Modifiers & Limitations

To further specify the nature of our structured bindings, we can use modifier types.

WE'LL COVER THE FOLLOWING

Several modifiers can be used with structured bindings.

`const` modifiers:

```
const auto [a, b, c, ...] = expression;
```

References:

```
auto& [a, b, c, ...] = expression;
auto&& [a, b, c, ...] = expression;
```

For example:

```
#include <iostream>
using namespace std;

int main() {
  std::pair a(0, 1.0f);
  auto& [x, y] = a;
  x = 10; // write access
  std::cout << a.first;// a.first is now 10
}
```

In the example, `x` binds to the element in the generated object, that is a reference to `a`.

Now it’s also quite easy to get a reference to a tuple member:

```
auto& [ refA, refB, refC, refD ] = myTuple;
```

Attributes

You can also add `[[attribute]]` to structured bindings:

```
[[maybe_unused]] auto& [a, b, c, ...] = expression;
```

> **Structured Bindings or Decomposition Declaration?** You might have seen another name used for this feature: “decomposition declaration”. During the standardisation process, both names were considered, but “structured bindings" was selected.

Structured Binding Limitations

There are several limitations related to structured bindings. They cannot be declared as `static` or `constexpr` and also they cannot be used in lambda captures. For example:

```
constexpr auto [x, y] = std::pair(0, 0);
// generates:
error: structured binding declaration cannot be 'constexpr'
```

It was also unclear about the linkage of the bindings. Compilers had a free choice to implement it (and thus some of them might allow capturing a structured binding in lambdas). Fortunately, those limitations might be removed due to C++20 proposal (already accepted): P1091: Extending structured bindings to be more like variable declarations.

Until now, we’ve dealt only with tuples. However, structured binding has a broader horizon than just tuples. We’ll talk more about this in the next lesson broader horizon than just tuples. We'll talk more about this in the next lesson.