# Type Safe C++? - LOL! :-)

Björn Fahller

# Type Safe C++? - LOL! :-)

Björn Fahller

# Type Safe C++? - LOL! :-)

What is type safety?

# What is type safety?

**type safety** *(Noun)*

the extent to which a programming language discourages or prevents type errors

-- *Wiktionary*

A type safe system discourages or prevents...

- ... use of one type when another is intended
- ... operations that do not make sense
- ... use of values outside the defined space

# Type Safe C++? - LOL! :-)

- Introduction to type safety
- **Type safety in C++**
- Simple library solution for strong types
- Sophisticated libraries – scouting github!
- What strong types does with your code

# My story begins

```
using request_id = uint32_t;
using receiver_id = uint32_t;

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

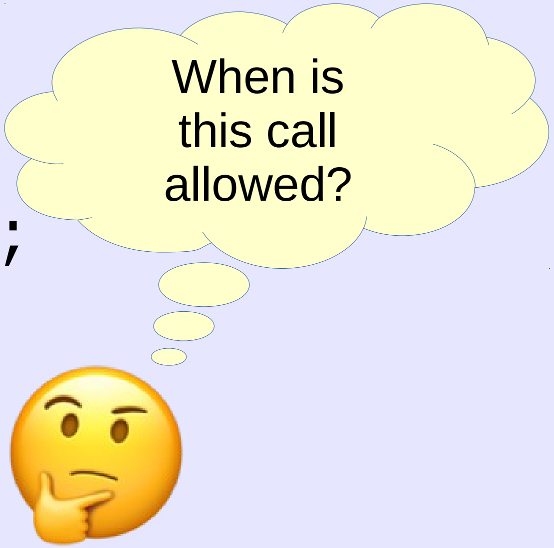
# My story begins

```
using request_id = uint32_t;
using receiver_id = uint32_t;

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

# My story begins

```
using request_id = uint32_t;
using receiver_id = uint32_t;

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

```
void other(A const& a);

void func(B b)
{
  other(b);
}
```



```cpp
using A = double;
using B = enum { aa, bb, cc };






void other(A const& a);

void func(B b)
{
  other(b);
}
```

```
struct A {
  int value;
};

struct B {
  int value;
};
void other(A const& a);

void func(B b)
{
  other(b);
}
```

```cpp
struct A {
  int value;
};

struct B {
  int value;
};
void other(A const& a);

void func(B b)
{
  other(b);
}
```

If we want this to compile, we can add:

```
struct A {
  int value;
};

struct B {
  int value;
};
void other(A const& a);

void func(B b)
{
  other(b);
}
```

If we want this to compile, we can add:

```
A::A(B const&); // not explicit
```

```
struct A {
  int value;
};

struct B {
  int value;
};
void other(A const& a);

void func(B b)
{
  other(b);
}
```

If we want this to compile, we can add:

```
A::A(B const&); // not explicit
B::operator A(); // not explicit
```

```
struct A {
  int value;
};

struct B {
  int value;
};
void other(A const& a);

void func(B b)
{
  other(b);
}
```

If we want this to compile, we can add:

```
A::A(B const&); // not explicit
B::operator A(); // not explicit
```

A as a public base class to B

# A different story begins

```
struct request_id { uint32_t value; };
struct receiver_id { uint32_t value; };

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  request_id req = new_request();
  return remove(receiver, req);
}
```

```
struct request_id { uint32_t value; };
struct receiver_id { uint32_t value; };

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
```

```
error: no matching function for call to 'remove'
  return remove(receiver, req);
         ^~~~~~
note: candidate function not viable:
 no known conversion from 'receiver_id' to 'request_id' for 1st argument
token remove(request_id req, receiver_id rec);
      ^
```


We have control over
when the compiler
will allow a conversion!

```
class receiver_id
{
public:
  explicit receiver_id(uint32_t v) : value{v} {}
  operator uint32_t() const { return value; }




private:
  uint32_t value;
};
```

```
class receiver_id
{
public:
  explicit receiver_id(uint32_t v) : value{v} {}
  operator uint32_t() const { return value; }
  bool operator==(receiver_id v) const {
    return value == v.value;
  }
  bool operator!=(receiver_id v) const;


private:
  uint32_t value;
};
```

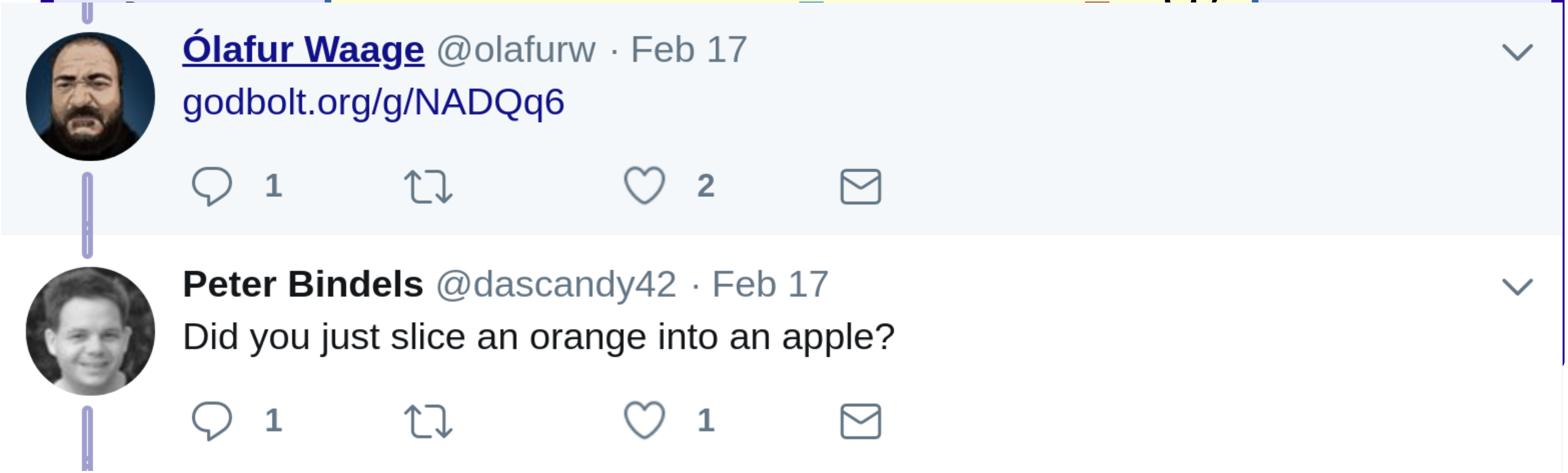
```
class receiver_id
{
public:
  explicit receiver_id(uint32_t v) : value{v} {}
  operator uint32_t() const { return value; }
  bool operator==(receiver_id v) const {
    return value == v.value;
  }
  bool operator!=(receiver_id v) const;
  bool operator<(receiver_id v) const;
  ...
private:
  uint32_t value;
};
```

