electricity is really just organized lightning.

— George Carlin
THE RULES

subjects are open!
five minutes (max)
have fun
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjorn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
FAMOUS PHYSICIST'S FAVOURITE FOODSTUFF
STEPHEN HAWKING

❤️

SAUSSAGE ROLLS
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here be is a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
A Year in Diversity

12 months by J Guy Davidson
Taking a pledge
Grill the committee
Aha, ahahaha ah ha

J. Guy Davidson @hatcat01 · Jul 14
Why isn’t there a C++ diversity group called #Include?
Ooopsy...

J. Guy Davidson
@hatcat01

Why isn’t there a C++ diversity group called Include?
10:23 PM - 14 Jul 2017

Kate Gregory @gregcoss - 17 Jul 2017
Replying to @hatcat01
Let’s start it. First get-together at @CppCor; another at @meetingcpp

@hatcat01
Well, Kate Gregory said I should do this, so...
Let’s organise
MEETING C++

What makes a good C++ programmer?

- Intellectual acuity
- Rigour
- Perseverance
Slack channel fills up

@hatcat01
Private discord group

Let's talk in private

@hatcat01
Ready for you to join in
Code of conduct
Or don’t join in, that’s fine
Disagreement is fine also

Disagreement is something normal.

— Dalai Lama —
TROLLING AND SEA-LIONING THOUGH...
Look! Here! Now!

https://www.includecpp.org/

@include_cpp

https://discord.gg/Sy9r7P9

https://github.com/include-cpp
MAKE THE POOL BIGGER

@hatcat01
**Thank you!**

You know what to do.
FAMOUS PHYSICIST’S FAVOURITE FOODSTUFF
LOOKS TASTY
ALBERT EINSTEIN

❤️

EGGS BENEDICT
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Björn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
FizzBuzz in the C PreProcessor

jon@jaggersoft.com
This was just for fun! I'm not suggesting you actually use the pre-processor like this...

Kudos to Paul Fultz II
https://github.com/pfultz2/Cloak
EVAL(REPEAT(100, INC, 0))

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
EVAL(REPEAT(100, INC, 0))
```c++
#define REPEAT(n, macro, i) \
    WHEN(n) \
    ( \
        macro( i ), \
        REPEAT( DEC(n), macro, macro(i) ) \
    )
```
#define REPEAT(n, macro, i) \
    WHEN(n) \
    ( \
        macro( i ), \
        REPEAT( DEC(n), macro, macro(i) ) \
    )
<table>
<thead>
<tr>
<th>DEC(x)</th>
<th>CAT(DEC_, x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC(0)</td>
<td>0</td>
</tr>
<tr>
<td>DEC(1)</td>
<td>0</td>
</tr>
<tr>
<td>DEC(2)</td>
<td>1</td>
</tr>
<tr>
<td>DEC(3)</td>
<td>2</td>
</tr>
<tr>
<td>DEC(4)</td>
<td>3</td>
</tr>
<tr>
<td>DEC(5)</td>
<td>4</td>
</tr>
<tr>
<td>DEC(6)</td>
<td>5</td>
</tr>
<tr>
<td>DEC(7)</td>
<td>6</td>
</tr>
<tr>
<td>DEC(8)</td>
<td>7</td>
</tr>
<tr>
<td>DEC(9)</td>
<td>8</td>
</tr>
<tr>
<td>DEC(10)</td>
<td>9</td>
</tr>
<tr>
<td>DEC(11)</td>
<td>10</td>
</tr>
<tr>
<td>DEC(12)</td>
<td>11</td>
</tr>
<tr>
<td>DEC(13)</td>
<td>12</td>
</tr>
<tr>
<td>DEC(14)</td>
<td>13</td>
</tr>
<tr>
<td>DEC(15)</td>
<td>14</td>
</tr>
<tr>
<td>DEC(16)</td>
<td>15</td>
</tr>
</tbody>
</table>
#define REPEAT(n, macro, i) \
    WHEN(n) \ 
    ( \ 
        macro(i), \ 
        REPEAT(DEC(n), macro, macro(i)) \ 
    )
#define WHEN(c) IF(c)(EXPAND, EAT)
#define IF(c) IIF(BOOL(c))
#define EXPAND(...) __VA_ARGS__
#define EAT(...) 

