sign up log in tour help stack overflow careers

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free. Sign up ×

How to use Enums in C++

Suppose we have an `enum` like the following:

```
enum Days { Saturday,Sunday,Tuesday,Wednesday,Thursday,Friday };
```

I want to create an instance of this `enum` and initialize it with a proper value, so I do:

```
Days day = Days.Saturday;
```

Now I want to check my variable or instance with an existing `enum` value, so I do:

```
if(day == Days.Saturday)
{
    std::cout<<"Ok its Saturday";
}
```

Which gives me a compilation error:

> error: expected primary-expression before '.' token

So to be clear, what is the difference between saying:

```
if(day == Days.Saturday)        //causes compilation error
```

and

```
if(day == Saturday)
```

What do these two actually refer to, in that one is OK and one causes a compilation error?

c++ enums

edited May 19 '14 at 3:28

asked Aug 29 '12 at 17:02

Hossein

2,654 6 39 63

8 It's not hard to Google "using enums in C++" to find the proper syntax. It's worth noting that C++11 has scoped enums as well as the ability to specify the underlying type. – chris Aug 29 '12 at 17:03

i know, i want o know why its giving me the error! – Hossein Aug 29 '12 at 17:04

Its Wednesday here. You have too lot of syntax errors for C++ compiler. Starting from 'Enum'. – Öö Tiib Aug 29 '12 at 17:06

@Hossein, Because enums aren't the same syntax (and semantics) in both languages. The first thing I do after getting an error when trying to use a feature in a new language is look up the syntax (or if it's possible) in that language. – chris Aug 29 '12 at 17:08

7 "*as far as i know the enums declaration and usage in these two languages are the same.*". There's your problem, right there. C# is **not** the same language as C++. Particularly, they have different syntax for enums. – Rob$_{\varphi}$ Aug 29 '12 at 17:26

10 Answers

This code is wrong:

```
enum Days { Saturday,Sunday,Tuesday,Wednesday,Thursday,Friday};
Days day = Days.Saturday;
if(day == Days.Saturday)
```

Because days is not a scope, nor object. It is a type. And Types themselves don't have members. What you wrote is the equivalent to `std::string.clear`. `std::string` is a type, so you can't use `.` on it. You use `.` on an *instance* of a class.

Unfortunately, enums are magical and so the analogy stops there. Because with a class, you can

do `std::string::clear` to get a pointer to the member function, but in C++03, `Days::Sunday` is invalid. (Which is sad). This is because C++ is (somewhat) backwards compatable with C, and C had no namespaces, so enumerations had to be in the global namespace. So the syntax is simply:

```
enum Days { Saturday,Sunday,Tuesday,Wednesday,Thursday,Friday};
Days day = Saturday;
if(day == Saturday)
```

edited Feb 24 '14 at 3:47

Eric Leschinski
28.9k 17 150 147

answered Aug 29 '12 at 17:26

Mooing Duck
34.9k 9 54 93

91 Fortunately, your complaint has been addressed in C++11. Change `enum` to `enum class` and it gets its own scope; so `Days::Sunday` is not only valid, but is the only way to access `Sunday`. Happy days! – Mike Seymour Aug 29 '12 at 22:57

This will be sufficient to declare your enum variable and compare it:

```
enum Days { Saturday,Sunday,Tuesday,Wednesday,Thursday,Friday};
Days day = Saturday;
if(day == Saturday){
    std::cout<<"Ok its Saturday";
}
```

edited Feb 24 '14 at 3:52

Eric Leschinski
28.9k 17 150 147

answered Aug 29 '12 at 17:04

mathematician1975
14.9k 3 22 56

why is it wrong to say if (day== Days.Satudday) ? they must be the same,so why is compiler complaining about it? – Hossein Aug 29 '12 at 17:05

1 @Hossein the values declared in your enum do not behave like class or struct member variables. This is not the correct syntax to use – mathematician1975 Aug 29 '12 at 17:07

2 @Hossein: because `Days` is not a scope, nor object. It is a type. And Types *themselves* don't have members. `std::string.clear` also fails to compile for the same reason. – Mooing Duck Aug 29 '12 at 17:07

5 @Hossein: Because that's not how enums in C++ work. Unscoped enumerations put their values into the surrounding namespace; scoped ones (`enum class` , new in 2011) have their own scope, and are accessed using the scope operator, `Days::Saturday` . The member access operator (`.`) is only used to access class members. – Mike Seymour Aug 29 '12 at 17:08

@MooingDUck and MikeSeymour Would one of you guys post your answer as an answer? because that is exactly what i was after by issuing this question ;) – Hossein Aug 29 '12 at 17:15

Much of this should give you compilation errors.

```
// note the lower case enum keyword
enum Days { Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday };
```

Now, `Saturday` , `Sunday` , etc. can be used as top-level bare constants,and `Days` can be used as a type:

```
Days day = Saturday;   // Days.Saturday is an error
```

And similarly later, to test:

```
if (day == Saturday)
    // ...
```

These `enum` values are like bare constants - they're *un*-scoped - with a little extra help from the compiler: (unless you're using C++11 *enum classes*) they *aren't* encapsulated like object or structure members for instance, and you can't refer to them as *members* of `Days` .

