CAMP

Tom Scogland

Apr 30, 2020

CONTENTS

1 Class Hierarchy **3**

2 File Hierarchy **5**

3 Full API **7**

3.1 Namespaces 7
3.2 Classes and Structs 18
3.3 Enums 107
3.4 Functions 108
3.5 Variables 114
3.6 Defines 114
3.7 Typedefs 116

Index **129**

CAMP

CHAPTER

ONE

CLASS HIERARCHY

CHAPTER

TWO

FILE HIERARCHY

CHAPTER

THREE

FULL API

3.1 Namespaces

3.1.1 Namespace camp

Page Contents

- *Namespaces*
- *Classes*
- *Functions*
- *Typedefs*
- *Variables*

Namespaces

- *Namespace camp::concepts*
- *Namespace camp::resources*
- *Namespace camp::type*
- *Namespace camp::type_traits*

Classes

- *Template Struct accumulate*
- *Template Struct accumulate< Op, Initial, list< Elements... > >*
- *Template Struct append*
- *Template Struct append< list< Elements... >, T >*
- *Template Struct apply_l*
- *Template Struct apply_l< Lambda, list< Args... > >*
- *Template Struct arg*
- *Template Struct as_list_s*

- *Template Struct as_list_s< tagged_tuple< camp::list< Tags... >, Args... > >*
- *Template Struct at*
- *Template Struct at< T, num< Val > >*
- *Template Struct at_key_s*
- *Template Struct bind*
- *Template Struct bind_front*
- *Template Struct extend*
- *Template Struct extend< list< Elements... >, list< NewElements... > >*
- *Template Struct filter*
- *Template Struct filter< Op, list< Elements... > >*
- *Template Struct find_if*
- *Template Struct find_if< Cond, list< Elements... > >*
- *Template Struct flatten*
- *Template Struct flatten< list< Elements... > >*
- *Template Struct idx_seq_from*
- *Template Struct idx_seq_from< int_seq< T, Args... > >*
- *Template Struct idx_seq_from< T< Args... > >*
- *Template Struct if_cs*
- *Template Struct if_cs< false, Then, Else >*
- *Template Struct if_s*
- *Template Struct if_s< nil, Then, Else >*
- *Template Struct index_of*
- *Template Struct index_of< T, list< Elements... > >*
- *Template Struct int_seq*
- *Template Struct integral_constant*
- *Template Struct invoke_l*
- *Template Struct is_same_s*
- *Template Struct is_same_s< T, T >*
- *Template Struct is_value_s*
- *Template Struct join*
- *Template Struct join< Seq1 >*
- *Template Struct join< Seq1, Seq2, Rest... >*
- *Template Struct join<>*
- *Template Struct lambda*
- *Template Struct list*
- *Template Struct make_idx_seq*

- *Template Struct make_int_seq*
- *Template Struct not_*
- *Template Struct prepend*
- *Template Struct prepend< list< Elements... >, T >*
- *Template Struct seq_at*
- *Template Struct seq_at< 0, camp::int_seq< T, Idx0, IdxRest... > >*
- *Template Struct seq_at< N, camp::int_seq< T, Idx0, IdxRest... > >*
- *Template Struct size*
- *Template Struct size< int_seq< T, Args... > >*
- *Template Struct size< list< Args... > >*
- *Template Struct tagged_tuple::is_pack_this_tuple*
- *Template Struct tagged_tuple::is_pack_this_tuple< That >*
- *Template Struct transform*
- *Template Struct transform< Op, list< Elements... > >*
- *Template Struct tuple*
- *Template Struct tuple::is_pack_this_tuple*
- *Template Struct tuple::is_pack_this_tuple< That >*
- *Template Struct tuple_element*
- *Template Struct tuple_size*
- *Template Struct tuple_size< tagged_tuple< L, Args... > & >*
- *Template Struct tuple_size< tagged_tuple< L, Args... > >*
- *Template Struct tuple_size< tuple< Args... > & >*
- *Template Struct tuple_size< tuple< Args... > >*
- *Template Struct value*
- *Template Class tagged_tuple*
- *Template Class tuple<>*

Functions

- *Function camp::CAMP_MAKE_L(accumulate)*
- *Function camp::CAMP_MAKE_L(filter)*
- *Function camp::CAMP_MAKE_L(bind_front)*
- *Function camp::CAMP_MAKE_L(find_if)*
- *Template Function camp::cval*
- *Template Function camp::declptr*
- *Template Function camp::forward(type::ref::rem<T>&)*
- *Template Function camp::forward(type::ref::rem<T>&&)*

- *Template Function camp::forward_as_tuple*
- *Template Function camp::get(const Tuple&)*
- *Template Function camp::get(Tuple&)*
- *Template Function camp::make_tagged_tuple*
- *Template Function camp::make_tuple*
- *Template Function camp::make_unique*
- *Template Function camp::move*
- *Template Function camp::safe_swap(T&, T&)*
- *Template Function camp::safe_swap(T&, T&)*
- *Template Function camp::sink*
- *Template Function camp::tie*
- *Template Function camp::tuple_cat_pair(tuple<Lelem...> const&, camp::idx_seq<Lidx...>, tuple<Relem...> const&, camp::idx_seq<Ridx...>)*
- *Template Function camp::tuple_cat_pair(L const&, R const&)*
- *Template Function camp::val*

Typedefs

- *Typedef camp::_1*
- *Typedef camp::_2*
- *Typedef camp::_3*
- *Typedef camp::_4*
- *Typedef camp::_5*
- *Typedef camp::_6*
- *Typedef camp::_7*
- *Typedef camp::_8*
- *Typedef camp::_9*
- *Typedef camp::as_list*
- *Typedef camp::at_key*
- *Typedef camp::at_t*
- *Typedef camp::at_v*
- *Typedef camp::cartesian_product*
- *Typedef camp::decay*
- *Typedef camp::diff_between*
- *Typedef camp::diff_from*
- *Typedef camp::eval*
- *Typedef camp::false_type*

