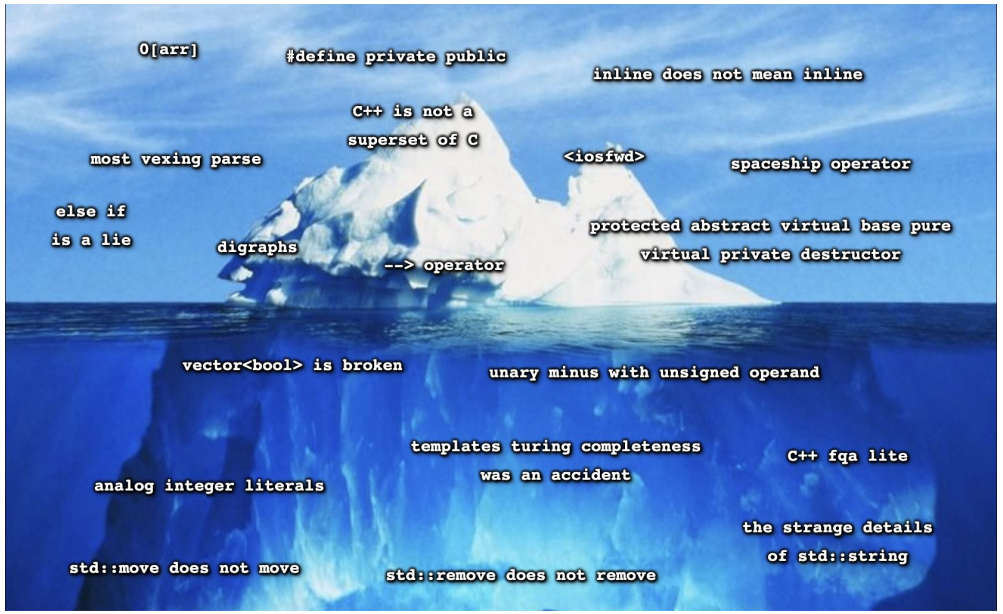
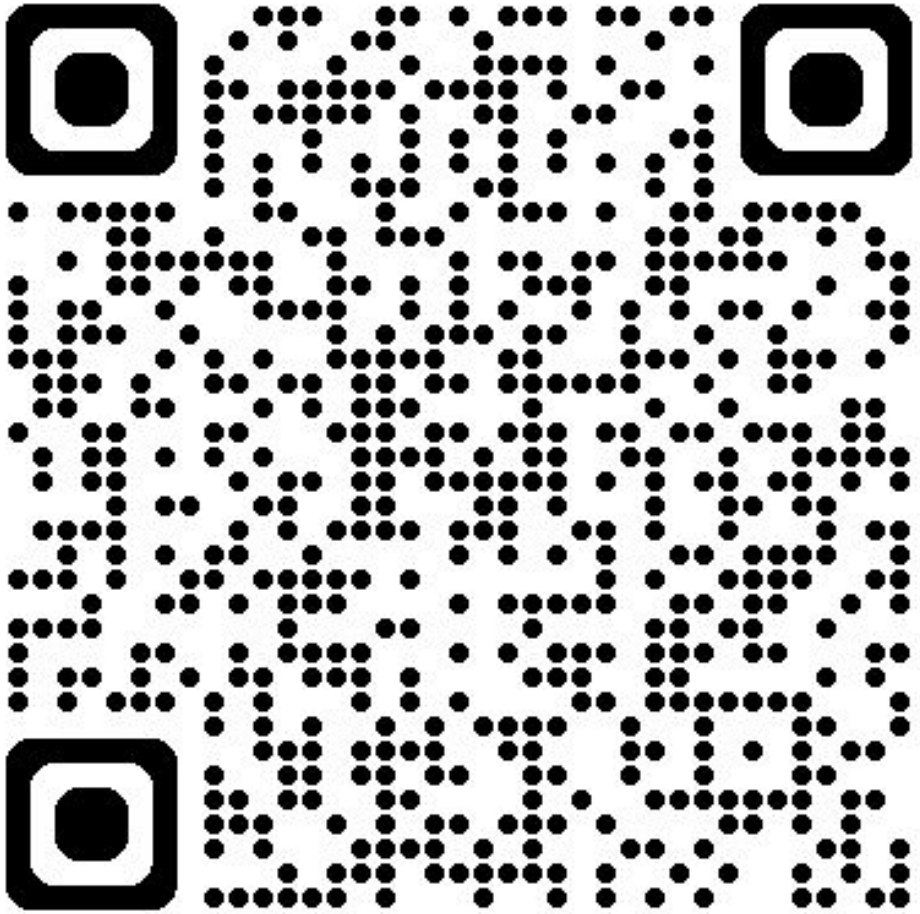
Lecture 17: C++ Iceberg



