

Forum:[Hack Forums](#)

Thread: [Would love some help on coding a 'simple' function \(counter\)](#)

Reporter - January 17, 2017, 9:30 pm

I have a code that calls an image to be visible

I would love to have a function that makes so every time the image gets called/'visible' it adds +1 to a label counter.

So lets say I have a label

Quote:Label = 0

Then the image gets called, and then it adds +1 to the label

Quote:Label = 1

Is this possible?

Can send over the project if needed ?

Mr Impact - January 17, 2017, 11:32 pm

In your function that calls the image to be visible add the increment operator to your label variable.

```
Code void CallImageFunction()
{
    // Call image to be visible
    Label++; // Increments the label every time the call image function runs
}
```

aeiro - January 26, 2017, 7:33 pm

(Jan 18, 2017 12:32 AM)Mr Impact Wrote: [CITE](#) In your function that calls the image to be visible add the increment operator to your label variable.

```
Code void CallImageFunction()
{
    // Call image to be visible
    Label++; // Increments the label every time the call image function runs
}
```

This is was i was thinking but instead of having it return void have it return Label so that way its possible to keep track of label outside of the function.

Code int counter = CallImageFunction();

```
int CallImageFunction()
{
    // Call image to be visible
    Label++; // Increments the label every time the call image function runs
    return Label;
}
```

caiveman - January 27, 2017, 3:02 am

```
Code int counter = 0;
void imagefunc(int* c){
    *c++; // haha im funny
    //image shit here
}
```

```
imagefunc(&counter);
```

Annabelle Lee - January 27, 2017, 5:53 am

(Jan 27, 2017 04:02 AM)caiveman Wrote: [\[CITE\]](#)

```
Code int counter = 0;
void imagefunc(int* c){
    *c++; // haha im funny
    //image shit here
}
```

imagefunc(&counter);This is probably the better way, since you don't need/shouldn't really be using a global variable.

0xDEAD10CC - January 28, 2017, 2:49 pm

(Jan 26, 2017 08:33 PM)aeiro Wrote: [\[CITE\]](#) This is was i was thinking but instead of having it return void have it return Label so that way its possible to keep track of label outside of the function.

```
Code int counter = CallImageFunction();
```

```
int CallImageFunction()
{
    // Call image to be visible
    Label++; // Increments the label every time the call image function runs
    return Label;
}
```

Honestly what's the point in doing this? You're exposing the label for no reason but encapsulating it within the function. That's a half-assed approach to writing manageable code.

I don't know why I'm seeing global variables either. And I also don't know what this "Label" is, sounds like Visual C++ to me.

What's wrong with a static local?

```
Code #include <iostream>
```

```
int increment_counter()
{
    static int x = 0;
    return ++x;
}

int main()
{
    std::cout << increment_counter() << '\n';
    std::cout << increment_counter() << '\n';
    std::cout << increment_counter() << '\n';
}
```

Maybe a bool reset condition in case you want to reset that. Doubt it needs to be thread-safe in this

case.

Praet0rian - January 30, 2017, 12:05 pm

(Jan 28, 2017 03:49 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Honestly what's the point in doing this? You're exposing the label for no reason but encapsulating it within the function. That's a half-assed approach to writing manageable code.

I don't know why I'm seeing global variables either. And I also don't know what this "Label" is, sounds like Visual C++ to me.

What's wrong with a static local?

Code `#include <iostream>`

```
int increment_counter()
{
    static int x = 0;
    return ++x;
}
```

```
int main()
{
    std::cout << increment_counter() << '\n';
    std::cout << increment_counter() << '\n';
    std::cout << increment_counter() << '\n';
}
```

Maybe a bool reset condition in case you want to reset that. Doubt it needs to be thread-safe in this case.

but aren't static variables a bad practice?

0xDEAD10CC - January 30, 2017, 1:41 pm

(Jan 30, 2017 01:05 PM)Praet0rian Wrote: [\[CITE\]](#) but aren't static variables a bad practice?

Definitely a better approach than a global variable. Btw, bad practice just about as much as a goto statement or any other debate that depends on context of usage. People who generalize the idea of bad practices just don't know why and when things are bad practice. If you don't need to thread this, then what would be the disadvantage? The variable is encapsulated, and if you need to modify how it gets incremented or add other things it's all within the function, rather than having to modify it in more than one place.

It's only an integer too.. Very small amount of memory.