Doc No: SC22/WG21/N1468
J16/03-0051
Date: 01-Apr-2003
Project: JTC1.22.32
Reply to: Daniel Gutson
danielgutson@hotmail.com

SELF METHODS

1. THE PROBLEM

There is no way to "force" and reflect in code when a method is "reflexive" by design, that is, when acts over the class itself and returns itself (this, or its self reference) (i.e. operators + =, ++, etc.).
In other words:

- There is no way to ensure that the implementation (or the sub classes') will return this (or *this as reference)
- From the interface (and API point of view), this intention is no explicit to the user.

Additionally, it is often common that self methods perform atomic operations (one statement length) that have to be followed (must) by a "return this" or "return*this" (in the case of returning the reference), which is redundant in the context of self methods.

- How are people addressing or working around the problem today?

All that can be done is to specify the class' type reference (or pointer) as return type, and add a comment about the intention, hoping that the implementers (either of the class and the subclasses) will respect it, returning "this" or "*this".

The categories this feature fits in are:

* Improve support for library building (provides self-documentation without the need of additional comments, saying: "this method is a self method: returns the same instance")

* Improve support for systems programming (ensuring that the implementations will always return the proper (self) instance, avoiding coding errors or design violations/misunderstandings) This is specially useful when the library provides a "visitor" interface to be implemented by the user.

* Remove embarrassments (self methods will contain the [atomic] operation only, without the need of any additional return).

2. THE PROPOSAL

Enable “this” or “*this” as return type keywords, for declaring a self method (returning “this” pointer and a reference to the instance respectively).

2.1. Basic cases

```
class IntWrapper
{
public
   *this operator ++ ()          { _value++; }

   *this operator += (int value) { _value+= value; }

   *this operator -> ()          { _calls++; }

private:
   int _value, _calls;
};
```

2.2. Advanced cases

Virtual self-methods

```
Struct Wrapper
{
   virtual *this operator ++ () = 0;
   virtual *this operator ++ (int) = 0;
   virtual *this operator = (int) = 0;
};

class MyWrapper: public Wrapper
{
   virtual *this operator++ ()
   { _value ++; }

   virtual MyWrapper& Operator ++ (int val) // error
   {…}

   virtual Wrapper& Operator = (int val) // error
   {…}
};
```

3. INTERACTIONS AND IMPLEMENTABILITY

3.1 INTERACTIONS

- Virtual self-methods are still self-methods in all the sub classes
- Self methods behave always in returning covariant types.
- Within a self method, the “return” keyword is optional, and when present, shall be alone (as far as “this” or “*this” is implicit.)
- Const self methods are declared as

```
    const *this [method] const;
```

or

```
const this [method] const;
```

The following declarations are invalid:

```
*this [method] const;

this [method] const;
```

3.2.IMPLEMENTABILITY

- The compiler should conceptually add the “return this” or “return *this” at the end of the method, or mimic such behavior through optimizations.

- This feature does not violate compatibility as far as the proposed syntax is currently invalid.