

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2008-01/msg00140.html>

- *From:* Richard Heathfield <rjh@xxxxxxxxxxxxxxxx>
 - *Date:* Wed, 02 Jan 2008 21:15:24 +0000
-

jellybean stonerfish said:

For the record, I was boasting when I posted my score.

<grin>

<snip>

Question 8 is broken. All of the suggested answers except A exhibit undefined behaviour, and answer A modifies the value of x. There is no correct answer.

I will disagree with you here. Digging out my copy of the c++ programming language on page 125, stroustrup says that $y=++x$ is equivalent to $y=(x+1)$

Yes, but the suggested answer isn't $y=++x$, but $x=++x$ (or some trivial variation thereof – I don't have the original to hand). This violates the requirement that a value shall not be modified more than once between sequence points.

<snip>

Question 10 fails to account for pointer comparison, but is otherwise trivial.

It also fails to account that with objects the comparison operators can compare other things than numbers. As only one answer returned a boolean, I picked that answer. I still don't quite understand, is there a difference between 'logical values' and 'boolean'?

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

<shrug> The phrase "logical value" doesn't appear in the language definition. Boolean values are either true or false.

Question 28 has no good answer. There's no reason why you can't write a linked list class (although there's no point, because there's one in the STL). Nor is there any reason why you couldn't use structs to implement a linked list. Nor is there any reason why you can't construct a linked list that can hold only a single type. Nor is there any point in constructing a linked list that can only have a fixed number of elements.

I don't understand your statement. Why is answer 'c' (can hold any type of data) not a good answer?

Because it's trivial to write a linked list that can hold only a single type of data. (Yes, it's also trivial to write a linked list that can hold any type of data.)

Question 44 has no clearly correct answer. (A) is wrong because it doesn't access the base class function. (B) is wrong because it uses a class name in an object context. (C) is wrong unless the code exists within the class of which the function is a member.

Base_class is an object of the base class?

If so, it's a lousy name, wouldn't you agree? But (A) uses a member of the *derived* class, not a member of the base class.

<snip>

Questions 48–49 seem to use the word "composition" in a way I'm not familiar with. I think of the term as applying to the moving of a subexpression evaluation into an object that stores the subexpression to make subsequent evaluation of the whole expression simpler and quicker. It is not obvious that the suggested answers have anything to do with this! This may simply be because I'm not as up on C++ as the author of the quiz. Given the brokenness of the rest of it, however, I do not find that possibility to be terribly likely.

I believe 'composition' in this context means to put together parts to make a bigger object, so I picked 'c', the correct answer.

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

The term is "aggregation", and struct types (and arrays, which do a similar trick) are called "aggregate types". Composition is something else (see Stroustrup, which explains it reasonably well-ish).

—

Richard Heathfield <<http://www.cpax.org.uk>>

Email: rh@cpax.org.uk

Google users: <<http://www.cpax.org.uk/prg/writings/googly.php>>

"Usenet is a strange place" – dmr 29 July 1999

.