CS106L, Spring 2026

Rachel Fernandez & Preston Seay

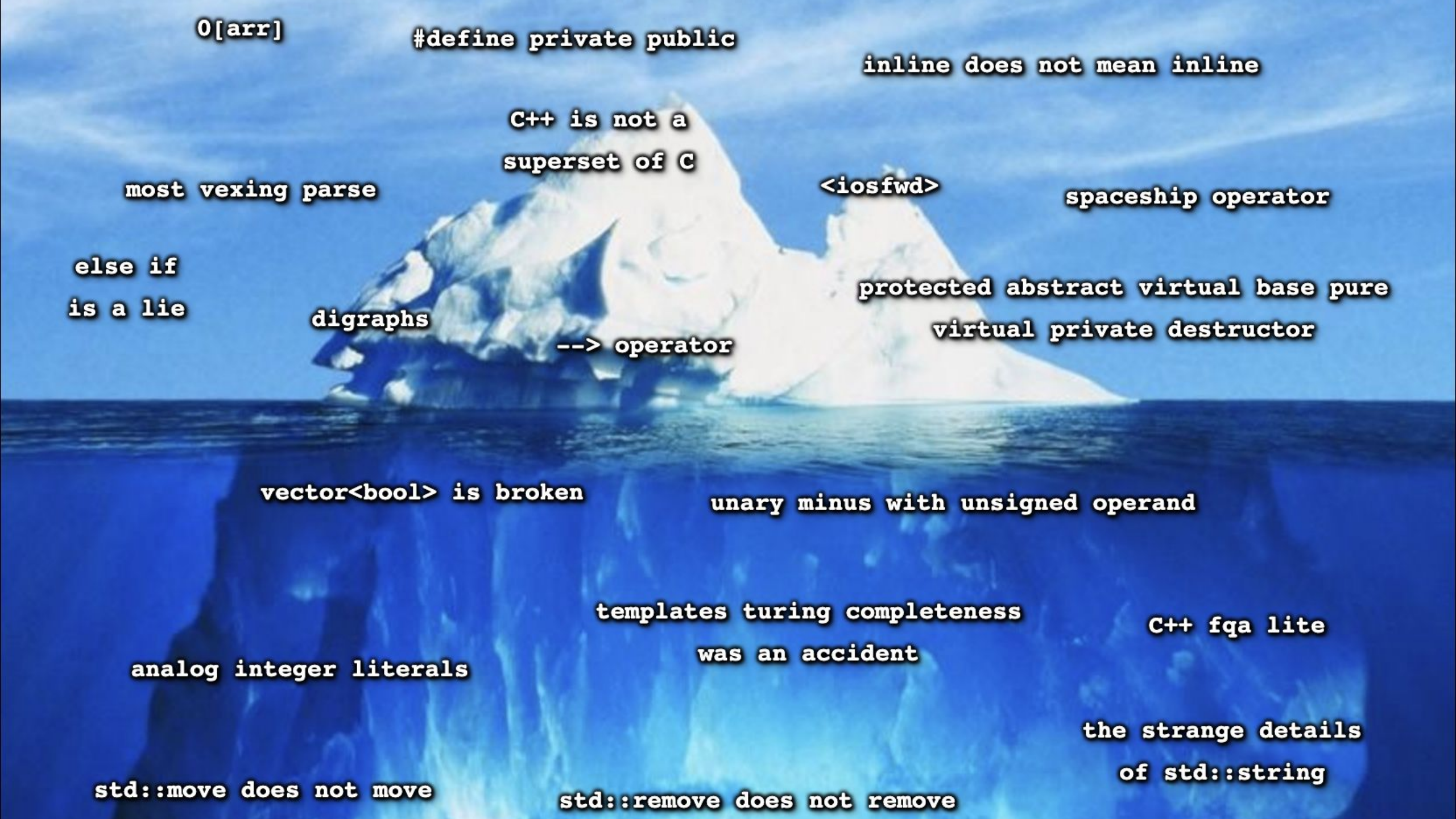
Attendance



https://forms.gle/UFMH3U3i5g1aau7u8

Today’s Agenda

- C++ Iceberg
- Kahoot


0[arr]
#define private public
inline does not mean inline
C++ is not a
superset of C
most vexing parse
<iosfwd>
spaceship operator
else if
is a lie
digraphs
--> operator
protected abstract virtual base pure
virtual private destructor
