Subject: Re: About Nuller and Null
Posted by Tom1 on Sun, 11 Oct 2020 08:34:47 GMT
View Forum Message <> Reply to Message

Hi,

I sure have been overthinking, and then some! :)

I just wanted to basically return a null pointer (instead of a pointer to the result) when a function cannot solve a valid result. Then by checking for a null pointer, I could determine if the function succeeded or not.

When working on container classes based on e.g. Vector or Array classes, I would obtain the result as a reference to the item. Or the solution might fail, in which case I would return a null reference. But null references are not allowed or their behavior is undefined. So using null references is likely just asking for trouble.

Then I (naively) figured out Upp::Null and Nuller are just right for the purpose. However, it seems this is not the case. I cannot easily/safely return a Null object in place of a reference. The problems I have encountered while trying to work around the issue include:

- 'warning: returning a reference to a local or temporary object' when returning a T(Null) for an object
- returning null references are generally undefined and should not exist in C++
- There shall be no pointers to references in C++, which prevents changing my function to return pointers and null pointers alternatively

After quite some hours of tinkering, I came up with the Tuple2<bool,T> based solution to get a feeling of returning a pointer/null.

```
template <typename T>
struct Optional : public Tuple2<bool, T>{
 typedef Tuple2<bool, T> Base;
 Optional(T data) : Base(true,data) { }
 Optional() : Base(false,(T)Null) { }
 inline operator bool() const { return (bool)Base::a; }
 inline operator T(){ return (T)Base::b; }
 inline bool IsOK() const { return (bool)Base::a; }
 inline T Get(){ return (T)Base::b; }
 inline bool IsNullInstance() const{ return !IsOK(); }
};

// Usage:
//
// Optional<T> func(){
// if(success) return Optional<T>(value);
// else return Optional<T>();
// }
```

```
//
// In the calling function:
//
// Optional<T> result = func();
// if(!result) Cout() << "Failed, returned null\n";
// else Cout() << "Success, returned " << result << "\n";
```

Please note that this can return real references, if T is a reference.

If you see any flaws in this approach, or have a cleaner way to do it, please let me know.

Best regards,

Tom