#define IIF(c) CAT(IIF_, c)
#define IIF_0(t, ...) __VA_ARGS__
#define IIF_1(t, ...) t
WHEN(7)(42)  -->  IF(7)(EXPAND,EAT)(42)
  -->  IIF(BOOL(7))(EXPAND,EAT)(42)
  -->  IIF(1)(EXPAND,EAT)(42)
  -->  IIF_1(EXPAND,EAT)(42)
  -->  EXPAND(42)
  -->  42

WHEN(0)(42)  -->  IF(0)(EXPAND,EAT)(42)
  -->  IIF(BOOL(0))(EXPAND,EAT)(42)
  -->  IIF(0)(EXPAND,EAT)(42)
  -->  IIF_0(EXPAND,EAT)(42)
  -->  EAT(42)
  -->
#define WHEN(c) IF(c)(EXPAND, EAT)
#define IF(c) IIF(BOOL(c))
#define EXPAND(...) __VA_ARGS__
#define EAT(...)

#define IIF(c) CAT(IIF_, c)
#define IIF_0(t, ...) __VA_ARGS__
#define IIF_1(t, ...) t
```c
#define BOOL(x) COMPL(NOT(x))
#define COMPL(b) CAT(COMPL_, b)
#define COMPL_0 1
#define COMPL_1 0

BOOL(0) 0
BOOL(1) 1
BOOL(42) 1
```
#define BOOL(x) COMPL(NOT(x))

#define COMPL(b) CAT(COMPL_, b)
#define COMPL_0 1
#define COMPL_1 0
#define NOT(x) CHECK(CAT(NOT_, x))
#define NOT_0 ~, 1,

#define CHECK(...) CHECK_N(__VA_ARGS__, 0,)
#define CHECK_N(_, n, ...) n

NOT(42) --> CHECK(CAT(NOT_, 42))
--> CHECK(NOT_42)
--> CHECK_N(NOT_42, 0,)

NOT(0) --> CHECK(CAT(NOT_, 0))
--> CHECK(NOT_0)
--> CHECK(~, 1,)
--> CHECK_N(~, 1, 0,)
Macros cannot be recursive...
A macro can be tricked into being recursive!

```c
#define REPEAT(n, macro, i) \ WHEN(n) \ ( \ ( \ macro(i), \ OBSTRUCT(REPEAT_INDIRECT)() \ ( \ DEC(n), macro, macro(i) \ ) \ ) \ )
#define REPEAT_INDIRECT() REPEAT
```

```c
#define OBSTRUCT(id) id DEFER(EMPTY)()
#define DEFER(id) id EMPTY()
#define EMPTY()
```
EVAL(REPEAT(100, INC, 0))

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,

#define INC(x) CAT(INC_, x)
#define INC_0 1
#define INC_1 2
#define INC_2 3
#define INC_3 4
#define INC_4 5
#define INC_5 6
#define INC_6 7
#define INC_7 8
#define INC_8 9
#define INC_9 10
#define INC_10 11
#define INC_11 12
#define INC_12 13
#define INC_13 14
EVAL(REPEAT(100,CYCLE3,0))

1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3

#define CYCLE3(f) CYCLE3_ ## f
#define CYCLE3_0 1
#define CYCLE3_1 2
#define CYCLE3_2 3
#define CYCLE3_3 1
EVAL(REPEAT(100,CYCLE5,0))

1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4,

#define CYCLE5(f) CYCLE5_  ##  f

#define CYCLE5_0 1
#define CYCLE5_1 2
#define CYCLE5_2 3
#define CYCLE5_3 4
#define CYCLE5_4 5
#define CYCLE5_5 1
NTH_ARG(1,a)
NTH_ARG(2,a,b)
NTH_ARG(3,a,b,c)
jon@jaggersoft.com: email
http://cyber-dojo.org: practice
50% off my consultancy rate for sites near an in-season salmon river!
#define FIZZ_BUZZ(n) \
  FIZZ_BUZZ_ \
  ( \
   n, \n   NTH_ARG(n, EVAL(REPEAT(100,CYCLE3,0))), \
   NTH_ARG(n, EVAL(REPEAT(100,CYCLE5,0))) \n  )