- *Typedef camp::first*
- *Typedef camp::idx_seq*
- *Typedef camp::idx_seq_for_t*
- *Typedef camp::idx_seq_from_t*
- *Typedef camp::idx_t*
- *Typedef camp::if_*
- *Typedef camp::if_c*
- *Typedef camp::is_same*
- *Typedef camp::is_same_t*
- *Typedef camp::is_tuple*
- *Typedef camp::is_value*
- *Typedef camp::iterator_from*
- *Typedef camp::make_idx_seq_t*
- *Typedef camp::make_int_seq_t*
- *Typedef camp::nil*
- *Typedef camp::nullptr_t*
- *Typedef camp::num*
- *Typedef camp::plain*
- *Typedef camp::second*
- *Typedef camp::t*
- *Typedef camp::true_type*
- *Typedef camp::tuple_ebt_t*
- *Typedef camp::tuple_element_t*

Variables

- *Variable camp::r*

3.1.2 Namespace camp::concepts

Page Contents

- *Namespaces*
- *Classes*
- *Functions*
- *Typedefs*

Namespaces

- *Namespace camp::concepts::metalib*

Classes

- *Template Struct all_of*
- *Template Struct any_of*
- *Template Struct Arithmetic*
- *Template Struct BidirectionalIterator*
- *Template Struct BidirectionalRange*
- *Template Struct Comparable*
- *Template Struct ComparableTo*
- *Template Struct EqualityComparable*
- *Template Struct FloatingPoint*
- *Template Struct ForwardIterator*
- *Template Struct ForwardRange*
- *Template Struct HasBeginEnd*
- *Template Struct Integral*
- *Template Struct Iterator*
- *Template Struct LessEqualComparable*
- *Template Struct LessThanComparable*
- *Template Struct none_of*
- *Template Struct RandomAccessIterator*
- *Template Struct RandomAccessRange*
- *Template Struct Range*
- *Template Struct requires_*
- *Template Struct Signed*
- *Template Struct Swappable*
- *Template Struct Unsigned*

Functions

- *Template Function camp::concepts::convertible_to*
- *Template Function camp::concepts::has_type*
- *Template Function camp::concepts::is*
- *Template Function camp::concepts::is_not*

Typedefs

- *Typedef camp::concepts::enable_if*
- *Typedef camp::concepts::negate*

3.1.3 Namespace camp::concepts::metalib

Page Contents

- *Classes*

Classes

- *Template Struct all_of*
- *Template Struct all_of_t*
- *Template Struct any_of*
- *Template Struct any_of_t*
- *Template Struct negate_t*
- *Template Struct none_of*
- *Template Struct none_of_t*

3.1.4 Namespace camp::resources

Page Contents

- *Namespaces*

Namespaces

- *Namespace camp::resources::v1*

3.1.5 Namespace camp::resources::v1

Page Contents

- *Classes*
- *Enums*

Classes

- *Template Struct resource_from_platform*
- *Template Struct resource_from_platform< Platform::host >*
- *Class Event*
- *Class Event::EventInterface*
- *Template Class Event::EventModel*
- *Class Host*
- *Class HostEvent*
- *Class Resource*
- *Class Resource::ContextInterface*
- *Template Class Resource::ContextModel*

Enums

- *Enum Platform*

3.1.6 Namespace camp::type

Page Contents

- *Namespaces*

Namespaces

- *Namespace camp::type::c*
- *Namespace camp::type::cv*
- *Namespace camp::type::ref*
- *Namespace camp::type::rvref*
- *Namespace camp::type::v*

3.1.7 Namespace camp::type::c

Page Contents

- *Classes*
- *Typedefs*

Classes

- *Template Struct rem_s*
- *Template Struct rem_s< const T >*

Typedefs

- *Typedef camp::type::c::add*
- *Typedef camp::type::c::rem*

3.1.8 Namespace camp::type::cv

Page Contents

- *Classes*
- *Typedefs*

Classes

- *Template Struct rem_s*
- *Template Struct rem_s< const T >*
- *Template Struct rem_s< const volatile T >*
- *Template Struct rem_s< volatile T >*

Typedefs

- *Typedef camp::type::cv::add*
- *Typedef camp::type::cv::rem*

3.1.9 Namespace camp::type::ref

Page Contents

- *Classes*
- *Typedefs*

Classes

- *Template Struct rem_s*
- *Template Struct rem_s< T & >*
- *Template Struct rem_s< T && >*

Typedefs

- *Typedef camp::type::ref::add*
- *Typedef camp::type::ref::rem*

3.1.10 Namespace camp::type::rvref

Page Contents

- *Typedefs*

Typedefs

- *Typedef camp::type::rvref::add*

3.1.11 Namespace camp::type::v

Page Contents

- *Classes*
- *Typedefs*

Classes

- *Template Struct rem_s*
- *Template Struct rem_s< volatile T >*

Typedefs

- *Typedef camp::type::v::add*
- *Typedef camp::type::v::rem*

3.1.12 Namespace camp::type_traits

Page Contents

- *Classes*
- *Typedefs*

Classes

- *Template Struct is_arithmetic*
- *Template Struct is_bidirectional_iterator*
- *Template Struct is_bidirectional_range*
- *Template Struct is_comparable*
- *Template Struct is_comparable_to*
- *Template Struct is_floating_point*
- *Template Struct is_forward_iterator*
- *Template Struct is_forward_range*
- *Template Struct is_integral*
- *Template Struct is_iterator*
- *Template Struct is_random_access_iterator*
- *Template Struct is_random_access_range*
- *Template Struct is_range*
- *Template Struct is_signed*
- *Template Struct is_unsigned*
- *Template Struct SpecializationOf*
- *Template Struct SpecializationOf< Expected, Actual< Args... > >*

Typedefs

- *Typedef camp::type_traits::IsSpecialized*
- *Typedef camp::type_traits::IterableValue*
- *Typedef camp::type_traits::IteratorValue*

