Bravely Default

Document number: P0481R0
Date: 2016-10-15
Audience: EWG
Reply-to: Tony Van Eerd. default at forecode.com

Acknowledgements

This proposal is based on the work of Bjarne Stroustrup and Jens Maurer, in particular P0221.

Summary - Differences with P0221

- for now, only proposing `==` (and `!=`) not `<` et al.
- new rules for when `==` (and `!=`) should be generated.

When to generate

The rule is simple: **Code that is OK with the default copy-constructor, is OK with default ==.**

Because copy is entwined with equality. ie

```
struct Foo { ... };

Foo foo;
Foo tmp = foo;

f(tmp);  // does the same as f(foo)
```

or

```
Foo Foo::operator++(int)
{
   Foo tmp = *this; // a copy!
   ++*this;
   return tmp; // copy is "same as" ie "equal to" old value, right???
}
```

Users expect copies to be substitutable with the original. Substitutability is a/the definition of equality.
Users expect copies to be the same (ie equal), that's why they are called "copies".

So, even if Foo contains raw pointers, if the default copy-ctor is "OK", then the default `==` is also OK.

I'm told a more technical wording might be
Generate == if the copy constructor is not user-provided.

When to generate `!=`

Whenever it is not user-provided, and the class has `==` (whether defaulted or user-provided).

What if I want default `==` but don't want `!=`? That's an odd situation. Declare `!=` as deleted.
What if I want default `==` but don't want `!=` and don't want it deleted? That's a really odd situation.

How to generate

Same as P0221

What about `<`

This proposal need not have much/any interaction with `<`. However, for those concerned about accepting one before the other, the stance from this proposal is that the ordering operators should only be opt-in (which EWG expressed interest in).

Straw-man syntax: `operator<() = default;`

Note that this is as similar to normal operator syntax and normal defaulting syntax as possible.
Also note the precedent of cast operators, which leave out the redundant return type (ie `operator int()() { return 17; }`) so the above syntax follows that lead (and extends it by leaving out the redundant input params as well).

Similar to when `!=` is generated, `>` would be generated if `<` exists.
`<=` would be generated when `<` and `==` exist.
`>=` would be generated when `>` and `==` exist.

They would be generated as in P0221.

Footnotes

- "if the copy constructor is not user-provided" is technical terminology from Jens, so I assume it is right
- "Bravely Default" is a video game for Nintendo 3DS. The title (for a video game) never made sense to me, but seems appropriate here. Since this proposal is a "sequel" of P0221, a more appropriate name might be "Bravely Second: End Layer" (Bravely Default's sequel), but then almost no one would get the reference.