 컴파일러에 의해 컴파일되며, 높은 호환성을 가지고 있습니다.

[정리노트 이수영]

C++ 표준은 스트림 입출력만 지원

- C++ '표준'은 '호환성'이 높다. 앞서 배운 스트림 입출력 방식은 C++ 표준인 반면, 저수준 입출력 방식은 C++ 표준이 아니다.
- 스트림 입출력 방식은 입력 데이터가 버퍼에 저장되고, Enter로 데이터가 프로그램에 읽히며, 버퍼가 차거나 \n가 있거나 강제 출력 명령이 있으면 데이터가 출력장치에 표시된다.
- 반면 저수준 입출력 방식은 데이터가 어딘가에 저장되지 않고 입력되는 '즉시' 프로그램에 데이터가 전달된다. 데이터를 수정하기 위한 <Backspace>키마저도 '즉시' 프로그램에 전달된다. 표준은 아니지만 게임같은 '실시간' 프로그램에 적합한 것이다.

[정리노트 원준서]

2003년 이전의 C++ 입출력 라이브러리 약점

- 대표적인 구 표준(C++03) 입출력 라이브러리 클래스로는 ios, istream, ostream, iostream, ifstream, ofstream, fstream이 존재합니다.
- 문자를 한 바이트의 char로 처리하기에, cin>>로 문자를 읽을 때, 한글 문자를 읽을 수 없습니다. 영어나 기호는 1바이트의 문자 코드를 가지고 있고, 한글 문자는 2바이트의 문자 코드를 가지고 있기 때문입니다.
- 지금도 마찬가지로 cin으로 한글을 문자 단위로는 읽을 수 없습니다.
- 현재의 표준 C++ 입출력 라이브러리는, 다양한 크기의 다국어 문자를 수용하기 위해, 입출력 라이브러리가 템플릿으로 작성됩니다.

[정리노트 이수영]

2003년 이전의 C++ 입출력 라이브러리 약점

- 문자 단위로 읽을 수 없는 데이터가 있다는 한계가 있었다.
- 예를 들어 영어/기호는 1바이트인 반면, 일본어/중국어/한글은 2바이트라서 유니코드로 변환해야 된다. 그런데 C++03은 모든 문자를 한 바이트의 char로 처리해서, 한 바이트가 아닌 일본어/중국어/한글을 읽을 수 없다. 물론 2003년 뿐만 아니라 지금도 cin을 가지고 한글을 문자 단위로 읽을 수 없다는 점이 놀랍다. 보통 그 다음 표준에서 개선됐을 텐데 왜 아직 읽을 수 없는 상태로 유지되고 있을까? 고민하던 찰나에 다음 장에서 현재 쓰고 있는 방안인 템플릿 개념을 배우니 내용의 흐름이 잘 이해됐다.

[정리노트 원준서]

입출력 클래스

- ios는 모든 입출력 스트림 클래스들의 기본 클래스로, 스트림 입출력에 필요한 공통 함수와 상수, 멤버 변수를 선언합니다.
- istream, ostream, iostream
- istream은 문자 단위 입력 스트림, ostream은 문자 단위 출력 스트림, iostream은 문자 단위로 입출력을 동시에 할 수 있는 스트림 클래스입니다.
- ifstream, ofstream, fstream
- 파일에서 읽고 쓰는 기능을 가진 파일 입출력 스트림 클래스, 파일에서 읽을 때는 ifstream 클래스를, 파일에 쓸 때는 ofstream 클래스를, 읽고 쓰기를 동시에 할 때 fstream 클래스를 이용합니다.

