- Example

Here, we'll see an example of how decltype can be used to automatically deduce the return type of a function.

The `add` function

```
#include <iostream>
#include <typeinfo>

template<typename T1, typename T2>
auto add(T1 first, T2 second) -> decltype(first + second){
    return first + second;
}

int main(){

  std::cout << std::endl;

  std::cout << "add(1, 1)= " << add(1, 1) << std::endl;
  std::cout << "typeid(add(1, 1)).name()= " << typeid(add(1, 1)).name() << std::endl;

  std::cout << std::endl;

  std::cout << "add(1, 2.1)= " << add(1, 2.1) << std::endl;
  std::cout << "typeid(add(1, 2.1)).name()= " << typeid(add(1, 2.1)).name() << std::endl;

  std::cout << std::endl;

  std::cout << "add(1000LL, 5)= " << add(1000LL, 5) << std::endl;
  std::cout << "typeid(add(1000LL, 5)).name()= " << typeid(add(1000LL, 5)).name() << std::end

  std::cout << std::endl;

}
```

Explanation

The example implements the `add` function, which takes two arguments and returns their sum. The return type of the function is deduced by the compiler by applying the `decltype` operator on the sum of the arguments.

The expression, `typeid(add(1, 2.1)).name()`, such as in line 19, returns a string representation of the return type.

In the next lesson, we'll be presented with an exercise related to automatic return type deduction.