JTC1/SC22/WG14 - N3122
Title: Composite Type for Structures and Unions
Author: Martin Uecker
Date: 2023-04-22

The standard does not explicitly define a composite type for structures or unions, although such a type is needed for determining the result type of the conditional operator and when the same identifier is declared multiple times in the same scope. Change 1 adds an explicit rule for this missing composite type and uses it for the conditional operator. Change 2 adds a clarification about when the composite type might be the "same type" or not – leaving it unspecified. Change 3a and Change 3b propose wording for enumerated types. Change 3b also defines the composite type formed from an enumerated type and a compatible integer type as the enumerated type, but we point out that existing compilers disagree about this. Finally, it is suggested to clarify conversion rules for enumerated types.

Change 1 (composite type for structure and union types)

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 type 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 the usual arithmetic conversions, were they applied to 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 the same type as such a type or a different type that satisfies all requirements.**

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

Change 3a (enumerated types)

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

Change 3b (enumerated types, also including the case of integer and enumerated type)

-- If at least one type is an enumerated type, the composite type is an enumerated type.

Change 4 (conversion for enumerated types)

6.3.1.8 Usual arithmetic conversions

Otherwise, **if any of the two types is an enumerated type it is converted to its compatible standard integer type. Then,** the integer promotions are performed on both operands. Then the following rules are applied to the promoted operands:

Acknowledgment: I want to thank Jens Gustedt for comments and suggestions. Errors are all mine.