JTC1/SC22/WG14 - N3141
Title: Composite Types v2
Author: Martin Uecker, Robert Seacord
Date: 2023-06-22

Change 1:

6.2.7 Compatible Type and Composite Type

3 A composite type can be constructed from two types that are compatible; **if both types are the same type, the composite type is this type. Otherwise,** it is a type that is compatible with both ~~**of the two types**~~ and satisfies the following conditions:

-- If both types are structure types or both types are union types, the composite type is determined recursively by forming the composite types of their members.
-- If both types are array types …
-- If both types are function types …
-- If one of the types has a standard attribute, the composite type also has that attribute.

These rules apply recursively to the types from which the two types are derived.

6.5.15

If ~~**both**~~ the second and third operands have arithmetic type, the result type ~~**that would be determined by**~~ **is the sames as if** the usual arithmetic conversions~~**,**~~ were ~~**they**~~ applied to **both** ~~**those two**~~ operands. ~~**, is the type of the result.**~~ If both the operands have structure or union type, the result **is the composite type**~~**has the type of one operand**~~. If both operands have void type, the result has void type

Change 2 (same type):

If any of the original types satisfies all requirements of the composite type, it is unspecified whether the composite type is one of these types or a different type that satisfies the requirements.**

****) The notion of "same type" affects redeclarations of typedef names and tags in the same scope.**

Change 3:

-- If both types are enumerated types, the composite type is an enumerated type.

Change 4 (new):

-- If at least one type is an enumerated type and the other type is an integer type, it is implementation-defined whether the composite type is an integer type or an enumerated type.