vector<bool> is broken
unary minus with unsigned operand
templates turing completeness
was an accident
C++ fqa lite
analog integer literals
the strange details
of std::string
std::move does not move
std::remove does not remove

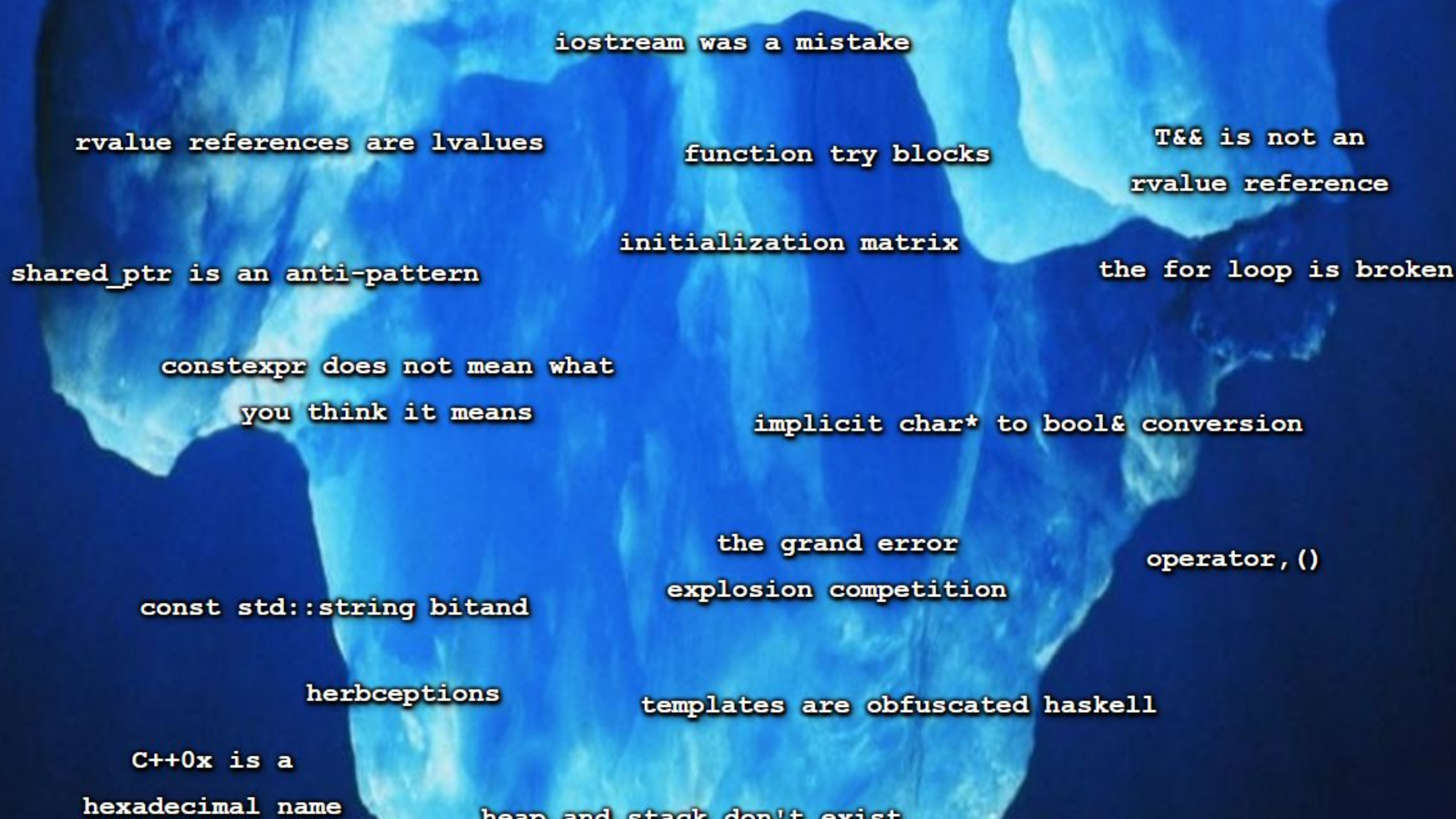
→ operator

```c
#include <stdio.h>
int main()
{
    int x = 10;
    while (x --> 0) // x goes to 0
    {
        printf("%d ", x);
    }
}
```

```
9 8 7 6 5 4 3 2 1 0
```

