- Example

Let's have a look at an example of constexpr if.

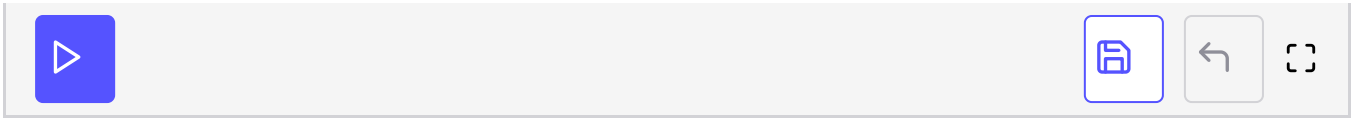
WE'LL COVER THE FOLLOWING

- Example: `constexpr if`
  - Explanation

Example: `constexpr if`

```
// constexprIf.cpp

#include <iostream>
#include <type_traits>

// SFINAE

template <typename T, std::enable_if_t<std::is_arithmetic<T>{}>* = nullptr>
auto get_value_SFINAE(T) {
    std::cout << "get_Value_SFINAE(5)" << std::endl;
}

template <typename T, std::enable_if_t<!std::is_arithmetic<T>{}>* = nullptr>
auto get_value_SFINAE(T) {
    std::cout << "get_Value_SFINAE(five)" << std::endl;

}

// Tag dispatch

template <typename T>
auto get_value_TAG_DISPATCH(T, std::true_type) {
     std::cout << "get_Value_TAG_DISPATCH(5)" << std::endl;
}

template <typename T>
auto get_value_TAG_DISPATCH(T, std::false_type) {
    std::cout << "get_Value_TAG_DISPATCH(five)" << std::endl;
}

template <typename T>
auto get_value_TAG_DISPATCH(T t) {
    return get_value_TAG_DISPATCH(t, std::is_arithmetic<T>{});
}
```

```
// constexpr if

template <typename T>

auto get_value_CONSTEXPR_IF(T) {
     if constexpr (std::is_arithmetic_v<T>) {
         std::cout << "get_Value_CONSTEXPR_IF(5)" << std::endl;
     }
     else {
         std::cout << "get_Value_CONSTEXPR_IF(five)" << std::endl;
     }
}

int main(){

    std::cout << std::endl;

    get_value_SFINAE(5);
    get_value_SFINAE("five");

    std::cout << std::endl;

    get_value_TAG_DISPATCH(5);
    get_value_TAG_DISPATCH("five");

    std::cout << std::endl;

    get_value_CONSTEXPR_IF(5);
    get_value_CONSTEXPR_IF("five");

    std::cout << std::endl;

}
```



Explanation

We have created `get_value` functions which use SFINAE, TAG_DISPATCH, and CONSTEXPR_IF. These functions use the `std::is_arithmetic` function from the type-traits library. `std::is_arithmetic` returns `true` only if `std::is_integral` or `std::is_floating_point` is true for a given type. All the calls from `main` verify that the passed argument falls in their required category.

- `get_value_SFINAE` uses the function `std::enable_if` from the type-traits library. `std::enable_if` is only true, if the given type is arithmetic.
- `get_value_TAG_DISPATCH` in line 33 dispatches on the result of `std::is_arithmetic`.
- `get_value_CONSTEXR_IF` creates another branch of the `if`-statement depending on the result of the expression `std::is_arithmetic_v<T>` which is a shorthand for `std::is_arithmetic<T>::value`.

We have learned about the techniques which we used in templates. In the next chapter, we'll see different design techniques used in C++ Templates.