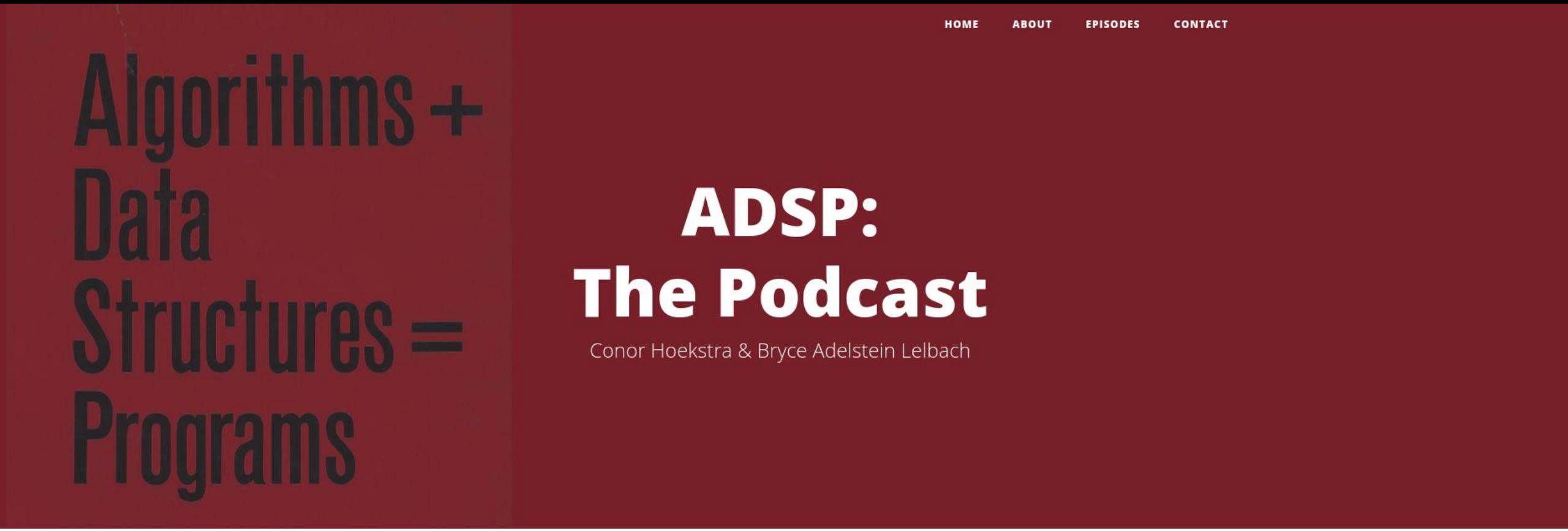
My Least Favorite Anti-Pattern

Conor Hoekstra

 code_report 

About Me

- I'm a Senior Library Software Engineer for NVIDIA
  - Working on the RAPIDS AI team (http://rapids.ai)
- I am a programming language enthusiast
- Founder and organizer of the **Programming Languages Virtual** Meetup
- I love algorithms and beautiful code
- I have a YouTube channel
  youtube.com/codereport
- My online alias is `code_report`

Episode 0: Coming Soon ...

A informal podcast about algorithms and everything software related...

