# STD::VARIANT & BLOCKCHAIN

THOMAS CATALANO

@tomsnode

THE ROLE & IMPORTANCE

TO VERIFY A PREDEFINED RANGE OF SUMS

$\log(n)$

TO ACCOMPLISH THIS

WE NEED TO ALLOCATE A SPECIFIED RANGE

FIND THE MAXIMUM VALUE OF THE UNARY METAFUNCTION F OVER SEQUENCE

COMPILE TIME POLYMORPHISM

CACHE MANAGING

## BOOST::VARIANT

Annoying runtime checks

Calling outside library boost::optional for templates on types and create new objects

```
template < class TIn, class TOut >

class TofferStreamBase
{
...
protected:
    Toffer<TOut>offer_;
    boost::optional<TOut>ownerFunds;
```

templates on types                create new object std::variant in C++17

Time denotes a strong pointer

Frequently used objects lock the cache

Cache holds strong & weak pointers

Tuple helper class

```
template <typename _Array_type, typename
    _Variant_type, typename _Index_seq>

struct  __gen_vtable_impl;

template <typename _Result _type, typename
    _Visitor, size_t… __dimensions...>,
    _Multi_array<_Result_type (*) (Visitor,
    _Variants…), __dimensions, typename..
    __Variants, size_t... __indices>

struct  __gen_vtable_impl<
        _Multi_arrary<_Result_type (*) (_Visitor,
        _Variants…) __dimensions...>,
        tuple<_Variants...>, std::index_sequence
```

*In the end key points to remember*

Compile time polymorphism -by- overloading -for- cache sync verification

Metaprograming utilities

Libraries like boost/mpl
or
bits/enable_special_member.h

thank you

C++Now 19

Aspen, Colorado