[정리노트 이수영]

입출력 클래스

- 다양한 크기(바이트)의 다국어 문자가 모두 정상적으로 입출력될 수 있도록 입출력 라이브러리가 템플릿 클래스로 작성된다.
- 계층은 ios-iostream-(io)fstream 순으로 진행된다고 보면 좋다.
- 스트림 입출력 기반 템플릿 클래스 ios
- 입출력 스트림 기반 템플릿 클래스 istream, ostream
- 파일 입출력 스트림 템플릿 클래스 ifstream, ofstream
- ios를 상속받는 istream, ostream을 상속받는 fstream의 구조!
- ios는 기본 클래스, istream은 문자 단위 입출력 스트림 클래스, fstream은 파일 입출력 스트림 클래스다.
- 여기서 '클래스'를 배웠으니 다음 장에선 '객체'를 배우는 연결고리가 있다.

[정리노트 원준서]

C++ 표준 입출력 스트림 객체

- C++ 프로그램이 실행될 때 자동으로 생겨나는 스트림은 다음과 같습니다.
- cin은 istream 타입의 스트림 객체로서 키보드 장치와 연결됩니다.
- cout은 ostream 타입의 스트림 객체로서 스크린 장치와 연결됩니다.
- cerr은 ostream 타입의 스트림 객체로서 스크린 장치와 연결됩니다. 오류 메시지를 출력할 목적으로 사용되며, 스트림 내부 버퍼를 거치지 않고 출력됩니다.
- clog는 ostream 타입의 스트림 객체로서 스크린 장치와 연결됩니다. 오류 메시지를 출력할 목적으로 사용되며, 스트림 내부에 버퍼를 거쳐 출력됩니다.

[정리노트 이수영]

C++ 표준 입출력 스트림 객체

- 프로그램이 '실행'될 때 '자동'으로 생겨나는 스트림 객체다.
- 입력 스트림 객체 cin (istream 타입-키보드 장치와 연결된다.)
- 출력 스트림 객체 cout (ostream 타입-스크린 장치와 연결된다.)
- 출력 스트림 객체 cerr (ostream 타입-스크린 장치와 연결된다.)
- 오류 메시지를 출력하기 위해 쓰이는데 **버퍼를 안 거친다.**
- 출력 스트림 객체 clog (ostream 타입-스크린 장치와 연결된다.)
- 오류 메시지를 출력하기 위해 쓰이는데 **버퍼를 거친다.**
- 객체의 타입에 istream/ostream 타입이 있다는 걸 새로 배웠다. 뒤의 예제를 풀 때, 이런 타입이 있는 걸 기억 못 하니 코드가 안 떠올랐다.

[정리노트 원준서]

문자열 입력

- `get()`이 읽는 도중 `<Enter>` 키(`'\n'`)를 만날 때 읽기를 중단하고 리턴합니다.
- `<Enter>` 키(`'\n'`)는 스트림 버퍼에 남아 있습니다. 다시 `get()`으로 문자열 읽기를 시도하면 입력 스트림에 남은 `'\n'`키를 만나게 되어 바로 리턴. 프로그램은 무한 루프에 빠질 수 있습니다.
- 이때 이 문제를 해결하기 위해서는 `cin.get()`이나 `cin.ignore(1);`를 통해 문자 1개(`'\n'`)를 스트림에서 읽어 버려야 합니다.

[정리노트 이수영]

문자열 입력

- ostream의 멤버 함수 put(), write(), flush()는 리턴타입이 ostream의 주소(reference)다.
- ostream& **put**(char ch)는 문자 ch를 스트림에 **출력**한다.
- ostream& **write**(char* str, int n)은 배열 str에 있는 문자 n개를 스트림에 **출력**한다. 배열을 char* 타입으로 한단 걸 잊지 말자!
- ostream& flush()는 현재 스트림 버퍼에 있는 데이터를 출력!
- istream의 멤버 함수 **get**()은 입력 스트림에서 데이터를 **읽는다**.
- istream& get(char* s, int n)은 입력 스트림에서 문자 (n-1)개를 배열 s에 저장하고, 배열 마지막에 '\0'를 삽입한다는 걸 뜻한다.
- get()은 <Enter> 키(='\n')를 만나면 읽기를 멈추고 리턴하는데, 이 때 입력됐던 '\n'가 버퍼에 남아있다는 특징이 있다. 그래서 cin.ignore(1);라는 코드를 입력해 무한루프에서 빠져나올 수 있게 해야 한다.
- 무한 루프를 보니 예제 11-4가 떠오른다. 무한 루프가 도는 과정은 예제 11-4를 통해 직접 눈으로 확인할 수 있었다. 확실히 교재엔 없지만 get()과 getline()을 비교해보라는 말씀을 듣고 실습을 해보니, 무한루프가 너무 빠르게 도는 과정이 충격적이라 오래 기억에 남는다. getline()은 get()과 다른 특징은 다 똑같지만, delim에 지정된 구분 문자를 스트림에서 제거한다는 점만 다른데, get()이 버퍼에 남아있는 '\n'을 버퍼에서 빼내야 한다는 점과 연관지어 같이 기억하기 좋았다.