```
class receiver_id
{
public:
  explicit receiver_id(uint32_t v) : value{v} {}
  operator uint32_t() const { return value; }
  bool operator==(receiver_id v) const {
    return
  }
  bool operator!=(receiver_id v) const;
  bool operator<(receiver_id v) const;
  ...
private:
  uint32_t value;
};
```

```
enum class receiver_id : uint32_t {};
```

```
class receiver_id
{
public:
  explicit receiver_id(uint32_t v) : value{v} {}
  operator uint32_t() const { return value; }
  bool ope
    return
```

```
enum class receiver_id : uint32_t {};
```



**Ólafur Waage** @olafurw · Feb 17
godbolt.org/g/NADQq6
1 2

**Peter Bindels** @dascandy42 · Feb 17
Did you just slice an orange into an apple?
1 1

```
  explicit receiver_ic
  operator uint32_t()
  bool operator==(rece
    return
  }
  bool operator!=(rece
  bool operator<(recei
  ...
private:
  uint32_t value;
};
```

```
enum class re
```

```
enum class Orange{};
enum class Apple{};

int main()
{
    Orange o{4};
    Apple a{3};
    Apple x{o}; // Oops
    // Apple y = o; // Fails

    return 0;
};
```

**Ólafur Waage** @olafurw · Feb 17

godbolt.org/g/NADQq6

1 2

**Peter Bindels** @dascandy42 · Feb 17

Did you just slice an orange into an apple?

1 1

```
1 enum class Orange{};
2 enum class Apple{};
3
4 int main()
```

```
explicit receiver_id
operator uint32_t()
```

**Shafik Yaghmour** @shafikyaghmour · Feb 18

Replying to @lefticus @bjorn_fahller and 4 others

Wow this seems rather undesirable, it looks like was introduced by this p0138r2 open-std.org/jtc1/sc22/wg21…

I may be reading it wrong but it seems like this effect w/ enums was unintended and it was only meant to deal w/ conversion from underlying type. Which I think makes sense.

> EWG did consider the request of extending the relaxation suggested in this paper to enumerations with declared enumerators, but ultimately rejected that suggestion.

**Ólafur Waage** @olafurw · Feb 17
godbolt.org/g/NADQq6

1 2

**Peter Bindels** @dascandy42 · Feb 17
Did you just slice an orange into an apple?

1 1

**Shafik Yaghmour** @shafikyaghmour · Feb 18
Replyi
Wow t
open-s

I may l
and it v
makes

EWG
decla

```
enum class Orange{};
enum class Apple{};

```

.s

**Richard Smith**
@zygoloid

Following

Replying to @shafikyaghmour

That appears to be an oversight in the wording; I don't think we intended to allow cases that require an explicit conversion to the enumeration's underlying type.

4:29 AM - 18 Feb 2018

**Ólafur Waage** @olafurw · Feb 17
godbolt.org/g/NADQq6

1 · 2

**Peter Bindels** @dascandy42 · Feb 17
Did you just slice an orange into an apple?

1 · 1

**Shafik Yaghmour** @shafikyaghmour · Feb 18

```
enum class Orange{};
enum class Apple{};
```

**Ólafur Waage**
@olafurw

Following

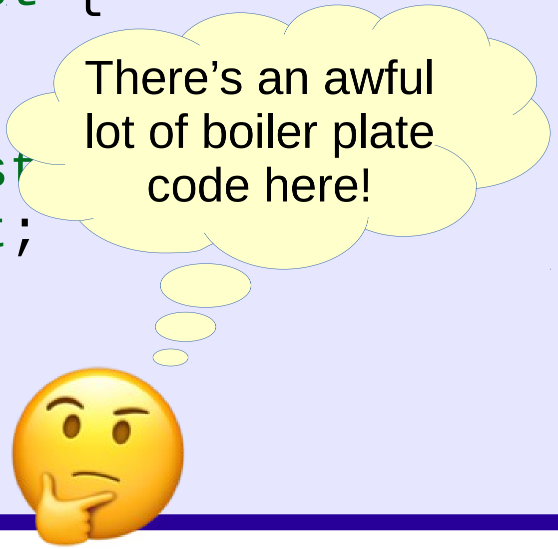
Replying to @shafikyaghmour @bjorn_fahller and 4 others

Did......did I just find a bug in the C++ standard?

2:21 AM - 18 Feb 2018

```cpp
class receiver_id
{
public:
  explicit receiver_id(uint32_t v) : value{v} {}
  operator uint32_t() const { return value; }
  bool operator==(receiver_id v) const {
    return value == v.value;
  }
  bool operator!=(receiver_id v) const;
  bool operator<(receiver_id v) const;
  ...
private:
  uint32_t value;
};
```

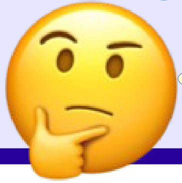
```
class receiver_id
{
public:
  explicit receiver_id(uint32_t v) : value{v} {}
  operator uint32_t() const { return value; }
  bool operator==(receiver_id v) const {
    return value == v.value;
  }
  bool operator!=(receiver_id v) const
  bool operator<(receiver_id v) const;
  ...
private:
  uint32_t value;
};
```



```
class receiver_id
{
public:
  explicit receiver_id(uint32_t v) : value{v} {}
  operator uint32_t() const { return value; }
  bool operator==(receiver_id v) const {
    return value == v.value;
  }
  bool operator!=(receiver_id v) const
  bool operator<(receiver_id v) const;
  ...
private:
  uint32_t value;
};
```

There's an awful lot of boiler plate code here!

Repeat once more for `request_id`



# Type Safe C++? - LOL! :-)

- Introduction to type safety
- Type safety in C++
- **Simple library solution for strong types**
- Sophisticated libraries – scouting github!
- What strong types does with your code