3.2 Classes and Structs

3.2.1 Template Struct accumulate

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Op`
2. `typename Initial`
3. `typename Seq`

Struct Documentation

template<template<typename...> class **Op**, typename **Initial**, typename **Seq**>
struct accumulate

3.2.2 Template Struct accumulate< Op, Initial, list< Elements... > >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Op`
2. `typename Initial`
3. `typename... Elements`

Struct Documentation

template<template<typename...> class **Op**, typename **Initial**, typename ...**Elements**>
struct accumulate<*Op*, *Initial*, *list*<*Elements*...>>

Public Types

template<>
using type = **typename** detail::accumulate_impl::type

3.2.3 Template Struct append

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Seq`
2. `typename T`

Struct Documentation

template<typename **Seq**, typename **T**>
struct append

3.2.4 Template Struct append< list< Elements... >, T >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Elements`
2. `typename T`

Struct Documentation

template<typename ...**Elements**, typename **T**>
struct append<*list*<*Elements*...>, *T*>

Public Types

template<>
using type = *list*<Elements..., T>

3.2.5 Template Struct apply_l

- Defined in file_camp_lambda.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Lambda`
2. `typename Seq`

Struct Documentation

template<typename **Lambda**, typename **Seq**>
struct apply_l

3.2.6 Template Struct apply_l< Lambda, list< Args... > >

- Defined in file_camp_lambda.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Lambda`
2. `typename... Args`

Struct Documentation

template<typename **Lambda**, typename ...**Args**>
struct apply_l<*Lambda*, *list*<*Args*...>>

Public Types

template<>
using type = **typename** Lambda::template expr::type

3.2.7 Template Struct arg

- Defined in file_camp_lambda.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `idx_t n`

Struct Documentation

template<*idx_t* **n**>
struct arg

Public Types

template<>
using expr = **typename** *at*<*list*<Ts...>, *num*<n - 1>>::type

3.2.8 Template Struct as_list_s

- Defined in file_camp_list_list.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public detail::_as_list::type< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename **`T`**>
`struct as_list_s` : **`public`** detail::_as_list::type<*T*>

3.2.9 Template Struct as_list_s< tagged_tuple< camp::list< Tags... >, Args... > >

- Defined in file_camp_tuple.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Tags`
2. `typename... Args`

Struct Documentation

template<typename ...**Tags**, typename ...**Args**>
struct as_list_s<tagged_tuple<camp::*list*<*Tags*...>, *Args*...>>

Public Types

template<>
using type = *list*<Args...>

3.2.10 Template Struct at

- Defined in file_camp_list_at.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Seq`
2. `typename Num`

Struct Documentation

template<typename **Seq**, typename **Num**>
struct at

3.2.11 Template Struct at< T, num< Val > >

- Defined in file_camp_list_at.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`
2. `idx_t Val`

Struct Documentation

template<typename **T**, *idx_t* **Val**>
struct at<*T*, *num*<*Val*>>

Public Types

template<>
using type = **typename** detail::_at::type

3.2.12 Template Struct at_key_s

- Defined in file_camp_map.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Seq`
2. `typename Key`

Struct Documentation

template<typename **Seq**, typename **Key**>
struct at_key_s

Public Types

template<>
using type = decltype(detail::lookup<Key>(*declptr*<detail::lookup_table<Seq>>()))

3.2.13 Template Struct bind

- Defined in file_camp_lambda.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Expr`
2. `typename... ArgBindings`

Struct Documentation

template<template<typename...> class **Expr**, typename ...**ArgBindings**>
struct bind

Public Types

template<>
using bindings = *list*<ArgBindings...>

template<>
using expr = **typename** Expr<**typename** detail::get_bound_arg<ArgBindings, Ts...>::type...>::type

template<>
using type = bind

3.2.14 Template Struct bind_front

- Defined in file_camp_lambda.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Expr`
2. `typename... BoundArgs`

Struct Documentation

template<template<typename...> class **`Expr`**, typename ...**`BoundArgs`**>
`struct bind_front`

Public Types

template<>
`using expr` = **`typename`** Expr<BoundArgs..., Ts...>::type

template<>
`using type` = bind_front

3.2.15 Template Struct all_of

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::metalib::all_of_t< Args... >` (*Template Struct all_of_t*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct all_of : **public** camp::concepts::metalib::*all_of_t*<*Args*...>
metaprogramming concept for SFINAE checking of aggregating concepts

3.2.16 Template Struct any_of

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::metalib::any_of_t< Args... >` (*Template Struct any_of_t*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct any_of : **public** camp::concepts::metalib::*any_of_t*<*Args*...>
metaprogramming concept for SFINAE checking of aggregating concepts

3.2.17 Template Struct Arithmetic

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- public decltype___valid_expr___ isstd::is_arithmetic< T >

Template Parameter Order

1. typename T

Struct Documentation

template<typename T> camp::concepts::Arithmetic : public decltype___valid_expr___ isstd::is

3.2.18 Template Struct BidirectionalIterator

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- public decltype___valid_expr___ ForwardIterator< T >
- public has_typeval< T &>
- public convertible_toval< T &>
- public val< T &>

Template Parameter Order

1. typename T

Struct Documentation

template<typename T> camp::concepts::BidirectionalIterator : public decltype___valid_expr_

3.2.19 Template Struct BidirectionalRange

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ HasBeginEnd< T >`
- `public camp::concepts::BidirectionalIterator< iterator_from< T > >` (*Template Struct BidirectionalIterator*)

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename T> camp::concepts::BidirectionalRange : public decltype___valid_expr___ H

3.2.20 Template Struct Comparable

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
  - *Derived Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::ComparableTo< T, T >` (*Template Struct ComparableTo*)

Derived Type

- `public camp::concepts::RandomAccessIterator< T >` (*Template Struct RandomAccessIterator*)

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename **T**>
struct Comparable : **public** camp::concepts::ComparableTo<*T*, *T*>
Subclassed by *camp::concepts::RandomAccessIterator< T >*