www.adspthepodcast.com

Nov 10, 2019
Sun, 12:44 PM GMT+00:00

Algorithms + Data Structures = Programs

Following

ADSP: The Podcast

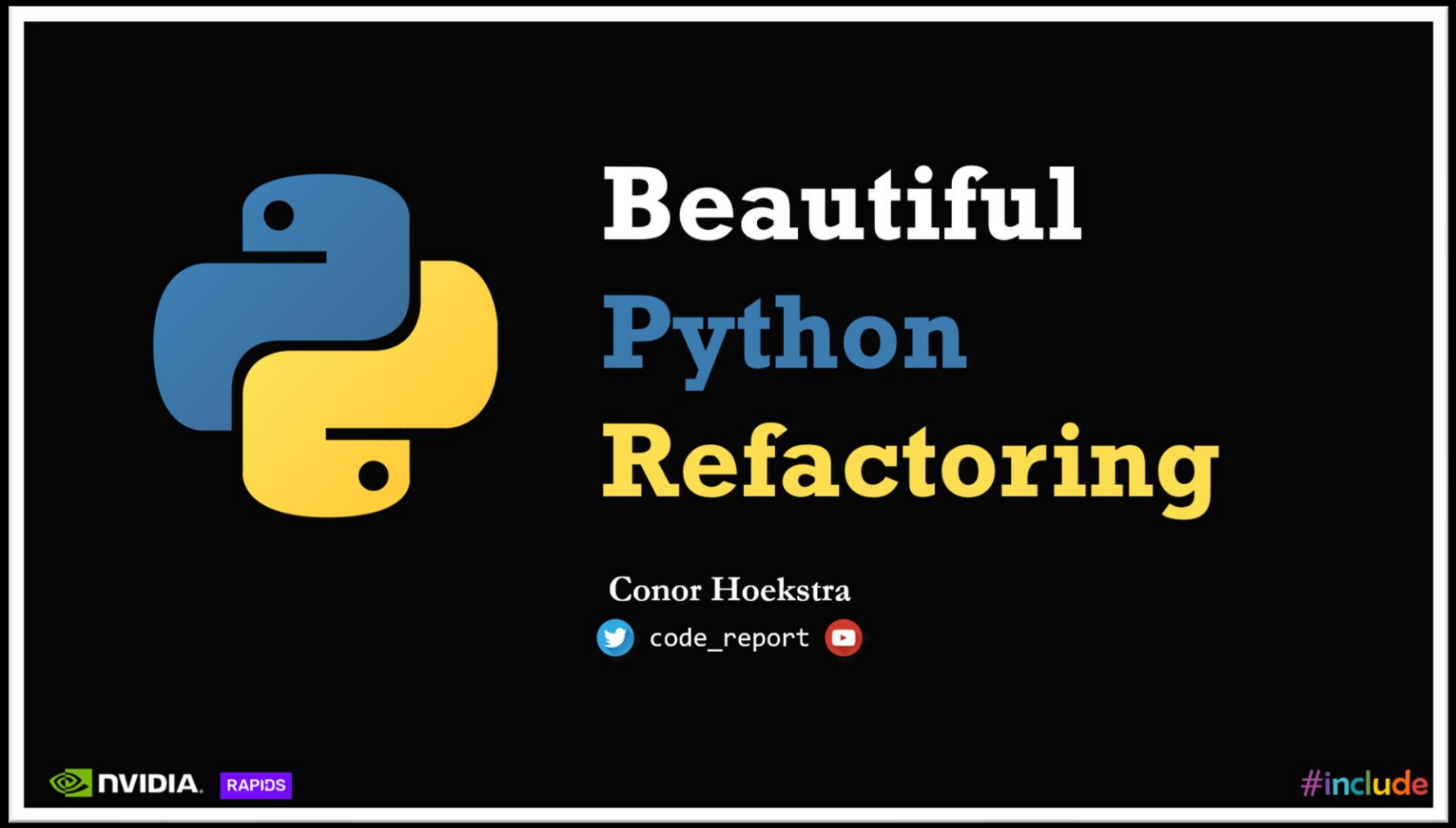
@adspthepodcast

ADSP: The Podcast. Hosted by @blelbach & @code_report
A podcast about algorithms and everything software related...

Joined November 2020

0 Following **1** Follower


2020
PYCON
ONLINE



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

Change 1
enumerate

```
col = []
i = 0

for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

Change 1
enumerate

```
col = []

for i, t in enumerate(tr_elements[0]):
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

	Rust	**enumerate**	trait.Iterator	Doc
	Python	**enumerate**	-	Doc
	Racket	**enumerate**	list-utils	Doc
	D	**enumerate**	range	Doc
	Ruby	**with_index**	Enumerable	Doc
	Kotlin	**withIndex**	collections	Doc
	Elixir	**with_index**	Enum	Doc
	Racket	**indexed**	data/collection	Doc
	Haskell	**indexed**	Data.List.Index	Doc
	Clojure	**map-indexed***	core	Doc
	C#	**Select**	Enumerable	Doc
	Scala	**zipWithIndex**	various	Doc

Change 1
enumerate

```
col = []

for i, t in enumerate(tr_elements[0]):
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

Change 2

Delete print & enumerate

```
col = []

for i, t in enumerate(tr_elements[0]):
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

Change 2

Delete print & enumerate

```
col = []

for t in tr_elements[0]:
    name = t.text_content()
    col.append((name, []))
```

Change 2

Delete print & enumerate

```
col = []

for t in tr_elements[0]:
    col.append((t.text_content(), []))
```

Change 3

List comprehension

```
col = []

for t in tr_elements[0]:
    col.append((t.text_content(), []))
```

Change 3

List comprehension

```
col = [(t.text_content(), []) for t in tr_elements[0]]
```

ITM

Initialize
Then
Modify

```
col = []
i = 0

for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('%d:"%s"'%(i, name))
    col.append((name, []))
```

```
col = []

for t in tr_elements[0]:
    name = t.text_content()
    col.append((name, []))
```

ITM

Initialize
Then
Modify

I did not “create” this anti-pattern

I learned this anti-pattern

How?

From 2 talks

#1

C++ Seasoning

Sean Parent

Going Native 2013

https://www.youtube.com/watch?v=W2tWOdzgXHA

“no raw loops”

Minute 2:10, C++ Seasoning

Sean Parent

Going Native 2013

What is a raw loop?

What is a Raw Loop?

- A *raw loop* is any loop inside a function where the function serves purpose larger than the algorithm implemented by the loop

Alternatives to raw loops?

One drawback of raw loops

“what variables are modified in that loop”

Minute 5:45, C++ Seasoning

Sean Parent

Going Native 2013

```cpp
// Example 1

std::vector v = { 1, 2, 3, 4, 5 };

auto ans = 0;
for (int i = 0; i < v.size(); ++i)
    ans += v[i];
```

https://godbolt.org/z/4Brqo_

```cpp
// Example 1

std::vector const v = { 1, 2, 3, 4, 5 };

auto ans = 0;
for (auto const& e : v)
    ans += e;
```

https://godbolt.org/z/uLCXFb

```cpp
// Example 1

std::vector const v = { 1, 2, 3, 4, 5 };

auto ans = std::accumulate(
    v.cbegin(),
    v.cend(),
    0);
```

https://godbolt.org/z/KwVY3h

```cpp
// Example 1

std::vector const v = { 1, 2, 3, 4, 5 };

auto const ans = std::accumulate(
    v.cbegin(),
    v.cend(),
    0);
```

https://godbolt.org/z/UZ6Vvb

```cpp
// Example 2

std::vector const v = { 1, 2, 3, 4, 5 };

auto ans = false;
for (auto const& e : v) {
    if (e % 2 == 0) {
        ans = true;
        break;
    }
}
```

https://godbolt.org/z/8sTpbj

```cpp
// Example 2

std::vector const v = { 1, 2, 3, 4, 5 };

auto const ans = std::any_of(
    v.cbegin(),
    v.cend(),
    [](auto const& e) { return e % 2 == 0; });
```

https://godbolt.org/z/MtGkHf

```cpp
// Example 3

std::vector const v = { 1, 2, 3, 4, 5 };

auto ans = 0;
for (auto const& e : v) {
    if (e % 2 == 0)
        ++ans;
}
```

https://godbolt.org/z/fGA7gP

```cpp
// Example 3

std::vector const v = { 1, 2, 3, 4, 5 };

auto const ans = std::count_if(
    v.cbegin(),
    v.cend(),
    [](auto const& e) { return e % 2 == 0; });
```

https://godbolt.org/z/WdK724

```cpp
// Example 4

std::vector const v = { 1, 2, 3, 4, 5 };

auto ans = 1;
for (auto const& e : v)
    ans *= e;
```

https://godbolt.org/z/yHwQvK

```cpp
// Example 4

std::vector const v = { 1, 2, 3, 4, 5 };

auto const ans = std::accumulate(
    v.cbegin(),
    v.cend(),
    1,
    std::multiplies{});
```

https://godbolt.org/z/2WH38z

[[digression]]

On The Virtues of `const`

cppcon | 2016

THE C++ CONFERENCE • BELLEVUE, WASHINGTON

Rich Code For Tiny Computers:

A Simple Commodore 64 Game in C++ 17

```
    Color{2,  0x88, 0x39, 0x32},
    Color{3,  0x67, 0xB6, 0xBD},
    Color{4,  0x8B, 0x3F, 0x96},
    Color{5,  0x55, 0xA0, 0x49},
    Color{6,  0x40, 0x31, 0x8D},
    Color{7,  0xBF, 0xCE, 0x72},
    Color{8,  0x8B, 0x54, 0x29},
    Color{9,  0x57, 0x42, 0x00},
    Color{10, 0xB8, 0x69, 0x62},
    Color{11, 0x50, 0x50, 0x50},
    Color{12, 0x78, 0x78, 0x78},
    Color{13, 0x94, 0xE0, 0x89},
    Color{14, 0x78, 0x69, 0xC4},
    Color{15, 0x9F, 0x9F, 0x9F}
  }};

  VIC_II vic;
  vic.background() = 1;
  vic.border() = 12;

  vic.border() = vic.nearest_color<255,0,0>(colors).num;


}
```

```
main:
  # @main
        pushl   %esi
        movb    $1, 53281
        movb    $12, 53280
        movzbl  _ZZ4mainE6colors+5, %eax
        addl    $-255, %eax
        imull   %eax, %eax
        movzbl  _ZZ4mainE6colors+6, %ecx
        imull   %ecx, %ecx
        addl    %eax, %ecx
        movzbl  _ZZ4mainE6colors+7, %eax
        imull   %eax, %eax
        addl    %ecx, %eax
        movzbl  _ZZ4mainE6colors+1, %ecx
        addl    $-255, %ecx
        imull   %ecx, %ecx
        movzbl  _ZZ4mainE6colors+2, %edx
        imull   %edx, %edx
        addl    %ecx, %edx
        movzbl  _ZZ4mainE6colors+3, %ecx
```

CppCon.org

```cpp
std::vector const v = { 1, 2, 3, 4, 5 };

auto ans = std::accumulate(
    v.cbegin(),
    v.cend(),
    0);
```

```rust
let v = vec![1, 2, 3, 4, 5];

let ans = v.iter()
           .fold(0, |a, b| a + b);
```

```rust
let mut v = vec![1, 2, 3, 4, 5];

let mut ans = v.iter()
               .fold(0, |a, b| a + b);
```

```
   Compiling playground v0.0.1 (/playground)
warning: variable does not need to be mutable
 --> src/main.rs:3:9
  |
3 |     let mut v = vec![1, 2, 3, 4, 5];
  |         ----^
  |         |
  |         help: remove this `mut`
  |
  = note: `#[warn(unused_mut)]` on by default

warning: variable does not need to be mutable
 --> src/main.rs:5:9
  |
5 |     let mut ans = v.iter()
  |         ----^^^
  |         |
  |         help: remove this `mut`

warning: 2 warnings emitted
```

[[end digression]]

#2

Best Presentation

Easy to Use, Hard to Misuse:
Declarative Style in C++

Ben Deane

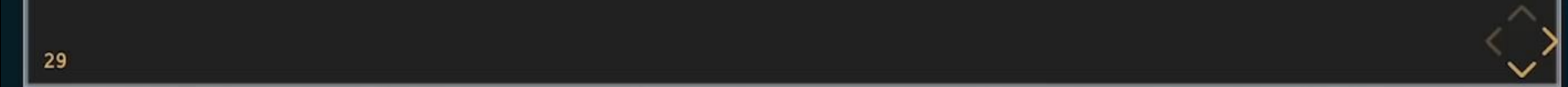
C++Now 2018

https://www.youtube.com/watch?v=2ouxETt75R4

GCC EXTENSION?

```
Bar b =
({
  auto sp = wp.lock();
  sp ? sp->bar() : Bar{};
});
```

Not ISO C++.



Easy to Use,
Hard to Misuse:
Declarative Style in C++

“I+LE … avoids the initialization-declaration split”

Minute 17:20, Easy to Use, Hard to Misuse

Ben Deane

CppCon 2018

```cpp
// Example 1

auto pet = ""s;
if (is_cool) {
    pet = "cat"s;
} else {
    pet = "dog"s;
}
```

https://godbolt.org/z/ViEDGx

```cpp
// Example 1

auto pet = is_cool ? "cat"s : "dog"s;
```

https://godbolt.org/z/CuormR

```cpp
// Example 1

auto const pet = is_cool ? "cat"s : "dog"s;
```

https://godbolt.org/z/CuormR

```cpp
// Example 1

auto pet = ""s;
if (is_cool) {
    // some other logic/stuff
    pet = "cat"s;
} else {
    // some other logic/stuff
    pet = "dog"s;
}
```

https://godbolt.org/z/ViEDGx

```cpp
// Example 1

auto const pet = [] {
    if (is_cool) {
        // some other logic/stuff
        return "cat"s;
    } else {
        // some other logic/stuff
        return "dog"s;
    }
} ();
```

https://godbolt.org/z/Wths0l

[[digression]]

Expressions

vs.

