Make Error Handling the Users' Problem

(They will thank you for it)

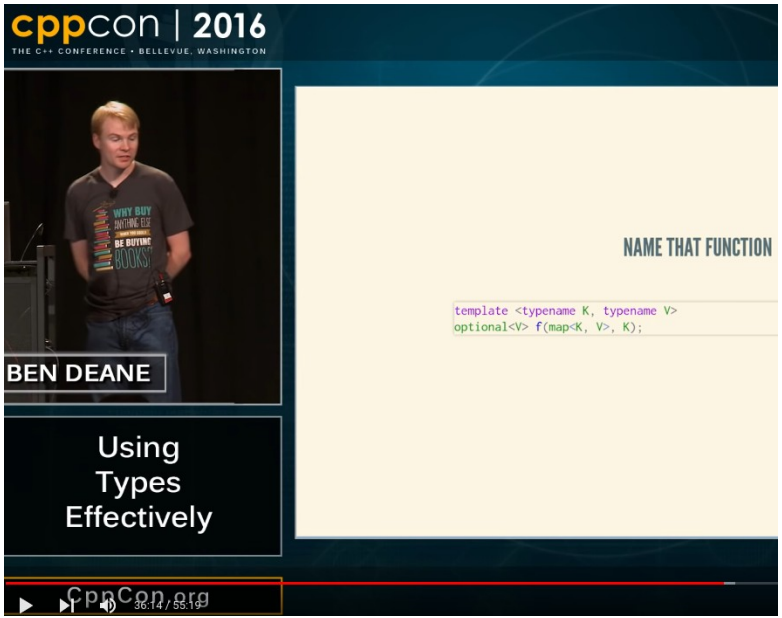
Søren Rune Nissen

soren.nissen+mcpp@gmail.com

Inspired by Ben Deane

(And a problem I had at work)

- Write total functions
- They’re easier to understand
- They’re easier *to write*



Example

```
auto count_vowels(char const *)
```

But what about - nullptr

Throw ?

std::pair<error, count> ?

UB ?

Example

```
size_t count_vowels(string_view)
```

Practically implements itself

No weird edge cases

Solid return type

No need to read documentation!

Example

```
size_t count_vowels(string_view)
```

```
auto reciprocal(double d)
{
    //precondition: d != 0
    //precondition: d != inf
    //precondition: d != nan

    return 1.0/d;
}
```

Example

```
size_t count_vowels(string_view)
```

```
auto reciprocal(Divisor d)
{
    return 1.0/d;
}
```

Does your codebase use exceptions?

- Throw in the constructor

Do it not not?

- Private constructor
- Static function `std::optional<Divisor> factory(double d);`

Nobody wants to read

all the documentation

for a whole new class

for every input parameter

Consider using a framework

ConstrainedMath.h

```
using Divisor = Constrained < double, Not<0.0>, Finite >;
double reciprocal(Divisor D);
```

ConstrainedMath.cpp

```
#include "ConstrainedMath.h"
double reciprocal(Divisor d)
{
    return 1.0/d;
}
```

In Conclusion

Enforce preconditions with invariants

Let your users select error handling using CTOR or factory method

Instead of writing and documenting a lot of parameter functions, consider using a framework to save time and to give the user a common type so they only have to read the documentation once.

Resources

Ben Deane – Using Types Effectively:

- https://www.youtube.com/watch?v=ojZbFIQSdI8

Generic constraints library

- https://github.com/SRNissen/snct-constraints

Feel free to contact me with work opportunities in C++

- soren.nissen+mcpp@gmail.com
- Søren Rune Nissen on linkedin