3.2.21 Template Struct ComparableTo

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ convertible_toval< U >< val< T >()), convertible_to< bool >(val< T >()< val< U >()), convertible_to< bool >(val< U >()<=val< T >()), convertible_to< bool >(val< T >()<=val< U >()), convertible_to< bool >(val< U >() > val< T >()), convertible_to< bool >(val< T >() > val< U >()), convertible_to< bool >(val< U >() >=val< T >()), convertible_to< bool >(val< T >() > val< U >`
- `public convertible_toval< U > val< T >`
- `public convertible_toval< T > val< U >`
- `public convertible_toval< U > val< T >`

- `public convertible_toval< T > val< U >`

Template Parameter Order

1. `typename T`
2. `typename U`

Struct Documentation

`template<typename T, typename U> camp::concepts::ComparableTo : public decltype___valid_exp`

3.2.22 Template Struct EqualityComparable

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public decltype___valid_expr___ convertible_toval< T > val< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::EqualityComparable : public decltype___valid_expr___ c`

3.2.23 Template Struct FloatingPoint

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public decltype___valid_expr___ isstd::is_floating_point< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::FloatingPoint : public decltype___valid_expr___ isstd`

3.2.24 Template Struct ForwardIterator

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ Iterator< T >`
- `public val< T &>`
- `public val< T &>`

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::ForwardIterator : public decltype___valid_expr___ Ite`

3.2.25 Template Struct ForwardRange

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ HasBeginEnd< T >`
- `public camp::concepts::ForwardIterator< iterator_from< T > >` (*Template Struct ForwardIterator*)

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename T> camp::concepts::ForwardRange : public decltype___valid_expr___ HasBeg

3.2.26 Template Struct HasBeginEnd

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ std::beginval< T >`
- `public std::endval< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename T> camp::concepts::HasBeginEnd : public decltype___valid_expr___ std::beg

3.2.27 Template Struct Integral

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public decltype___valid_expr___ isstd::is_integral< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::Integral : public decltype___valid_expr___ isstd::is_`

3.2.28 Template Struct Iterator

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ is_notIntegral< T >`
- `public val< T >`
- `public has_typeval< T &>`

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename T> camp::concepts::Iterator : public decltype___valid_expr___ is_notInteg

3.2.29 Template Struct LessEqualComparable

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public decltype___valid_expr___ convertible_toval< T ><=val< T >()))) { };template< T > struct GreaterEqualComparable :decltype(___valid_expr___(convertible_to< bool >(val< T >() > val< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename T> camp::concepts::LessEqualComparable : public decltype___valid_expr___

3.2.30 Template Struct LessThanComparable

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public decltype___valid_expr___ convertible_toval< T >< val< T >()))) { };template< T > struct GreaterThanComparable :decltype(___valid_expr___(convertible_to< bool >(val< T >() > val< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::LessThanComparable : public decltype___valid_expr___ c`

3.2.31 Template Struct all_of

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::is_same< list< t, num< Bs >... >, list< num< Bs >..., t > >`

Template Parameter Order

1. `bool... Bs`

Struct Documentation

template<bool... **Bs**>
struct all_of : **public** camp::*is_same*<*list*<*t*, *num*<*Bs*>...>, *list*<*num*<*Bs*>..., *t*>>
all_of metafunction of a value type list all must be “true”

3.2.32 Template Struct all_of_t

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::metalib::all_of< Bs::value... >` (*Template Struct all_of*)

Template Parameter Order

1. `typename... Bs`

Struct Documentation

template<typename ...**Bs**>
struct all_of_t : **public** camp::concepts::metalib::*all_of*<*Bs*::value...>
all_of metafunction of a bool list all must be “true”

3.2.33 Template Struct any_of

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::metalib::negate_t< none_of< Bs... > >` (*Template Struct negate_t*)

Template Parameter Order

1. `bool... Bs`

Struct Documentation

template<bool... **Bs**>
struct any_of : **public** camp::concepts::metalib::*negate_t*<*none_of*<*Bs*...>>
any_of metafunction of a value type list at least one must be “true””

3.2.34 Template Struct any_of_t

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::metalib::any_of< Bs::value... >` (*Template Struct any_of*)

Template Parameter Order

1. `typename... Bs`

Struct Documentation

template<typename ...**Bs**>
struct any_of_t : **public** camp::concepts::metalib::*any_of*<*Bs*::value...>
any_of metafunction of a bool list at least one must be “true””

3.2.35 Template Struct negate_t

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::integral_constant<!T::value >` (*Template Struct integral_constant*)

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename **T**>
struct negate_t : **public** camp::integral_constant<!*T*::value>
negation metafunction of a value type

3.2.36 Template Struct none_of

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::is_same< idx_seq< false, Bs... >, idx_seq< Bs..., false > >`

Template Parameter Order

1. `bool... Bs`

Struct Documentation

template<bool... **Bs**>
struct none_of : **public** camp::*is_same*<*idx_seq*<false, *Bs*...>, *idx_seq*<*Bs*..., false>>
none_of metafunction of a value type list all must be “false”

3.2.37 Template Struct none_of_t

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::metalib::none_of< Bs::value... >` (*Template Struct none_of*)

Template Parameter Order

1. `typename... Bs`

Struct Documentation

template<typename ...**Bs**>
struct none_of_t : **public** camp::concepts::metalib::*none_of*<*Bs*::value...>
none_of metafunction of a bool list all must be “false”

3.2.38 Template Struct none_of

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::metalib::none_of_t< Args... >` (*Template Struct none_of_t*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct none_of : **public** camp::concepts::metalib::*none_of_t*<*Args*...>
metaprogramming concept for SFINAE checking of aggregating concepts