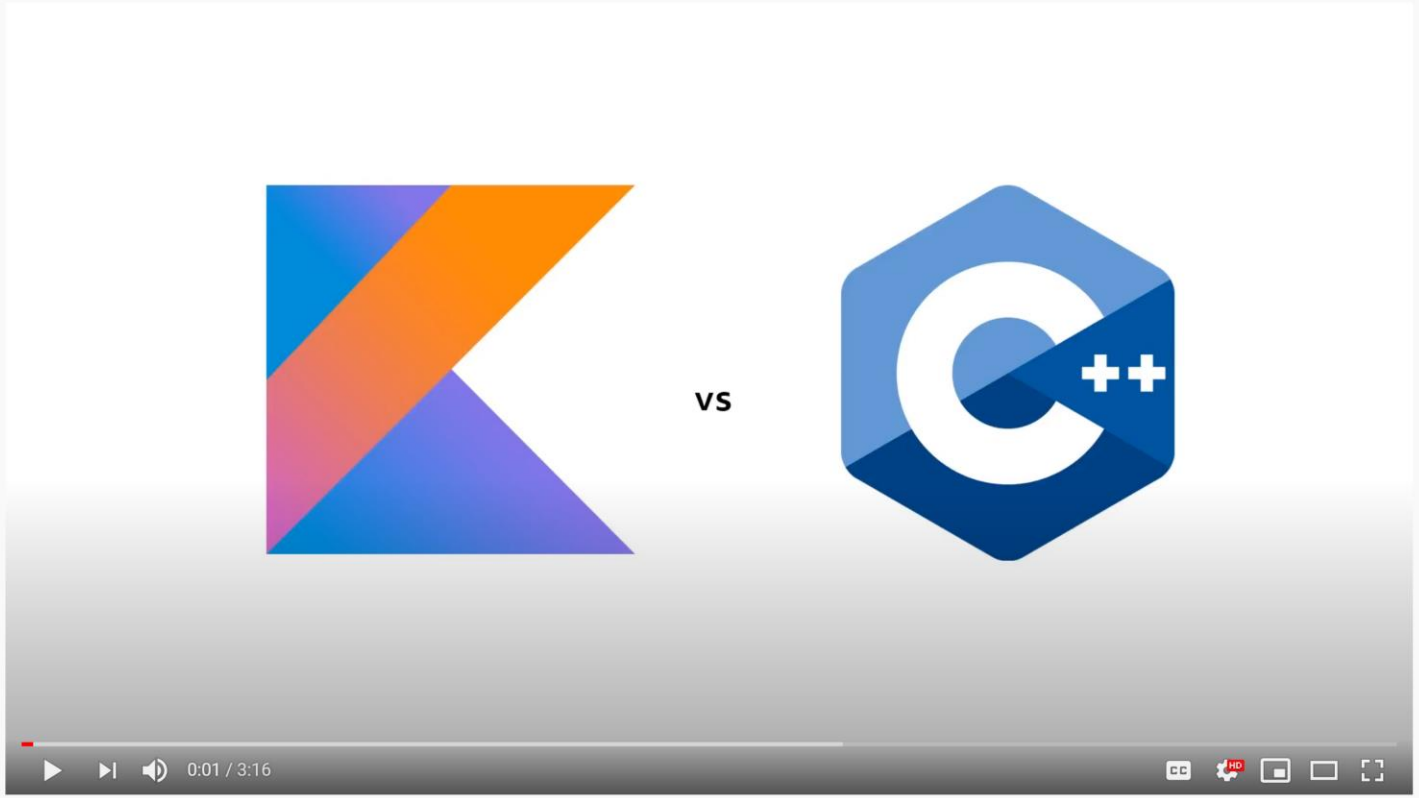
Statements

Expressions yield **values**

Statements **do not**

YouTube

kotlin vs c++



Kotlin vs C++: If Expression & Immediately Invoked Lambdas

1,168 views • Aug 15, 2018

33 2 SHARE SAVE

code_report
19.1K subscribers

ANALYTICS EDIT VIDEO

```
18      for ((key, value) in map.entries) {
19          println("$key -> $value")
20      }
21
22      var s: String
23      if (System.currentTimeMillis() % 2L == 0L) {
24          println("Yay!")
25          s = "Luck!"
26      } else {
27          s = "Not this time"
28      }
29      println(s)
30  }
31
32  fun test(e: Example): String {
33      when (e.a) {
34          1 -> return "Odd"
35          3 -> return "Odd"
36          5 -> return "Odd"
37          2 -> return "Even"
38          4 -> return "Even"
39          6 -> return "Even"
40          else -> return "Too big"
41      }
```

```kotlin
val s = if (condition) {
    "Luck"
} else {
    "Not this time"
}
```

```cpp
auto s = []() {
    if (condition) {
        return "Luck";
    } else {
        return "Not this time";
    }
} ();
```

```
a = if true  do 42 else 55 end
b = if false do 42 end

IO.puts a                          # 42
IO.puts (if b == nil do "nil" end) # nil
```

Functional C++

For Fun & Profit

JET BRAINS

@phil_nash
Developer Advocate

Expression
oriented programming

[[end digression]]

ITM so far …

1. “raw loops” (Sean Parent)
2. initialization-declaration split (Ben Deane)
3. Non-RAII code

```cpp
struct Rectangle {
    int height;
    int width;
};

int main () {

    Rectangle r;
    r.height = 2;
    r.width = 3;

    return 0;
}
```

https://godbolt.org/z/RK3FWQ

```cpp
struct Rectangle {
    int height;
    int width;
    Rectangle(int h, int w) :
        height{h}, width{w} {}
};

int main () {

    Rectangle r{2, 3};

    return 0;
}
```

https://godbolt.org/z/ByUYU4

```cpp
struct Rectangle {
    int height;
    int width;
    Rectangle(int h, int w) :
        height{h}, width{w} {}
};

int main () {

    Rectangle const r{2, 3};

    return 0;
}
```

