Returning a pointer of a local variable C++

Asked 8 years, 5 months ago Active 4 years ago Viewed 30k times

13

9

I need to create a function that returns a pointer to an int.

Like so:

```
int * count()
{
    int myInt = 5;

    int * const p = &myInt;

    return p;
}
```

Since a pointer is simply an address, and the variable myInt is destroyed after this function is called. How do I declare an int inside this method that will keep a place in the memory in order for me to access it later via the returned pointer? I know I could declare the int globally outside of the function, but I want to declare it inside the function.

Thanks in advance for any help!

c++ function pointers

Share Edit Follow Flag

asked Sep 27, 2013 at 4:09
user906357
4,243 • 6 • 24 • 38

You cannot (as described in you question). Why not pass an integer pointer into the function (or better still use a reference)? – Ed Heal Sep 27, 2013 at 4:12

@EdHeal he can use `new int(something)` but that's really dumb so ... – aaronman Sep 27, 2013 at 4:13

2 To get a really meaningful answer, you're probably going to have to tell us more about how you intend to use the variable. For example, if I called the function twice in a row, should it return the address of the same variable both times, or a unique variable each time? – Jerry Coffin Sep 27, 2013 at 4:21

Add a comment

Start a bounty

5 Answers

Active Oldest Votes

12

Use the new operator

```
int * count()
{
    int myInt = 5;

    int * p = new int;
    *p = myInt;

    return p;
}
```

As pointed out in other answers this is generally a bad idea. If you must do it this way then maybe you can use a smart pointer. See this question for how to do this What is a smart pointer and when should I use one?

Share Edit Follow Flag

edited May 23, 2017 at 11:53
Community Bot
1 • 1

answered Sep 27, 2013 at 4:18
Nathaniel Johnson
4,458 • 1 • 38 • 67

9 This will leak memory unless he remembers to free it. – Jonathan Potter Sep 27, 2013 at 4:18

Yeah, I should really let someone who is a const pointer expert answer this. – Nathaniel Johnson Sep 27, 2013 at 4:20

3 IMHO - The thing that creates memory should be responsible for freeing it. This answer breaks that pattern. – Ed Heal Sep 27, 2013 at 5:08

How can I do this same thing with an array? i.e. myInt[10] – user906357 Sep 27, 2013 at 17:41

1 This doesn't answer the question, you're creating a new int on the heap every time this is called. The OP wants "an int inside this method that will keep a place in the memory in order for me to access it later via the returned pointer?". I.e. a *single* int in memory. @JonathanPotter's answer should be the accepted answer. – Steve Folly Oct 27, 2017 at 8:50

Add a comment | Show **1** more comment

9

You could use smart pointers.

For example:

```
unique_ptr<int> count()
{
   unique_ptr<int> value(new int(5));
   return value;
}
```

Then you can access the integer as follows:

```
cout << "Value is " << *count() << endl;
```

Share Edit Follow Flag

edited Feb 26, 2018 at 0:10
Arnav Borborah
10.6k • 6 • 35 • 75

answered Sep 27, 2013 at 4:37
Ed Heal
57.5k • 16 • 81 • 119

Add a comment

8

You can do this by making the variable `static`:

```
int* count()
{
    static int myInt = 5;
    return &myInt;
}
```

Share Edit Follow Flag

answered Sep 27, 2013 at 4:15
Jonathan Potter
34.9k • 4 • 73 • 73

4 It answers the question. Why would you pretend it's not possible when it is? – Jonathan Potter Sep 27, 2013 at 4:16

5 just because it's true doesn't mean it's helpful, if he doesn't understand static vars this will confuse him for hours – aaronman Sep 27, 2013 at 4:17

2 Read the question: "How do I declare an int inside this method that will keep a place in the memory in order for me to access it later via the returned pointer?" This is exactly what the `static` keyword does. – Jonathan Potter Sep 27, 2013 at 4:17

2 It's technically true but dangerous to just say without any more context (like the risks when mixed with threads). – nobody Sep 27, 2013 at 4:30

4 Given that the OP's other option was a global variable this is no more dangerous or worrisome. It gives him a global var with private scope, and I assume since that's what he asked for that's what he actually wants. I think people tend to over-analyse questions a bit on here at times. – Jonathan Potter Sep 27, 2013 at 4:31

Add a comment | Show **6** more comments

3

It is an error to return a pointer to a local variable. x points to a variable allocated on the heap:

```
link x = new node(a[m]);
Thus x isn't pointing to a local variable.
```

The reason that returning a pointer to a local variable is an error is that such a variable exists for only as long as the function is active (i.e. between it is entered and exited). Variables allocated on the heap (e.g. with the use of the new operator) exist until they are deallocated (e.g. with the delete operator).

Share Edit Follow Flag

answered Sep 27, 2013 at 4:17
Anne Staley
35 • 1

Add a comment

3

If you want to return a pointer of a variable correctly you have to do something like.

`int * myInt = new int(5);`

This is not a local variable BTW, meaning it does not have automatic storage and you have to `delete` the memory yourself

However using pointers like this is generally unnecessary and unadvised. It's better to create an int outside the function and have the function take a reference.

```
void count(int & i)
{
    i = 5;
}
```

BTW I don't know how you are planning to use the variable but since you also suggested using a global variable you may want to use a `static` var which @JonathanPotter suggested first. In many ways a static variable is similar to a global variable (both have static storage durations)

Share Edit Follow Flag

edited Sep 27, 2013 at 4:23

answered Sep 27, 2013 at 4:18
aaronman
17.7k • 6 • 57 • 78

@JonathanPotter my bad typo – aaronman Sep 27, 2013 at 4:20

Add a comment