3.2.39 Template Struct RandomAccessIterator

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- public decltype___valid_expr___ BidirectionalIterator< T >
- public camp::concepts::Comparable< T > (*Template Struct Comparable*)
- public has_typeval< T &> val< diff_from< T > >
- public has_typeval< T > val< diff_from< T > >
- public has_typeval< diff_from< T > > val< T >
- public has_typeval< T &> val< diff_from< T > >
- public has_typeval< T > val< diff_from< T > >
- public valval< diff_from< T > >

Template Parameter Order

1. typename T

Struct Documentation

template<typename T> camp::concepts::RandomAccessIterator : public decltype___valid_expr__

3.2.40 Template Struct RandomAccessRange

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- public decltype___valid_expr___ HasBeginEnd< T >
- public camp::concepts::RandomAccessIterator< iterator_from< T > > (*Template Struct RandomAccessIterator*)

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::RandomAccessRange : public decltype___valid_expr___ Ha`

3.2.41 Template Struct Range

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ HasBeginEnd< T >`
- `public camp::concepts::Iterator< iterator_from< T > >` (*Template Struct Iterator*)

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::Range : public decltype___valid_expr___ HasBeginEnd< T`

3.2.42 Template Struct requires_

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*

- *Struct Documentation*

Inheritance Relationships

Base Type

- `public detail::detected< Op, detail::TL< Args... > >`

Template Parameter Order

1. `template< class... > class Op`
2. `class... Args`

Struct Documentation

template<template<class...> class **Op**, class ...**Args**>
struct requires_ : **public** detail::detected<*Op*, detail::TL<*Args*...>>
SFINAE concept checking.

3.2.43 Template Struct Signed

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ Integral< T >`
- `public isstd::is_signed< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::Signed : public decltype___valid_expr___ Integral< T >`

3.2.44 Template Struct Swappable

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ swapval< T >`
- `public val< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::Swappable : public decltype___valid_expr___ swapval< T`

3.2.45 Template Struct Unsigned

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Types*
- *Template Parameter Order*

- *Struct Documentation*

Inheritance Relationships

Base Types

- `public decltype___valid_expr___ Integral< T >`
- `public isstd::is_unsigned< T >`

Template Parameter Order

1. `typename T`

Struct Documentation

`template<typename T> camp::concepts::Unsigned : public decltype___valid_expr___ Integral< T`

3.2.46 Template Struct extend

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Seq`
2. `typename T`

Struct Documentation

template<typename **`Seq`**, typename **`T`**>
`struct extend`

3.2.47 Template Struct extend< list< Elements... >, list< NewElements... > >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Elements`
2. `typename... NewElements`

Struct Documentation

template<typename ...**Elements**, typename ...**NewElements**>
struct extend<*list*<*Elements*...>, *list*<*NewElements*...>>

Public Types

template<>
using type = *list*<Elements..., NewElements...>

3.2.48 Template Struct filter

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Op`
2. `typename Seq`

Struct Documentation

template<template<typename...> class **Op**, typename **Seq**>
struct filter

3.2.49 Template Struct filter< Op, list< Elements... > >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Op`
2. `typename... Elements`

Struct Documentation

template<template<typename...> class **Op**, typename ...**Elements**>
struct filter<*Op*, *list*<*Elements*...>>

Public Types

template<>
using append_if = *if_*<**typename** Op<T>::type, **typename** *append*<Seq, T>::type, Seq>

template<>
using type = **typename** *accumulate*::type

3.2.50 Template Struct find_if

- Defined in file_camp_list_find_if.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Cond`
2. `typename Seq`

Struct Documentation

template<template<typename...> class **Cond**, typename **Seq**>
struct find_if

3.2.51 Template Struct find_if< Cond, list< Elements... > >

- Defined in file_camp_list_find_if.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Cond`
2. `typename... Elements`

Struct Documentation

template<template<typename...> class **Cond**, typename ...**Elements**>
struct find_if<*Cond*, *list*<*Elements*...>>

Public Types

template<>
using type = **typename** detail::_find_if::type

3.2.52 Template Struct flatten

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Seq`

Struct Documentation

template<typename **Seq**>
struct flatten

3.2.53 Template Struct flatten< list< Elements... > >

- Defined in file_camp_camp.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::detail::flatten_impl< list<>, sizeof...(Elements), Elements... >`

Template Parameter Order

1. `typename... Elements`

Struct Documentation

template<typename ...**Elements**>
struct flatten<*list*<*Elements*...>> : **public** camp::detail::flatten_impl<*list*<>, sizeof...(*Elements*), *Elements*...>

3.2.54 Template Struct idx_seq_from

- Defined in file_camp_number.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename **T**>
struct idx_seq_from

3.2.55 Template Struct idx_seq_from< int_seq< T, Args... > >

- Defined in file_camp_number.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::make_idx_seq< sizeof...(Args) >` (*Template Struct make_idx_seq*)

Template Parameter Order

1. `typename T`
2. `T... Args`

Struct Documentation

template<typename **T**, *T*... **Args**>
struct idx_seq_from<*int_seq*<*T*, *Args*...>> : **public** camp::*make_idx_seq*<sizeof...(*Args*)>

3.2.56 Template Struct idx_seq_from< T< Args... > >

- Defined in file_camp_number.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*

- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::make_idx_seq< sizeof...(Args) >` (*Template Struct make_idx_seq*)

Template Parameter Order

1. `template< typename... > class T`
2. `typename... Args`

Struct Documentation

template<template<typename...> class **T**, typename ...**Args**>
`struct idx_seq_from`<*T*<*Args*...>> : **`public`** camp::*make_idx_seq*<sizeof...(*Args*)>

3.2.57 Template Struct if_cs

- Defined in file_camp_number_if.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `bool Cond`
2. `typename Then`
3. `typename Else`

Struct Documentation

template<bool **Cond**, typename **Then** = camp::*true_type*, typename **Else** = camp::*false_type*>
`struct if_cs`

Public Types

template<>
using type = Then

3.2.58 Template Struct if_cs< false, Then, Else >

- Defined in file_camp_number_if.hpp

Page Contents

- *Inheritance Relationships*
  - *Derived Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Derived Type

