TODO:

A Lightning Talk

Daniel Frey

2018-02-15

Typical TODOs

```
int main()
{
    // TODO: World domination
    // TODO: Get a cat
}
```

Typical Problem

- “We’ll fix them later.”
- …and “later” never happens.
- In my experience any sufficiently large code base contains a large number of TODOs that just rot (until someone deletes them without fixing anything).

Observe!

```
#include <iostream>

int main()
{
    std::cout << __DATE__ << std::endl;
}
```

```
$ g++ t.cpp && ./a.out
Feb 15 2018
$
```

Prepare Stuff 1/5

```
namespace todo
{
    class ct_string
    {
        const char* const begin_;
        const unsigned size_;

    };
```

Prepare Stuff 1/5

```
namespace todo
{
    class ct_string
    {
        const char* const begin_;
        const unsigned size_;

    public:
        template< unsigned N >
        constexpr ct_string( const char ( &arr )[ N ] )
            : begin_( arr ), size_( N - 1 )
        {
        }




};
```

Prepare Stuff 1/5

```
namespace todo
{
   class ct_string
   {
      const char* const begin_;
      const unsigned size_;

   public:
      template< unsigned N >
      constexpr ct_string( const char ( &arr )[ N ] )
         : begin_( arr ), size_( N - 1 )
      {
      }

      constexpr unsigned size() const
      {
         return size_;
      }



};
```

Prepare Stuff 1/5

```
namespace todo
{
   class ct_string
   {
      const char* const begin_;
      const unsigned size_;

   public:
      template< unsigned N >
      constexpr ct_string( const char ( &arr )[ N ] )
         : begin_( arr ), size_( N - 1 )
      {
      }

      constexpr unsigned size() const
      {
         return size_;
      }

      constexpr char operator[]( const unsigned i ) const
      {
         return begin_[ i ];
      }
   };
```

Prepare Stuff 2/5

```
constexpr unsigned check( const unsigned i, const unsigned l )
{
   return ( i < l ) ? i : throw "out of range";
}
```

Prepare Stuff 2/5

```
constexpr unsigned check( const unsigned i, const unsigned l )
{
   return ( i < l ) ? i : throw "out of range";
}

constexpr unsigned digit( const ct_string s, const unsigned i )
{
   return ( s[ i ] >= '0' && s[ i ] <= '9' )
        ? ( s[ i ] - '0' )
        : throw "invalid character";
}
```

Prepare Stuff 2/5

```
constexpr unsigned check( const unsigned i, const unsigned l )
{
   return ( i < l ) ? i : throw "out of range";
}

constexpr unsigned digit( const ct_string s, const unsigned i )
{
   return ( s[ i ] >= '0' && s[ i ] <= '9' )
        ? ( s[ i ] - '0' )
        : throw "invalid character";
}

constexpr unsigned two( const ct_string s, const unsigned i )
{
   return digit( s, i ) * 10 + digit( s, i + 1 );
}
```

Prepare Stuff 2/5

```
constexpr unsigned check( const unsigned i, const unsigned l )
{
   return ( i < l ) ? i : throw "out of range";
}

constexpr unsigned digit( const ct_string s, const unsigned i )
{
   return ( s[ i ] >= '0' && s[ i ] <= '9' )
        ? ( s[ i ] - '0' )
        : throw "invalid character";
}

constexpr unsigned two( const ct_string s, const unsigned i )
{
   return digit( s, i ) * 10 + digit( s, i + 1 );
}

constexpr unsigned four( const ct_string s, const unsigned i )
{
   return two( s, i ) * 100 + two( s, i + 2 );
}
```

Prepare Stuff 3/5

```
// __DATE__ => "Feb 15 2018", note: format guaranteed by the C++ standard!
```

Prepare Stuff 3/5

```
// __DATE__ => "Feb 15 2018", note: format guaranteed by the C++ standard!

constexpr unsigned day_cpp( const ct_string s )
{
   return two( s, 4 ) - 1;
}
```

Prepare Stuff 3/5

```
// __DATE__ => "Feb 15 2018", note: format guaranteed by the C++ standard!

constexpr unsigned day_cpp( const ct_string s )
{
   return ( ( s[ 4 ] == ' ' ) ? digit( s, 5 ) : two( s, 4 ) ) - 1;
}
```

Prepare Stuff 3/5

```
// __DATE__ => "Feb 15 2018", note: format guaranteed by the C++ standard!

constexpr unsigned day_cpp( const ct_string s )
{
   return ( ( s[ 4 ] == ' ' ) ? digit( s, 5 ) : two( s, 4 ) ) - 1;
}

constexpr unsigned month_cpp( const ct_string s )
{
      return ( s[ 0 ] == 'J' && s[ 1 ] == 'a' && s[ 2 ] == 'n' ) ? 0
           : ( s[ 0 ] == 'F' && s[ 1 ] == 'e' && s[ 2 ] == 'b' ) ? 1
           : ( s[ 0 ] == 'M' && s[ 1 ] == 'a' && s[ 2 ] == 'r' ) ? 2
           : ( s[ 0 ] == 'A' && s[ 1 ] == 'p' && s[ 2 ] == 'r' ) ? 3
           : ( s[ 0 ] == 'M' && s[ 1 ] == 'a' && s[ 2 ] == 'y' ) ? 4
           : ( s[ 0 ] == 'J' && s[ 1 ] == 'u' && s[ 2 ] == 'n' ) ? 5
           : ( s[ 0 ] == 'J' && s[ 1 ] == 'u' && s[ 2 ] == 'l' ) ? 6
           : ( s[ 0 ] == 'A' && s[ 1 ] == 'u' && s[ 2 ] == 'g' ) ? 7
           : ( s[ 0 ] == 'S' && s[ 1 ] == 'e' && s[ 2 ] == 'p' ) ? 8
           : ( s[ 0 ] == 'O' && s[ 1 ] == 'c' && s[ 2 ] == 't' ) ? 9
           : ( s[ 0 ] == 'N' && s[ 1 ] == 'o' && s[ 2 ] == 'v' ) ? 10
           : ( s[ 0 ] == 'D' && s[ 1 ] == 'e' && s[ 2 ] == 'c' ) ? 11
           : throw "invalid month";
   }
```