#define FIZZ_BUZZ_(n,f,b) FIZZ_BUZZ_PRIMITIVE(n,f,b)
#define FIZZ_BUZZ_PRIMITIVE(n,f,b) FIZZ_BUZZ_ ## f ## _ ## b (n)
#define FIZZ_BUZZ_1_1(n) n
#define FIZZ_BUZZ_1_2(n) n
#define FIZZ_BUZZ_1_3(n) n
#define FIZZ_BUZZ_1_4(n) n
#define FIZZ_BUZZ_1_5(,) Buzz

#define FIZZ_BUZZ_2_1(n) n
#define FIZZ_BUZZ_2_2(n) n
#define FIZZ_BUZZ_2_3(n) n
#define FIZZ_BUZZ_2_4(n) n
#define FIZZ_BUZZ_2_5(,) Buzz

#define FIZZ_BUZZ_3_1(,) Fizz
#define FIZZ_BUZZ_3_2(,) Fizz
#define FIZZ_BUZZ_3_3(,) Fizz
#define FIZZ_BUZZ_3_4(,) Fizz
#define FIZZ_BUZZ_3_5(,) FizzBuzz
Testing!

```
# -E == preprocess only
# -P == dont show line's

gcc -std=c99 -E -P fizz_buzz_tests.h

#include "fizz_buzz.h"
#define ASSERT(e,a) ASSERT_(e,a)
#define ASSERT_(e,a) STR(ASSERT_ ## e ## _ ## a)

#include ASSERT(FIZZ_BUZZ_1_EQUALS, FIZZ_BUZZ(1))
#include ASSERT(FIZZ_BUZZ_3_EQUALS, FIZZ_BUZZ(3))
#include ASSERT(FIZZ_BUZZ_5_EQUALS, FIZZ_BUZZ(5))
#include ASSERT(FIZZ_BUZZ_15_EQUALS, FIZZ_BUZZ(15))

... fizz_buzz_tests.h
```
Create one empty *file* per assertion...

```
ASSERT_FIZZ_BUZZ_1_EQUALS_1
ASSERT_FIZZ_BUZZ_3_EQUALS_Fizz
ASSERT_FIZZ_BUZZ_5_EQUALS_Buzz
ASSERT_FIZZ_BUZZ_15_EQUALS_FizzBuzz
...
```
failing

```c
#include ASSERT(FIZZ_BUZZ_1_EQUALS, FIZZ_BUZZ(99))
```

diers.err

fizz_buzz.tests.h:6:1: fatal error: ASSERT_FIZZ_BUZZ_1_EQUALS_Fizz: No such file or directory
#include ASSERT(FIZZ_BUZZ_1_EQUALS, FIZZ_BUZZ(99))

~~~~~~~~~~~~~
compilation terminated.

passing

```c
```

stdout
50% off consultancy near an in-season salmon river

jon@jaggersoft.com
FAMOUS PHYSICIST’S FAVOURITE FOODSTUFF
MY FAVOURITE!
NIKOLA TESLA

❤️

TIRAMISU
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
Here beis a dragons

Magic!
Frances Buontempo
@fbuontempo
Lindenmayer Systems

- L-systems
- Recursion
- Grammars
- Trees, ferns...
- Self-similar
  - fractals
def X(n):
    if n>0:    L("X+YF+",n)

def Y(n):
    if n>0:    L("-FX-Y",n)

def L(s,n):
    for c in s:
        if   c=='-': lt(90)
        elif c=='+': rt(90)
        elif c=='X': X(n-1)
        elif c=='Y': Y(n-1)
        elif c=='F': fd(12)

bgcolor('black')
pencolor('red')
up()
goto(-20, 120)
down()
X(10)
hideturtle()

mainloop()
FAMOUS PHYSICIST’S FAVOURITE FOODSTUFF
Isaac Newton

❤

NECTARINES
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
{The Problem

```cpp
std::unique_ptr<char, decltype(std::free) *> t_copy { strdup(t), std::free }; // takes the address of std::free!
// function to function pointer conversion
```

Is Unspecified in the std!

see also:
http://stackoverflow.com/questions/27440953/stdunique-ptr-for-c-functions-that-need-free/
Why?

