Using variable templates on a tiny problem

By Paul Dreik
https://www.pauldreik.se/

Sweden C++ 2023-08-31

I had this code

```
1  void handle_data(uint32_t raw) {
2      interpret(raw & 0x3FFFFFFF);
3  }
```

Are we sure the mask contains the intended number of bits?
Can it be made more readable?

Digit separators (C++14)

```
1  void handle_data(uint32_t raw) {
2      interpret(raw & 0x3FFF'FFFF);
3  }
```

Better, but not really helping

Add a function

```
1  uint32_t make_mask(int Nbits) {
2      return (1ULL << Nbits) - 1;
3  }
4  void handle_data(uint32_t raw) {
5      interpret(raw & make_mask(30));
6  }
```

Much more readable!
Could we prevent misuse?
Can we avoid a function call?

Unfriendly on misuse

```
1  constexpr // does not help!
2  uint32_t make_mask(int Nbits) {
3     return (1ULL << Nbits) - 1;
4  }
5  make_mask(-1);      // no warning!
6  make_mask(33);      // no warning!
7  make_mask(0xFF); // no warning!
```

Using -Wall -Wextra with gcc and clang

Variable template (C++14)

```
1  template <int Nbits>
2  uint32_t mask_of{ (1ULL << Nbits) - 1 };
3
4  void handle_data(uint32_t raw) {
5     interpret(raw & mask_of<11>);
6  }
```

Error handling

```
1  template <int Nbits>
2  uint32_t mask_of{ (1ULL << Nbits) - 1 };
```

```
1  // gcc warns on narrowing, but compiles
2  // clang warns on narrowing+shift, compile error
3  auto x = mask_of<-1>;
```

```
1  // gcc and clang warns on narrowing and gives
2  // compile error, good!
4  auto y = mask_of<33>;
```

Can we get an even better error message?

Add constexpr!

```
1  template <int Nbits>
2  constexpr uint32_t mask_of{ (1ULL << Nbits) - 1 };
```

```
1  auto x = mask_of<-1>;        // compile error!
2  auto y = mask_of<33>;        // compile error!
3  auto z = mask_of<0xFF>;   // compile error!
```

```
<source>:34:28: error: non-constant-expression cannot be
narrowed from type 'unsigned long long' to 'uint32_t' (aka 'unsigned
int') in initializer list [-Wc++11-narrowing]
```

Can we get an even better error message?

Concepts (C++20)

```
1  template <int N>
2  concept is_within_32 = (N >= 0 && N <= 32);
3
4  template <int Nbits>
5  requires is_within_32<Nbits>
6  constexpr uint32_t mask_of{(1ULL << Nbits) – 1};
7
8  void handle_data(uint32_t raw) {
9      interpret(raw & mask_of<30>);
10 }
```

Error message (gcc)

```
<source>: In function 'void slide5::demo_misuse()':
<source>:57:33: error: use of invalid variable template 'mask_of<-1>'
   57 |       [[maybe_unused]] auto x = mask_of<-1>;
      |                                 ^~~~~~~~~~
<source>:57:33: note: constraints not satisfied
<source>: In substitution of 'template<int Nbits>  requires  is_within_32<Nbits> constexpr const uint32_t slide5::mask_of<Nbits> [with int Nbits = -1]':
<source>:57:33:   required from here
<source>:49:9:   required for the satisfaction of 'is_within_32<Nbits>' [with Nbits = -1]
<source>:49:27: note: the expression 'N >= 0 [with N = -1]' evaluated to 'false'
   49 | concept is_within_32 = (N >= 0 && N <= 32);
      |                         ~~^~~~
Compiler returned: 1
```

Error message (clang)

```
<source>:57:33: error: constraints not satisfied for variable template 'mask_of' [with Nbits = -1]
   57 |       [[maybe_unused]] auto x = mask_of<-1>;
      |                                 ^~~~~~~~~~
<source>:52:14: note: because -1 does not satisfy 'is_within_32'
   52 |     requires is_within_32<Nbits>
      |              ^
<source>:49:25: note: because '-1 >= 0' (-1 >= 0) evaluated to false
   49 | concept is_within_32 = (N >= 0 && N <= 32);
      |                         ^
1 error generated.
Compiler returned: 1
```

Links

- Code: https://godbolt.org/z/v7GTGvzdE
- Personal website: https://www.pauldreik.se/
- My company: https://www.dreik.se/
- Github: https://github.com/pauldreik