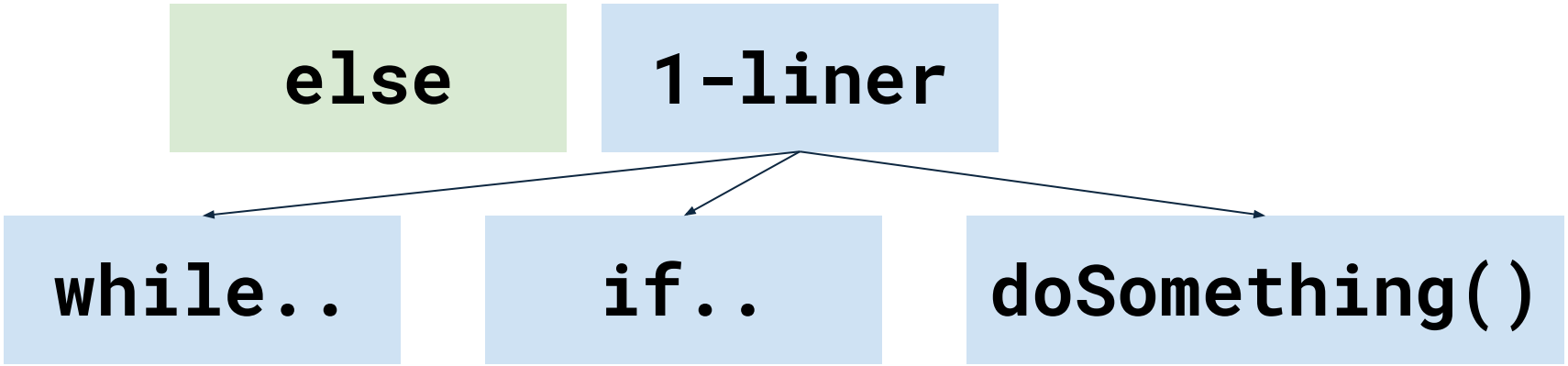
Yeah, it’s actually this :(

```c
(x-- > 0)
```


iostream was a mistake
rvalue references are lvalues
function try blocks
T&& is not an
rvalue reference
initialization matrix
shared_ptr is an anti-pattern
the for loop is broken
constexpr does not mean what
you think it means
implicit char* to bool& conversion
the grand error
explosion competition
operator,()
const std::string bitand
herbceptions
templates are obfuscated haskell
C++0x is a
hexadecimal name
heap and stack don't exist

else if is a lie

```
 9  #include <iostream>
10
11  int main() {
12      int i = 10;
13
14      if (false) {
15          std::cout << "hello" << std::endl;
16      } else while (i > 0) {
17          std::cout << i << std::endl;
18          --i;
19      }
20  }
```



