Accredited Standards Committee X3
Information Processing Systems
Operating under the procedures of
American National Standards Institute

Doc No: X3J16/96-0123 WG21/N0941
Date: May 28th, 1996
Project: Programming Language C++
Ref Doc:
Reply to: Josee Lajoie
(josee@vnet.ibm.com)

When is a function/object used?

556 - What does "An object/function is used..." mean?
======

1)FUNCTIONS:

3.2[basic.def.odr] para 2 says:

"A function is used if it is called, its address is taken, it is used to form a pointer to member, or it is a virtual member function that is not pure (_class.abstract_)."

It is not clear if the situations listed above refer to uses in the source code or refer to actual behaviors that happen at runtime. That is, does "if it is called" mean:

1) if a function call expression that calls the function is seen in the source code of the program (even if the program does not call the function at runtime), or
2) if the program actually calls the function at runtime.

```
int main() {
        extern int x;
        extern int f();
        return 0 ? x+f() : 0;
}
```

Are 'x' and 'f' used?

If meaning 1) is intended, we need to say what it means for functions "to be used" if calls to these functions are implicitly generated by the implementation (i.e. calls to conversion functions, special member functions, ...).

Proposed Resolution:

I believe solution 1) should be adopted.
It is the solution that guarantees consistency among different implementations. Adopting solution 2) means that a function would need to be defined only if it was actually called by the program at runtime. Whether a function is actually called by the program at runtime depends on the optimizations the implementation performs. This is implementation specific. This would mean that whether or not a function needs to be defined in a program would be implementation specific. I don't think this is very helpful for portability.

So, the words in 3.2[basic.def.odr] para 2 needs to be reworked to make it clear that "used" means "used in the source code".

Proposed new words:

"A function is used if:
- the function name, or an lvalue or a pointer that refers to the function, is the operand of a function call expression (5.2.2, _expr.call_),
- the function name or an lvalue that refers to the function is the operand of the unary & operator (5.3.1, _expr.unary.op_),
- the function is a member function and its name is used to form a pointer to member (5.3.1, _expr.unary.op_), or
- it is a virtual member function that is not pure

```
      (_class.abstract_).
  "

  Some additional wording is needed to take into account calls that are
  implicitly generated by the implementation (i.e. calls to conversion
  functions, special member functions, ...).

  Proposed new words:
    "A function is used if:
     -- it is a constructor and it is selected
        -- to implicitly create a class object of static or automatic
           storage duration (3.7.1, 3.7.2),
        -- to implicitly create a class object of dynamic storage
           duration (3.7.3) created by a new-expression (5.3.4),
        -- when the explicit type conversion syntax (5.2.3) is used,
        -- by function overload resolution to perform a conversion
           (13.3),

     -- it is a copy constructor and it is selected to perform the
        initialization
        -- in argument passing and in a function return (5.2.2),
        -- of the exception object in a throw-expression (15.1),
        -- of the exception-declaration in a catch handler (15.3),
        -- of a temporary object (12.2),

     -- it is a destructor and it is selected
        -- to destroy an object with static storage duration (3.7.1) at
           program termination (3.6.3)
        -- to destroy an object with automatic storage duration (3.7.2)
           when the block in which the object is created exits (6.7),
        -- to destroy a temporary object when the lifetime of the
           temporary object ends (12.2),
        -- to destroy an object allocated by a new-expression (5.3.4),
           through use of a delete-expression (5.3.5),
        -- in several situations due to the handling of exceptions
           (15.3),

     -- it is an assignment operator and it is selected
        -- to assign a value of its class type or a value of a class
           type derived from its class type to an object of its class
           type (see 12.8).
    "

  Did I forget anything?

  I know some standard purists do not like these kinds of lists because
  there is always the chance that the list is incomplete.  However, I
  believe such lists improve the readability and accessibility of the
  standard greatly.

2)OBJECTS:
  -------
  3.2[basic.def.odr] does not say anything about what it means for an
  object to be used.

  Proposed Resolution:
  --------------------
  Here again, I believe a definition for an object should be required
  if the source code for a the program "uses" the object.

  Proposed new words:
    "A non-local object with static storage duration is used if an
     expression uses the name of the object or uses an lvalue that
     refers to the object (other than as the operand of the sizeof
     operator)."
```

. . . . . .

