The :: Operator
SC22WG14 N654 — NCITS/J11 97-017

7 January 1997

Bill Seymour
wseymour@email.usps.gov

This paper proposes a **::** operator that would be a very limited version of the C++ scope resolution operator. It was originally part of the Member Functions proposal (N652 / 97-015), but was removed from that paper so that it could be accepted or rejected indepentently. Furthermore, this paper is in two parts which can be accepted or rejected indepentently of each other.

Part 1 proposes a unary **::** operator that may be used only within the body of a member function definition. Its use is limited to referencing objects or functions in the name space of ordinary identifiers that would otherwise be hidden by member objects or member functions. If the Member Functions proposal is not accepted, then part 1 of this proposal will be withdrawn.

Part 2 proposes a binary **::** operator which does not depend on acceptance of the Member Functions proposal. Because C structures and unions do not contain C++-style static members, the only use for this operator is in an expression that can be used as the operand of the **sizeof** operator. This is quite useful by itself, however, because it allows writing

```
sizeof type::member
```

instead of

```
sizeof ((type*)0)->member
```

the latter of which looks dangerously like the old **offsetof** hack until it is remembered that **sizeof**'s operand is not evaluated.

There is also a part 3 which includes all the changes to C9X that are included in parts 1 and 2 so that, if both the unary and binary versions of the operator are accepted, the redactor can find all the necessary changes in one place.

Part 1, the unary :: operator

6.1.5 Operators

Add the operator **::**

6.3.1 Primary expressions

[2] Change the grammar to:

primary-expression:
| **::**$_{opt}$ *identifier*
constant
string-literal
(*expression* **)**

Add the constraint:

A primary expression of the form **::identifier** shall not be used except within the block that forms the body of a member function definition.

Add to semantics:

An expression of the form **::identifier** is a primary expression, provided that the unadorned identifier would also be a primary expression, in which case **::identifier** designates an object or function in the name space of ordinary identifiers.[footnote]

And add the footnote:

Within the block that forms the body of a member function definition, **::identifier** designates an object or function that would, without the **::** operator, be hidden by a member object or member function.

Part 1 (continued)

6.3.1.2 The identifier this (added in Member Functions paper)

The Member Functions paper added the language:

> Within the body of a member function definition, any identifier that could denote a member of a structure of type *T*, is not already immediately preceded by **this->**, and is not otherwise in scope, shall be translated as if it were the primary expression, **(this->*identifier*)**.[footnote]
>
> [*footnote*] Thus, within a member function body, a structure member may be referenced without being preceded by **this->** unless it is hidden by another identifier.

This should be changed to:

> Within the body of a member function definition, any identifier that could denote a member of a structure of type *T* and is not already immediately preceded either by **this->** or by the **::** operator, shall be translated as if it were the primary expression, **(this->*identifier*)**.[footnote]
>
> [*footnote*] Thus, within a member function body, a structure member may be referenced without being preceded by **this->**; and structure members hide other identifiers that are not preceded by the **::** operator.

Part 2, the binary :: operator

6.1.5 Operators

Add the operator **::**

6.3.1 Primary expressions

[2] Change the grammar to:

primary-expression:
| *scope-specifier$_{opt}$ identifier*
constant
string-literal
(*expression* **)**

| *scope-specifier:*
| *struct-or-union$_{opt}$ identifier* **::**

Add the constraint:

A primary expression of the form **identifier::identifier**, **struct identifier::identifier** or **union identifier::identifier** shall not be used except as the operand of the **sizeof** operator.

Add to semantics:

An expression of the form **identifier::identifier**, **struct identifier::identifier** or **union identifier::identifier** is a primary expression, provided that the text to the left of the **::** operator designates a structure or union type and the identifier to the right designates a member object of such a structure or union, in which case the expression may be used as the operand of the **sizeof** operator.

Part 3, changes required if both forms of the operator are accepted

6.1.5 Operators

Add the operator **::**

6.3.1 Primary expressions

[2] Change the grammar to:

primary-expression:
| *scope-specifier*$_{opt}$ *identifier*
constant
string-literal
(*expression* **)**

| *scope-specifier:*
| **::**
| *struct-or-union*$_{opt}$ *identifier* **::**

Add the constraints:

A primary expression of the form **::identifier** shall not be used except within the block that forms the body of a member function definition.

A primary expression of the form **identifier::identifier**, **struct identifier::identifier** or **union identifier::identifier** shall not be used except as the operand of the **sizeof** operator.

Add to semantics:

An expression of the form **::identifier** is a primary expression, provided that the unadorned identifier would also be a primary expression, in which case **::identifier** designates an object or function in the name space of ordinary identifiers.[footnote]

An expression of the form **identifier::identifier**, **struct identifier::identifier** or **union identifier::identifier** is a primary expression, provided that the text to the left of the **::** operator designates a structure or union type and the identifier to the right designates a member object of such a structure or union, in which case the expression may be used as the operand of the **sizeof** operator.

And add the footnote to the **::identifier** paragraph above:

Within the block that forms the body of a member function definition, **::identifier** designates an object or function that would, without the **::** operator, be hidden by a member object or member function.

Part 3 (continued)

6.3.1.2 The identifier this (added in Member Functions paper)

The Member Functions paper added the language:

> Within the body of a member function definition, any identifier that could denote a member of a structure of type *T*, is not already immediately preceded by **this->**, and is not otherwise in scope, shall be translated as if it were the primary expression, **(this->*identifier*)**.[footnote]
>
> [*footnote*] Thus, within a member function body, a structure member may be referenced without being preceded by **this->** unless it is hidden by another identifier.

This should be changed to:

> Within the body of a member function definition, any identifier that could denote a member of a structure of type *T* and is not already immediately preceded either by **this->** or by the **::** operator, shall be translated as if it were the primary expression, **(this->*identifier*)**.[footnote]
>
> [*footnote*] Thus, within a member function body, a structure member may be referenced without being preceded by **this->**; and structure members hide other identifiers that are not preceded by the **::** operator.