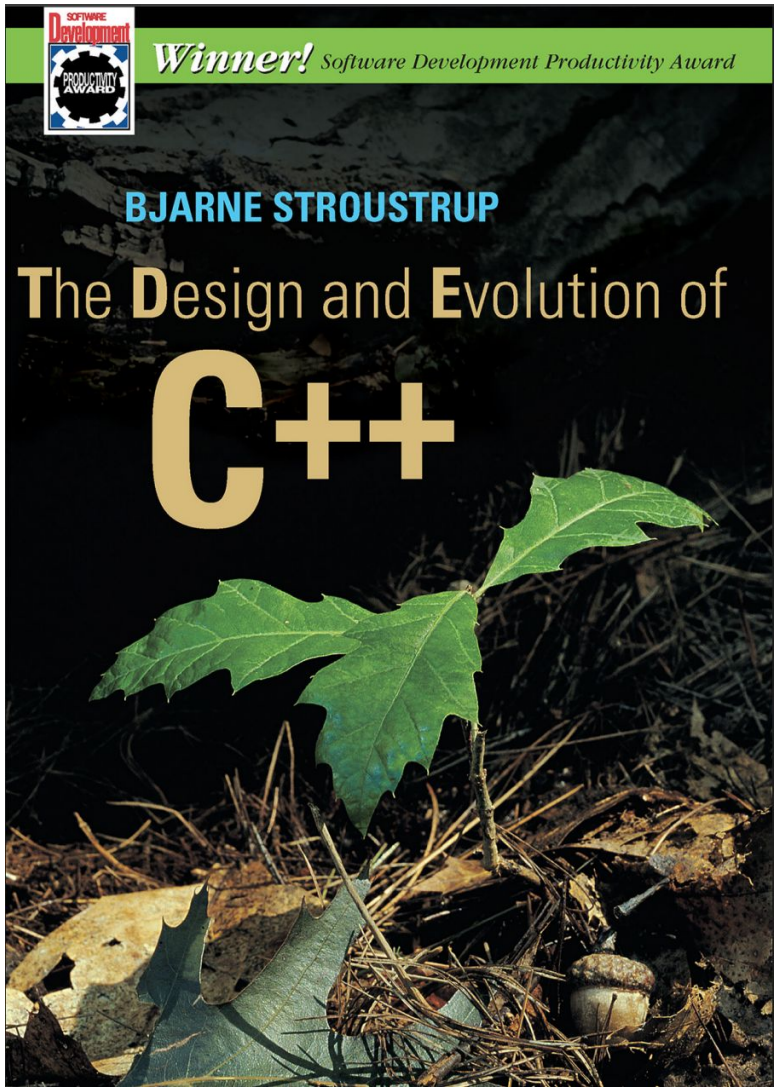
iostream is bad?

- “C’s printf family of functions is effective…”
- “It is not, however, type-safe or extensible…”
- “I started looking for a type-safe, terse, extensible, and efficient alternative…”

- Douglas McIlroy suggested it be similar to Unix streams: >>, >, and |
  - They considered = and < and >
- Andrew Koenig’s idea of manipulators

```
int i = 1234;

cout << i << ' '              // decimal by default: 1234
     << hex << i << ' '       // hexadecimal: 4d2
     << oct << i << '\n';     // octal: 2322
```



Section 8.3.1: The I/O Stream Library

iostream is bad?