https://godbolt.org/z/ByUYU4

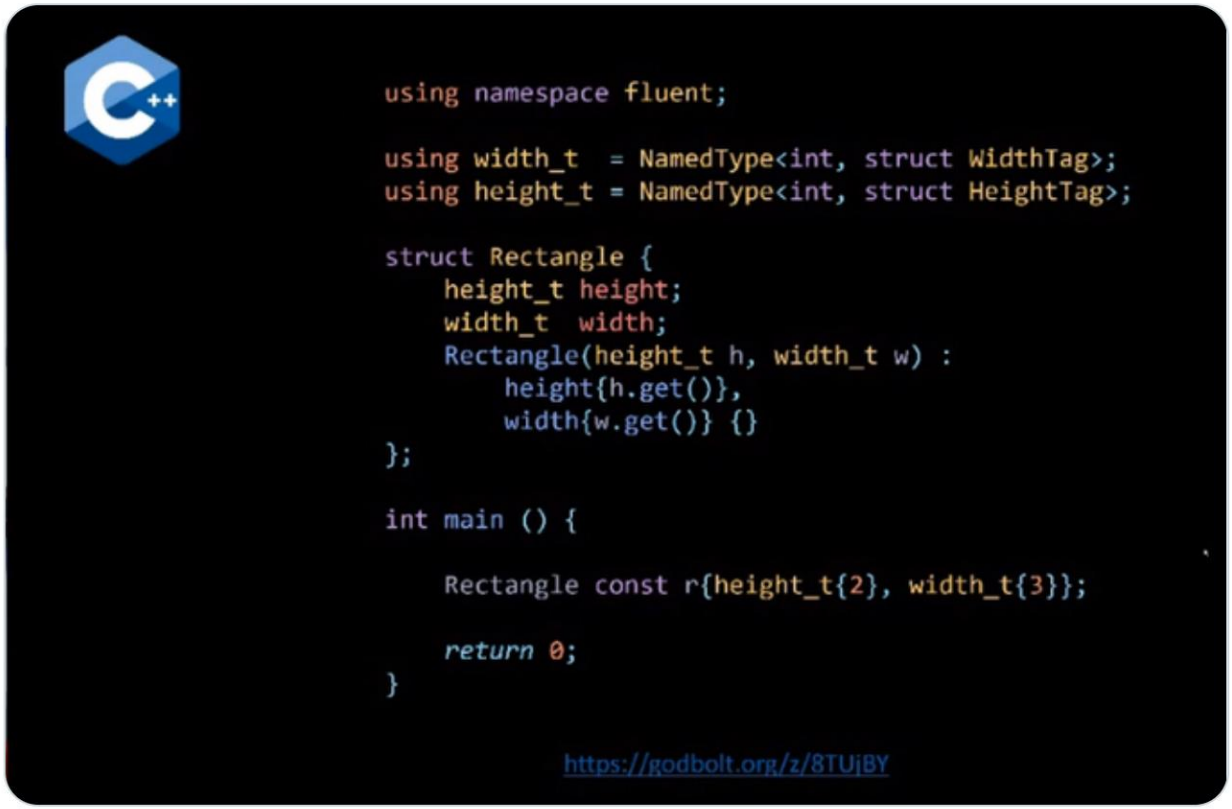
```cpp
using namespace fluent;

using width_t  = NamedType<int, struct WidthTag>;
using height_t = NamedType<int, struct HeightTag>;

struct Rectangle {
    height_t height;
    width_t  width;
    Rectangle(height_t h, width_t w) :
        height{h.get()},
        width{w.get()} {}
};

int main () {

    Rectangle const r{height_t{2}, width_t{3}};

    return 0;
}
```

https://godbolt.org/z/8TUjBY

Barry Revzin
@BarryRevzin

I'm sorry, but using strong types to implement named parameters is one of my least favorite anti-patterns.

This is a lot of code for the author to write to force the user to understand a larger API surface and write more code. We could've just had Rectange{.height=2, .width=3}!



```cpp
using namespace fluent;

using width_t  = NamedType<int, struct WidthTag>;
using height_t = NamedType<int, struct HeightTag>;

struct Rectangle {
    height_t height;
    width_t  width;
    Rectangle(height_t h, width_t w) :
        height{h.get()},
        width{w.get()} {}
};

int main () {

    Rectangle const r{height_t{2}, width_t{3}};

    return 0;
}
```

https://godbolt.org/z/8TUjBY

```cpp
using namespace fluent;

using width_t  = NamedType<int, struct WidthTag>;
using height_t = NamedType<int, struct HeightTag>;

struct Rectangle {
    height_t height;
    width_t  width;
    Rectangle(height_t h, width_t w) :
        height{h.get()},
        width{w.get()} {}
};

int main () {

    Rectangle const r{height_t{2}, width_t{3}};

    return 0;
}
```

https://godbolt.org/z/8TUjBY

```
struct Rectangle {
    int height;
    int width;
};

int main () {

    Rectangle const r{.height = 2, .width = 3};

    return 0;
}
```

https://godbolt.org/z/qras4c

```cpp
std::vector<int> v;
v.push_back(10);
v.push_back(20);
v.push_back(30);
v.push_back(40);
```

https://godbolt.org/z/9BcwCm

```
std::vector v = { 10, 20, 30, 40 };
```

https://godbolt.org/z/UGQVjq

```
std::vector const v = { 10, 20, 30, 40 };
```

https://godbolt.org/z/UGQVjq

ITM: Initialize Then Modify

1. “raw loops” (Sean Parent)
2. initialization-declaration split (Ben Deane)
3. Non-RAII code

How to Avoid the ITM Anti-Pattern?

1. Use algorithms 
2. Use I+LE
3. Use RAII

All of these enable more `const`

“sometimes I just need a for loop”

- Many developers

no raw loops ≠ no loops

What is a Raw Loop?

- A *raw loop* is any loop inside a function where the function serves purpose larger than the algorithm implemented by the loop