Prepare Stuff 4/5

```
// ISO date => "2018-02-15"
```

Prepare Stuff 4/5

```
// ISO date => "2018-02-15"

constexpr unsigned month_iso( const ct_string s )
{
   return check( two( s, 5 ) - 1, 12 );
}
```

Prepare Stuff 4/5

```
// ISO date => "2018-02-15"

constexpr unsigned month_iso( const ct_string s )
{
   return check( two( s, 5 ) - 1, 12 );
}

constexpr bool is_leap_year( const unsigned y )
{
   return ( y % 4 == 0 ) && ( ( y % 400 == 0 ) || ( y % 100 != 0 ) );
}
```

Prepare Stuff 4/5

```
// ISO date => "2018-02-15"

constexpr unsigned month_iso( const ct_string s )
{
   return check( two( s, 5 ) - 1, 12 );
}

constexpr bool is_leap_year( const unsigned y )
{
   return ( y % 4 == 0 ) && ( ( y % 400 == 0 ) || ( y % 100 != 0 ) );
}

constexpr unsigned days( const unsigned y, const unsigned m )
{
   return ( m == 1 )
        ? ( is_leap_year( y ) ? 29 : 28 )
        : ( ( m == 3 || m == 5 || m == 8 || m == 10 ) ? 30 : 31 );
}
```

Prepare Stuff 4/5

```
// ISO date => "2018-02-15"

constexpr unsigned month_iso( const ct_string s )
{
   return check( two( s, 5 ) - 1, 12 );
}

constexpr bool is_leap_year( const unsigned y )
{
   return ( y % 4 == 0 ) && ( ( y % 400 == 0 ) || ( y % 100 != 0 ) );
}

constexpr unsigned days( const unsigned y, const unsigned m )
{
   return ( m == 1 )
        ? ( is_leap_year( y ) ? 29 : 28 )
        : ( ( m == 3 || m == 5 || m == 8 || m == 10 ) ? 30 : 31 );
}

constexpr unsigned day_iso( const ct_string s )
{
   return check( two( s, 8 ) - 1, days( four( s, 0 ), month_iso( s ) ) );
}
```

Prepare Stuff 5/5

```
constexpr unsigned total_cpp( const ct_string s )
{
   return ( s.size() == 11 && s[ 3 ] == ' ' && s[ 6 ] == ' ' )
        ? four( s, 7 ) * 366 + month_cpp( s ) * 31 + day_cpp( s )
        : throw "invalid string";
}

constexpr unsigned total_iso( const ct_string s )
{
   return ( s.size() == 10 && s[ 4 ] == '-' && s[ 7 ] == '-' )
        ? four( s, 0 ) * 366 + month_iso( s ) * 31 + day_iso( s )
        : throw "invalid string";
   }

}  // namespace todo

#define TODO( DATE, MSG ) \
   static_assert( todo::total_cpp( __DATE__ ) < todo::total_iso( DATE ), MSG )
```

Using TODO

```
#include "todo.hpp"

int main()
{
    TODO( "2081-11-10", "World domination!" );
    TODO( "2018-01-01", "Get a cat" );
}
```

- Works at global scope, namespace scope, class scope, function scope, etc.

Using TODO (GCC)

```cpp
#include "todo.hpp"

int main()
{
    TODO( "2081-11-10", "World domination!" );
    TODO( "2018-01-01", "Get a cat" );
}
```

```
$ g++ -std=c++11 example.cpp && ./a.out
In file included from example.cpp:1:0:
example.cpp: In function ‘int main()’:
todo.hpp:123:4: error: static assertion failed: Get a cat
    static_assert( todo::total_cpp( __DATE__ ) < todo::total_iso( DATE ), MSG )
    ^
example.cpp:6:4: note: in expansion of macro ‘TODO’
    TODO( "2018-01-01", "Get a cat" );
    ^
$
```

Using TODO (Clang)

```cpp
#include "todo.hpp"

int main()
{
    TODO( "2081-11-10", "World domination!" );
    TODO( "2018-01-01", "Get a cat" );
}
```

```
$ clang++-5.0 -std=c++11 example.cpp && ./a.out
example.cpp:6:4: error: static_assert failed "Get a cat"
   TODO( "2018-01-01", "Get a cat" );
   ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
./todo.hpp:123:4: note: expanded from macro 'TODO'
   static_assert( todo::total_cpp( __DATE__ ) < todo::total_iso( DATE ), MSG )
   ^              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 error generated.
$
```

Thank you!

https://github.com/d-frey/todo

Questions?

https://github.com/d-frey/todo