- `public camp::if_s< nil, Then, Else >` (*Template Struct if_s< nil, Then, Else >*)

Template Parameter Order

1. `typename Then`
2. `typename Else`

Struct Documentation

template<typename **Then**, typename **Else**>
struct if_cs<false, *Then*, *Else*>
Subclassed by *camp::if_s< nil, Then, Else >*

Public Types

template<>
using type = Else

3.2.59 Template Struct if_s

- Defined in file_camp_number_if.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::if_cs< Cond::value, Then, Else >` (*Template Struct if_cs*)

Template Parameter Order

1. `typename Cond`
2. `typename Then`
3. `typename Else`

Struct Documentation

template<typename **Cond**, typename **Then** = camp::*true_type*, typename **Else** = camp::*false_type*>
struct if_s : **public** camp::*if_cs*<*Cond*::value, *Then*, *Else*>

3.2.60 Template Struct if_s< nil, Then, Else >

- Defined in file_camp_number_if.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::if_cs< false, Then, Else >` (*Template Struct if_cs< false, Then, Else >*)

Template Parameter Order

1. `typename Then`
2. `typename Else`

Struct Documentation

template<typename **`Then`**, typename **`Else`**>
`struct if_s`<*nil*, *Then*, *Elses*> : **`public`** camp::if_cs<false, *Then*, *Else*>

3.2.61 Template Struct index_of

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`
2. `typename L`

Struct Documentation

template<typename **`T`**, typename **`L`**>
`struct index_of`
Get the index of the first instance of T in L.

3.2.62 Template Struct index_of< T, list< Elements... > >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`
2. `typename... Elements`

Struct Documentation

template<typename **T**, typename ...**Elements**>
struct index_of<*T*, *list*<*Elements*...>>

Public Types

template<>
using inc_until = *if_*<**typename** std::is_same<T, Item>::type, *if_c*<*size*<Seq>::value == 1, **typename** *prepend*<Seq, *nu*

template<>
using indices = **typename** *accumulate*<*inc_until*, *list*<*num*<0>>, *list*<Elements...>>::type

template<>
using type = **typename** *if_c*::type

3.2.63 Template Struct int_seq

- Defined in file_camp_number.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`
2. `T... vs`

Struct Documentation

template<typename **T**, *T*... **vs**>
struct int_seq

Public Types

template<>
using type = int_seq

3.2.64 Template Struct integral_constant

- Defined in file_camp_number_number.hpp

Page Contents

- *Inheritance Relationships*
  - *Derived Types*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Derived Types

- `public camp::concepts::metalib::negate_t< T >` (*Template Struct negate_t*)
- `public camp::is_same_s< T, U >` (*Template Struct is_same_s*)
- `public camp::is_same_s< T, T >` (*Template Struct is_same_s< T, T >*)
- `public camp::type_traits::SpecializationOf< class, T >` (*Template Struct SpecializationOf*)
- `public camp::concepts::metalib::negate_t< none_of< Bs... > >` (*Template Struct negate_t*)

Template Parameter Order

1. `class NumT`
2. `NumT v`

Struct Documentation

template<class **NumT**, *NumT* **v**>
struct integral_constant

Subclassed by *camp::concepts::metalib::negate_t< T >*, *camp::is_same_s< T, U >*, *camp::is_same_s< T, T >*, *camp::type_traits::SpecializationOf< class, T >*, *camp::concepts::metalib::negate_t< none_of< Bs... > >*

Public Types

template<>
using value_type = NumT

template<>
using type = integral_constant

Public Functions

constexpr operator value_type() **const**

constexpr value_type **operator()**() **const**

Public Static Attributes

constexpr NumT **value** = v

3.2.65 Template Struct invoke_l

- Defined in file_camp_lambda.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Lambda`
2. `typename... Args`

Struct Documentation

template<typename **Lambda**, typename ...**Args**>
struct invoke_l

Public Types

template<>
using type = **typename** Lambda::template expr::type

3.2.66 Template Struct is_same_s

- Defined in file_camp_type_traits_is_same.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::integral_constant< false >` (*Template Struct integral_constant*)

Template Parameter Order

1. `typename T`
2. `typename U`

Struct Documentation

template<typename **T**, typename **U**>
`struct is_same_s` : **`public`** camp::integral_constant<false>

3.2.67 Template Struct is_same_s< T, T >

- Defined in file_camp_type_traits_is_same.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::integral_constant< true >` (*Template Struct integral_constant*)

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename **T**>
struct is_same_s<*T*, *T*> : **public** camp::integral_constant<true>

3.2.68 Template Struct is_value_s

- Defined in file_camp_value.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Val`

Struct Documentation

template<typename **Val**>
struct is_value_s

Test whether a type is a valid camp value.

Public Types

template<>
using type = camp::*t*

3.2.69 Template Struct join

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Seqs`

Struct Documentation

template<typename ...**Seqs**>
struct join

3.2.70 Template Struct join< Seq1 >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Seq1`

Struct Documentation

template<typename **Seq1**>
struct join<*Seq1*>

Public Types

template<>
using type = Seq1

3.2.71 Template Struct join< Seq1, Seq2, Rest... >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Seq1`
2. `typename Seq2`
3. `typename... Rest`

Struct Documentation

template<typename **Seq1**, typename **Seq2**, typename ...**Rest**>
struct join<*Seq1*, *Seq2*, *Rest*...>

Public Types

template<>
using type = **typename** join::type

3.2.72 Template Struct join<>

- Defined in file_camp_camp.hpp

Page Contents

- *Struct Documentation*

Struct Documentation

template<>
struct join<>

Public Types

template<>
using type = *list*<>

3.2.73 Template Struct lambda

- Defined in file_camp_lambda.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Expr`

Struct Documentation

template<template<typename...> class **Expr**>
struct lambda

Public Types

template<>
using expr = **typename** Expr<Ts...>::type

3.2.74 Template Struct list

- Defined in file_camp_list_list.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Ts`