Now What? A vignette in three parts

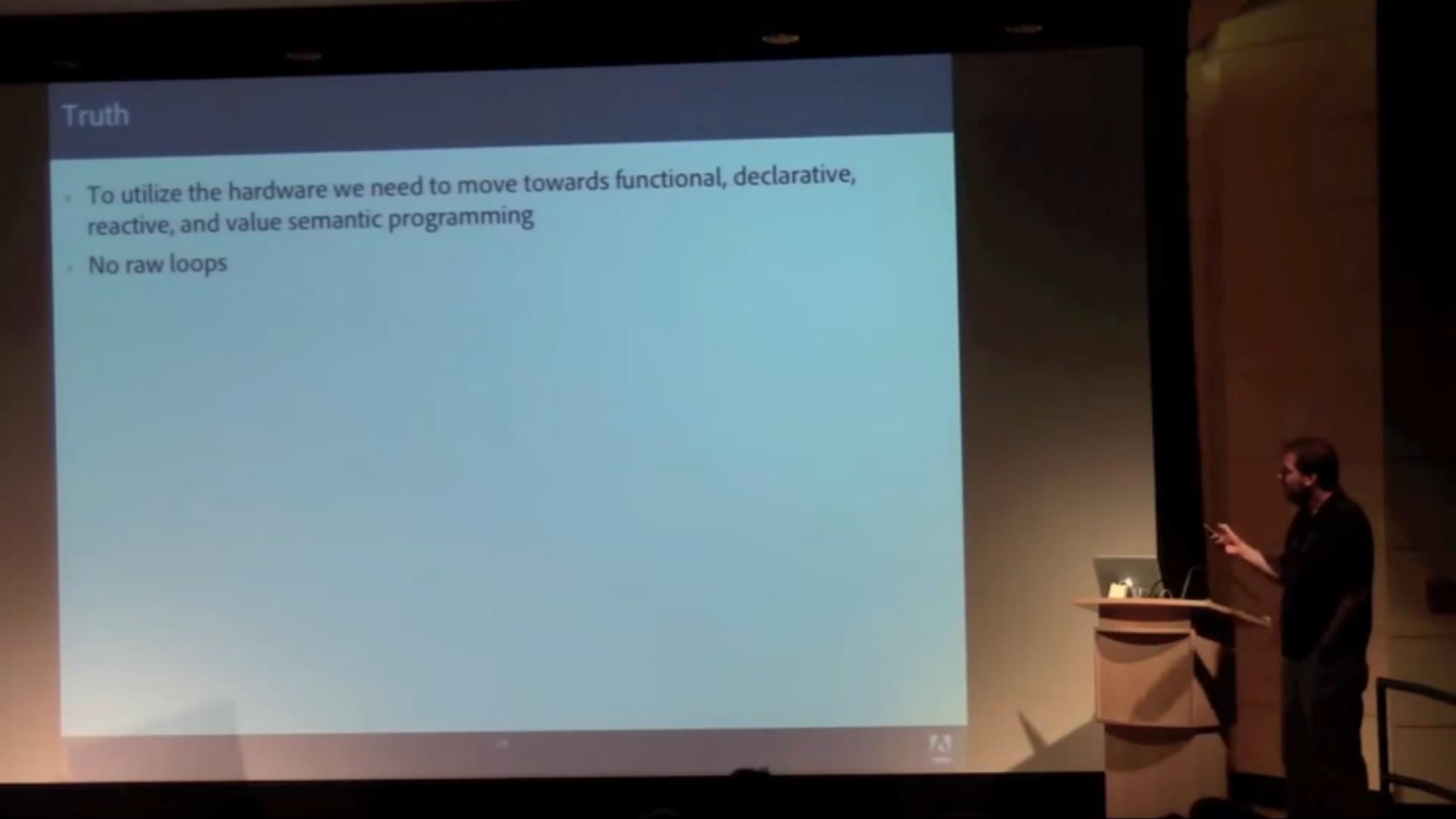
Sean Parent

BoostCon 2012

https://www.youtube.com/watch?v=iGenpw2NeKQ

Truth

- To utilize the hardware we need to move towards functional, declarative, reactive, and value semantic programming
- No raw loops



“don’t stick a loop in the middle of your function UNLESS that function is explicitly an algorithm”

Minute 35:30, Now What?

Sean Parent

BoostCon 2012

Unintentional Support for ITM

C++ Code Smells

Jason Turner

CppCon 2019

https://www.youtube.com/watch?v=f_tLQl0wLUM

What Do We Think?

```
1  #include <string>
2
3  void do_work()
4  {
5    std::string str;
6    // do some stuff
7    str = "Hello World";
8    // work with str
9  }
```

ITM

https://godbolt.org/z/7zl9t_

C++ Code Smells

What Do We Think?

```
1  #include <string>
2
3  void get_value(std::string &out_param);
4
5  int main()
6  {
7    std::string value;
8    get_value(value);
9    // use value
10 }
```

ITM

https://godbolt.org/z/egT7ec

C++ Code Smells

What Do We Think?

```
1  #include <vector>
2
3  void process_more(const std::vector<double> &);
4
5  void process_data(const std::vector<double> &values) {
6    bool in_range = true;
7    for (const auto &v : values) {
8      if (v < 5.0 || v > 100.0) {
9        in_range = false;
10       break;
11     }
12   }
13
14   if (in_range) {
15     process_more(values);
16   }
17 }
```

ITM

https://godbolt.org/z/PXsqPk

C++ Code Smells

What Do We Think?

```
1  double Data::total_area()
2  {
3    int value = 0;
4
5    // step 1: pipe area
6    for (int i = 0; i < pipes.size(); ++i) {
7      value += pipes[i].radius * pipes[i].radius * M_PI;
8    }
9
10   // step 2: hose area
11   for (int i = 0; i < hose.size(); ++i) {
12     value += hose[i].radius * pipes[i].radius * M_PI;
13   }
14
15   // and many more
16
17   return value;
18 }
```

ITM

https://godbolt.org/z/X30YWl

C++ Code Smells

Let's Update This Code Sample

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    int length;
7    string greet1 = "Hello";
8    string greet2 = ", world!";
9    string greet3 = greet1 + greet2;
10
11   length = greet3.size();
12 }
```

ITM

https://godbolt.org/z/hJEDYV

C++ Code Smells

Let's Update This Code Sample #2

```
1  #include <iostream>
2
3  int main()
4  {
5    int i, n, fact = 1;
6
7    std::cout << "Enter a whole number: ";
8    std::cin >> n;
9
10   for (i = 1; i <= n; ++i) {
11     fact *= i;
12   }
13
14   std::cout << "\nFactorial of " << n << " = " << fact << std::endl;
15   return 0;
16 }
```

ITM

https://godbolt.org/z/Q2D71b

C++ Code Smells

Code Smells

- Constructions Separate from Assignment
- Out Variables
- Raw Loops
- Multi-Step Functions
- Non-Connonical Operators
- Code With Conversions
- Casting Away `const`
- Code With Warnings
- `static const`
- `extern const`
- Raw `new` and `delete`

C++ Code Smells

Conclusions

C++ Code Smells


cppcon

Avoiding **ITM** enables more `const`

Two Final Examples

#1

Michael ☕🍷🤯
@_MRogers

Replying to @code_report @pycon and 2 others

very enjoyable

trying to overcome the anti-pattern of ITM, straight forward for lists but struggle with dictionaries

for example, if I have a nested dictionary of str:[list] and I want to turn each element of the list into a key and count how often it occurs

```
# this is an anti-pattern ITM (initialise then modify)
nom_count_dict = {}
for i in nominated:
    for name in nominated[i]:
        if name not in nom_count_dict:
            nom_count_dict[name] = 1
        else:
            nom_count_dict[name] += 1
```

4:21 PM · Jun 5, 2020 · Twitter Web App