You'll have what you're looking for with C++11, which introduces an `enum class` :

```
enum class Days
{
    SUNDAY,
    MONDAY,
    // ... etc.
```

```
}

// ...

if (day == Days::SUNDAY)
    // ...
```

Note that this C++ is a little different from C in a couple of ways, one is that C requires the use of the `enum` keyword when declaring a variable:

```
// day declaration in C:
enum Days day = Saturday;
```

edited Aug 29 '12 at 20:13

answered Aug 29 '12 at 17:07

pb2q

32.6k 9 68 91

I have updated the question,I think its now clearer what i'm exactly after:) By the way thankyou:) – Hossein Aug 29 '12 at 17:08

You can use a trick to use scopes as you wish, just declare enum in such way:

```
struct Days
{
   enum type
   {
      Saturday,Sunday,Tuesday,Wednesday,Thursday,Friday
   };
};

Days::type day = Days::Saturday;
if (day == Days::Saturday)
```

answered Sep 11 '14 at 11:39

ataman1x

71 1 1

This should not work in C++:

```
Days.Saturday
```

Days is not a scope or object that contains members you can access with the dot operator. This syntax is just a C#-ism and is not legal in C++.

Microsoft has long maintained a C++ extension that allows you to access the identifiers using the scope operator:

```
enum E { A, B, C };

A;
E::B; // works with Microsoft's extension
```

But this is non-standard before C++11. In C++03 the identifiers declared in an enum exist only in the same scope as the enum type itself.

```
A;
E::B; // error in C++03
```

C++11 makes it legal to qualify enum identifiers with the enum name, and also introduces enum classes, which create a new scope for the identifiers instead of placing them in the surrounding scope.

```
A;
E::B; // legal in C++11

enum class F { A, B, C };

A; // error
F::B;
```

edited Aug 29 '12 at 17:31

answered Aug 29 '12 at 17:18

bames53

48.8k 4 64 128

Sadly, elements of the enum are 'global'. You access them by doing `day = Saturday` . That means that you cannot have `enum A { a, b } ;` and `enum B { b, a } ;` for they are in conflict.

answered Aug 29 '12 at 17:10

Grzegorz
2,116 1 4 20

2 Until you use `enum class` in C++11, that is. Before that, you have to make dummy classes. – chris Aug 29 '12 at 17:11

Don't know C++11. I am assuming the question refers to C++. Yes, using classes or namespaces will do the trick. – Grzegorz Aug 29 '12 at 17:13

@Grzegorz: i think chris is referring to the newly introduced enum class which provides strongly typed enums. – Hossein Aug 29 '12 at 17:20

@Hossein: Thank you for pointing it out. I have found explanation of the num class, and I know what Chris was talking about. Thanks a lot. – Grzegorz Aug 29 '12 at 17:24

@Grzegorz:I didn't mean to disrespect,just thought i might be helping,sorry for any probable misunderstanding.I again Thank you for your time and helping me ;) – Hossein Aug 29 '12 at 17:27

Rather than using a bunch of if-statements, enums lend themselves well to switch statements

I use some enum/switch combinations in the level builder I am building for my game.

EDIT: Another thing, I see you want syntax similar to;

```
if(day == Days.Saturday)
etc
```

You can do this in C++:

```
if(day == Days::Saturday)
etc
```

Here is a very simple example:

EnumAppState.h

```
#ifndef ENUMAPPSTATE_H
#define ENUMAPPSTATE_H
enum eAppState
{
    STARTUP,
    EDIT,
    ZONECREATION,
    SHUTDOWN,
    NOCHANGE
};
#endif
```

Somefile.cpp

```
#include "EnumAppState.h"
eAppState state = eAppState::STARTUP;
switch(state)
{
case STARTUP:
    //Do stuff
    break;
case EDIT:
    //Do stuff
    break;
case ZONECREATION:
    //Do stuff
    break;
case SHUTDOWN:
    //Do stuff
    break;
case NOCHANGE:
    //Do stuff
    break;
}
```

edited Aug 29 '12 at 17:16 answered Aug 29 '12 at 17:10


Dean Knight
452 3 13

The nice thing here is that compilers will tell you if you missed putting a case in. – chris Aug 29 '12 at 17:11

Shouldnt you use class enum in this case? – Hossein Aug 29 '12 at 17:12

1 enum is just a datatype in C++ So declaring an enum like I did above in a .h file, and then including that file in whatever .cpp file you want to use it in will give you access to the enum. Just noticed I forgot to add the #include in my .cpp example. Editing. – Dean Knight Aug 29 '12 at 17:15

Also, I see someone else saying that enums in C++ are global. In my experience, using enums the way I have above, I can only access them when I have included the .h. So this seems to stop global access too, which is always good. EDIT: It seems like I am unknowingly using enums in a C++11 way if I am reading things right... – Dean Knight Aug 29 '12 at 17:20

Enums in c++ are like integers masked by the names you give them, when you declare your enum-values (this is not a definition only a hint how it works).

But there are two errors in your code:

1. spell `enum` all lower case
2. you don't need the `Days.` before Saturday.
3. If this enum is declared in a class, then use `if(day==YourClass::Saturday){}`

answered Aug 29 '12 at 17:09


bali182
4,581 1 13 37

I think your root issue is the use of `.` instead of `::`, which will use the namespace.

Try:

```
enum Days {Saturday, Sunday, Tuesday, Wednesday, Thursday, Friday};
Days day = Days::Saturday;
if(Days::Saturday == day)  // I like literals before variables :)
{
    std::cout<<"Ok its Saturday";
}
```

answered Jun 6 at 19:30


VenomFangs
3,928 5 28 60

---

first of all make 'E' in enum, 'e' as a lower case. second drop type name 'Days' in 'Days.Saturday'. Third ...buy yourself a good c++ book.

answered Aug 29 '12 at 17:11

vikramjitSingh
26 5
