# Naming convention for params of ctors and setters

Asked 15 years, 11 months ago Modified 15 years, 4 months ago Viewed 2k times

8

For those of you who name you member variables with no special notation like `m_foo` or `foo_`, how do you name parameters to your ctors and setters?

Some options I've tried so far...

```
Obj(int foo) : foo(foo) { }
void set_foo(int foo) { this->foo = foo; }

Obj(int _foo) : foo(_foo) { }
void set_foo(int _foo) { foo = _foo; }

Obj(int a_foo) : foo(a_foo) { } // a for "argument"
void set_foo(int a_foo) { foo = a_foo; }

Obj(int init_foo) : foo(init_foo) { }
void set_foo(int new_foo) { foo = new_foo; }
```

c++ naming-conventions

Share
Improve this question
Follow

edited Jan 26, 2009 at 16:53

community wiki
3 revs, 3 users 100%
Iraimbilanja

## 12 Answers

Sorted by: Highest score (default)

9

I'm using foo_, it's better than _foo since it won't conflict with implementation specific function names and keywords.

Share Improve this answer Follow

answered Jan 16, 2009 at 16:00
vava
25.3k ● 11 ● 66 ● 79

indeed, underscores are reserved for C++ compiler implementations (+1) – xtofl Jan 16, 2009 at 16:01

it won't conflict anyway, because the only reserved names starting with _ are *global* ones ;)
– Iraimbilanja Jan 16, 2009 at 16:05

Some implementation specific stuff implemented with #define and macros can't tell is it suppose to be global or not. – vava Jan 16, 2009 at 16:13

Only double underscores at the start of names are reserved, a single underscore is guaranteed to never conflict – 1800 INFORMATION Jan 17, 2009 at 10:02

3 Wrong; see 17.4.3.1.2 in the Standard. Single underscores followed by uppercase letters are reserved, and anything that begins with an underscore in the global namespace is reserved. I prefer to remember a simple rule: don't start identifiers with underscores. – David Thornley Jan 26, 2009 at 17:16

6

I'm going with

```
Obj(int foo) : mFoo(foo) { }
void setFoo(int foo) { mFoo = foo; }
```

in my programs. For copy constructors and operator=, i tend to call it

```
Obj(Obj const& that):mFoo(that.mFoo) { }
```

For operators, i'm going with

```
Obj operator+(Obj const& lhs, Obj const& rhs) { ... }
```

Because those are the **l**eft **h**and **s**ide and the **r**ight **h**and **s**ide of it.

Share Improve this answer Follow

answered Jan 16, 2009 at 16:29

Johannes Schaub - litb
506k ● 131 ● 917 ● 1.2k

2

I avoid (by avoid mean never use) underscore as the first character of any identifier. I know its overkill but worth the effort.

Read this: [What are the rules about using an underscore in a C++ identifier?]()

Though not a rule I limit the use of underscore and prefer camel case to make my variables readable. But that's just a personal preference and I don't mind reading code that uses it.

Additionally I never name parameters the same as my member variables. The compiler will not help you catch the kind of errors this can generate (and this is all about getting the compiler to do the real work so you can do the expressive work the compiler can't do).

```
int X::setWork(int work)
{
    this->work = work;  // OK. But more Java like.

    work = work;        // Compiler won't see that problem.
}

int X::setWork(int newWork)
{
    work = newWork;    // A bit harder to get wrong.
}
```

edited May 23, 2017 at 11:48
Community Bot
1 ● 1

answered Jan 17, 2009 at 9:23
Loki Astari
264k ● 86 ● 342 ● 571

1 +1 for highlighting that naming the parameters the same as the members is just setting the code up for unnecessary bugs in the future. – Richard Corden Jan 21, 2009 at 9:40

1

I tend to follow the first letter of the parameter that is being set, and disambiguate with this...

```
void setFoo(int f) { foo = f; }
```

For a simple setter, for one variable, it is pretty well clear.

