Verifying a CRTP parameter

Tony Thompson

tony.thompson@sig.com

C++@SIG

1/16/2020

SIG
SUSQUEHANNA

About Me

- C++ since 2013, industry since 2016, SIG since 2017
  - Physics data acquisition/analysis, radar systems, distributed monitoring systems, library design
- Software Developer on SMI team in Enterprise Technology
  - LibSMI (Cross-platform C++ messaging middleware library with C# and Python bindings)
  - SMI Agent (running on >10,000 SIG servers/desktops)
  - SMI AlertEngine, SMI Ganglia Adapter, SMI Router, SMI JsonAdapter, etc.
  - Maintainer/user of Jolt (STL/boost extensions for SIG), UnitTest libraries in plus toolchains
- Resident elasticsearch/Kibana power user

Have you ever written code like this?

```
VariantLikeType variant;
switch (variant.GetType()) {
    case INT: {
        auto value = variant.as<int>();
        // do something with value as int
    }
    break;
    case STRING {
        auto const & value = variant.as<std::string>();
        // do something with string
    }
    break;
    // etc...
}
```

Curiously Recurring Template Pattern

```
template <class Derived>
struct base {
   void apply(VariantLikeType const & _variant){
      auto & derived(static_cast<Derived &>(*this));
      switch (_variant.GetType()){
         case INT: {
            auto const & value = _variant.as<int>();
            derived.template on_apply<int>(value);}
         break;
         // other types...
      }
   }
};

struct A : base<A> {
   template <class T>
   void on_apply(T const & _value) {
      //... do generic things with _value
   }
};
```

Usage

```
VariantLikeType variant;
//...
A decoder;
//calls A::on_apply(T const &) with the relevant T
decoder.apply(variant);
```

But wait! What does the following do?

```
struct A : base<A> {
   template <class T>
   void on_apply(T const & _value) {
      //... do generic things with _value
   }
};

struct B : base<A> { // can the compiler/you detect this?
   //...
};

VariantLikeType variant;
//...
B decoder;
decoder.apply(variant); // ??
```

Solution for detecting in Debug builds with RTTI

```
template <class Derived>
struct base {
   void apply(VariantLikeType const & _variant){
      ASSERT(typeid(*this) == typeid(Derived));
      auto & derived(static_cast<Derived &>(*this));
      // ...
   }

#ifdef DEBUG
   virtual void _() {}
#endif
};
```

Questions?

SIG
SUSQUEHANNA