```
nominated = { "Bob": ["Sam", "Jen", "Cat"],
              "Sam": ["Bob", "Cat"],
              "Jen": ["Bob", "Cat"] }

nom_count_dict = {}
for i in nominated:
    for name in nominated[i]:
        if name not in nom_count_dict:
            nom_count_dict[name] = 1
        else:
            nom_count_dict[name] += 1
```

```
nominated = { "Bob": ["Sam", "Jen", "Cat"],
              "Sam": ["Bob", "Cat"],
              "Jen": ["Bob", "Cat"] }

nom_count_dict = {}
for _, votes in nominated.items():
    for name in votes:
        if name not in nom_count_dict:
            nom_count_dict[name] = 1
        else:
            nom_count_dict[name] += 1
```

```
nominated = { "Bob": ["Sam", "Jen", "Cat"],
              "Sam": ["Bob", "Cat"],
              "Jen": ["Bob", "Cat"] }

nom_count_dict = {}
for votes in nominated.values():
    for name in votes:
        if name not in nom_count_dict:
            nom_count_dict[name] = 1
        else:
            nom_count_dict[name] += 1
```

```python
import collections as c

nominated = { "Bob": ["Sam", "Jen", "Cat"],
              "Sam": ["Bob", "Cat"],
              "Jen": ["Bob", "Cat"] }

nom_count_dict = c.defaultdict(int)
for votes in nominated.values():
    for name in votes:
        nom_count_dict[name] += 1
```

```python
import collections as c
import itertools as it

nominated = { "Bob": ["Sam", "Jen", "Cat"],
              "Sam": ["Bob", "Cat"],
              "Jen": ["Bob", "Cat"] }

nom_count_dict = c.defaultdict(int)
for name in it.chain(*nominated.values()):
    nom_count_dict[name] += 1
```

```python
import collections as c
import itertools as it

nominated = { "Bob": ["Sam", "Jen", "Cat"],
              "Sam": ["Bob", "Cat"],
              "Jen": ["Bob", "Cat"] }

nom_count_dict = c.Counter(it.chain(*nominated.values()))
```

```python
import collections as c
import itertools as it

nominated = { "Bob": ["Sam", "Jen", "Cat"],
              "Sam": ["Bob", "Cat"],
              "Jen": ["Bob", "Cat"] }

nom_count_dict = c.Counter(it.chain(*nominated.values()))

# Counter({'Cat': 3, 'Bob': 2, 'Sam': 1, 'Jen': 1})
```

```clojure
(def nominated
  {"Bob" ["Sam", "Jen", "Cat"]
   "Sam" ["Bob", "Cat"]
   "Jen" ["Bob", "Cat"]})

(def nom-count-dict
  (-> nominated
      vals
      flatten
      frequencies))
```

```clojure
(def nom-count-dict
  (-> nominated
      vals
      flatten
      frequencies))
```

Hoogle Translate

Counter

	pandas	**value_counts**	Series	Doc
	RAPIDS (Python)	**value_counts**	Series	Doc
	Clojure	**frequencies**	core	Doc
	Racket	**frequencies**	list-utils	Doc
	Python	**Counter***	collections	Doc
	Haskell	**count**	Data.List.Unique	Doc

#2

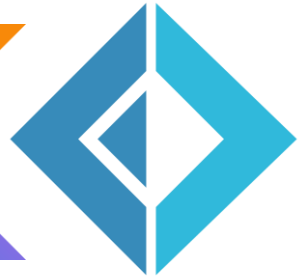


LeetCode



Contest 176


Problem 1
Count Negative Numbers in a Sorted Matrix



Contest 176: Problem 1 – Count Negative Numbers in a Sorted Matrix

Given a `m * n` matrix `grid` which is sorted in non-increasing order both row-wise and column-wise.

Return the number of **negative** numbers in `grid`.

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 100`
- `-100 <= grid[i][j] <= 100`













Contest 176: Problem 1 – Count Negative Numbers in a Sorted Matrix

Given a `m * n` matrix `grid` which is sorted in non-increasing order both row-wise and column-wise.

Return the number of **negative** numbers in `grid`.

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 100`
- `-100 <= grid[i][j] <= 100`













Contest 176: Problem 1 – Count Negative Numbers in a Sorted Matrix

```
[-2, -1, 0]
[-1,  1, 3]
[-1,  2, 4]
```

Contest 176: Problem 1 – Count Negative Numbers in a Sorted Matrix

```
[-2, -1, 0]
[-1,  1, 3]
[-1,  2, 4]
```












Contest 176: Problem 1 – Count Negative Numbers in a Sorted Matrix

```
[-2, -1, 0, -1, 1, 3, -1, 2, 4]
```

Contest 176: Problem 1 – Count Negative Numbers in a Sorted Matrix

```
[-2, -1, -1, -1]
```













Contest 176: Problem 1 – Count Negative Numbers in a Sorted Matrix

```
[-2, -1, -1, -1]
   length = 4
```












	flatten	filter	length
	-	filter	len
	flatten	count	
	join	filter	distance
	join	count	
	flatten	filter	count
	flatMap	filter	count
	reduce + append	filter	length
	flatten	count	
	concat	filter	count
	concat	filter	length
	flatten	filter	count
	flatten	filter	length
	, (ravel)	0>	+/ (plus reduce)

ITM: Initialize Then Modify


```python
def countNegatives(self, grid: List[List[int]]) -> int:
    ans = 0
    for row in grid:
        for i in row:
            if i < 0:
                ans += 1
    return ans
```


```python
def countNegatives(self, grid: List[List[int]]) -> int:
    return sum(i < 0 for j in grid for i in j)
```













https://old.reddit.com/r/programming/comments/f9p6ds/13_different_programming_languages_solving_1/

[–] **Academic_Childhood** 13 points 3 months ago

The unidiomatic, old-fashioned, mutating Python method is the one that every developer from every language will immediately be able to decipher what it does.

[–] **arkethos** [S] 7 points 3 months ago

I agree with your statement. The largest number of programmers across every language will be able to read the "ITM" code. However, you don't state whether you think this code is objectively better - which I would adamantly argue it is not.

It is an "artifact" of our CS education system and therefore the implicit "internet" education that is available that more people are able to read the nested `for` loops in Python compared to the Ruby

