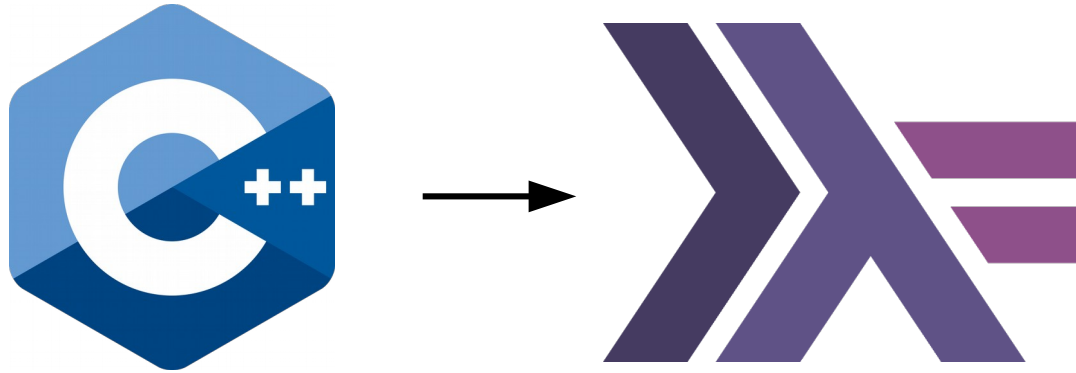


C++ Algorithms in Haskell and the Haskell Playground

Conor Hoekstra

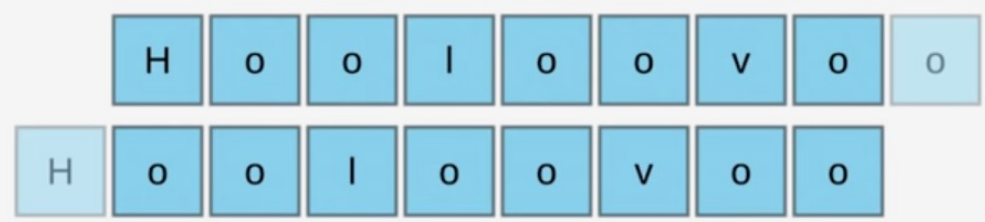
 code_report

 codereport



Composition

Task: Count repeated values





Functional Programming in

How to improve your
C++ programs using
functional techniques

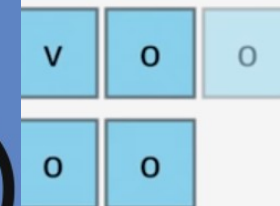
Ivan Čukić

Implementation

oo

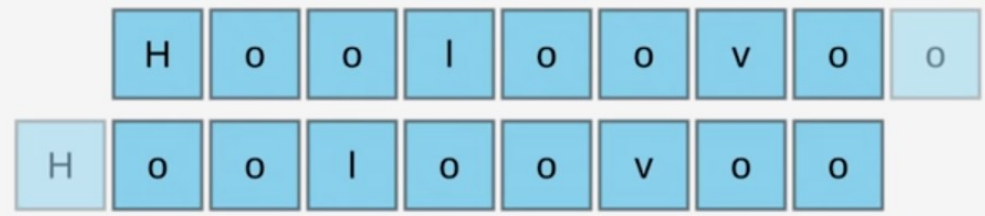
The End

oo



Composition

Task: Count repeated values



Task: Count repeated values

The diagram illustrates the process of counting repeated values in an array. It shows two rows of boxes representing the array "Hoolooloo". The first row has 9 boxes, and the second row has 9 boxes. The 8th box in both rows contains "o". Arrows point from these "o" boxes to a third row of boxes representing a frequency array. The 8th box in this row contains "1". An arrow points from this "1" to the number "3", indicating the total count of "o's".




```
template <typename T>
int count_adj_equals(const T& xs)
{
    return std::inner_product(
        std::cbegin(xs), std::cend(xs) - 1, // To the penultimate el.
        std::cbegin(xs) + 1,
        0,
        std::plus{},
        std::equal_to{});
}
```

© Ivan Čukić, 2019

10

○○○○○○●○○○○○○○○○○

○○○○○○○○○○○○○○○○○○○○○○○○○○○○

○○○○○○

oo

00

Composition

**“inner_product is a
really cool example that
you should never use”
- Ivan Cukic**

© Ivan Čukić, 2019

10

```
plus{} previously plus<>()  
[](auto a, auto b) { return a + b; }
```




```
plus{} previously plus<>()  
[](auto a, auto b) { return a + b; }
```



```
(\a b → a + b)  
(+)
```



```
plus{} previously plus<>()  
[](auto a, auto b) { return a + b; }
```



```
(\a b → a + b)  
(+)
```



```
[](auto e) { return e == 1; }
```



```
plus{} previously plus<>()  
[](auto a, auto b) { return a + b; }
```



```
(\a b → a + b)  
(+)
```



```
[](auto e) { return e == 1; }
```



```
(\e → e == 1)  
(==1)
```





Generic library functions

- `equals(value) ...`
- `compose(function...)`
- `if_then(pred, action)`
- `when_all(p...`
- `when_none(p...`
- `when_any(p...`
- `do_all(action...`
- `LIFT(overload_set)`

Domain specific functions

- `ip_matches(ip, mask)`
- `select_address(ipif)`
- `select_gateway(ipif)`
- `address_matches(ip)`
- `state_is(state_type)`

As of Boost 1.67.0 (April 14th, 2018)

`boost::hof` (higher order functions)

https://www.boost.org/doc/libs/1_68_0/libs/hof/ tions



<https://github.com/rollbear/lift>



Meeting C++ 2018
Björn Fahller
Higher Order Functions
for Ordinary C++
Developers



Thank You!

 code_report
 codereport



+



=