[정리노트 원준서]

포맷 입출력

- C++에서도 입출력 시 포맷 지정이 가능합니다. C언어의 printf()와 유사합니다.
- 포맷 입출력의 3가지 방법으로는 포맷 플래그, 포맷 함수, 조작자가 있습니다.
- 포맷 플래그는 입출력 스트림에서 입출력 형식을 저장하기 위한 플래그입니다.

[정리노트 이수영]

포맷 입출력(1) 포맷 '플래그'

- ios 클래스에 정의된 포맷 플래그는 ios::뒤에 의미를 나타내는 플래그가 적히는 형태로 구성된다.
- ios::uppercase는 대문자로 바꿔 출력하고, ios::showbase는 16진수면 0x를, 8진수면 0을 앞에 붙이고, ios::showpos는 양수일 때 + 기호를 출력하고, ios::left/right는 왼쪽/오른쪽맞춤, ios::dec/oct/hex는 10/8/16진수로 출력한다. ios::boolalpha는 true/false를 1/0이 아닌 "true"/"false"로 출력되게 한다.
- 설정하는 방법이 중요한데 cout.**setf**(ios::~); 형태다.
- 해제하는 방법은 cout.**unsetf**(ios::~); 형태다.

[정리노트 이수영]

포맷 입출력(2) 포맷 '함수'

- `cout.width(int minwidth);`
- `cout.fill(char cFill);`
- `cout.precision(int np);`
- 형태는 이렇지만 실제 코드를 이해하는 게 훨씬 쉬웠다.
- `cout.width(10);`은 앞으로 출력될 모든 문자열을 10칸으로 지정
- `cout.fill('^');`은 width로 지정된 칸 중 "Hello"를 제외하고 ^로 채운다. width가 10이고 Hello가 5글자니, 남은 5문자는 ^^^^^ 그래서 출력은 ^^^^^Hello가 되겠지.
- `cout.precisiton(5);`는 유효숫자 자리수가 5이므로 3.6667만 출력된다. 특히 소수점 아래자리와 정수부분을 포함한 숫자라는 점에 유의하자. `precision(5)`는 소수점 아래를 5자리까지 표현하는 게 아니라, 정수를 포함해서 5자리다. 즉, 36.667777이라면 36.667만 출력될 것이다.

[정리노트 원준서]

조작자

- manipulator, 스트림 조작자입니다.
- 조작자는 함수입니다.
- C++ 표준 라이브러리에 구현된 조작자: 입출력 포맷 지정 목적
- 개발자 만의 조작자 작성 가능 : 다양한 목적
- 매개 변수 없는 조작자와 매개 변수를 가진 조작자로 구분
- 조작자는 항상 << 나 >> 연산자와 함께 사용됩니다.

