F.21

Giuseppe D'Angelo giuseppe.dangelo@kdab.com F.21: To return multiple "out" values, prefer returning a tuple or struct.

Reason A return value is self-documenting as an "output-only" value. Note that C++ does have multiple return values, by convention of using a tuple (including pair), possibly with the extra convenience of tie at the call site. [...]

With C++17 we should be able to use "structured bindings" to declare and initialize the multiple variables:

```
if (auto [ iter, success ] = my_set.insert("Hello");
    success) { do_something_with(iter); }
```

F.21: To return multiple "out" values, return a struct.[†]

[†] ... unless you need to return a variadic struct, for which we don't really have a good syntax for you to define one, so use tuple.

Thank you

Bonus slides

What's wrong with tuples?

- Nothing
- (no, seriously: nothing)
- However, tuples: hammer = multiple return arguments: nail

Unwarranted tuples weaken the type system

```
tuple<float, float> calculateSize();
tuple<float, float> calculateComplexNumber();
static_assert(is_same_v<decltype(calculateSize()),</pre>
                         decltype(calculateComplexNumber())>);
auto c1 = calculateComplexNumber(),
     c2 = calculateComplexNumber();
auto s = calculateSize();
if (c1 == c2) f();
if (c1 == s) q();
if (c1 < c2) h();
```

What about structs?

- Structs enforce proper type safety
- However they are slightly more tedious to use

Structs are more tedious to use: reason #1

Cannot declare anonymous structs as return parameter

```
struct { int x, y; bool success; } foo() { ... }
error: new types may not be defined in a return type
```

Workaround (not general; YMMV):

```
auto foo() {
    struct { int x, y; bool success; } v;
    ...
    return v;
}
```

• Otherwise, must **name** the struct

Structs are more tedious to use: reason #2

Must find names for the data members:

```
struct MyType { int ???, ???; bool ???; };
MyType foo();
```

 Not an issue! When using structured binding to decompose the return value, we already have the names to give:

```
auto [lastValue, offset, success] = foo();
// ^^^^^^ ^^ ^^^^ ^^^ ^^^^ // here you have the names of your data members
```

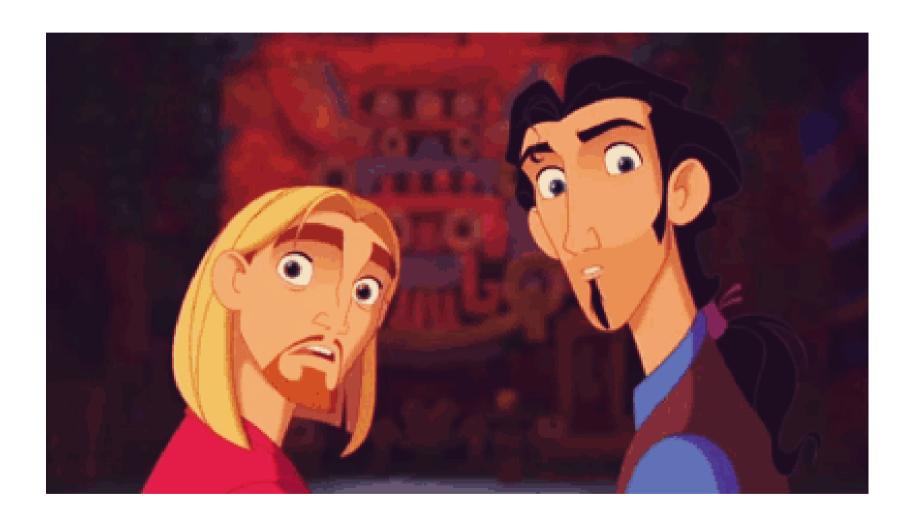
Thank you

Bonus bonus slides

C++17

Look, ma!

A struct as return value, and an out parameter!



Thank you