Random guy on the internet:

- “You get to an interface with fairly cryptic and confusing member function names, e.g. `getloc`/`imbue`, `uflow`/`underflow`, `snextc`/`sbumpc`/`sgetc`/`sgetn`, `pbase`/`pptr`/`epptr`”

Long story short:

- Yes, there are some **shortcomings**.
- It has been **improved** across versions by many ideas.
- They made it **pretty far**: Ada Rationale [Ichbiah, 1979] argued that it was impossible to have terse, type-safe I/O without special language features.

the for loop is broken

The Promise:
The range expression looks like it stays alive for the whole loop. It doesn't always though!

```
for (auto e : getCollection())
```

`getCollection()` returns a temporary. C++ goes okay! I'll keep this alive for the whole loop

```
for (auto e : getCollection().getRef())
```

`.getCollection()` creates a temporary collection

`.getRef()` returns a reference within the temp collection

temporary collection = destroyed before loop begins. getRef() pointing to dead memory

the for loop is broken

The Promise:
The range expression looks like it stays alive for the whole loop. It doesn't always though!

```
for (auto e : getCollection())
```

`getCollection()` returns a temporary. C++ goes okay! I'll keep this alive for the whole loop

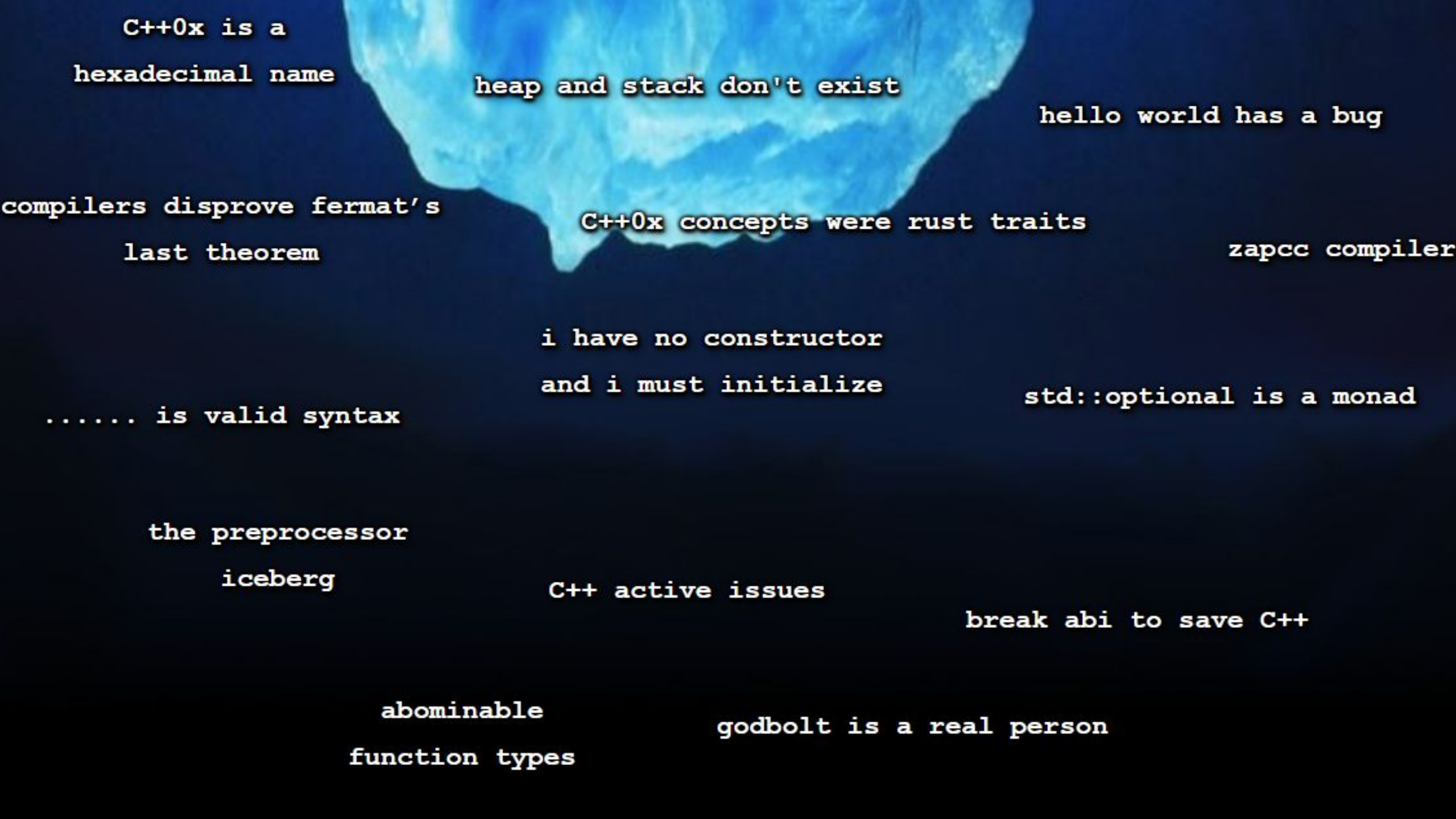
```
for (auto e : getCollection().getRef())
```

