

# Re: enum safety

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2008-02/msg01125.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2008-02/msg01125.html)

---

- *From:* Ian Collins <[ian-news@xxxxxxxxxxxx](mailto:ian-news@xxxxxxxxxxxx)>
  - *Date:* Mon, 11 Feb 2008 07:07:05 +1300
- 

Keith Thompson wrote:

Ian Collins <[ian-news@xxxxxxxxxxxx](mailto:ian-news@xxxxxxxxxxxx)> writes:

Army1987 wrote:

Ian Collins wrote:

Sard wrote:

Page 39 K&R2 says

'Although variables of enum types may be declared, compilers need not check that what you store in such a variable is a valid value for the enumeration'

This is why (in my opinion) enums are horribly broken in C.

Not necessarily. For example, you can have an object which can contain a number up to 10 as numbers, and 11, 12 and 13 with special meanings. E.g.  
enum rank { ACE = 1, JACK = 11, QUEEN = 12, KING = 13 };  
enum rank foo = 7;

That illustrates the problem perfectly, you declare a valid range of values for rank and then legally assign some other arbitrary value to it. Useless.

It seems perfectly useful to me. An object of type enum range can sensibly hold any value from 1 to 13. The values 1, 11, 12, and 13

## Re: enum safety

happen to have names associated with them; the others are merely numbers. The language guarantees that this will work (in fact, it guarantees that values in the range 0..127 are valid).

A C enumerated type doesn't act like an enumerated type in, say, Pascal or Ada, but it's not supposed to. It merely provides a set of names for specific values, and an integral type that can hold all those values.

Or in C++, where they are first class types. A C enum is little different from a typedef and a set of #defines. It's the lack of type safety I don't like. Maybe it as was a bad experience with come code that tried to use enems as types, but ended up abusing them (assigning wrong values) that made me feel this way.

—  
Ian Collins.

.