If structured binding

Zhihao Yuan

2024/5/1

The code I’m able to write

Symantec

```
if (auto res = std::to_chars(first, last, 42))
    return std::string(first, res.ptr);
else
    throw std::system_error(std::make_error_code(res.ec));
```

The code I want to write

Symantec

```
if (auto [ec, ptr] = std::to_chars(first, last, 42))
    return std::string(first, ptr);
else
    throw std::system_error(std::make_error_code(ec));
```

What I'm proposing?

Symantec

- A grammar that allows structured binding declaration to appear in place of a *condition*
- In this specific example,

```
if (auto [a, b, c] = fn())
{
    statements;
}
```

condition

is equivalent to

```
if (auto [a, b, c] = fn(); underlying-object)
{
    statements;
}
```

init-statement

Where can I use this?

Symantec

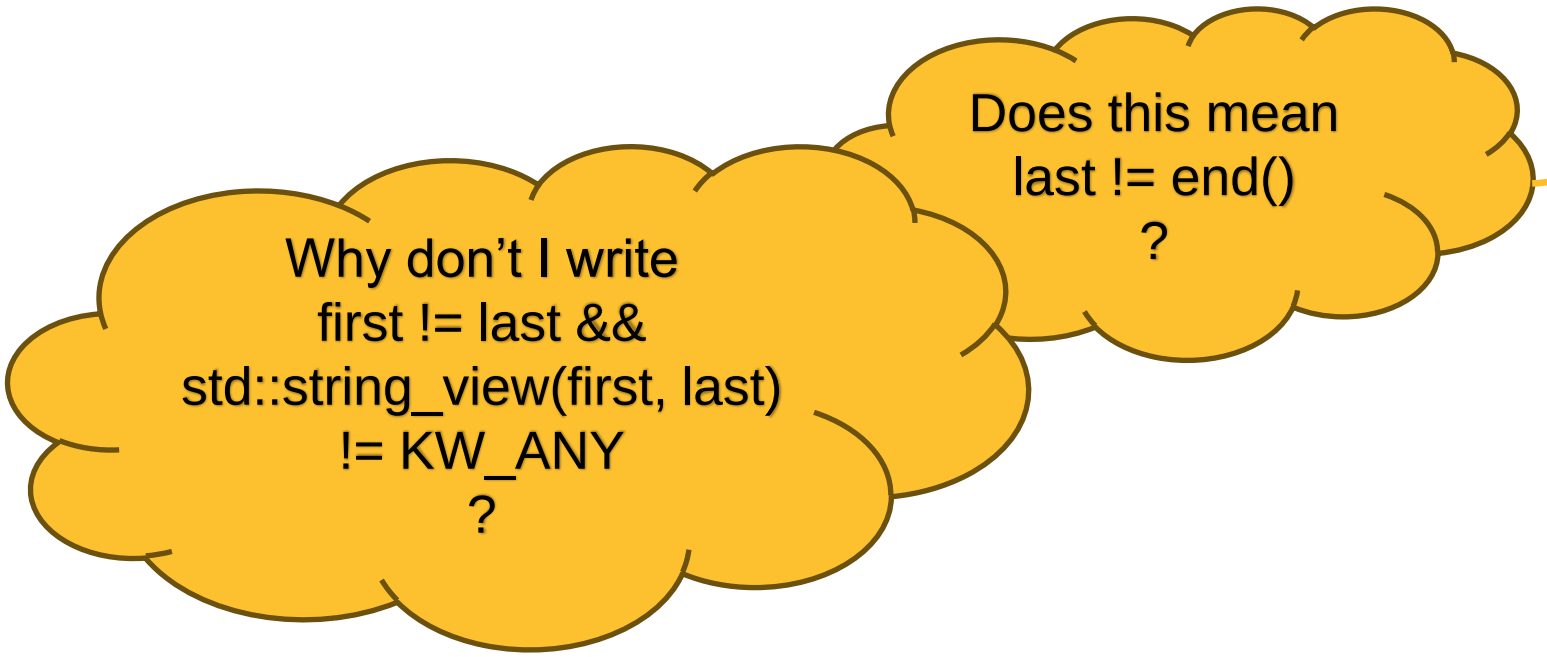
- if constexpr$_{opt}$ (*init-statement*$_{opt}$ *condition*) *statement* else *statement*
- switch (*init-statement*$_{opt}$ *condition*) *statement*
- while (*condition*) *statement*
- for (*init-statement condition*$_{opt}$; *expression*$_{opt}$) *statement*

