

# 20191121 실습 9주차

박건호

KITRI BoB 8<sup>th</sup> 5422 RTDCS

devgunho.github.io



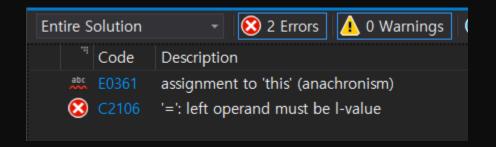
## this (C++)

The this pointer is a pointer accessible only within the nonstatic member functions of a class, struct, or union type. It points to the object for which the member function is called. Static member functions do not have a this pointer.



#### 1. this Exercise

#### 주어진 코드의 출력을 예측해보고 컴파일 오류를 해결하기



```
Int main()
{
    Test obj(5);
    Test* ptr = new Test(10);
    obj.change(ptr);
    obj.print();
    return 0;
}

Microsoft Visual Studio Debug Console
x = 10

D:\(\pi\)_\(\text{Workspace}\(\pi\)19_\(\text{ComputerProgramming}\(\pi\)19_\(\text{CP}\)_\(\text{Press any key to close this window . . . .}
```



### this

The expression \*this is commonly used to return the current object from a member function: C++ Copy Copy return \*this; The **this** pointer is also used to guard against self-reference: Copy Copy C++ if (&Object != this) { // do not execute in cases of self-reference



### this

```
The expression *this is commonly used to return the current object from a member function:

C++

return *this;

The this pointer is also used to guard against self-reference:

C++

if (&Object != this) {
// do not execute in cases of self-reference
```

① Note

Because the **this** pointer is nonmodifiable, assignments to **this** are not allowed. Earlier implementations of C++ allowed assignments to **this**.

```
∃#include <iostream>
 #include <string.h>
 using namespace std;
∃class Buf
 public:
     Buf(char* szBuffer, size t sizeOfBuffer);
     Buf& operator=(const Buf&);
     void Display() { cout << buffer << endl; }</pre>
 private:
     char* buffer;
     size t sizeOfBuffer;
□Buf::Buf(char* szBuffer, size_t sizeOfBuffer)
     sizeOfBuffer++; // account for a NULL terminator
     buffer = new char[sizeOfBuffer];
     if (buffer)
         strcpy_s(buffer, sizeOfBuffer, szBuffer);
         sizeOfBuffer = sizeOfBuffer;
```



```
Buf& Buf::operator=(const Buf& otherbuf)
{
    if (&otherbuf != this)
    {
        if (buffer)
            delete[] buffer;

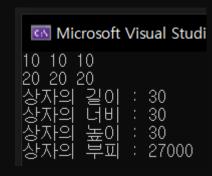
        sizeOfBuffer = strlen(otherbuf.buffer) + 1;
        buffer = new char[sizeOfBuffer];
        strcpy_s(buffer, sizeOfBuffer, otherbuf.buffer);
    }
    return *this;
}
```



#### 2. Box Class Plus

Box 객체에 대하여 + 연산을 하면 상자의 길이, 폭, 높이가 서로 합쳐진다고 가정한다.

+ 연산자 중복 함수를 Box 클래스의 멤버 함수로 작성하기





## 3. Box Class ==(equal), !=

== 연산에 대하여 2개의 상자가 동일한 부피인지 검사한다.

!= 연산에 대하여 2개의 상자가 다른 부피인지 검사한다.

```
Microsoft Visual Studio

10 10 10
10 10 10
Box1 == Box2 : true
Box1 != Box2 : false
```

```
Microsoft Visual Studio Debug Console

10 10 10
20 20 20

Box1 == Box2 : false

Box1 != Box2 : true
```



#### 4. Reference Return

앞선 예제에서 보다시피 C++ 에서는 자기 자신에 대한 참조자를 리턴하는 키워드는 없다.

주석을 참고하여 다음 코드를 정상 동작하게 코드를 추가해보자.

```
#include <iostream>
 using namespace std;
⊟class A {
 public:
     int a;
                // 리턴 타입이 자기자신의 참조자
     A& B() {
        // 자기 자신에 대한 참조자를 리턴
⊡int main()
     int aaa;
     cin >> aaa;
     A a;
     A\& b = a.B();
     a.B().a = aaa; // 이와 같은 문법이 가능
     cout << a.a << endl;</pre>
     return 0;
```



## 5. Find Average Operation

```
#include <iostream>
 using namespace std;
□class average {
                                      객체를 복사하여 리턴하는 것이 아니라 참조자이기 때문에 대입연산의
    double sum;
                                      좌항에 적용할 수 있다.
    int count;
 public:
                                      주석을 참고하여 코드가 정상 동작하게 코드를 추가하여라.
    average() {
        sum = count = 0;
    average& operator[](double num) {
        count++;
        sum += num;
                            ⊡void main() {
                                                                                               Micros
        return *this;
                                 double d1, d2, d3, d4, d5, d6;
                                 cin >> d1 >> d2 >> d3 >> d4 >> d5 >> d6;
    double operator() () {
                                 average avg;
        // 333
                                 avg[d1][d2][d3][d4][d5][d6]; // [] 연산자를 통해 값을 누적한다.
                                 std::cout << avg() << std::endl; // () 연산자를 통해 평균을 얻어낸다.
                                                                                              4.01667
```



## 6. Find Max Operation

앞선 예제를 활용하여 최댓값을 찾는 연산자를 정의하여라.

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

class max {
   int value;
   public:
   max() { value = 0; }
}
```

```
void main() {
    int d1, d2, d3, d4, d5, d6;
    cin >> d1 >> d2 >> d3 >> d4 >> d5 >> d6;
    max m;
    m[d1][d2][d3][d4][d5][d6];
    std::cout << m() << std::endl;
}

C:\(\mathrightarrow\)
C:\(\mathrightarrow\)
C:\(\mathrightarrow\)
Press any key to clo</pre>
```



## 7. Go Upstream

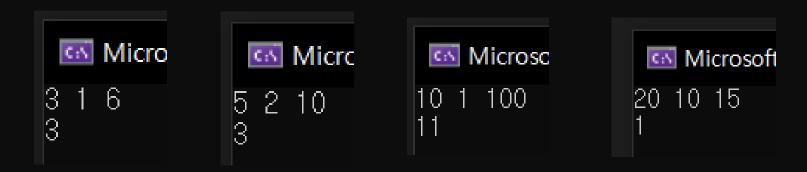
강을 거슬러 올라가서 N km에 위치한 선착장에 도착해야 한다.

낮에는 A km 올라갈 수 있지만, 밤에 잠을 자면서 B km 다시 하류로 흘러간다.

마지막 목표지점(선착장)인 N km에 도달한 후에는 더 이상 하류로 흘러 내려가지 않는다.

선착장에 도착하려면 며칠이 걸리는지 구하시오.

첫째 줄에 세 정수 A, B, N이 공백으로 구분되어 주어진다. (1<=B<A<=1,000,000,000)



실행시간제한은 없으나 반복문없이 구현하는 방법을 생각해보기