```cpp
std::unique_ptr<std::FILE, decltype(&std::fclose)> fp(std::fopen("demo.txt", "r"),
&std::fclose);
if(fp) // fopen could have failed; in which case fp holds a null pointer
std::cout << (char)std::fgetc(fp.get()) << 'n';
```

**Thou Shalt Not Specialize std Function Templates!**
The details: addressable functions

6 Let $F$ denote a standard library function ([global.functions]), a standard library static member function, or an instantiation of a standard library function template. Unless $F$ is designated an addressable function, the behavior of a C++ program is unspecified (possibly ill-formed) if it explicitly or implicitly attempts to form a pointer to $F$. [Note: Possible means of forming such pointers include application of the unary & operator ([expr.unary.op]), offsetof ([specialized.addressof]), or a function-to-pointer standard conversion ([conv.func]). — end note] Moreover, the behavior of a C++ program is unspecified (possibly ill-formed) if it attempts to form a reference to $F$ or if it attempts to form a pointer-to-member designating either a standard library non-static member function ([member.functions]) or an instantiation of a standard library member function template.

Exception (so far): iostream manipulators
A Workaround Proposal?

P0984R0 - All (*)() - Pointers Replaced by Ideal Lambdas

<table>
<thead>
<tr>
<th>Document Number:</th>
<th>P0984R0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>2018-04-01</td>
</tr>
<tr>
<td>Project:</td>
<td>Programming Language C++</td>
</tr>
<tr>
<td>Audience:</td>
<td>EWG/LEWG</td>
</tr>
<tr>
<td>Target:</td>
<td>C++20</td>
</tr>
</tbody>
</table>

The closure type for a non-generic lambda-expression with no lambda-capture whose constraints (if any) are satisfied is called a Ideal Lambda. An Ideal Lambda has a conversion function to pointer to function with C++ language linkage(10.5) having the same parameter and return types call operator template specialization. An Ideal Lambda furthermore defines an overload for the unary operator&() that returns the result of the said conversion to function pointer. [Note: That operator overload guarantees that existing code bases that invalidly take the address of a standard library function continue to work as expected. — end note]

Names that are defined as functions in C shall be defined as functions constexpr inline auto variables initialized from an Ideal Lambda in the C++ standard library, unless the C++ standard defines overloads of said function. In that case the names defined as functions in C shall be defined as functions. \[1 \]
It is unspecified whether any non-overloaded non-template non-member functions in the C++ standard library shall be defined as `constexpr inline auto` variables initialized from an Ideal Lambda. For the purpose of this standard these variables are called *FOOL (Function ObsOleted by Lambda).* [Note: This mechanism allows many wrong programs that take the address of a standard library function to conform to this standard. —end note] It is unspecified whether any overloaded or templated non-member functions are defined as inline(10.1.6).
const std::string filename = "./hello1.txt";
auto close = [](auto fd){ ::close(fd); };
{
    auto file = unique_resource (::open (filename.c_str(),
                                  O_CREAT|O_RDWR, 06)
                               , close);
    ::write (file.get (), "Hello World!\n", 12u);
    ASSERT (file.get () != -1);
}
Solution for unique_ptr:

```cpp
struct free_deleter{
    template <typename T>
    void operator()(T *p) const {
        std::free(const_cast<std::remove_const_t<T>*>(p));
    }
};

template <typename T>
using unique_C_ptr=std::unique_ptr<T,free_deleter>;

static_assert(sizeof(char *)==sizeof(unique_C_ptr<char>),"");
// compiles!
```

Wrap the call in a class!
lambdas/decltype(lambda) works in the future
Stay tuned for FOOL!
FAMOUS PHYSICIST'S FAVOURITE FOODSTUFF
FFFEED ME!
MARIE CURIE

❤️

VERY HOT CURRY
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
Shaderator
Compute Shader debugging and Unit Testing with C++

by Cezary Bloch
https://github.com/cezbloch/shaderator
Shaders

- Programs run on GPU
- Executed in parallel
- Originally for shading polygons eg. in games

Compute Shaders/Kernels

- Skip rendering pipeline
- Used for GPGPU
- Used a lot outside gaming industry - research, finance, AI
- Performance boost
StructuredBuffer<unsigned int> Input : register( t0 );
RWStructuredBuffer<unsigned int> Data : register( u0 );

groupshared unsigned int shared_data[BITONIC_BLOCK_SIZE];

[numthreads(BITONIC_BLOCK_SIZE, 1, 1)]
void BitonicSort( uint3 Gid : SV_GroupID, uint3 DTid : SV_DispatchThreadID,
                uint3 GTid : SV_GroupThreadID, uint GI : SV_GroupIndex )
{
    shared_data[GI] = Data[DTid.x];
    GroupMemoryBarrierWithGroupSync();

    for (unsigned int j = g_iLevel >> 1 ; j > 0 ; j >>= 1)
    {
        unsigned int result = ((shared_data[GI & ~j] <= shared_data[GI | j]) ==
                                (bool)(g_iLevelMask & DTid.x)) ? shared_data[GI ^ j] : shared_data[GI];
        GroupMemoryBarrierWithGroupSync();
        shared_data[GI] = result;
        GroupMemoryBarrierWithGroupSync();
    }
    Data[DTid.x] = shared_data[GI];
}
In C++

Full debugging support

➔ Step through
➔ Data Conditional Breakpoints
➔ Thread freeze
➔ Memory & variable look-up
➔ Assertions/Exceptions
➔ Unit testing
➔ Logging/Tracing/Tracepoints
On GPU

Complicated to set-up and limited debugging

- **Step through**
  Draw calls required, Record executing and ‘replay’

- **Breakpoints/Tracepoints**
  On one kernel only

- **Memory & variable look-up**
  Some values not available

- **Unit testing**
  Check the output buffer

- **Assertions/Exceptions**

- **Logging/Tracing**
How to bring all the IDE and C++ language features to Compute Shaders on GPU?

Tip
If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.
HLSL → GLSL → C++
Shaderator

Macro Magic
Dispatch Engine
Vector operations
HLSSL Types
GLSSL Types
The diagram shows a Test Fixture connected to the Executor. The Executor has the following functions:

- set_constants(value_1, value_2)
- set_input_buffer(buffer)
- create_output_buffer(size)
- dispatch()

The Executor also involves the header file `shaderator.h` and the HLSL code.
structuredBuffer<unsigned int> SHADERATOR_REGISTER_I(Input, 0);
RWStructuredBuffer<unsigned int> SHADERATOR_REGISTER_U(Data, 0);

groupshared unsigned int shared_data[BITONIC_BLOCK_SIZE];

SHADERATOR_NUM_THREADS(BITONIC_BLOCK_SIZE, 1, 1)
void BitonicSort(SHADERATOR_SV_DispatchThreadId(DTid),
                 SHADERATOR_SV_GroupID(Gid),
                 SHADERATOR_SV_GroupThreadId(GTid),
                 SHADERATOR_SV_GroupIndex(GI))
{
    shared_data[GI] = Data[DTid.x];
    GroupMemoryBarrierWithGroupSync();

    for (unsigned int j = g_level >> 1; j > 0; j >>= 1)