`.getCollection()` creates a temporary collection

`.getRef()` returns a reference within the temp collection

Caveat:

getCollection() allocates memory, then it expires, but unless the memory gets taken, the data still is there and the "loop" works


C++0x is a
hexadecimal name
heap and stack don't exist
hello world has a bug
compilers disprove fermat's
last theorem
C++0x concepts were rust traits
zapcc compiler
i have no constructor
and i must initialize
std::optional is a monad
..... is valid syntax
the preprocessor
iceberg
C++ active issues
break abi to save C++
abominable
function types
godbolt is a real person

hello world has a bug

we wrote to /dev/full and it failed, which is what we wanted to see!

```
$ echo "Hello World!" > /dev/full
bash: echo: write error: No space left on device
$ echo $?
1
```

BUG! writes to the buffer, thinks its a success, and then errors after

```
$ gcc hello.c -o hello
$ ./hello > /dev/full
$ echo $?
0
```

the error definitely happened but the program ignored it!

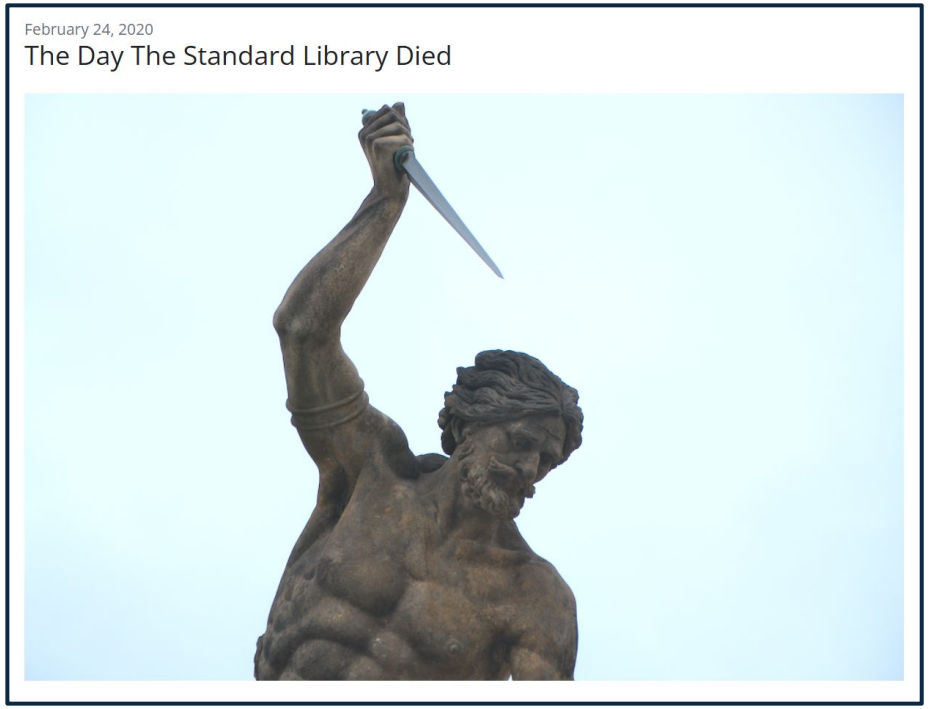
```
$ strace -etrace=write ./hello > /dev/full
write(1, "Hello World!\n", 13)          = -1 ENOSPC (No space
+++ exited with 0 +++
```

break abi to save c++

- ABI: Application Binary Interface
  - A low-level contract for code, such as: Calling Conventions, Data Representation, System Calls, Name Mangling, and Exceptions.