Struct Documentation

template<typename ...**Ts**>
struct list

Public Types

template<>
using type = list

3.2.75 Template Struct make_idx_seq

- Defined in file_camp_number.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `idx_t Upper`

Struct Documentation

template<*idx_t* **Upper**>
struct make_idx_seq

Public Types

template<>
using type = **typename** detail::gen_seq::type

3.2.76 Template Struct make_int_seq

- Defined in file_camp_number.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*

- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public detail::gen_seq::type< T, integral_constant< T, Upper > >`

Template Parameter Order

1. `typename T`
2. `T Upper`

Struct Documentation

template<typename **`T`**, *T* **`Upper`**>
`struct make_int_seq` : **`public`** detail::gen_seq::type<*T*, integral_constant<*T*, *Upper*>>

3.2.77 Template Struct not_

- Defined in file_camp_number.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename **`T`**>
`struct not_`

Public Types

template<>
using type = **typename** *if_s*::type

3.2.78 Template Struct prepend

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Seq`
2. `typename T`

Struct Documentation

template<typename **Seq**, typename **T**>
struct prepend

3.2.79 Template Struct prepend< list< Elements... >, T >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Elements`
2. `typename T`

Struct Documentation

template<typename ...**Elements**, typename **T**>
struct prepend<*list*<*Elements*...>, *T*>

Public Types

template<>
using type = *list*<Elements..., T>

3.2.80 Template Struct resource_from_platform

- Defined in file_camp_resource.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `Platform p`

Struct Documentation

template<*Platform* **p**>
struct resource_from_platform

3.2.81 Template Struct resource_from_platform< Platform::host >

- Defined in file_camp_resource.hpp

Page Contents

- *Struct Documentation*

Struct Documentation

template<>
struct resource_from_platform<*Platform*::host>

Public Types

template<>
using type = camp::resources::Host

3.2.82 Template Struct seq_at

- Defined in file_camp_number.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `idx_t N`
2. `typename IdxSeg`

Struct Documentation

template<*idx_t* **N**, typename **IdxSeg**>
struct seq_at

3.2.83 Template Struct seq_at< 0, camp::int_seq< T, Idx0, IdxRest... > >

- Defined in file_camp_number.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`
2. `T Idx0`
3. `T... IdxRest`

Struct Documentation

template<typename **T**, *T* **Idx0**, *T*... **IdxRest**>
struct seq_at<0, camp::*int_seq*<*T*, *Idx0*, *IdxRest*...>>

Public Static Attributes

constexpr T **value** = Idx0

3.2.84 Template Struct seq_at< N, camp::int_seq< T, Idx0, IdxRest... > >

- Defined in file_camp_number.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `idx_t N`
2. `typename T`
3. `T Idx0`
4. `T... IdxRest`

Struct Documentation

template<*idx_t* **N**, typename **T**, *T* **Idx0**, *T*... **IdxRest**>
struct seq_at<*N*, camp::*int_seq*<*T*, *Idx0*, *IdxRest*...>>

Public Static Attributes

constexpr T **value** = seq_at<N - 1, camp::*int_seq*<T, IdxRest...>>::value

3.2.85 Template Struct size

- Defined in file_camp_size.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`

Struct Documentation

template<typename **T**>
struct size

3.2.86 Template Struct size< int_seq< T, Args... > >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename T`
2. `T... Args`

Struct Documentation

template<typename **T**, *T*... **Args**>
struct size<*int_seq*<*T*, *Args*...>>

Public Types

template<>
using type = *num*<sizeof...(Args)>

Public Static Attributes

constexpr *idx_t* **value** = {sizeof...(Args)}

3.2.87 Template Struct size< list< Args... > >

- Defined in file_camp_list_list.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct size<*list*<*Args*...>>

Public Types

template<>
using type = *num*<sizeof...(Args)>

Public Static Attributes

constexpr *idx_t* **value** = {sizeof...(Args)}

3.2.88 Template Struct tagged_tuple::is_pack_this_tuple

- Defined in file_camp_tuple.hpp

Page Contents

- *Nested Relationships*
- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Nested Relationships

This struct is a nested type of *Template Class tagged_tuple*.

Inheritance Relationships

Base Type

- `public camp::integral_constant< false >` (*Template Struct integral_constant*)

Template Parameter Order

1. `typename... Ts`

Struct Documentation

template<typename ...**Ts**>
`struct is_pack_this_tuple` : **`public`** camp::integral_constant<false>

3.2.89 Template Struct tagged_tuple::is_pack_this_tuple< That >

- Defined in file_camp_tuple.hpp

Page Contents

- *Nested Relationships*
- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Nested Relationships

This struct is a nested type of *Template Class tagged_tuple*.

Inheritance Relationships

Base Type

- `public std::is_same< tagged_tuple, decay< That > >`

Template Parameter Order

1. `typename That`

Struct Documentation

template<typename **That**>
struct is_pack_this_tuple<*That*> : **public** std::is_same<tagged_tuple, *decay*<*That*>>

3.2.90 Template Struct transform

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Op`
2. `typename T`

Struct Documentation

template<template<typename...> class **Op**, typename **T**>
struct transform

3.2.91 Template Struct transform< Op, list< Elements... > >

- Defined in file_camp_camp.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `template< typename... > class Op`
2. `typename... Elements`

Struct Documentation

template<template<typename...> class **Op**, typename ...**Elements**>
struct transform<*Op*, *list*<*Elements*...>>

Public Types

template<>
using type = *list*<**typename** Op<Elements>::type...>

3.2.92 Template Struct tuple

- Defined in file_camp_tuple.hpp

Page Contents

- *Nested Relationships*
  - *Nested Types*
- *Template Parameter Order*
- *Struct Documentation*

Nested Relationships

Nested Types

- *Template Struct tuple::is_pack_this_tuple*
- *Template Struct tuple::is_pack_this_tuple< That >*

