Doc. no.: X3J16 / 95-0033
WG21 / N0633
Date: 31 January 1995
Project: Programming Language C++
Reply to: Dag Brück
E-mail: `dag@dynasim.se`

Banning implicit int in C++ and C

1. Proposal

An object declared without an explicit type specifier gets a default type specifier of int. This is generally known as the "implicit int" rule of C and C++. I would like to investigate the possibilities to ban implicit int in C++ *and C.* The main concern is not to introduce any incompatibilities between C and C++, but to change the languages synchronously.

There are several reasons to ban implicit int:

- In C++, implicit int creates several opportunities for ambiguity between expressions involving function-like casts and declarations.
- Some novices now think that void is the default type of a function declared without a type
- Explicit declaration is widely recommended for clarity and overtness.
- It may help us produce better compiler diagnostics.
- The current C++ Working Paper deprecates the use of implicit int.

Banning of implicit int has already been discussed in WG21, but was not adopted primarily (my interpretation) because it would create an incompatibility with C. WG21 decided instead to follow C as closely as possible, except for banning it in `typedef`.

With C++ nearing completion and C undergoing revision, this seems to be a very good point in time to initiate common revisions in areas such as banning implicit int. The key concern is not to introduce any incompatibilities with the other language.

I have through Tom Plum asked for feedback on how WG14 would receive a formal request from Sweden to both WG21 and WG14 to ban implicit int, with the understanding that it must be done in both languages or none. I have already received informal support from several members of X3J16 and WG21. The issue was discussed at the December 1994 meeting of WG14, and Tom's report was:

> There is considerable support in WG14 for *either* banning implicit int or deprecating it, but it is too soon to tell which. There was definite overwhelming agreement that if C++ wants to ban implicit int that C will have no complaint.

Tom Plum and I agree that this is very positive feedback.

2. Impact on standardization processes

I think the synchronization between WG14 and WG21 can be achieved by the proposer (e.g., Sweden): if we see strong opposition in either group, we will withdraw it from the other group as well. I would like to emphasize that I believe consensus between the groups is essential.

With the expressed support from WG14, and the expected support from WG21 and X3J16, I think implicit int can be banned with little effort. The technical aspects have already been discussed by the core working group and to some extent in the extensions working group.

3. Changes to working paper

The central part of the working paper is 7.1.5 **[dcl.type]** paragraph 3, in which:

> If there is no type-specifier or is the only type-specifiers present in a decl-specifier-seq are cv-qualifiers, then the `int` specifier is assumed as default. Regarding the prohibition of default `int` specifier in `typedef` declarations, see 7.1.3; in all other instances, the use of decl-specifier-seqs which contain no simple-type-specifier (and thus default to plain `int`) is deprecated.

should be removed. There is a corresponding change to 7.1.3 **[dcl.typedef]** paragraph 1 that says that implicit int is not allowed in typedefs.