```
grid.flatten.count { |x| x < 0 }
```

In my opinion, the Ruby code (without any knowledge of computer science or programming in general) is objectively more readable and understandable. However, algorithms and algorithm composition is not something taught often enough, where as `for` loops are taught in every CS 101 course.

[–] **arkethos** [S] 10 points 3 months ago

Also, you almost argue for the FP style when you say

> I want to count something

Yes the Haskell solution doesn't have `count`, but Scala, Ruby, D and so many of the solutions are literally calling an algorithm called `count`! That is exactly what we want. And to defend Haskell, you can easily define

```
count = length . filter
```

There isn’t agreement on this topic

ITM: **I**nitialize **T**hen **M**odify

1. “raw loops” (Sean Parent)
2. initialization-declaration split (Ben Deane)
   construction-assignment split
3. Non-RAII code

How to Avoid the ITM Anti-Pattern?

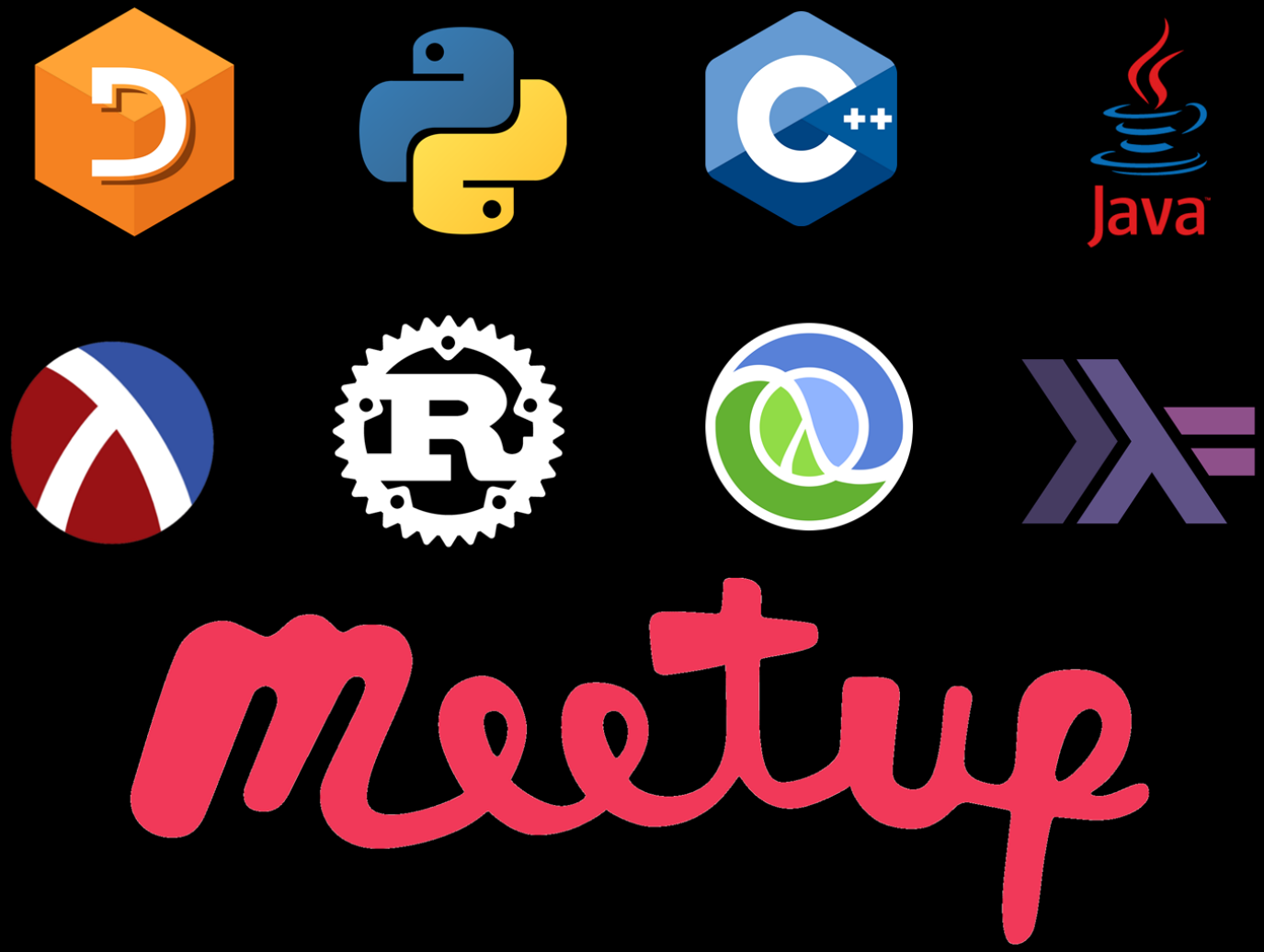
1. Use algorithms
2. Use I+LE
3. Use RAII

All of these enable more `const`

YouTube Video Links:

Speaker	Conference/Meetup	Year	Talk
Conor Hoekstra	PyCon	2020	Beautiful Python Refactoring
Sean Parent	Going Native	2013	C++ Seasoning
Jason Turner	CppCon	2016	Rich Code for Tiny Computers
Ben Deane	C++Now	2018	Easy to Use, Hard to Misuse: Declarative Style in C++
Andrey Breslav	Google I/O	2018	How to Kotlin - from the Lead Kotlin Language Designer
Sean Parent	BoostCon	2012	Now What? A vignette in three parts
Jason Turner	CppCon	2019	C++ Code Smells
Kate Gregory	CppCon	2014	Modernizing Legacy C++ Code

Sean Parent	@SeanParent
Jason Turner	@lefticus
Ben Deane	@ben_deane
Kate Gregory	@gregcons
Phil Nash	@phil_nash



https://www.meetup.com/Programming-Languages-Toronto-Meetup/

Thank You

https://github.com/codereport/Talks/

Conor Hoekstra

code_report

codereport

Questions?

https://github.com/codereport/Talks/

Conor Hoekstra

code_report

codereport