Examples

Example 1

Symantec

```
if (auto [first, last] = parse(begin(), end()); first != last)
{
    // interpret [first, last) into a value
}
```



Symantec

```
struct parse_window
{
    char const *first, *last;
};
```

```
parse_window parse(char const*, char const*);
```

Adding operator bool

Symantec

```
struct parse_window
{
    char const *first, *last;
    explicit operator bool() const noexcept
    {
        return first != last;
    }
};

parse_window parse(char const*, char const*);
```

Example 1

Symantec

```
if (auto [first, last] = parse(begin(), end()); first != last)
{
    // interpret [first, last) into a value
}
```

Example 1 simplified with this proposal

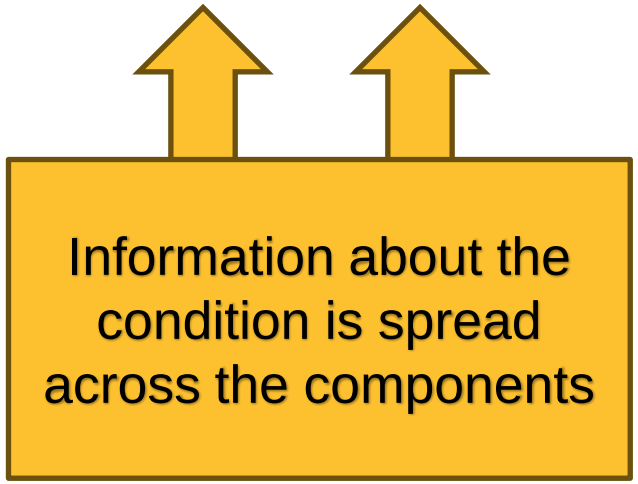
Symantec

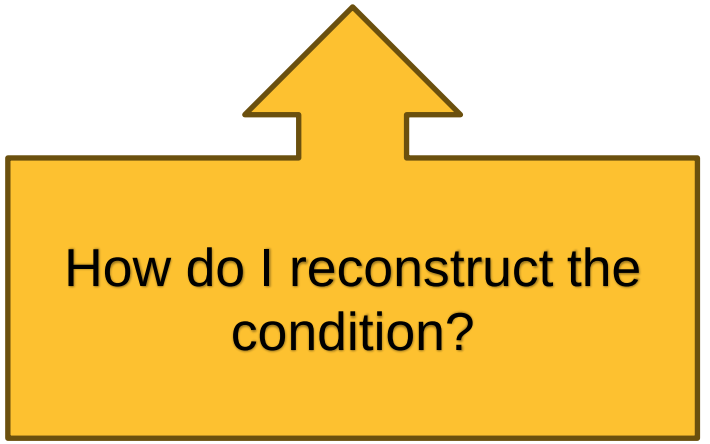
```
if (auto [first, last] = parse(begin(), end()))
{
    // interpret [first, last) into a value
}
```

Example 1

Symantec

```
if (auto [first, last] = parse(begin(), end()); first != last)
{
```

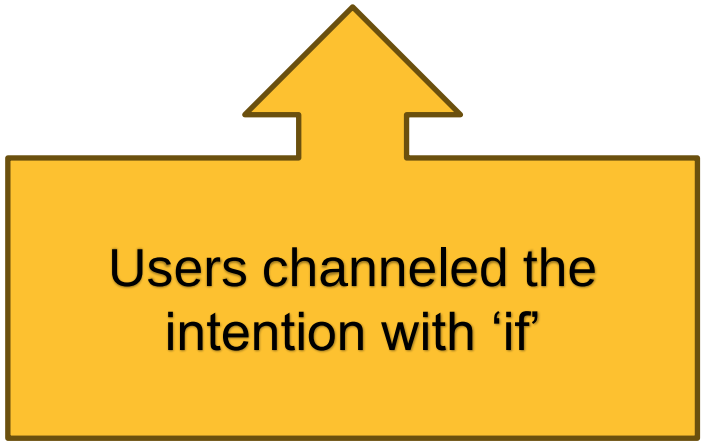




Example 1 simplified with this proposal

Symantec

```
if (auto [first, last] = parse(begin(), end()))
{
```



Example 2...?

Symantec

```
if (auto result = std::to_chars(p, last, 42))
{
    // okay to proceed
}
```

Why don’t you…?

