Semicolon after class declaration braces

Asked 10 years, 9 months ago Active 10 months ago Viewed 35k times

74

17

In C++ classes, why the semi-colon after the closing brace? I regularly forget it and get compiler errors, and hence lost time. Seems somewhat superfluous to me, which is unlikely to be the case. Do people really do things like:

```
class MyClass
{
.
.
.
} MyInstance;
```

I get it from a C compatibility point of view for structs and enums, but since classes aren't part of the C language I guess it's primarily there the keep consistency between similar declaration constructs.

What I was looking for was more related to design rationale rather than being able to change anything, although a good code completion IDE might trap this before compilation.

c++ class oop declaration

edited Mar 24 '19 at 11:03
double-beep
3,273 ●8 ●22 ●33

asked Apr 24 '09 at 12:47
SmacL
20.9k ●10 ●83 ●141

4 This might help: cpptalk.net/… – Michael Haren Apr 24 '09 at 12:51

@Michael, thanks for the link. From a historical perspective it makes good sense, and if C++ allows all C grammer, and C++ classes are synonomous with structs, we are left with the necessary semi-colon at the end of the class. – SmacL Apr 24 '09 at 13:15

3 @Brian, yup serious question. I'm well aware I have to live with it but I curious about the rationale behind the design and implementation. – SmacL Apr 24 '09 at 14:20

Okay, but you perhaps should edit your question to include you wanted design rationale. As it is, it encourages people to ask questions like "why the curly brace"? :) You may be interested in reading Stroustrup's Design & Evolution of C++, although it covers more weightier matters than semi-colons at the end of classes. – Brian Neal Apr 24 '09 at 14:26

@Brian, fair enough, and it was borderline as to whether or not to wiki it. The question was asked after leaving out a semi-colon in a regularly used header in a large build. It cost me half an hour, hence the visit to SO. Question edited as per your suggestion. – SmacL Apr 24 '09 at 14:30

7 Answers

43

The semi-colon after the closing brace in a type declaration is required by the language. It's been that way since the earliest versions of C.

And yes, people do indeed do the declaration you just put up there. It's useful for creating scoped types inside of methods.

```
void Example() {
  struct { int x; } s1;
  s1.x = 42;
  struct ADifferentType { int x; };
}
```

In this case, I think it's clear why the semi-colons are needed. As to why it's needed in the more general case of declaring in the header file I'm unsure. My *guess* is that it's historical and was done to make writing the compiler easier.

edited Apr 24 '09 at 12:56

answered Apr 24 '09 at 12:50
JaredPar
620k ● 129 ● 1131 ● 1380

Why can't I just create scoped type without specifying MyInstance? It seams strange as you combine two actions: declaring new type and declaring new variable. – Mykola Golubyev Apr 24 '09 at 12:56

@Mykola you can do both. See the sample I added – JaredPar Apr 24 '09 at 12:58

60

The link provided by @MichaelHaren appears to provide the *root cause*. The semicolon (as others have pointed out) is inherited from C. But that doesn't explain why C used it in the first place. The discussion includes this gem of an example:

```
struct fred { int x; long y; };
main()
{
  return 0;
}
```

Older versions of C had an implicit int return type from a function unless declared otherwise. If we omit the `;` at the end of the structure definition, we're not only defining a new type `fred`, but also declaring that `main()` will return an instance of `fred`. I.e. the code would be parsed like this:

```
struct fred { int x; long y; } main()
{
  return 0; /* invalid return type, expected fred type */
}
```

edited May 10 '18 at 13:57

answered May 22 '14 at 14:53
Nathan
3,787 ● 1 ● 24 ● 34

Yeah the implicit int return type would eff everything up here. Nice gem – Gaspa79 Jul 25 '19 at 18:24

17

I guess it's because classes are declarations, even when they need braces for grouping. And yes, there's the historical argument that since in C you could do

```
struct
{
  float x;
  float y;
} point;
```

you should in C++ be able to do a similar thing, it makes sense for the `class` declaration to behave in the same way.

answered Apr 24 '09 at 12:51
unwind
340k ●56 ●424 ●553

10

It's short for

```
class MyClass
{
.
.
.
};

// instance declaration
MyClass MyInstance;  // semicolon here
```

The semicolon after the curly braces of the class declaration is actually overkill, but it is how C++ is defined. The semicolon after the variable declaration is always needed and makes sense.

edited Apr 24 '09 at 19:27

answered Apr 24 '09 at 12:52
Stefan Steinegger
58.3k ●14 ●119 ●183

1 So, does C++ require a semi-colon after each declaration ? – Loai Nagati May 25 '09 at 12:18

1 Note that, this way, you can't create an object of an anonymous class, while you can the other way. – Kevin Nov 9 '14 at 2:32

5

I do not use such declarations

```
class MyClass
{
.
.
.
} MyInstance;
```

But in this case I can understand why is semicolon there.
Because it is like `int a;` - variable declaration.

Probably for consistence as you can omit 'MyInstance' semicolon stays there.

answered Apr 24 '09 at 12:53
Mykola Golubyev
47.7k ●14 ●75 ●99

3

It is needed after a `struct` for compatibility reasons, and how would you like this:

```
struct MyStruct { ... };
class  MyClass  { ... }    //inconsistency
```

answered Apr 24 '09 at 14:30
Zifre
23.6k ●8 ●76 ●101

But what about `namespace myNamespace { ... } // inconsistent but valid` ? – Chris K. Jan 25 '19 at 12:12

3

In C/C++ the ; is a statement terminator. All statements are terminated with ; to avoid ambiguity (and to simplify parsing). The grammar is consistent in this respect. Even though a class declaration (or any block for that matter) is multiple lines long and is delimited with {} it is still simply a statement (the { } is part of the statement) hence needs to be terminated with ; (The ; is not a separator/delimitor)

In your example

```
class MyClass{...} MyInstance;
```

is the complete statement. One could define multiple instances of the declared class in a single statement

```
class MyClass{...} MyInstance1, MyInstance2;
```

This is completely consistent with declaring multiple instances of a primitive type in a single statement:

```
int a, b, c;
```

The reason one does not often see such desclaration of class and instance, is the instance could ?only? be a global variable, and you don't really often want global objects unless they are static and/or Plain Old Data structures.

answered Apr 25 '09 at 8:00
Roger Nelson
1,784 ●2 ●13 ●20

1 But function definition is a statement too but without semicolons. – Jiapeng Zhang Mar 2 '18 at 12:44