    {
        unsigned int result = ((shared_data[GI & ~j] <= shared_data[GI & (j - 1)])
                          ? shared_data[GI & ~j] : shared_data[GI & (j - 1)]);
        GroupMemoryBarrierWithGroupSync();
        shared_data[GI] = result;
        GroupMemoryBarrierWithGroupSync();

        Data[DTid.x] = shared_data[GI];
    }
}
Faster development
Less errors
Quick problem diagnosis
Protection against regressions
Same code for C++ and HLSL

Enhancement of existing tools
What people are saying?

What a great idea! I've not heard of anybody else doing this.

I have been using the same approach for over 6 years and it's by far the best way to develop HLSL shaders.

If we don't have tests build with Shaderator I'm not changing our shader!
FAMOUS PHYSICIST'S FAVOURITE FOODSTUFF
GREEN GORGEOUSNESS...
GALILEO GALILEI

❤️

GUACAMOLE

❤️🍽️😍

GUACAMOLE
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
Literal *misdirection*
Previously on ... ACCU lightning talks
Previously on ... 

ACCU lightning talks

UK
1 gallon = 8 pints

US
1 gallon = 8 pints
Previously on ...  
**ACCU lightning talks**

**UK**

1 gallon = 8 pints  
1 pint = 20 fl. oz.

**US**

1 gallon = 8 pints  
1 pint = 16 fl. oz.
Previously on ...  

**ACCU lightning talks**

**UK**

1 gallon = 8 pints

1 pint = 20 fl. oz.

1 fl. oz. = 28.41 ml

**US**

1 gallon = 8 pints

1 pint = 16 fl. oz.

1 fl. oz. = 29.57 ml
Previously on ... ACCU lightning talks

**UK**

- 1 gallon = 8 pints
- 1 pint = 20 fl. oz.
- 1 fl. oz. = 28.41 ml
- 1 gallon = 4,545.6 ml

**US**

- 1 gallon = 8 pints
- 1 pint = 16 fl. oz.
- 1 fl. oz. = 29.57 ml
- 1 gallon = 3,785.0 ml
- Chuck drinks 6 pints at his local bar,
- Reggie drinks 5 pints at the pub.

Who drank fewer pints?

Who drank less beer?
“I’m literally bursting for a pee,” says Reggie.

Is a “Meaning of Life” moment coming?

Has Reggie failed a BBC R4 test?
Literally or figuratively

**literally**, adv. ... 3.b. Used as an intensive before a figurative expression.

- The American Heritage Dictionary of the English Language, 2016
The canning, processing, preserving, freezing, drying, marketing, storing, packing for shipment or distribution of:

(1) Agricultural produce;
(2) Meat and fish products; and
(3) Perishable foods.

The drivers argued, due to a lack of a comma between “packing for shipment” and “or distribution”, the law refers to the single activity of “packing”, not to “packing” and “distribution” as two separate activities. As the drivers distribute – but do not pack – the goods, this would make them eligible for overtime pay.

US Court of Appeals, First Circuit, March 2017
Example mapping

- Story
  - Rule
    - Example
      - The one where...
  - Rule
  - Rule

- Question
  - Question
  - Question

https://cucumber.io/blog/2015/12/08/example-mapping-introduction
https://www.youtube.com/watch?v=VwvrGfWmG_U
ARCHIMEDES

❤️

ANCHOVIES
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
Two Small Corrections

Anna-Jayne Metcalfe
@annajayne
anna@riverblade.co.uk

Riverblade Ltd
www.riverblade.co.uk
Two Small Corrections

What Stories Can You Tell?

What about the people you know?

Are you listening?

#caffeinedrivendev
FAMOUS PHYSICIST'S FAVOURITE FOODSTUFF
MY STOMACH’S RUMBLING
RICHARD FEYNMAN

❤️

FISH

(ANY, HE'S NOT FUSSY)
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
My favourite memory leak