Symantec

```
if (auto result = std::to_chars(p, last, 42))
{
    auto [ptr, ec] = result;
    // okay to proceed
}
```

Example 2

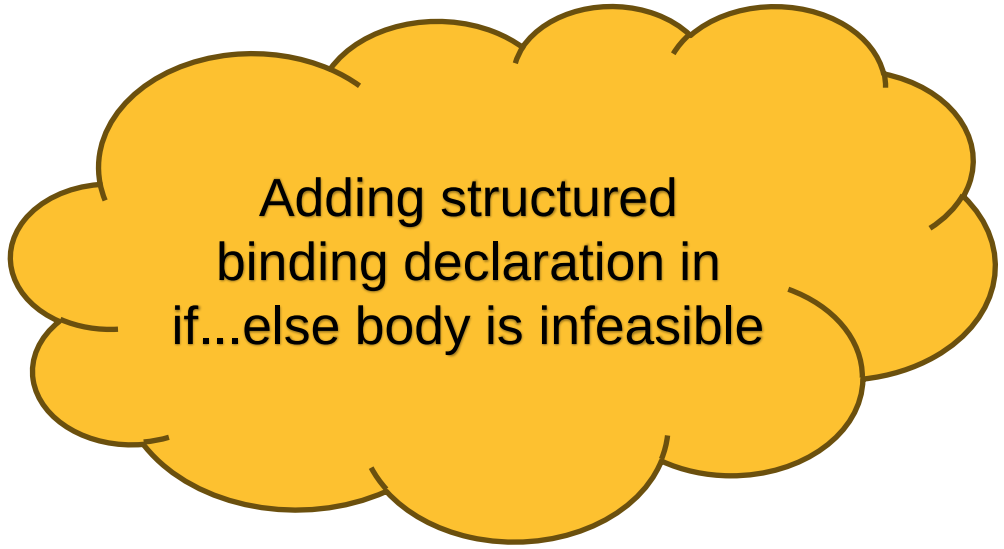
Symantec

```
if (auto result = std::to_chars(p, last, 42))
{
    // okay to proceed
}
else
{
    // handle errors
}
```

Example 2

Symantec

```
if (auto result = std::to_chars(p, last, 42))
{
    auto [ptr, _] = result;
    // okay to proceed
}
else
{
    auto [ptr, ec] = result;
    // handle errors
}
```



Why don’t you…?

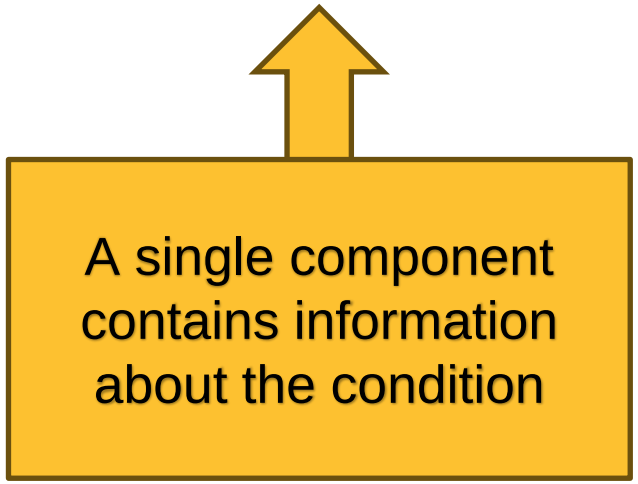
Symantec

```
if (auto [ptr, ec] = std::to_chars(p, last, 42); ec != std::errc{})
{
    // okay to proceed
}
else
{
    // handle errors
}
```

Why don’t you…?

Symantec

```
if (auto [ptr, ec] = std::to_chars(p, last, 42); ec != std::errc{})
{
```



C++26 <charconv> (P2497R0)

Symantec

operator bool

```
constexpr explicit operator bool() const noexcept;    (since C++26)
```

Checks whether the conversion is successful. Returns `ec == std::errc{}`.