Also, I often do this

```
int setFoo(const int f) { foo = f; }
```

so I can string things together.

answered Jan 16, 2009 at 16:02
Arcane
1,228 ● 1 ● 8 ● 15

hmm I don't get how the latter lets you "string things together". It doesn't look like an implementation of the chaining idiom – Iraimbilanja Jan 16, 2009 at 16:06

I suppose it's kind of like the assignment operator, in that you can assign the value of setFoo to something else.. – Eclipse Jan 16, 2009 at 16:33

2 I assume that you meant to write { return foo = f; } – KeithB Jan 16, 2009 at 16:56

1

I always do this:

```
Obj(int foo) : foo(foo) {}
```

I used to play games with appending funny symbols until one time I got hit by this:

```
Obj(Bar& b) : b_(b_) {}
```

Can you see the mistake? (Yes, b_ is a private member reference variable). It compiled without a warning. Cost me 3 days of debugging (I was a green programmer then).

Now I always use the same name to avoid typos (and subsequent crashes) like that. There is no ambiguity inside the initialization list. This part of the language was designed just for this case, so take advantage of it.

Share Improve this answer Follow

answered Mar 20, 2009 at 13:11
Brian Neal
32.3k ● 7 ● 57 ● 60

0

I always go for a Param or Arg suffix but only when disambiguation is necessary.

```
Obj(int fooArg) : foo(fooArg)
```

Share Improve this answer Follow

answered Jan 16, 2009 at 15:58
Yann Semet
623 ● 2 ● 5 ● 12

But there is no ambiguity here. – Brian Neal Mar 20, 2009 at 13:12

0

I used to follow the Microsoft convention of prefixing member variables with `m_`, like `m_foo`. At my current company the convention is a trailing underscore for member variables: `foo_`.

Generally, if you are working by yourself, then use whatever convention you like. If you are working in a team, use whatever the team agrees upon. Overall consistency in a code base is what is important, not the particular convention.

Share Improve this answer Follow

answered Jan 16, 2009 at 17:37
Dima
39.3k ● 14 ● 78 ● 116

yeah i'm looking for something that's both (a) pretty and (b) not confusing for the average coder who'll be reading it. foo(foo) and foo(a_foo) probably fail the latter, while foo(init_foo) and foo(foo_) fail the former. heh – Iraimbilanja Jan 16, 2009 at 18:43

---

0

Number two has problems as a convention, although in your case it could be harmless. A name that has a leading underscore followed by an uppercase character is reserved for the implementation, and all names with leading underscores are reserved in a global context. If you never have class members beginning with uppercase letters (I don't), you're safe with the convention as shown (using _foo only as a function argument), but I dislike naming conventions that skirt anywhere near the limits.

Share Improve this answer Follow

answered Jan 26, 2009 at 17:22
David Thornley
57k ● 9 ● 95 ● 158

---

0

For classes:

```
Obj(int foo) : _foo(foo) {};
```

For structs:

```
obj_t(int foo_) : foo(foo_) {};
```

Setter:

```
Obj.setFoo(int foo) { _foo = foo; }
```

I'm with litb on the use of `lhs` and `rhs` for operator calls.

I use `camelCase` for class member functions, and `names_with_underscores` for struct fields and methods.

Share
Improve this answer
Follow

edited Mar 20, 2009 at 16:40

answered Jan 16, 2009 at 16:32
mskfisher
3,406 ● 4 ● 37 ● 49

I like this, except for the last one, why is it *foo = foo, and not foo* = foo? – Frank Mar 20, 2009 at 13:09

Because I made a mistake. :) The de-facto standard at my workplace is that private variables are prefixed with an underscore, so I got the setter code backward. I've fixed it. – mskfisher Mar 20, 2009 at 16:39

0

I name the actual members with trailing underscores, so I do this:

```
Foo(int bar) : bar_(bar) { }
```

The reason is so I can use getter functions without anything like getBar() (bar() is better).

Share
Improve this answer
Follow

answered Jul 29, 2009 at 0:07

community wiki
jkeys

0

I do it like this:

```
obj(int foo) : _foo(foo) { }
int foo() const { return _foo; }
void foo(int value) { _foo = value; }
```

The only trick here is to make sure that the letter following the underscore is lowercase. However, I avoid uppercase in identifier names everywhere, as it is inconsistent with conventions used by the standard library (which uses `foo_bar_baz` for all identifiers).

Share
Improve this answer
Follow

answered Jul 29, 2009 at 0:12

community wiki
Pavel Minaev

0

I follow the **Google C++ Style Guide**

> Variable names are all lowercase, with underscores between words. Class member variables have trailing underscores. For instance: my_exciting_local_variable, my_exciting_member_variable_.

Share

Improve this answer

Follow

answered Jul 29, 2009 at 0:52

community wiki
xian