Static variables in C++

Asked 9 years, 1 month ago Active 2 years, 6 months ago Viewed 132k times

62

33

I would like to know what is the difference between static variables in a header file vs declared in a class. When static variable is declared in a header file is its scope limited to .h file or across all units. Also generally static variable is initialized in .cpp file when declared in a class right? So that does mean static variable scope is limited to 2 compilation units?

c++ static scope

edited Sep 13 '10 at 6:41
peSHIr
5,433 ● 1 ● 28 ● 44

asked Sep 13 '10 at 5:43
brett
2,159 ● 11 ● 33 ● 44

9 The 'static' keyword is very overloaded. It means differntly at different places. That's why it makes a fun question to ask at interviews. – vrdhn Sep 13 '10 at 7:40

Definitely among the top, together with abstract functions / abstract classes, and stuff like public / protected / private inheritance. ;-) – DevSolar Sep 13 '10 at 8:09

3 Answers

88

Excuse me when I answer your questions out-of-order, it makes it easier to understand this way.

> When static variable is declared in a header file is its scope limited to
>
> .h file or across all units.

There is no such thing as a "header file scope". The header file gets *included* into source files. The translation unit is the source file *including* the text from the header files. Whatever you write in a header file gets *copied* into each including source file.

As such, a static variable declared in a header file is like a static variable in each individual source file.

Since declaring a variable `static` this way means internal linkage, every translation unit `#include`ing your header file gets its **own**, **individual** variable (which is not visible outside your translation unit). This is usually not what you want.

> I would like to know what is the difference between static variables in a header file vs declared in a class.

In a class declaration, `static` means that all instances of the class *share* this member variable; i.e., you might have hundreds of objects of this type, but whenever one of these objects refers to the `static` (or "class") variable, it's the same value for all objects. You could think of it as a "class global".

> Also generally static variable is initialized in .cpp file when declared in a class right ?

Yes, *one* (and only *one*) translation unit must initialize the class variable.

> So that does mean static variable scope is limited to 2 compilation units ?

As I said:

- A header is not a compilation unit,
- `static` means completely different things depending on context.

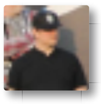
Global `static` limits scope to the translation unit. Class `static` means global to all instances.

I hope this helps.

PS: Check the last paragraph of Chubsdad's answer, about how you shouldn't use `static` in C++ for indicating internal linkage, but anonymous namespaces. (Because he's right. ;-))

edited Sep 13 '10 at 7:32

answered Sep 13 '10 at 6:19



DevSolar
50.5k ● 14 ● 104 ● 175

10 "static means completely different things depending on context." --> the very source of most confusion about it. This "not adding keywords" mindset is really annoying :(– Matthieu M. Sep 13 '10 at 9:51

@Matthieu M. Depends on your standpoint. For keeping some compatibility between C and C++, it's very beneficial. I admit, though, that they've overdone it a bit with `static`. – DevSolar Sep 13 '10 at 10:38

I know that compatibility was necessary, otherwise the language wouldn't have been as popular. However they could have used a new keyword for the C++ meaning, they could have made it "non-keyword" outside of a class/struct scope to preserve backward compatibility. I am glad that they did introduce the `nullptr` keyword in C++0x. – Matthieu M. Sep 13 '10 at 11:21

1 Well... how `static` is used in class scope *is* somewhat similar to how it is used in *function* scope (variable persistent across multiple uses). And they *did* introduce a way to get rid of another use of `static` (anon namespaces)... all in all, not too bad a job. Better than Java, in any case. :-D – DevSolar Sep 13 '10 at 11:24

47

Static variable in a header file:

say `'common.h'` has

```
static int zzz;
```

This variable `'zzz'` has internal linkage (This same variable can not be accessed in other translation units). Each translation unit which includes `'common.h'` has it's own unique object of name `'zzz'`.

Static variable in a class:

Static variable in a class is not a part of the subobject of the class. There is only one copy of a static data member shared by all the objects of the class.

class.

> $9.4.2/6 - "Static data members of a class in namespace scope have external linkage (3.5).A local class shall not have static data members."

So let's say `'myclass.h'` has

```
struct myclass{
   static int zzz;        // this is only a declaration
};
```

and `myclass.cpp` has

```
#include "myclass.h"

int myclass::zzz = 0           // this is a definition,
                               // should be done once and only once
```

and `"hisclass.cpp"` has

```
#include "myclass.h"

void f(){myclass::zzz = 2;}    // myclass::zzz is always the same i
                               // translation unit
```

and `"ourclass.cpp"` has

```
#include "myclass.h"
void g(){myclass::zzz = 2;}    // myclass::zzz is always the same i
                               // translation unit
```

So, class static members are not limited to only 2 translation units. They need to be defined only once in any one of the translation units.

> Note: usage of 'static' to declare file scope variable is deprecated and unnamed namespace is a superior alternate

edited Apr 30 '13 at 7:08

Mark Pearl
4,305 ● 7 ● 38 ● 55

answered Sep 13 '10 at 6:09

Chubsdad
19.8k ● 4 ● 51 ● 97

1 thank you very much for a nice answer. – brett Sep 13 '10 at 6:15

1 "Each file which includes"... is quite inexact, seeing that header files can include other header files. Better to stick with the phrase *compilation unit* or *translation unit*. – Ben Voigt Sep 13 '10 at 6:25

3 and +1 for pointing out that anonymous namespaces supersede the `static` modifier for global variables. – Ben Voigt Sep 13 '10 at 6:28

@Ben Voight: Yup, I will change it to translation unit. Old habits die hard...Thanks – Chubsdad Sep 13 '10 at 6:29

14

A static variable declared in a header file outside of the class would be `file-scoped` in every .c file which includes the header. That means separate copy of a variable with same name is accessible in each of the .c files where you include the header file.

A static class variable on the other hand is `class-scoped` and the same static variable is available to every compilation unit that includes the header containing the class with static variable.

answered Sep 13 '10 at 6:14

Goutham
579 ● 3 ● 10