```cpp
template <typename T, typename tag>
class safe_type
{
public:




private:
  T value_;
};
```

```
template <typename T, typename tag>
class safe_type
{
public:
  safe_type(T t) : value_(std::move(t)) {}


  operator T() const { return value_; }
  // operators...
private:
  T value_;
};
```

```
template <typename T, typename tag>
class safe_type
{
public:
  safe_type(T t) : value_(std::move(t)) {}
  template <typename T2, typename tag2>
  safe_type(safe_type<T2, tag2> const&) = delete;
  operator T() const { return value_; }
  // operators...
private:
  T value_;
};
```

```cpp
template <typename T, typename tag>
class safe_type
{
public:
  safe_type(T t) : value_(std::move(t)) {}
  template <typename T2, typename tag2>
  safe_type(safe_type<T2, tag2> const&) = delete;
  operator T() const { return value_; }
  // operators...
private:
  T value_;
};

using int1 = safe_type<int, struct int1_>;
using int2 = safe_type<int, struct int2_>;
```

```
using request_id = safe_type<uint32_t, struct request_id_tag>;


using receiver_id = safe_type<uint32_t, struct receiver_id_tag>;



token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

```
using request_id = safe_type<uint32_t, struct request_id_tag>;


using receiver_id = safe_type<uint32_t, struct receiver_id_tag>;



token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

```
using request_id = safe_type<uint32_t, struct request_id_tag>;


using receiver_id = safe_type<uint32_t, struct receiver_id_tag>;



token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

```
using request_id = safe_type<uint32_t, struct request_id_tag>;

using receiver_id = safe_type<uint32_t, struct receiver_id_tag>;

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
```

```
error: no matching function for call to 'remove'
  remove(receiver, req);
  ^~~~~~
note: candidate function not viable: no known conversion
 from 'safe_type<[...], struct receiver_id_tag>'
 to 'safe_type<[...], struct request_id_tag>' for 1st argument
token remove(request_id req, receiver_id rec);
      ^
```

```
using request_id = safe_type<uint32_t, struct request_id_tag>;


using receiver_id = safe_type<uint32_t, struct receiver_id_tag>;



token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

```
struct request_id : safe_type<uint32_t, request_id> {
  using safe_type::safe_type;
};
struct receiver_id : safe_type<uint32_t, receiver_id> {
  using safe_type::safe_type;
};

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

```
struct request_id : safe_type<uint32_t, request_id> {
  using safe_type::safe_type;
};
struct receiver_id : safe_type<uint32_t, receiver_id> {
  using safe_type::safe_type;
};

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
```

```
error: no matching function for call to 'remove'
  remove(receiver, req);
  ^~~~~~
note: candidate function not viable: no known conversion
 from 'receiver_id' to 'request_id' for 1st argument
token remove(request_id req, receiver_id rec);
      ^
```

```
struct request_id : safe_type<uint32_t, request_id> {
  using safe_type::safe_type;
};
struct receiver_id : safe_type<uint32_t, receiver_id> {
  using safe_type::safe_type;
};

token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

```
#define SAFE_TYPE(name, base_type)             \
struct name : safe_type<base_type, name> { \
  using safe_type::safe_type;                  \
}
```

```
SAFE_TYPE(request_id, uint32_t);


SAFE_TYPE(receiver_id, uint32_t);



token remove(request_id req, receiver_id rec);

token initiate_remove(receiver_id receiver)
{
  auto req = new_request();
  return remove(receiver, req);
}
```

```
#define SAFE_TYPE(name, base_type)           \
struct name : safe_type<base_type, name> { \
  using safe_type::safe_type;               \
}
```

```cpp
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_name const& ifname,
                     customer_name const& customer);

interface_name lookup_interface(MAC_address mac);

void setup_customer(MAC_address mac,
                    customer_name const& customer)
{
  assert(!customer.empty());
  auto if_name = lookup_interface(mac);
  assert(if_name.find(':') != std::string::npos);
  label_interface(customer, if_name);
}
```

```
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_name const& ifname,
                     customer_name const& customer);

interface_name lookup_interface(MAC_address mac);

void setup_customer(MAC_address mac,
                    customer_name const& customer)
{
  assert(!customer.empty());
  auto if_name = lookup_interface(mac);
  assert(if_name.find(':') != std::string::npos);
  label_interface(customer, if_name);
}
```

```cpp
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_name const& ifname,
                     customer_name const& customer);

interface_name lookup_interface(MAC_address mac);

void setup_customer(MAC_address mac,
                    customer_name const& customer)
{
  assert(!customer.empty());
  auto if_name = lookup_interface(mac);
  assert(if_name.find(':') != std::string::npos);
  label_interface(customer, if_name);
}
```

Accidental swap!

```
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_name const& ifname,
                     customer_name const& customer);
```

```
template <typename T,
          typename tag,
          bool = std::is_class<T>{} && !std::is_final<T>{}>
class safe_type { /* as before */};
```

```
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_name const& ifname,
                     customer_name const& customer);
```

```
template <typename T,
          typename tag,
          bool = std::is_class<T>{} && !std::is_final<T>{}>
class safe_type { /* as before */};

template <typename T, typename tag>
struct safe_type<T, tag, true> : T
{
  using T::T;
  template <typename T2, typename tag2>
  safe_type(safe_type<T2,tag2> const&) = delete;
};
```

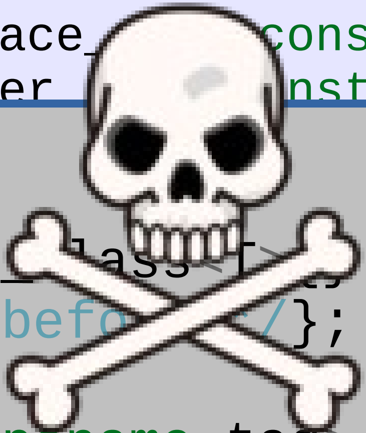
```
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_name const& ifname,
                     customer_name const& customer);

interface_name lookup_interface(MAC_address mac);

void setup_customer(MAC_address mac,
                    customer_name const& customer)
{
```

