Subject: One vs std::experimental::optional
Posted by piotr5 on Mon, 30 Jun 2014 19:03:00 GMT
View Forum Message <> Reply to Message

I discovered an interesting discussion about classes storing optional values. so I thought I might try the same problem-case with Upp::One

```
struct A
{
    constexpr A(int &x) : ref(x) {}
    int &ref;
};

int toptional()
{
    int n1 = 0, n2 = 0;
    One<A> opt(new A(n1));
    A* a=new A(n2);
    opt=a;
    opt->ref = 1;
    Cout() << n1 << " " << n2 << EOL;
}
```

the result is "0 1" as it should be. since upp isn't using a union, and it only works with values on heap (thereby being useless for constexpr values because of the required destructor), it's quite safe to use. I wonder why stdc++ wont implement it that way. what's the use of constexpr optional values anyway?

unfortunately I don't quite understand the things posted in above link. why isn't 1<2?