Subject: Three problems with enumerations and one with type aliases
From: Van Snyder

1 Introduction

The purpose of this paper is to draw attention to three problems with enumerations, and a problem with type aliases that is related to the third problem with enumerations. The first problem is that there are several typos in section 4.7. The other three problems could be solved with minor syntax changes that would have no effect on semantics.

The second problem with enumerations is that one expects to declare far more enumerators than enumerations. The syntax is inverted on this issue, however: The keyword to introduce an enumerator is much longer than the keyword to introduce the enumeration of which it is a member. I propose to replace the enumerator declaration with a shorter keyword. Since enumerator declarations appear only within an enumeration definition, there is in the "asymptotic limit" no need for a keyword at all. I therefore propose that *enumerator-def-stmt* be replaced by *enumerator-list*. If this is unacceptable, change "ENUM" to "ENUMERATION" and change "ENUMERATOR" to "ENUM".

The third problem with enumerations and the problem with type aliases are related. Both enumerations and type aliase declarations introduce type aliases. I argued that they should both introduce new types, but I don't propose to re-examine those decisions at this time. I do propose, however, that the syntax of enumeration declarations should be modified to make it clear that they introduce type aliases, and type alias declarations should be modified to be parallel. I propose in the case of the *enum-def-stmt* that a mandatory attribute ", ALIAS" be added, and in the case of type aliases that they be spelled "TYPE, ALIAS" instead of "TYPEALIAS". An obvious syntactic mutation that might be applied in the future so that they declare new types is simply to allow the "ALIAS"attribute to be omitted.

2 Edits – typos in section 4.7

Edits refer to 00-007r1. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by immediately following text, while a page and line number followed by + indicates that immediately following text is to be inserted after the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

[Editor: This isn't really a type, but it's nearly in the same category. The terms "enumeration" and "enumerator" appear in the index, but the keywords don't. Put the keywords in the index.]	4.7
[Editor: There's a frame-ism (724) here.]	59:18
[Editor: `PARAMATER` ⇒ `PARAMETER`.]	59:38

3 Edits – to get rid of the ENUMERATOR keyword

[Editor: Replace *enumerator-def-stmt* by *enumerator-list* twice.]	58:18-19
[Editor: Delete.]	58:23

[Editor: Delete.] 58:26-27

```
ZERO, ONE, TWO
```

59:29

```
RED=4, BLUE=9
YELLOW
```

59:32-33

4 Edits – to make it clear that enumerations are aliases

These edits make it clear from the syntax that enumerations are aliases. It also makes it easy for a future revision of the standard to re-examine the question whether there ought to be a variety of enumeration declaration that introduces a new type, and provides a suggestion for the syntax: leave off the ALIAS attribute.

---

enum-def-stmt	**is** ENUM[*kind-selector*], ALIAS :: *type-alias-name* **or** ENUM, BIND(C), ALIAS :: *type-alias-name* **or** ENUM, ALIAS, BIND(C) :: *type-alias-name*	58:21-22

OR

enum-def-stmt	**is** ENUM[*kind-selector*], *enum-attr-list* :: ■ ■ *type-alias-name*	
enum-attr	**is** BIND(C) **or** ALIAS	

[Editor: Put the constraints at [58:25+] if you prefer.]

Constraint: If BIND(C) appears, the kind selector shall not appear.

Constraint: ALIAS shall appear.

5 Edits – minor change to spelling of TYPEALIAS

These edits change the syntax of type alias declarations to make it easy for a future revision of the standard to re-examine the question whether there ought to be a variety of type renaming declaration that introduces a new type, and provides a suggestion for the syntax: leave off the ALIAS attribute.

[Editor: "TYPEALIAS" ⇒ "TYPE, ALIAS".] 57:35

[Editor: "TYPEALIAS" ⇒ "TYPE, ALIAS".] 58:3

6 Why do the last two?

If we don't make the last two minor changes in syntax, and a future committee decides it would be desirable to have varieties of enumeration and type renaming declarations that introduce new types, we will have the following kludge:

"ENUM" and "TYPEALIAS" introduce type aliases, but one says "ALIAS" and the other does not. The natural syntax to introduce a new type identical to an existing one is "TYPE *new-type-name => old-type-spec*" but one couldn't simply use "ENUM" to create a new enumerated type, because that means to introduce an alias for integers. So something like "ENUM, NEWTYPE" would be needed to introduce new types. The "ENUM" and "TYPE" would then be weirdly antisymmetric. Syntactic structures that do similar things ought to be parallel. It would be better to use "ENUM, ALIAS" and "TYPE, ALIAS" to introduce type aliases, and leave the door open to allow later use of "ENUM" and "TYPE" to introduce new types.