Definition of a Concepts

1. Combination of concepts and compile time predicates

   - Combinations
     - Conjunction (`&&`)
     - Disjunction (`||`)
     - Negation (`!`)

   - Compile time predicate
     - Functions that return a boolean at compile time.
     - The type-traits library provides many compile-time predicates.

Definition of a Concept

```
template <typename T>
concept Integral = std::is_integral<T>::value;

template <typename T>
concept SignedIntegral = Integral<T> && std::is_signed<T>::value;

template <typename T>
concept UnsignedIntegral = Integral<T> && !SignedIntegral<T>;
```

signedUnsignedIntegrals.cpp

Definition of a Concept

2. Requires Expression

    1. Simple requirements
    2. Type requirements
    3. Compound requirements
    4. Nested requirements

Definition of a Concept

- Simple requirements

```
template<typename T>
concept Addable = requires (T a, T b) {
    a + b;
};
```

`T` fulfills the concept `Addable`:

- The addition of two values of type `T` is valid.

Definition of a Concept

- Type requirements

```
template<typename T>
concept TypeRequirement = requires {
    typename T::value_type;
    typename Other<T>;
};
```

`T` fulfills the concept `TypeRequirement`:

- `T` has the member `value_type`.
- The class template `Other` can be instantiated with `T`.

typeRequirements.cpp

Definition of a Concept

- Compound requirements

```
template<typename T>
concept Equal = requires(T a, T b) {
    { a == b } -> std::convertible_to<bool>;
    { a != b } -> std::convertible_to<bool>;
};
```

`T` fulfills the concept `Equal`:

- `T` supports equality and inequaliy.
- Equal and non-equal return a in a `bool` convertible value.

conceptsDefinitionEqual.cpp

Definition of a Concept

- Nested requirements

```
template <typename T>
concept Integral = std::is_integral<T>::value;

template <typename T>
concept SignedIntegral = Integral<T> && std::is_signed<T>::value;

template <typename T>
concept UnsignedIntegral = Integral<T> &&
requires(T) {
    requires !SignedIntegral<T>;
};
```

`T` fulfills the `UnsignedIntegral` concept if it supports the `Integral` concept and not the `SignedIntegral` concept.