```
error: no matching function for call to 'label_interface'
  label_interface(customer, if_name);
  ^~~~~~~~~~~~~~~
note: candidate function not viable: no known conversion
 from 'customer_name'
 to 'const interface_name' for 1st argument
void label_interface(const interface_name& ifname,
     ^
```

```
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_name const& ifname,
                     customer_name const& customer);
```

```
template <typename T,
          typename tag,
          bool = std::is_class<T>{} && !std::is_final<T>{}>
class safe_type { /* as before */};

template <typename T, typename tag>
struct safe_type<T, tag, true> : T
{
  using T::T;
  template <typename T2, typename tag2>
  safe_type(safe_type<T2,tag2> const&) = delete;
};
```

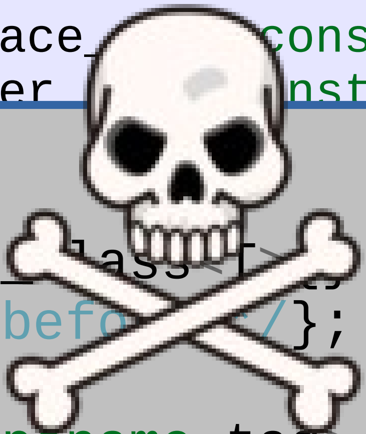
```
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_name const& ifname,
                     customer_name const& customer);
```



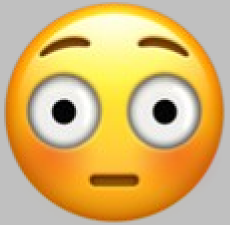
```
template <typename T,
          typename tag,
          bool = std::is_class<T>{} && !std::is_final<T>{}>
class safe_type { /* as before */};

template <typename T, typename tag>
struct safe_type<T, tag, true> : T
{
  using T::T;
  template <typename T2, typename tag2>
  safe_type(safe_type<T2,tag2> const&) = delete;
};
```

```
SAFE_TYPE(interface_name, std::string);
SAFE_TYPE(customer_name, std::string);

void label_interface(interface_     const&
                     customer_     nst&
```

```
template <typename T,
          typename tag,
          bool = std::is_class<T>{} && !std::is_final<T>{}>
class safe_type { /* as before */};

template <typename T, typename tag>
struct safe_type<T, tag, true> : T
{
  using T::T;
  template <typename T2, typename tag2>
  safe_type(safe_type<T2,tag2> const&) = delete;
};
```



Oh, no, I violated the
Liskov Substitution Principle!



**Liskov Substitution Principle**

Subtype Requirement:
Let *φ(x)* be a property provable about objects *x* of type *T*.
Then *φ(y)* should be true for objects *y* of type *S*
where *S* is a subtype of *T*.

```
;
nst& ifname,
st& customer);
template <typename T,
          typename tag,
          bool = std::is_class<T>{} && !std::is_final<T>{}>
class safe_type { /* as before */};

template <typename T, typename tag>
struct safe_type<T, tag, true> : T
{
  using T::T;
  template <typename T2, typename tag2>
  safe_type(safe_type<T2,tag2> const&) = delete;
};
```

**Liskov Substitution Principle**

Subtype Requirement:
Let *φ(x)* be a property provable about objects *x* of type *T*.
Then *φ(y)* should be true for objects *y* of type *S*
where *S* is a subtype of *T*.

**Robert (Uncle Bob) Martin**

Functions that use pointers or references
to base classes must be able to use objects
of derived classes without knowing it.

```
;
nst& ifname,
st& customer);
} && !std::is_final<T>{}>
;
>
struct safe_type<T, tag, true> : T
{
  using T::T;
  template <typename T2, typename tag2>
  safe_type(safe_type<T2,tag2> const&) = delete;
};
```

**Liskov Substitution Principle**

Subtype Requirement:
Let *φ(x)* be a property provable about objects *x* of type *T*. Then *φ(y)* should be true for objects *y* of type *S* where *S* is a subtype of *T*.

**customer_name** and **interface_name** are different types implemented in terms of strings

**Robert (Uncle Bob) Martin**

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.

```
  auto if_name = lookup_interface(mac);
  assert(if_name.find(':') != std::string::npos);
  label_interface(customer, if_name);
}
```

**Liskov Substitution Principle**

Subtype Requirement:
Let *φ(x)* be a property provable about objects *x* of type *T*. Then *φ(y)* should be true for objects *y* of type *S* where *S* is a subtype of *T*.

**Robert (Uncle Bob) Martin**

Functions that use pointers of reference to base classes must be able to use objects of derived classes without knowing it.

**`customer_name`** and **`interface_name`** are different types <u>implemented in terms of strings</u>

```
;
nst& ifname,
  assert(!customer.empty());
  auto if_name = lookup_interface(mac);
  assert(if_name.find(':') != std::string::npos);
  label_interface(customer, if_name);
}
```

**Liskov Substitution Principle**

Subtype Requirement:
Let *φ(x)* be a property provable about objects *x* of type *T*.
Then *φ(y)* should be true for objects *y* of type *S*
where *S* is a subtype of *T*.

**Robert (Uncle Bob) Martin**

Functions that use pointers of reference
to base classes must be able to use objects
of derived classes without knowing it.

**`customer_name`** and **`interface_name`**
are different types implemented in terms
of strings, <u>but they are not strings</u>.

```
;
nst& ifname,
  assert(!customer.empty());
  auto if_name = lookup_interface(mac);
  assert(if_name.find(':') != std::string::npos);
  label_interface(customer, if_name);
}
```

**Liskov Substitution Principle**

Subtype Requirement:
Let *φ(x)* be a property provable about objects *x* of type *T*.
Then *φ(y)* should be true for objects *y* of type *S*
where *S* is a subtype of *T*.

**Robert (Uncle Bob) Martin**

Functions that use pointers of reference
to base classes must be able to use objects
of derived classes without knowing it.

**`customer_name`** and **`interface_name`**
are different types implemented in terms
of strings, but they are not strings.
Maybe it makes sense to allow unlimited access
to the non-mutating functions of
**`std::string`**, but not to all mutating ones.