427 - When is a diagnostic required when a function/variable with
      static storage duration is used but not defined?
======

1)FUNCTIONS:

  3.2[basic.def.odr] para 2 says:
    "Every program shall contain at least one definition of every
     function that is used in that program. That definition can appear
     explicitly in the program, it can be found in the standard or a
     user-defined library, or (when appropriate) it is implicitly
     defined (see _class.ctor_, _class.dtor_ and _class.copy_). If a
     non-virtual function is not defined, a diagnostic is required only
     if an attempt is actually made to call that function. If a virtual
     function is not defined and it is neither called nor used to form a
     pointer to member, no diagnostic is required."

  The sentence: "If a non-virtual function is not defined, a diagnostic
  is required only if an attempt is actually made to call that
  function." This seems to be hinting that, for cases such as the ones
  above, a diagnostic is not required.

  [Jerry Schwarz, core-6173:]
   I think we should be talking about undefined behaviors, not required
   diagnostics. That is, if a program references (calls it or takes its
   address) an undefined non-virtual function then the program has
   undefined behavior.

  [Fergus Henderson, core-6175, on Jerry's proposal:]
   I think that would be a step backwards. If a variable or function
   is used but not defined, all existing implementations will report a
   diagnostic. What is to be gained by allowing implementations to
   do something else (e.g. delete all the users files, etc.) instead?

  [Mike Ball, core-6183:]
   Then you had better not put the function definition in a shared
   library, since this isn't loaded until runtime. Sometimes linkers
   will detect this at link time and sometimes they won't.

  [Sean Corfield, core-6182:]
   I'd like it worded so that an implementation can still issue a
   diagnostic here (example above) AND REFUSE TO EXECUTE THE PROGRAM.
   If 'x' and 'f' were not mentioned in the program (except in their
   declarations) I would be quite happy that no definition is required.
   But unless an implementation can refuse to execute the program, you
   are REQUIRING implementations to make the optimisation and that is
   definitely a Bad Thing(tm), IMO. It seems the only way to allow that
   is to make the program ill-formed (under the ODR) but say no
   diagnostic is required.

  [Fergus Henderson, core-6174:]
   ObjectCenter reports a diagnostic only if an attempt is actually
   made to use the function or variable; in other words, link errors
   are not reported until runtime. In an interpreted environment, this
   is quite desirable.

  Proposed Resolution:

   There is another situation similar to this one already properly
   covered by the WP: access to copy constructors. The WP says that
   even if the call to a copy constructor is elided, the copy
   constructor must still be accessible.

   I believe the WP should have the same requirement for a function

definition if the function is "used" (i.e. as defined in resolution for issue 556) in a program, i.e., it must be defined. Of course, this should be a "no diagnostic is required" kind of rule since it is not always possible for the implementation to generate an error (dynamically linked libraries, etc.).

Proposed new words:

Add to the sentence in 3.2 para 2:
"Every program shall contain at least one definition of every function that is used in that program"

"; no diagnostic required."

And delete the last two sentences of the paragraph"
"If a non-virtual function is not defined, a diagnostic is required only if an attempt is actually made to call that function. If a virtual function is not defined and it is neither called nor used to form a pointer to member, no diagnostic is required."

2)VARIABLES:

3.2[basic.def.odr] para 3 says:
"A non-local variable with static storage duration shall have exactly one definition in a program unless the variable either has a built-in type or is an aggregate and unless it is either unused or used only as the operand of the sizeof operator."

Joe Coha mentioned in private email:
"Do I really need to have one definition of the static data member in the program? Even if it's unused? 9.4.2 says yes. However, this seems contradictory to the rules in 3.2. If a program is not required to define a non-local variable with static storage duration if the variable is not used, why is the WP requiring that the static data member be defined if it is not used?"

Proposed Resolution:

The requirements for variables with static storage duration (including static data members) should be similar to those for functions, i.e. if a variable is "used" (i.e. as defined in resolution for issue 556) in a program, it must be defined. Just like it is the case for functions, this should be a "no diagnostic is required" kind of rule.

Proposed new words:

Replace 3.2 para 3 with:
"A non-local object with static storage duration that is used in a program shall be defined; no diagnostic required. Only one definition shall be provided in the program."

9.4.2 para 5 should be modified to say:
"If a static data member is used in a program, exactly one definition of the static data member shall be provided (3.2)."