```cpp
#include <vector>

struct V : std::vector<V> {}

int main()
{
    V v;
    v.emplace_back();
    v.swap(v.front());
}
```
My favourite memory leak
My favourite memory leak

```
end_store
end
begin
```

```v
end_store
end
begin
```

```
v2
end_store
end
begin
```
My favourite memory leak

My favourite memory leak – ACCU 2018 – © Björn Fahller
@bjorn_fahller
My favourite memory leak

```cpp
#include <vector>

struct V : std::vector<V> {}

int main()
{
    V v;
    v.emplace_back();
    v.swap(v.front());
}
```
#include <vector>

struct V : std::vector<V> {}

int main()
{
    V v;
    v.emplace_back();
    v.swap(v.front());
}
My favourite memory leak

```cpp
#include <vector>

struct V : std::vector<V> {};

int main()
{
    V v;
    v.emplace_back();
    v.swap(v.front());
}
```
My favourite memory leak

```cpp
#include <vector>

struct V :
std::vector<V> {}

int main()
{
    V v;
    v.emplace_back();
    v.swap(v.front());
}
```
My favourite memory leak

Björn Fahller

bjorn@fahller.se

@bjorn_fahller

@rollbear  cpplang, swedencpp
FAMOUS PHYSICIST’S FAVOURITE FOODSTUFF
MICHAEL FARADAY

❤️

FOCACIA
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
for i := 0; i <= 5; i++ {
    fmt.Printf("%d\n", i)
}
const FIVE = 5 // Five

for i := 0; i <= FIVE; i++ {
    fmt.Printf("%d
", i)
}
const FIVE = 6 // Tau

for i := 0; i <= FIVE; i++ {
    fmt.Printf("%d\n", i)
}
const maxIterations = 5

for i := 0; i <= maxIterations; i++ {
    fmt.Printf("%d\n", i)
}
for i := 0; i <= maxIterations; i++ {
    fmt.Printf("%d, %d\n", i, maxIterations)
}
FAMOUS PHYSICIST’S FAVOURITE FOODSTUFF
A PARTICULARLY FINE FOODSTUFF
MAX PLANCK

❤️

PANETTONE
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
Care of Magical Creatures

By Gail Ollis
I really enjoyed researching my paper!
I really enjoyed researching my paper!
I really enjoyed researching my paper!

It was definitely uncomfortable to speak in front of a lot of people especially in a very “male” orientated course which sort of makes me feel like I shouldn’t be there!
#metoo
#notsecretary
#notprincess
STORIES FOR BOYS WHO DARE TO BE DIFFERENT
TRUE TALES OF AMAZING BOYS WHO CHANGED THE WORLD WITHOUT KILLING DRAGONS
FAMOUS PHYSICIST’S FAVOURITE FOODSTUFF
AMEDEO
AVOGADRO
❤
APPLES
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here beis a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjørn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
RANKING HACKERS

Steve Love // essennell.love@gmail.com // @IAmSteveLove
FAMOUS PHYSICIST’S FAVOURITE FOODSTUFF
MMMMMmmmm...
NIELS BOHR

❤️

BURRITOS
Guy Davidson - A year in diversity
Jon Jagger - FizzBuzz in the C pre-processor
Frances Buontempo - Here be is a dragons
Peter Sommerlad - APRIL
Cezary Bloch - Shaderator
Seb Rose - Literal Misdirection
Anna-Jayne - Two Small Corrections
Bjorn Fahller - My favourite memory leak
Dom Davis - Putting the away into go
Gail Ollis - Care of Magical Creatures
Steve Love - </rant>
Pete Goodliffe - The New C++ Interview
THE NEW C++ INTERVIEW

(WITH APOLOGIES TO YOUR SANITY)

Pete Goodliffe
pete@goodliffe.net
@petegoodliffe
try
{
        
        // aaaghr!
}
catch (...)
{
         
}
Boo!
NAME

char
times 2

100 miles
invitation

this is a party only for people in the club
1. substitution principle
2. substitution principle
3. substitution principle
4. substitution principle
5. substitution principle
6. substitution principle
7. substitution principle
8. substitution principle
9. substitution principle
10. substitution principle
function — function — function — function — function — function
you swapped it!

this is not a problem
NO!!!

that way...
gluttony
lust
avarice
pride
wrath
vanity
sloth

Bill: $1000
FAMOUS PHYSICIST'S FAVOURITE FOODSTUFF
BLAISE PASCAL

❤

PEAS
THANKS!