Arithmetic Operation

Mnemonic	Instruction	Type	Description
ADD rd, rs1, rs2	Add	R	rd ← rs1 + rs2
SUB rd, rs1, rs2	Subtract	R	rd ← rs1 - rs2
ADDI rd, rs1, imm12	Add immediate	I	rd ← rs1 + imm12
SLT rd, rs1, rs2	Set less than	R	rd ← rs1 < rs2 ? 1 : 0
SLTI rd, rs1, imm12	Set less than immediate	I	rd ← rs1 < imm12 ? 1 : 0
SLTU rd, rs1, rs2	Set less than unsigned	R	rd ← rs1 < rs2 ? 1 : 0
SLTIU rd, rs1, imm12	Set less than immediate unsigned	I	rd ← rs1 < imm12 ? 1 : 0
LUI rd, imm20	Load upper immediate	U	rd ← imm20 << 12
AUIP rd, imm20	Add upper immediate to PC	U	rd ← PC + imm20 << 12



https://cor3ntin.github.io/posts/abi/

break abi to save c++

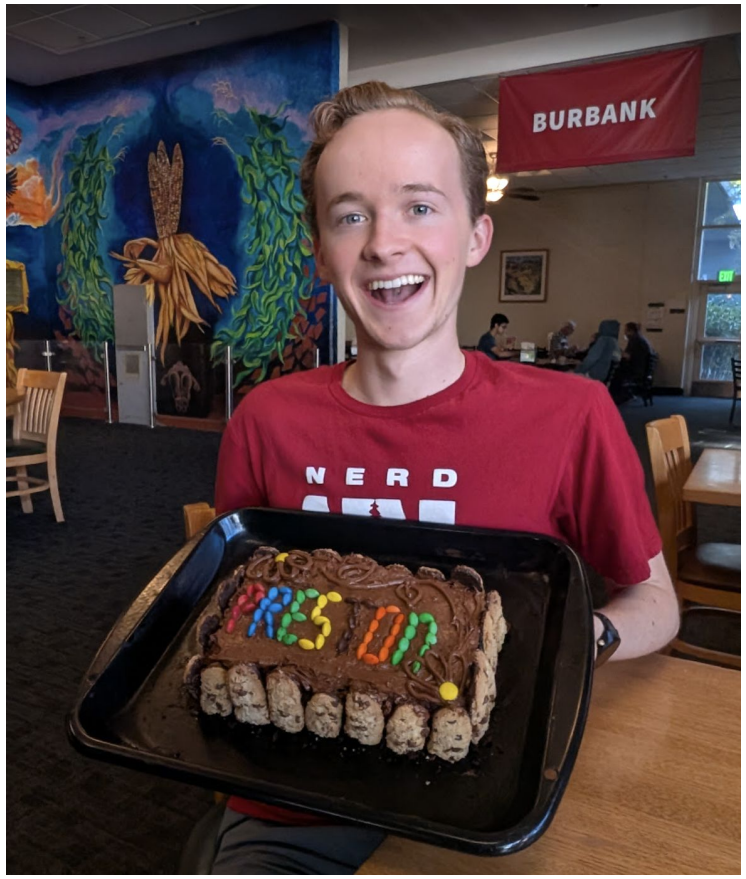
- ***Improvements*** to the ***current std*** would break the ABI, so they aren't implemented:
  - "Making associative container (much) faster
  - "Making `std::regex` faster (it is currently faster to launch PHP to execute a regex than it is to use `std::regex`)"

break abi to save c++

- ***Additions*** to the ***std*** would break the ABI, so they aren't implemented:
  - "unique_ptr could fit in register with language modifications, which would be needed to make it zero-overhead, compared to a pointer
  - "Adding UTF-8 support to regex is an ABI break 
  - "There seems to be a lot of people who believe that cost of exceptions could be greatly reduced as a quality of implementation matter but that might require breaking ABI."

KAHOOT TIME

Thank you for a great quarter!



pseay@stanford.edu

rfern@stanford.edu