```
                                        ;

                                        nst& ifname,

  assert(!customer.empty());
  auto if_name = lookup_interface(mac);
  assert(if_name.find(':') != std::string::npos);
  label_interface(customer, if_name);
}
```

# Type Safe C++? - LOL! :-)

- Introduction to type safety
- Type safety in C++
- Simple library solution for strong types
- **Sophisticated libraries – scouting github!**
- What strong types does with your code

# Jonathan Müller @foonathan

type_safe

Zero overhead utilities for preventing bugs at compile time

https://github.com/foonathan/type_safe

# Jonathan Müller @foonathan

type_safe

Zero overhead utilities for preventing bugs at compile time

https://github.com/foonathan/type_safe

A rich type library, with which you can piece together the exact behaviour of a type that you want.

It also includes a number of predefined neat type templates, and other features like improved `optional<T>` and `variant<T...>`

# Jonathan Müller @foonathan

## type_safe

Zero overhead utilities for preventing bugs at compile time

https://github.com/foonathan/type_safe

A rich type library, with which you can piece together the exact behaviour of a type that you want.

It also includes a number of predefined neat type templates, and other features like improved optional<T> and variant<T...>

Since October 2016

https://github.com/foonathan/type_safe

```
// type_safe/strong_typedef.hpp

template <class Tag, typename T>
class type_safe::strong_typedef {
public:
  constexpr strong_typedef();

  explicit constexpr strong_typedef(const T& value);
  explicit constexpr strong_typedef(T&& value);

  explicit constexpr operator T&() & noexcept;
  explicit constexpr operator const T&() const & noexcept;
  explicit constexpr operator T&&() && noexcept;
  explicit constexpr operator const T&&() const && noexcept;
};
```

https://github.com/foonathan/type_safe

```
#include <type_safe/strong_typedef.hpp>
```

https://github.com/foonathan/type_safe

```
#include <type_safe/strong_typedef.hpp>
namespace ts = type_safe;
```

https://github.com/foonathan/type_safe

```
#include <type_safe/strong_typedef.hpp>
namespace ts = type_safe;

struct my_handle : ts::strong_typedef<my_handle, int>



{
  using strong_typedef::strong_typedef;
};
```

https://github.com/foonathan/type_safe

```cpp
#include <type_safe/strong_typedef.hpp>
namespace ts = type_safe;
namespace op = type_safe::strong_typedef_op;
struct my_handle : ts::strong_typedef<my_handle, int>
                 , op::equality_comparison<my_handle>

{
  using strong_typedef::strong_typedef;
};
```

https://github.com/foonathan/type_safe

```
#include <type_safe/strong_typedef.hpp>
namespace ts = type_safe;
namespace op = type_safe::strong_typedef_op;
struct my_handle : ts::strong_typedef<my_handle, int>
                 , op::equality_comparison<my_handle>
                 , op::output_operator<my_handle>
{
  using strong_typedef::strong_typedef;
};
```

https://github.com/foonathan/type_safe

```cpp
#include <type_safe/strong_typedef.hpp>
namespace ts = type_safe;
namespace op = type_safe::strong_typedef_op;
struct my_handle : ts::strong_typedef<my_handle, int>
                 , op::equality_comparison<my_handle>
                 , op::output_operator<my_handle>
{
  using strong_typedef::strong_typedef;
};

struct my_int : ts::strong_typedef<my_int, int>
              , op::integer_arithmetic<my_int>
{
  using strong_typedef::strong_typedef;
};
```

https://github.com/foonathan/type_safe

```cpp
#include <type_safe/strong_typedef.hpp>
namespace ts = type_safe;
namespace op = type_safe::strong_typedef_op;
struct my_handle : ts::strong_typedef<my_handle, int>
                 , op::equality_comparison<my_handle>
{
  using strong_typedef::strong_typedef;

  friend std::ostream&
  operator<<(std::ostream& os, my_handle const& h)
  {
    return os << "H{" << static_cast<const int&>(h) << "}";
  }
};
```

https://github.com/foonathan/type_safe

```cpp
#include <type_safe/strong_typedef.hpp>
namespace ts = type_safe;
namespace op = type_safe::strong_typedef_op;
struct my_handle : ts::strong_typedef<my_handle, int>
                 , op::equality_comparison<my_handle>
{
  using strong_typedef::strong_typedef;

  friend std::ostream&
  operator<<(std::ostream& os, my_handle const& h)
  {
    return os << "H{" << ts::get(h) << "}";
  }
};
```

# Jonathan Boccara @joboccara

NamedType

Implementation of strong types in C++

https://github.com/joboccara/NamedType

# Jonathan Boccara `@joboccara`

## NamedType

Implementation of strong types in C++

`https://github.com/joboccara/NamedType`

A small type library with a simpler aim, but which still allows you to piece together the strong types with your desired behaviour.

It also supports conversions between different types of the same kind, for example meters to feet, or non-linear like Watt to dB.

# Jonathan Boccara @joboccara

## NamedType

Implementation of
strong types in C++

https://github.com/joboccara/NamedType

A small type library with a simpler aim, but which still allows you to piece together the strong types with your desired behaviour.

It also supports conversions between different types of the same kind, for example meters to feet, or non-linear like Watt to dB.

MeetingC++ [https://www.youtube.com/watch?v=WVleZqzTw2k](https://www.youtube.com/watch?v=WVleZqzTw2k)

https://github.com/joboccara/NamedType

```
// NamedType/named_type.hpp

using my_handle =
    fluent::NamedType<
      int, struct my_handle_tag
    >;
```

https://github.com/joboccara/NamedType

```
// NamedType/named_type.hpp

using my_handle =
    fluent::NamedType<
      int, struct my_handle_tag,
      fluent::comparable,
      fluent::printable,
      fluent::hashable
    >;
```

```cpp
// NamedType/named_type.hpp

struct my_handle
  : fluent::NamedType<
      int, my_handle,
      fluent::comparable,
      fluent::printable,
      fluent::hashable
    >

