

My Courses

All Courses

About Contact



Help

vitorpbarbosa7



Course **Progress**

Course > Readings/Videos > Reading 9: Abstract Data Types > Questions



Questions

□ Bookmark this page

abstract data types

1/1 point (graded)

Consider an abstract data type Bool . The type has the following operations:

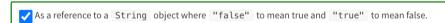
```
true : void → Bool
false : void → Bool
and : Bool \times Bool \rightarrow Bool
or : Bool × Bool → Bool
not : Bool → Bool
```

...where the first two operations construct the two values of the type, and last three operations have the usual meanings of logical and, logical or, and logical not on those values.

Which of the following are possible ways that Bool might be implemented, and still be able to satisfy the specs of the operations? Check all that apply.

✓ As a single bit, where 1 means true and 0 means fals	
	ρ

As an int value where 5 means true and 8 means false.



As a long value in which all possible values mean true.



Explanation

Bool can be implemented by virtually any kind of data structure, as long as it distinguishes the two values true and false. We just have to implement the five operations so that they satisfy their specs: for example, <code>and(true, false)</code> must return <code>false</code> .

That's the essence of what makes an abstract data type. The operations themselves (and their specs) completely define the data type, abstracting away from the details of data structure, memory storage, or implementation. It's a Bool data type because it provides these operations, not because of how it's actually stored inside the machine.

Submit

0

Answers are displayed within the problem

Previous Next >

△ ③ Some Rights Reserved

Open Learning Library Connect

About Contact Accessibility Twitter

All Courses Facebook

Why Support MIT Open Learning?

Help

© Massachusetts Institute of Technology, 2024