Example

```
#include <array>
#include <charconv>
#include <iostream>
#include <string_view>
#include <system_error>

void show_to_chars(auto... format_args)
{
    std::array<char, 10> str;

#if __cpp_lib_to_chars >= 202306L
    // use C++26 operator bool() for error checking
    if (auto res = std::to_chars(str.data(), str.data() + str.size(), format_args...))
        std::cout << std::string_view(str.data(), res.ptr) << '\n';
    else
        std::cout << std::make_error_code(res.ec).message() << '\n';
#else
```

Between this

Symantec

```
if (auto [ptr, ec] = std::to_chars(p, last, 42); ec != std::errc{})
{
    // okay to proceed
}
else
{
    // handle errors
}
```

...and this

Symantec

```
if (auto result = std::to_chars(p, last, 42))
{
    auto [ptr, _] = result;
    // okay to proceed
}
else
{
    auto [ptr, ec] = result;
    // handle errors
}
```

I prefer

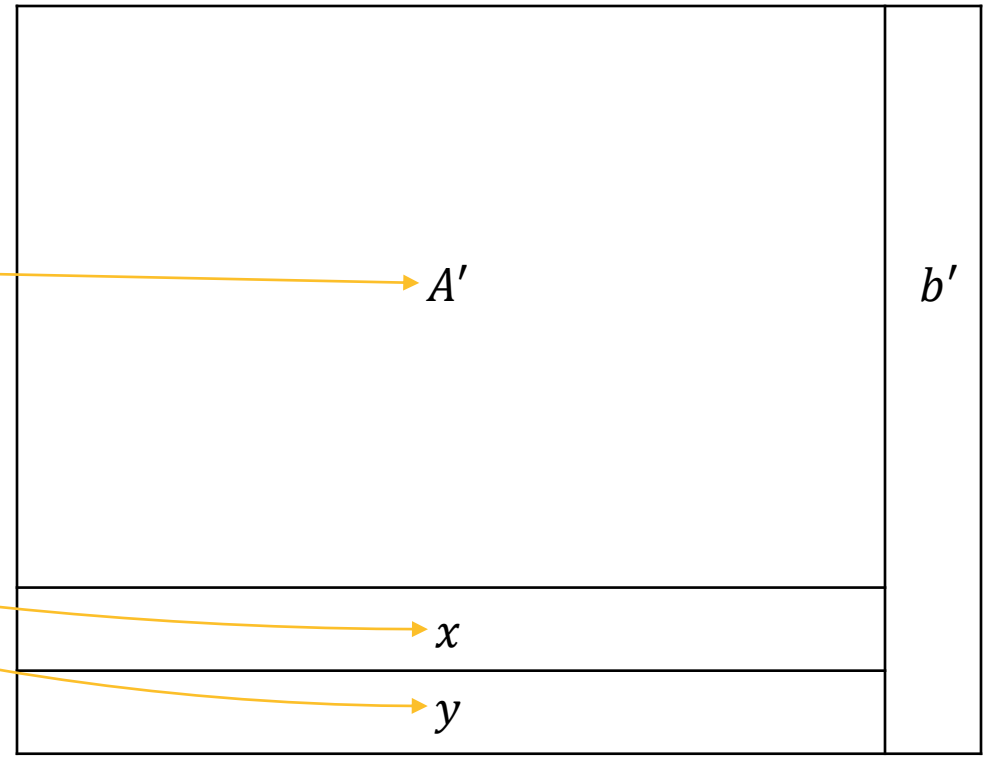
Symantec

```
if (auto [ptr, ec] = std::to_chars(p, last, 42))
{
    // okay to proceed
}
else
{
    // handle errors
}
```

Example 3

iterative solver

```
struct tableau
{
    matrix Ap;
    vector bp, x, y;
};
```



Example 3

iterative solver

```
struct tableau
{
    matrix Ap;
    vector bp, x, y;
};
```

```
class Solver
{
  public:
    tableau solve();
    bool is_optimal(vector const&);
};
```

Example 3

Symantec

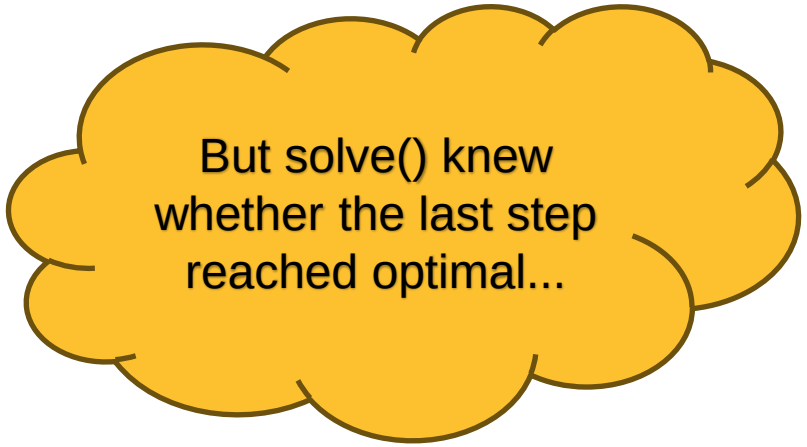
```
for (;;)
{
    // ...
    auto [Ap, bp, x, y] = solve();

    // stop iteration
}
```