{
  using NamedType::NamedType;
};
```

https://github.com/joboccara/NamedType

```
// NamedType/named_type.hpp

struct my_handle
 : fluent::NamedType<
      int, my_handle,
      fluent::comparable,
      fluent::printable,
      fluent::hashable,
      fluent::ImplicitlyConvertibleTo<int>::templ
   >
{
  using NamedType::NamedType;
};
```

# Type Safe C++? - LOL! :-)

- Introduction to type safety
- Type safety in C++
- Simple library solution for strong types
- Sophisticated libraries – scouting github!
- **What strong types does with your code**

# Network capacity utilisation

A slot is a network capacity quanta

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

# Network capacity utilisation

A slot is a network capacity quanta

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | SlotCount | SlotIndexes | SlotRanges |
|---|---|---|---|
| | 8 | 3,4,7,8,13,14,15,16 | {3-4},{7-8},{13-16} |
| | 3 | 5,9,10 | {5},{9-10} |
| | 13 | 0,1,2,6,11,12,17,18, 19,20,21,22,23 | {0-2},{6},{11-12},{17-23} |

# Network capacity utilisation

A slot is a network capacity quanta

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | SlotCount | SlotIndexes | SlotRanges |
|---|---|---|---|
| | 8 | 3,4,7,8,13,14,15,16 | {3-4},{7-8},{13-16} |
| | 3 | 5,9,10 | {5},{9-10} |
| | 13 | 0,1,2,6,11,12,17,18, 19,20,21,22,23 | {0-2},{6},{11-12},{17-23} |

```
typename SlotIndex;
```

```
typename SlotCount;
```

```
struct SlotRange {
  SlotIndex start;
  SlotCount length;
};
```

# *Magic Numbers*

```
SlotCount availableCapacity();

...

if (availableCapacity() == 0) {
  ...
}
```

```
SlotCount availableCapacity();

...

if (availableCapacity() == SlotCount{0}) {
  ...
}
```

```
SlotCount availableCapacity();
constexpr SlotCount noSlots{0};
...

if (availableCapacity() == noSlots) {
  ...
}
```

# *Encapsulation*

```
class MessageBuffer
{
public:




  template <size_t bits>
  void serialize_bits(unsigned value);
};






SlotCount capacity = ...
MessageBuffer buffer ...
buffer.serialize_bits<24>(capacity);
```

```
class MessageBuffer
{
public:



   template <size_t bits>
   void serialize_bits(unsigned value);
};





SlotCount capacity = ...
MessageBuffer buffer ...
buffer.serialize_bits<24>(capacity);
```

```
class MessageBuffer
{
public:




  template <size_t bits>
  void serialize_bits(unsigned value);
};
void serialize_data(MessageBuffer& b, SlotCount const& c)
{
  b.serialize_bits<24>(c);
}

SlotCount capacity = ...
MessageBuffer buffer ...
serialize_data(buffer, capacity);
```

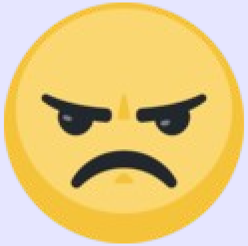
```
class MessageBuffer
{
public:
  template <typename T>
  void serialize(T const& t) { serialize_data(*this, t); }

  template <size_t bits>
  void serialize_bits(unsigned value);
};
void serialize_data(MessageBuffer& b, SlotCount const& c)
{
  b.serialize_bits<24>(c);
}

SlotCount capacity = ...
MessageBuffer buffer ...
buffer.serialize(capacity);
```

# *Type Semantics*

```
class SlotPool
{
public:
  void releaseCapacity(std::vector<SlotRange> const& ranges)








  SlotCount availableCapacity() const { return unusedSlots;}
  ...
private:
  SlotCount unusedSlots;
  ...
};
```

```cpp
class SlotPool
{
public:
  void releaseCapacity(std::vector<SlotRange> const& ranges)
  {
    for (auto& range : ranges)
    {
      ususedSlots += range.length;
    }
    ...
  }
  SlotCount availableCapacity() const { return unusedSlots;}
  ...
private:
  SlotCount unusedSlots;
  ...
};
```

```
class SlotPool
{
public:
  void releaseCapacity(std::vector<SlotRange> const& ranges)
  {
    for (auto& range : ranges)
    {
      ususedSlots += range.length;
    }
    ...
  }
  SlotCount availableCapac      ) const { return unusedSlots;}
  ...
private:
  SlotCount unusedSlots;
  ...
};
```



Does not compile!
No operator += for SlotCount

# Which operations makes sense?

Frame with 24 slots

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

`SlotCount+SlotCount?`

# Which operations makes sense?

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

```
SlotCount+SlotCount->SlotCount
```

# Which operations makes sense?

| Frame with 24 slots | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

```
SlotCount+SlotCount->SlotCount

SlotCount-SlotCount?
```

# Which operations makes sense?

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

```
SlotCount+SlotCount->SlotCount
```

```
SlotCount-SlotCount->SlotCount
```

# Which operations makes sense?

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

```
SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

SlotCount*SlotCount?
```

# Which operations makes sense?

Frame with 24 slots

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

`SlotCount+SlotCount->SlotCount`

`SlotCount-SlotCount->SlotCount`

~~SlotCount*SlotCount~~

# Which operations makes sense?

Frame with 24 slots

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

`SlotCount+SlotCount->SlotCount`

`SlotCount-SlotCount->SlotCount`

~~SlotCount*SlotCount~~

`SlotCount*Ratio?`

# Which operations makes sense?

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

`SlotCount+SlotCount->SlotCount`

`SlotCount-SlotCount->SlotCount`

~~`SlotCount*SlotCount`~~

`SlotCount*Ratio->SlotCount`

# Which operations makes sense?

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

```
SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount
```

~~SlotCount*SlotCount~~

```
SlotCount*Ratio->SlotCount

SlotCount/SlotCount?
```

# Which operations makes sense?

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

```
SlotCount+SlotCount->SlotCount
SlotCount-SlotCount->SlotCount
```

~~`SlotCount*SlotCount`~~

```
SlotCount*Ratio->SlotCount
SlotCount/SlotCount->Ratio
```

# Which operations makes sense?

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

```
SlotCount+SlotCount->SlotCount
SlotCount-SlotCount->SlotCount
```

~~`SlotCount*SlotCount`~~

