Doc No: X3J16 / 95-0109
WG21 / N0709
Date: May 30, 1995
Reply to: Ben Schreiber
Microsoft Corp.
bensc@microsoft.com

References in Unions

The Issues

* References are not required to have storage

Although all implementations I know of implement references as pointers, this is not required: "It is unspecified whether or not a reference requires storage." (8.3.2p3 [dcl.ref]). However, the definition of unions explicitly speaks of storage and allocation of members. Allowing references in unions implies that they do in fact have storage.

* References: Objects or Not?

The problem of whether references have storage extends to all class types. References are not object types (3.9 [basic.types] says that types describe objects, references and functions), so reference members are not sub-objects.

* What does it mean for a reference to have an indeterminate value?

What is the meaning of a reference member of a union when a value is assigned to a different member? Presumably its lifetime has ended (the storage has been re-used), thus according to 3.8p3 [basic.life] it has indeterminate value. This is the only case where you can get a reference with indeterminate value.

Option 1: Ban References in Unions

The simplest solution is to disallow references in unions. I am aware that this is an unpopular solution (certain national bodies have informed me that such a change would change their `Yes' vote to a `No' on the CD as a whole), and I am no longer convinced that this is the way to go, but I am presenting this for perspective.

Working Paper Changes If Adopted:

Add the following text to the end of paragraph 1 of 9.6 ([class.union]): "A union shall have no members of reference type."

Option 2: Minor Fixes to Handle References in Unions

Since this issue first came up, the object and memory models have been clearly defined. It is now possible to clear up these issues with small fixes, which is what I recommend.

* References: Objects or Not?

Change 8.3.2 [dcl.ref] paragraph 3 from:

> It is unspecified whether or not a reference requires storage (3.7).

to:

> References are not object types (3.9); in particular, it is not specified whether or not a reference requires storage (3.7). References have lifetimes (3.8), and are allowed as class data members (9.2); for these purposes references behave like objects.

* References are not required to have storage

Change the beginning of 9.6 [class.union] paragraph 1 from:

> A union can be thought of as a class whose member objects all begin at offset zero and whose size is sufficient to contain any of its member objects. At most one of the member objects can be stored in a union at any time.

to:

> A union is a class in which at most one of the data members can be stored at any time, i.e. the lifetimes (3.8) of the data members are disjoint. The size of a union shall be sufficient to contain the largest of its data members. All data members of a union are allocated at offset zero.

This change of wording serves three purposes: it explicitly refers to object lifetimes for the meaning of `inactive' members (members other than the one in use at a particular time), it uses the term "data member" in place of "member object," to be consistent with 9.2, thus allowing references, which do not necessarily have storage, and it eliminates the vague wording "can be thought of."

* What does it mean for a reference to have an indeterminate value?

Add the following text to the end of 3.8 [basic.life] paragraph 3:

> Before the lifetime of a reference starts or after its lifetime ends, accessing the reference results in undefined behavior. [Note: the only case where this can happen is if the reference is a member of a union (9.6).]

This wording explicitly separates the case of references from object types.