Example 3

```
for (;;)
{
    // ...
    auto [Ap, bp, x, y] = solve();

    if (is_optimal(x))
        break;
}
```



Example 3 improved with the proposal

Symantec

```
for (;;)
{
    // ...
    if (auto [Ap, bp, x, y] = solve())
        break;
}
```

Saves the cost of reconstructing the information from the components

But solve() knew whether the last step reached optimal...

Migrate from an aggregate

Symantec

```
template<std::size_t i>
decltype(auto) get(this auto&& self)
{
    if constexpr (i == 0)
        return std::forward_like<decltype(self)>(self.Ap);
    else if constexpr (i == 1)
        return std::forward_like<decltype(self)>(self.bp);
    else if constexpr (i == 2)
        return std::forward_like<decltype(self)>(self.x);
    else if constexpr (i == 3)
        return std::forward_like<decltype(self)>(self.y);
}
```

Example 4

Symantec

```
if (auto [all, city, state, zip] =
        ctre::match<"(\\w+), (\\w+) (\\d+)">(s); all)
{
    return location{ city, state, zip };
}
```

Example 4

```
if (auto [all, city, state, zip] =
        ctre::match<"(\\w+), (\\w+) (\\d+)">(s); all)
{
```

All components but one have similar roles

Example 4 designed differently with the proposal


```
if (auto [city, state, zip] = ctre2::match<"(\\w+), (\\w+) (\\d+)">(s))
{
    return location{ city, state, zip };
}
```

Technical Details

Can decomposition be conditional?

Symantec

```
auto consume_int() -> std::optional<int>;


    if (auto [i] = consume_int()) // let e be the underlying object
    {
        // i = *e
    }
    else
    {
        // *e is not evaluated
    }
```

Can decomposition be conditional?

Symantec

- No

To destructure	Abusing condition
`optional<T>`	`[x]`

Can decomposition be conditional?

Symantec

- No

To destructure	Abusing condition
`optional<T>`	`[x]`
`optional<tuple<T, U>>`	`[x, y]` ✕

Can decomposition be conditional?

Symantec

- No

To destructure	Abusing condition	Pattern matching
`optional<T>`	`[x]`	`let ?x`
`optional<tuple<T, U>>`	`[x, y]` ✕	`let ?[x, y]`

Pattern matching's solution (P2688R1)

Symantec

```
if (consume_int() match let ?i)
{
    // use(i)
}
else
{
    // has no value
}
```

When does the Boolean test happen?

Symantec

- After initializing the bindings
- So that

```
if (auto [a, b, c] = fn())
{
    statements;
}
```

is equivalent to

```
if (auto [a, b, c] = fn(); underlying-object)
{
    statements;
}
```

Can I decompose an array in a condition?

- No.

Implementation Experience

Implemented in Clang

Symantec

C++

```
9
10  struct format_status
11  {
12      format_errc ec;
13      char *bp;
14
15      explicit operator bool() const noexcept
16      {
17          return ec == format_errc::no_error;
18      }
19  };
20
21  format_status
22
23  int main()
24  {
25      if (auto [ok, ptr] = readint())
26      {
27          printf("stopped at %p\n", ptr);
28      }
29  }
```

warning: ISO C++17 does not permit structured binding declaration in a condition [-Wbinding-in-condition] x86-64 clang (trunk) #1

View Problem (Alt+F8) No quick fixes available

x86-64 clang (trunk) -Wall -std=c++17

```
1   main: # @main
2     push rax
3     call readint()@PLT
4     test eax, eax
5     jne .LBB0_2
6     lea rdi, [rip + .L.str]
7     mov rsi, rdx
8     xor eax, eax
9     call printf@PLT
10  .LBB0_2:
11    xor eax, eax
12    pop rcx
13    ret
14  .L.str:
15    .asciz "stopped at %p\n"
```

Thank you