```
SlotCount*Ratio->SlotCount
SlotCount/SlotCount->Ratio
SlotCount/Ratio?
```

# Which operations makes sense?

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

# Which operations makes sense?

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SlotCount+SlotCount->SlotCount

SlotIndex+SlotIndex?

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

# Which operations makes sense?

Frame with 24 slots

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

`SlotCount+SlotCount->SlotCount`

~~`SlotIndex+SlotIndex`~~

`SlotCount-SlotCount->SlotCount`

~~`SlotCount*SlotCount`~~

`SlotCount*Ratio->SlotCount`

`SlotCount/SlotCount->Ratio`

`SlotCount/Ratio->SlotCount`

# Which operations makes sense?

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

`SlotCount+SlotCount->SlotCount`

`SlotCount-SlotCount->SlotCount`

~~`SlotCount*SlotCount`~~

`SlotCount*Ratio->SlotCount`

`SlotCount/SlotCount->Ratio`

`SlotCount/Ratio->SlotCount`

~~`SlotIndex+SlotIndex`~~

`SlotIndex+SlotCount?`

# Which operations makes sense?

Frame with 24 slots

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

`SlotCount+SlotCount->SlotCount`

`SlotCount-SlotCount->SlotCount`

~~SlotCount*SlotCount~~

`SlotCount*Ratio->SlotCount`

`SlotCount/SlotCount->Ratio`

`SlotCount/Ratio->SlotCount`

~~SlotIndex+SlotIndex~~

`SlotIndex+SlotCount->SlotIndex`

# Which operations makes sense?

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

`SlotCount+SlotCount->SlotCount`

`SlotCount-SlotCount->SlotCount`

~~`SlotCount*SlotCount`~~

`SlotCount*Ratio->SlotCount`

`SlotCount/SlotCount->Ratio`

`SlotCount/Ratio->SlotCount`

~~`SlotIndex+SlotIndex`~~

`SlotIndex+SlotCount->SlotIndex`

`SlotIndex-SlotIndex?`

# Which operations makes sense?

Frame with 24 slots

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

# Which operations makes sense?

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

`SlotCount+SlotCount->SlotCount`

`SlotCount-SlotCount->SlotCount`

~~`SlotCount*SlotCount`~~

`SlotCount*Ratio->SlotCount`

`SlotCount/SlotCount->Ratio`

`SlotCount/Ratio->SlotCount`

~~`SlotIndex+SlotIndex`~~

`SlotIndex+SlotCount->SlotIndex`

`SlotIndex-SlotIndex->SlotCount`

`SlotIndex/SlotIndex?`

# Which operations makes sense?

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

# Which operations makes sense?

Frame with 24 slots

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

SlotIndex/SlotCount?

# Which operations makes sense?

Frame with 24 slots

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

# Which operations makes sense?

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

SlotIndex/Ratio?

# Which operations makes sense?

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

~~SlotIndex/Ratio~~

# Which operations makes sense?

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

~~SlotIndex/Ratio~~

SlotIndex*SlotIndex?

# Which operations makes sense?

| Frame with 24 slots | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

~~SlotIndex/Ratio~~

~~SlotIndex*SlotIndex~~

# Which operations makes sense?

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

~~SlotIndex/Ratio~~

~~SlotIndex*SlotIndex~~

SlotIndex*SlotCount?

# Which operations makes sense?

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

~~SlotIndex/Ratio~~

~~SlotIndex*SlotIndex~~

~~SlotIndex*SlotCount~~

# Which operations makes sense?

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

~~SlotIndex/Ratio~~

~~SlotIndex*SlotIndex~~

~~SlotIndex*SlotCount~~

SlotIndex*Ratio?

# Which operations makes sense?

| Frame with 24 slots | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

SlotCount+SlotCount->SlotCount

SlotCount-SlotCount->SlotCount

~~SlotCount*SlotCount~~

SlotCount*Ratio->SlotCount

SlotCount/SlotCount->Ratio

SlotCount/Ratio->SlotCount

~~SlotIndex+SlotIndex~~

SlotIndex+SlotCount->SlotIndex

SlotIndex-SlotIndex->SlotCount

~~SlotIndex/SlotIndex~~

~~SlotIndex/SlotCount~~

~~SlotIndex/Ratio~~

~~SlotIndex*SlotIndex~~

~~SlotIndex*SlotCount~~

~~SlotIndex*Ratio~~

# Which operations makes sense?

Frame with 24 slots

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

`SlotCount+SlotCount->SlotCount` ~~`SlotIndex+SlotIndex`~~

`SlotCount-SlotCount->SlotCount` `SlotIndex+SlotCount->SlotIndex`

~~`SlotCount*Sl`~~ `>SlotCount`

`SlotCount*Ra`

`SlotCount/Sl`

`SlotCount/Ra`

**Affine Space**

In mathematics, an affine space is a geometric structure that generalizes the properties of Euclidean spaces in such a way that these are independent of the concepts of distance and measure of angles, keeping only the properties related to parallelism and ratio of lengths for parallel line segments...

-- Wikipedia

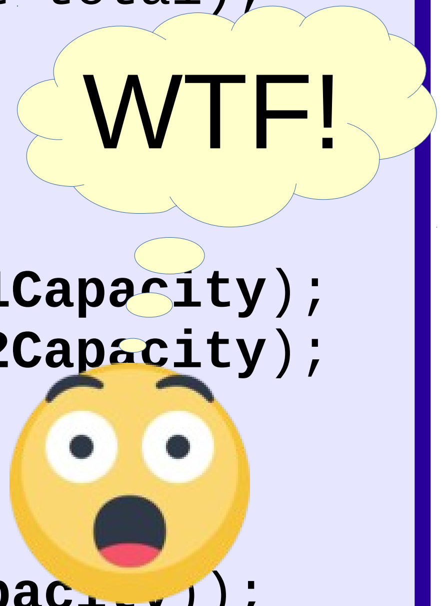
# *Test Code*

```
DestClient::newCapacity(RequestId, SlotCount);
TestNode::throttleCapacityTo(RequestId, SlotCount total);

TEST(capacity_decrease_is_notified_to_clients) {
  TestNode node;

  DestClient client1 = node.clientWithCapacity(5);
  DestClient client2 = node.clientWithCapacity(8);



  REQUIRE_CALL(client1, newCapacity(4, 2));
  REQUIRE_CALL(client2, newCapacity(4, 3));

  node.throttleCapacityTo(4, 5);
}
```

```
DestClient::newCapacity(RequestId, SlotCount);
TestNode::throttleCapacityTo(RequestId, SlotCount total);

TEST(capacity_decrease_is_notified_to_clients) {
  TestNode node;

  DestClient client1 = node.clientWithCapacity(5);
  DestClient client2 = node.clientWithCapacity(8);

  RequestId req{4};

