Doc No: X3J16/95-0104
WG21/N0704
Date: May 29, 1995
Project: Programming Language C++
Reply-To: Jim Welch
jww@watcom.on.ca

Implied Accessibility

and

Clarified Semantics for Catch Handlers

James W. Welch
WATCOM International
jww@watcom.on.ca

Abstract

This paper proposes additional accessibility requirements to be added to the working paper. The first two proposals are not considered to be controversial and may be editorial in nature. The third proposal, while substantive, is not considered to be controversial. The fourth proposal is substantive and may be controversial to some people. It adds accessibility requirements for object passed to catch handlers, specifies the order of destruction for such objects, and clarifies the effect to changes to these objects.

Proposal(1): Clause 5.3.4 [expr.new]

Add a new paragraph (paragraph 23) at the end of the section:

"The deallocation function to be used to free memory shall be accessible˝and not ambiguous."

Discussion: I consider this addition to be editorial.

Proposal(2): Clause 13.3 [over.match]

Add a new sentence to the end of paragraph 4:

"When overload resolution succeeds, the best viable function must be˝accessible in the context in which it is used."

Discussion: There exist many cases where accessibility must be checked and˝no explicit statement of such is given. The proposal attempts to place one such statement in a central˝location. If it is determined that this is too severe a restriction to place at this point, then the˝equivalent text must be inserted many times into the Working Draft. I consider this proposal to be editorial.

Proposal(3): Clause 15.1 [except.throw]

Add a new sentence to the end of paragraph 4:

"When the thrown object is a class object, the copy constructor used to initialize the temporary copy must be accessible. Similarly, the destructor for that object, if˝required, must be accessible. The copy constructor and destructor must be accessible even when the use of a˝temporary object can be eliminated."

Discussion: The first two sentences are redundant and are implied by the˝existing text. The sentences are included to add clarity. The third sentence adds consistency by requiring˝uniform accessibility rules independent of a particular implementation. I believe the addition of the˝third sentence is substantive.

Proposal(4): Clause 15.3 [except.handle]

Add two new paragraphs to the end of the section:

"When the catch handler specifies a class object, a copy constructor is˝used to initialize a temporary object which is bound to the optionally specified name in the˝exception-declaration for the catch handler. The object cannot be an abstract class (10.9) since these˝objects cannot be instantiated. That object is destructed when the handler is exited, after the destruction˝of any automatic objects initialized within the handler. The copy constructor and destructor must˝be accessible in the context of the catch handler and need not be accessible in the context of a˝throw-expression which is a match for the type of the catch handler. If the use of a temporary object can be˝eliminated without changing the meaning of the program except for execution of constructors and destructors˝associated with the use of the temporary object, then the optional name can be bound directly to the˝temporary (or original) object specified in a throw-expression causing the catch handler to be executed.˝ The copy constructor and destructor associated with the object must be accessible even when the˝temporary object is eliminated.

When the catch handler specifies a non-constant object, any changes to that˝object which are effected while the handler has not exited, are changes to the temporary˝copy for the handler and will not affect the temporary (or original) object that was initialized by˝execution of the throw-expression. When the catch handler specifies a reference to a non-constant object, any˝changes to the referenced object are changes to the temporary (or original) object initialized when the throw˝expression was executed and will have effect should that object be rethrown."

Discussion: The current draft does not specify any requirements regarding˝accessibility of copy constructors or destructors associated with the objects passed to catch˝handlers. The draft does not provide any rules for what happens when such handlers modify the objects˝passed to them. I believe the proposed requirements are consistent with existing practice and with user˝expectations; at least they should provide a starting point for the establishment of such rules. This˝third proposal is substantive in my opinion.

Conclusions

I believe this paper cleans up the remaining areas in the draft where˝accessibility requirements were understood but not yet stated. If I have overlooked any spots, I would be˝agreeable to revising this paper to cover them also. I am, of course, not invested in the specific wording˝in any cases above and expect those more qualified in drafting will be able to restate the proposals in˝an improved manner.