Do static members of a class occupy memory if no object of that class is created?

Say I have a class and I have a static member in it, but I don't create any objects of that type. Will the memory be occupied for the static variable? If it would be occupied, what is the point of putting it in a class?

c++ class static

edited Jan 30 '11 at 9:15
stakx
34.7k ●6 ●75 ●134

asked Jan 30 '11 at 9:04
Shen Xu
3,007 ●21 ●85 ●153

3 Answers

No.

static members don't belong to the instances of class. they don't increase instances and class size even by 1 bit!

```
struct A
{
    int i;
    static int j;
};
struct B
{
    int i;
};
std::cout << (sizeof(A) == sizeof(B)) << std::endl;
```

Output:

```
1
```

That is, size of `A` and `B` is exactly same. static members are more like global objects accessed through `A::j`.

See demonstration at ideone : http://www.ideone.com/YeYxe

$9.4.2/1 from the C++ Standard (2003),

> **A static data member is not part of the subobjects of a class. There is only one copy of a static data member shared by all the objects of the class.**

$9.4.2/3 and 7 from the Standard,

> **once the static data member has been defined, it exists even if no objects of its class have been created.**
>
> **Static data members are initialized and destroyed exactly like non-local objects (3.6.2, 3.6.3)**.

As I said, static members are more like global objects!

edited Jan 30 '11 at 9:27

answered Jan 30 '11 at 9:18
Nawaz
161k ●39 ●339 ●548

1 +1 for linking to definitions from the standard, basically outlines the answer in concrete. – Daniel Jan 30 '11 at 9:22

1 Thanks very informative – Shen Xu Jan 30 '11 at 10:45

The C++ standard doesn't explicitly state when static memory is allocated, as long as it is available on first use. That said, it is most likely allocated during program initialization, thus guaranteeing its presence as soon as it is required, without needing special-case code to detect and perform allocation on access.

The purpose of putting static data into a class is the same as putting any other data into classes. By putting the data into a class structure, you are defining an encapsulating namespace, as well as being able to control access using accessor and mutator methods; this, in turn, will allow you to validate data going into the static memory store, and to ensure consistency throughout the use of this data.

answered Jan 30 '11 at 9:08

lacqui
3,895 ● 12 ● 26

Actually, I think the standard treats static members like all globals, and their instanciations is submitted, as far as I understand, to the same law that the instanciations of other globals. That is the initialization order is required to be consistent within a translation unit and there is nothing said about the multiplexing with other translation units. – Matthieu M. Jan 30 '11 at 14:22

Static variables are stored in a special memory area called BSS, while instances of a class are stored in a heap or on a stack. So, static members are stored separately.

answered Jan 30 '11 at 9:10

Kos
336 ● 2 ● 3

6 Actually, the existence of BSS is system-dependent. The C++ standard doesn't define any such beast, nor any details of the implementation of data allocation. – lacqui Jan 30 '11 at 9:13