[정리노트 이수영]

조작자

- 조작자는 함수이며, '항상' << 또는 >> 연산자와 함께 쓰인다.
- C++ 표준에 구현된 게 있는데, 없다면 사용자 지정이 가능하다.
- 매개변수가 있는 조작자는 `#include <iomanip>`를 선언한다. `setfill('^');`이 있다.
- 매개변수가 없는 조작자엔 `endl` 등이 있다. `oct/dec/hex`는 정수 필드를 8/10/16진수로 출력. `left/right`는 왼쪽/오른쪽 맞춤 출력. `flush`는 스트림 버퍼 강제 출력. `showbase`는 16/8진수일 때 정수 앞에 `0x/0`을 붙인다. `noshowbase`는 `showbase`에서 설정한 값을 해제한다.
- 예제 11-8이 조작자의 장점을 가장 잘 보여준다고 생각했다. 출력 결과처럼 딱 형식에 맞춘 형태로 출력하고 싶을 때가 있는데, 이 때 가장 좋은 조작자가 `setw(8)`이다. `setfill('.')`과 같이 쓰니 한눈에 확인하기 쉽다. 마치 1차시, 2차시, 3차시로 파일명을 저장하다가 11차시로 저장하면 11차시가 1차시 뒤에 정렬되는 것을 막기 위해 01차시, 02차시로 저장해서 11차시가 02차시 앞이 아니라 10차시 뒤에 저장되도록 하는 것처럼 0, 5, 10이 십의자리/일의자리가 일렬로 정렬되도록 하는 조작자가 `setw(n)`, `setfill('c')`인 것이다.

[정리노트 원준서]

삽입 연산자(<<)와 추출 연산자(>>)

- 삽입 연산자(<<)
- insertion operator, 삽입자라고도 부릅니다. << 연산자는 C++의 기본 연산자로 정수 시프트 연산자이며, ostream 클래스에 중복 작성되어 있습니다.
- 사용자 삽입 연산자 만들기
- 개발자가 작성한 클래스의 객체를 << 연산자로 출력할 수 있습니다.
- 추출 연산자(>>)
- extraction operator, >> 연산자는 C++의 기본 연산자로 정수 시프트 연산자이며, ostream 클래스에 중복 작성되어 있습니다.
- 추출 연산자의 실행 과정은 삽입 연산자의 실행 과정과 유사합니다.
- 사용자 추출 연산자 만들기
- 개발자가 작성한 클래스의 객체에 >> 연산자로 입력합니다.

[정리노트 이수영]

삽입 연산자(<<)와 추출 연산자(>>)

- 삽입 연산자는 시프트 연산자가 오버로딩된 것으로, 중복연산자 개념을 끌어와서 해당 내용을 이해하면 좋다. 7장에 배운 중복 연산자 개념을 가져온다면, 이 때 <<는 시프트 연산자가 아닌 삽입 연산자로 쓰인 것이다. 따라서 ostream 클래스에 중복 작성된 연산자이므로, 출력과 관련된 연산자이지 시프트 연산처리하는 연산자가 아님에 유의하면 좋다.
- 특히 해당 내용을 배울 때 삽입 연산자를 사용자가 직접 만드는 부분이 여러 번 반복되는 과정에서 가장 기억에 남는다. 특히 포인트 클래스로 연습하다 보니 기억나는데, C++ 표준에는 Point와 같은 클래스가 설정되어 있지 않아 직접 사용자가 구현해야 한다.
- `cout << p;`는 컴파일러에 의해 경우의 수 2가지로 변형될 수 있는데, 첫번째는 `cout . << (Point p)`다. 그러나 C++ 표준에 Point 클래스가 없으므로, 두번째로 변형될 수 있다. `<< (cout, p);`로 변형되며, <<에 해당하는 연산자를 개발자가 직접 작성하면 된다. `ostream& operator << (ostream& stream, Point a)` 조작자(함수)를 쓴다. 그리고 Point 클래스의 x, y는 private 멤버이므로 main()에서 private 멤버에 접근하기 위해서는 friend로 선언해야 한다. 그게 바로 `friend ostream& operator << (ostream& stream, Point a);`다. 예제 11-9는 수업내용을 완벽하게 이해하지 못한 채로 작성해 곧바로 떠오르지 않은 부분이라 확실히 틀린 후에 보니 더 오래 기억하고 싶다고 느꼈다.