CPPNOW 2019

TEMPLATE

<TEMPLATE...>

Kris Jusiak, Quantlab Financial

KRIS@JUSIAK.NET | @KRISJUSIAK | LINKEDIN.COM/IN/KRIS-JUSIAK

BACKGROUND

BACKGROUND

```
template<class>                         // type          (C++98)
template<typename>                      // type          (C++98)
```

BACKGROUND

```
template<class>                    // type        (C++98)
template<typename>                 // type        (C++98)
```

```
template<int>                      // value       (C++98)
```

BACKGROUND

```
template<class>                    // type          (C++98)
template<typename>                 // type          (C++98)

template<int>                      // value         (C++98)

template<auto>                     // non-type      (C++17)
```

BACKGROUND

```
template<class>                          // type          (C++98)
template<typename>                       // type          (C++98)

template<int>                            // value         (C++98)

template<auto>                           // non-type      (C++17)

template<std::basic_fixed_string>        // class type    (C++20)
```

BACKGROUND

```
template<class>                          // type          (C++98)
template<typename>                       // type          (C++98)

template<int>                            // value         (C++98)

template<auto>                           // non-type      (C++17)

template<std::basic_fixed_string>        // class type    (C++20)

template<Concept>                        // concept       (C++20)
```

PROBLEM 1

PROBLEM 1

```
template<typename...> struct type_list{};
```

PROBLEM 1

```
template<typename...> struct type_list{};
```

```
type_list<>;              // 👍 Okay
type_list<int, class A>; // 👍 Okay
```

PROBLEM 1

```
template<typename...> struct type_list{};
```

```
type_list<>;                // 👍 Okay
type_list<int, class A>; // 👍 Okay
```

```
type_list<int, 42>;         // 👎 Oops
```

PROBLEM 1

```
template<typename...> struct type_list{};
```

```
type_list<>;                // 👍 Okay
type_list<int, class A>; // 👍 Okay
```

```
type_list<int, 42>;         // 👎 Oops
```

Error: template argument for template type parameter must be a type

PROBLEM 2

PROBLEM 2

```
template<template<typename...> typename T>
constexpr auto to();
```

PROBLEM 2

```
template<template<typename...> typename T>
constexpr auto to();
```

```
/**
 * template<class T, class Allocator> class vector;
 */
to<std::vector>(); // 👍 Okay
```

PROBLEM 2

```
template<template<typename...> typename T>
constexpr auto to();
```

```
/**
 * template<class T, class Allocator> class vector;
 */
to<std::vector>(); // 👍 Okay
```

```
/**
 * template<class T, std::size_t N> struct array;
 */
to<std::array>();  // 👎 Oops
```

PROBLEM 2

```
template<template<typename...> typename T>
constexpr auto to();
```

```
/**
 * template<class T, class Allocator> class vector;
 */
to<std::vector>(); // 👍 Okay
```

```
/**
 * template<class T, std::size_t N> struct array;
 */
to<std::array>();  // 👎 Oops
```

Error: candidate template ignored: invalid explicitly-specified argument for template parameter 'Ts'

PROBLEM 3

PROBLEM 3

```
template<typename> concept Concept = true;
```

PROBLEM 3

```
template<typename> concept Concept = true;
```

```
template<Concept T>  // requires Concept<T>
constexpr auto to(); // 👍 Okay
```

PROBLEM 3

PROBLEM 3

```
template<template<typename...> typename T> constexpr auto to();
```

PROBLEM 3

```
template<template<typename...> typename T> constexpr auto to();
```

```
to<Concept>(); // 👎 Oops
```

PROBLEM 3

```
template<template<typename...> typename T> constexpr auto to();
```

```
to<Concept>(); // 👎 Oops
```

Error: use of concept 'Concept' requires template arguments

PROBLEM 3

PROBLEM 3

```
/**
 * With Class Types in Non-Type Template Parameters (C++20)
 */
template<class T> constexpr auto to();
```

PROBLEM 3

```
/**
 * With Class Types in Non-Type Template Parameters (C++20)
 */
template<class T> constexpr auto to();
```

```
to<Concept>(); // 👎 Oops
```

PROBLEM 3

```
/**
 * With Class Types in Non-Type Template Parameters (C++20)
 */
template<class T> constexpr auto to();
```

```
to<Concept>(); // 👎 Oops
```

Error: invalid use of 'Concept' in template argument

SOLUTION?

SOLUTION?

```
template<template...> class template_list {};
```

SOLUTION?

```
template<template...> class template_list {};
```

```
template_list<>;                    // 👍 Okay
```

SOLUTION?

```
template<template...> class template_list {};

template_list<>;                  // 👍 Okay

template_list<int, class A>; // 👍 Okay
```

SOLUTION?

```
template<template...> class template_list {};

template_list<>;                 // 👍 Okay

template_list<int, class A>; // 👍 Okay

template_list<int, 42>;          // 👍 Okay
```

SOLUTION?

SOLUTION?

```
template<template<template...> typename T>
constexpr auto to();
```

SOLUTION?

```
template<template<template...> typename T>
constexpr auto to();
```

```
to<std::vector>(); // 👍 Okay
```

SOLUTION?

```
template<template<template...> typename T>
constexpr auto to();
```

```
to<std::vector>(); // 👍 Okay
```

```
to<std::array>();  // 👍 Okay
```

SOLUTION?

SOLUTION?

```
template<template<template...> typename T>
constexpr auto to();
```

SOLUTION?

```
template<template<template...> typename T>
constexpr auto to();
```

```
to<Concept>(); //  👍 Okay
```

LET'S TEMPLATE TEMPLATE ALL THE THINGS?

KRIS@JUSIAK.NET | @KRISJUSIAK | LINKEDIN.COM/IN/KRIS-JUSIAK