

## Re: operator function

**Source:** <http://coding.derkeiler.com/Archive/C/ CPP/alt.comp.lang.learn.c-cpp/2004-06/0031.html>

---

**From:** Karl Heinz Buchegger (*kbuchegg\_at\_gascad.at*)

**Date:** 06/01/04

Date: Tue, 01 Jun 2004 11:31:17 +0200

Kenny wrote:

```
>
>>
>> Strange. There are a number of problems with your operator
>> (and with the GetTotalTime() function), but none
>> of them have anything to do with what the error message seems
>> to imply.
>>
>> To remember, here is your operator
>>
>> Setting operator+(const Setting &a, const Setting &b )
>> {
>> return a.GetTotalTime() + b.GetTotalTime();
>>
>> }
>>
>> When compiling this, my compiler emits:
>>
>> error C2662: 'GetTotalTime' : cannot convert 'this' pointer from 'const
> class Setting' to 'class
> Setting &'
>> Conversion loses qualifiers
>>
>> Does that help you? Hint: A member function can be const. Under which
> circumstances
>> is that important?
>
> Well I guess you are talking about a copy constructor right?
```

Not really. The object is const! The member function is not!  
Since the member function doesn't alter the state of that object, it could be used on a const object – if only the compiler would know that it does no harm. Marking the function as const is exactly the way to do that:

```
class Setting
{
public:
```

```
Setting ()
{
    _hours = 0;
    _minutes = 0;
    _seconds = 0;
    _total_time = 0;
}
```

```
Setting (int h, int m, int s, int TT)
{
    _hours = h;
    _minutes = m;
    _seconds = s;
    _total_time = TT;
}
```

```
~Setting(){}
```

```
int GetTotalTime() const
```

```
*****
```

Now the compiler knows, that it is safe to use that function on a const object.

```
>
>>
>> When fixing this, the next error in the operator is:
>>
>> error C2664: '__thiscall Setting::Setting(const class Setting &)': cannot
> convert parameter 1 from
>> 'int' to
>> 'const class Setting &'
>> Reason: cannot convert from 'int' to 'const class Setting'
>> No constructor could take the source type, or constructor overload
> resolution was ambiguous
>>
>> Hint: The compiler is talking about the result of the addition, which is
> an int. But the
>> operator returns a Setting object. In order to gap that bridge, the
> compiler needs to create
>> a Setting object from a single int, which requires a constructor that
> takes ...
>
> well this part I can solve by making the function void and just print out
> the value so it will not return anything.
> cout<< a.GetTotalTime() + b.GetTotalTime();
```

Not really. Now this function doesn't do any longer what it is supposed to do.

The purpose of an operator+ is to add some objects (whatever add may mean for them), not to print something.

Just because it is easier for you doesn't mean that it is correct. Imagine a program where I (not you) use your Settings objects. For some reasons I use operatot+ to do some calculations. And now your operator+ doesn't do the addition but instead does some output. Can you imagine what I am going to tell you :-)

Take the hint about the constructor serious, it is the path to the solution.

As for your error message:

Honestly I have no idea why your compiler emits exactly that message. It may be because there is some other error in your code prior to operator+ and the compiler got confused.

There is only one way we (the group) can help any further. Post the complete program (not something that looks close or similar) which demonstrates exactly that error message (compile it before you post to make sure that your compiler emits exactly that error message) and we can look further. Otherwise we have to use our crystal balls and that is not something you want us to do :-)

--

Karl Heinz Buchegger  
kbuchegg@gascad.at