  REQUIRE_CALL(client1, newCapacity(req, 2));
  REQUIRE_CALL(client2, newCapacity(req, 3));

  node.throttleCapacityTo(req, 5);
}
```

```
DestClient::newCapacity(RequestId, SlotCount);
TestNode::throttleCapacityTo(RequestId, SlotCount total);

TEST(capacity_decrease_is_notified_to_clients) {
  TestNode node;
  SlotCount c1Capacity{5}, c2Capacity{8};
  DestClient client1 = node.clientWithCapacity(c1Capacity);
  DestClient client2 = node.clientWithCapacity(c2Capacity);

  RequestId req{4};
  SlotCount newC1Capacity{2}, newC2Capacity{3};
  REQUIRE_CALL(client1, newCapacity(req, newC1Capacity));
  REQUIRE_CALL(client2, newCapacity(req, newC2Capacity));
  SlotCount newTotalCapacity{5};
  node.throttleCapacityTo(req, newTotalCapacity);
}
```

```
DestClient::newCapacity(RequestId, SlotCount);
TestNode::throttleCapacityTo(RequestId, SlotCount total);

TEST(capacity_decrease_is_notified_to_clients) {
  TestNode node;
  SlotCount c1Capacity{5}, c2Capacity{8};
  DestClient client1 = node.clientWithCapacity(c1Capacity);
  DestClient client2 = node.clientWithCapacity(c2Capacity);

  RequestId req{4};
  SlotCount newC1Capacity{2}, newC2Capacity{3};
  REQUIRE_CALL(client1, newCapacity(req, newC1Capacity));
  REQUIRE_CALL(client2, newCapacity(req, newC2Capacity));
  SlotCount newTotalCapacity{5};
  node.throttleCapacityTo(req, newTotalCapacity);
}
```



```
DestClient::newCapacity(RequestId, SlotCount);
TestNode::throttleCapacityTo(RequestId, SlotCount total);

TEST(capacity_decrease_is_notified_to_clients) {
  TestNode node;

  DestClient client1 = node.clientWithCapacity(SlotCount{5});
  DestClient client2 = node.clientWithCapacity(SlotCount{8});

  RequestId req{4};

  REQUIRE_CALL(client1, newCapacity(req, SlotCount{2}));
  REQUIRE_CALL(client2, newCapacity(req, SlotCount{3}));

  node.throttleCapacityTo(req, SlotCount{5});
}
```

```
constexpr SlotCount operator"" _slots(unsigned long long v)
{
  auto cv = static_cast<unsigned>(v);

  return SlotCount{cv};
}
  DestClient client1 = node.clientWithCapacity(SlotCount{5});
  DestClient client2 = node.clientWithCapacity(SlotCount{8});

  RequestId req{4};

  REQUIRE_CALL(client1, newCapacity(req, SlotCount{2}));
  REQUIRE_CALL(client2, newCapacity(req, SlotCount{3}));

  node.throttleCapacityTo(req, SlotCount{5});
}
```

```
constexpr SlotCount operator"" _slots(unsigned long long v)
{
  auto cv = static_cast<unsigned>(v);

  return SlotCount{cv};
}
```

```
  DestClient client1 = node.clientWithCapacity(5_slots);
  DestClient client2 = node.clientWithCapacity(8_slots);

  RequestId req{4};

  REQUIRE_CALL(client1, newCapacity(req, 2_slots));
  REQUIRE_CALL(client2, newCapacity(req, 3_slots));

  node.throttleCapacityTo(req, 5_slots);
}
```

# Type Safe C++? - LOL! :-)

- Introduction to type safety
- Type safety in C++
- Simple library solution for strong types
- Sophisticated libraries – scouting github!
- What strong types does with your code

# Type Safe C++? - LOL! :-)

- Safety for built in types is abysmal in C++

# Type Safe C++? - LOL! :-)

- Safety for built in types is abysmal in C++
- Structs/classes are as strong as you wish
  - You **must add** the functionality you want

# Type Safe C++? - LOL! :-)

- Safety for built in types is abysmal in C++
- Structs/classes are as strong as you wish
  - You **must add** the functionality you want
- Libraries exist that makes this easier

# Type Safe C++? - LOL! :-)

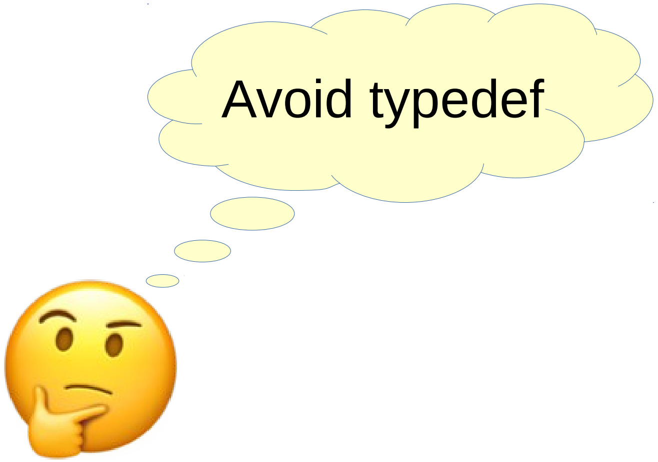
- Safety for built in types is abysmal in C++
- Structs/classes are as strong as you wish
  - You **must add** the functionality you want
- Libraries exist that makes this easier
- Thinking about what operations your types should support makes you understand the problem better!
  - Beware of the urge for convencience!

# Type Safe C++? - LOL! :-)

- Safety for built in types is abysmal in C++
- Structs/classes are as strong as you wish
  - You **must add** the functionality you want
- Libraries exist that makes this easier
- Thinking about what operations your types should support makes you understand the problem better!
  - Beware of the urge for convencience!
- Strong types leads to more expressive code
  - Fewer magical numbers
  - More encapsulation
  - Explicit tests that express intent

# Type Safe C++? - LOL! :-)

- Safety for built in types is abysmal in C++
- Structs/classes are as strong as you wish
  - You **must add** the functionality you want
- Libraries exist that makes this easier
- Thinking about what operations your types should support makes you understand the problem better!
  - Beware of the urge for convencience!
- Strong types leads to more expressive code
  - Fewer magical numbers
  - More encapsulation
  - Explicit tests that express intent



# Type Safe C++? - LOL! :-)

Björn Fahller

bjorn@fahller.se

@bjorn_fahller

@rollbear *cpplang, swedencpp*