Template Parameter Order

1. `typename... Elements`

Struct Documentation

template<typename ...**Elements**>
struct tuple

Public Types

template<>
using TList = camp::*list*<Elements...>

template<>
using TMap = **typename** internal::tag_map<camp::*list*<Elements...>, camp::*make_idx_seq_t*<sizeof...(Elements)>>::type

template<>
using type = tuple

Public Functions

template<bool B = concepts::metalib::all_of< std::is_default_constructi

constexpr tuple(*tuple* **const** &*o*)

constexpr tuple(*tuple* &&*o*)

tuple &**operator=**(tuple **const** &*rhs*)

tuple &**operator=**(tuple &&*rhs*)

template<typename... Args, typename std::enable_if< !is_pack_this_tuple< Args... >::va

template<typename ...**RTypes**>
constexpr Self &**operator=**(**const** tuple<*RTypes*...> &*rhs*)

3.2.93 Template Struct tuple::is_pack_this_tuple

- Defined in file_camp_tuple.hpp

Page Contents

- *Nested Relationships*
- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Nested Relationships

This struct is a nested type of *Template Struct tuple*.

Inheritance Relationships

Base Type

- `public camp::integral_constant< false >` (*Template Struct integral_constant*)

Template Parameter Order

1. `typename... Ts`

Struct Documentation

template<typename ...**Ts**>
`struct is_pack_this_tuple` : **`public`** camp::integral_constant<false>

3.2.94 Template Struct tuple::is_pack_this_tuple< That >

- Defined in file_camp_tuple.hpp

Page Contents

- *Nested Relationships*
- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Nested Relationships

This struct is a nested type of *Template Struct tuple*.

Inheritance Relationships

Base Type

- `public std::is_same< tuple, decay< That > >`

Template Parameter Order

1. `typename That`

Struct Documentation

template<typename **That**>
struct is_pack_this_tuple<*That*> : **public** std::is_same<tuple, *decay*<*That*>>

3.2.95 Template Struct tuple_element

- Defined in file_camp_tuple.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `camp::idx_t i`
2. `typename T`

Struct Documentation

template<camp::*idx_t* **i**, typename **T**>
struct tuple_element

Public Types

template<>
using type = camp::*at_v*<**typename** T::TList, i>

3.2.96 Template Struct tuple_size

- Defined in file_camp_tuple.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename Tuple`

Struct Documentation

template<typename **`Tuple`**>
`struct tuple_size`

3.2.97 Template Struct tuple_size< tagged_tuple< L, Args... > & >

- Defined in file_camp_tuple.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename L`
2. `typename... Args`

Struct Documentation

template<typename **`L`**, typename ...**`Args`**>
`struct tuple_size`<tagged_tuple<*L*, *Args*...>&>

Public Static Attributes

`constexpr` size_t **`value`** = sizeof...(Args)

3.2.98 Template Struct tuple_size< tagged_tuple< L, Args... > >

- Defined in file_camp_tuple.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename L`
2. `typename... Args`

Struct Documentation

template<typename **L**, typename ...**Args**>
struct tuple_size<tagged_tuple<*L*, *Args*...>>

Public Static Attributes

constexpr size_t **value** = sizeof...(Args)

3.2.99 Template Struct tuple_size< tuple< Args... > & >

- Defined in file_camp_tuple.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct tuple_size<tuple<*Args*...>&>

Public Static Attributes

constexpr size_t **value** = sizeof...(Args)

3.2.100 Template Struct tuple_size< tuple< Args... > >

- Defined in file_camp_tuple.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct tuple_size<tuple<*Args*...>>

Public Static Attributes

constexpr size_t **value** = sizeof...(Args)

3.2.101 Template Struct rem_s

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s

Public Types

template<>
using type = T

3.2.102 Template Struct rem_s< const T >

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s<const *T*>

Public Types

template<>
using type = T

3.2.103 Template Struct rem_s

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s

Public Types

template<>
using type = T

3.2.104 Template Struct rem_s< const T >

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s<const *T*>

Public Types

template<>
using type = T

3.2.105 Template Struct rem_s< const volatile T >

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s<volatile const *T*>

Public Types

template<>
using type = T

3.2.106 Template Struct rem_s< volatile T >

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s<volatile *T*>

Public Types

template<>
using type = T

3.2.107 Template Struct rem_s

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s

Public Types

template<>
using type = T

3.2.108 Template Struct rem_s< T & >

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s<*T*&>

Public Types

template<>
using type = T

3.2.109 Template Struct rem_s< T && >

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s<*T*&&>

Public Types

template<>
using type = T

3.2.110 Template Struct rem_s

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s

Public Types

template<>
using type = T

3.2.111 Template Struct rem_s< volatile T >

- Defined in file_camp_helpers.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `class T`

Struct Documentation

template<class **T**>
struct rem_s<volatile *T*>

Public Types

template<>
using type = T

3.2.112 Template Struct is_arithmetic

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*

- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::Arithmetic, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
`struct is_arithmetic` : **`public`** camp::concepts::*requires_*<camp::concepts::Arithmetic, *Args*...>

3.2.113 Template Struct is_bidirectional_iterator

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::BidirectionalIterator, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_bidirectional_iterator : **public** camp::concepts::*requires_*<camp::concepts::BidirectionalIterator, *Args*...>

3.2.114 Template Struct is_bidirectional_range

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::BidirectionalRange, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_bidirectional_range : **public** camp::concepts::*requires_*<camp::concepts::BidirectionalRange, *Args*...>

3.2.115 Template Struct is_comparable

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*

- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::Comparable, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
`struct is_comparable` : **`public`** camp::concepts::*requires_*<camp::concepts::*Comparable*, *Args*...>

3.2.116 Template Struct is_comparable_to

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::ComparableTo, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_comparable_to : **public** camp::concepts::*requires_*<camp::concepts::ComparableTo, *Args*...>

3.2.117 Template Struct is_floating_point

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::FloatingPoint, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_floating_point : **public** camp::concepts::*requires_*<camp::concepts::FloatingPoint, *Args*...>

3.2.118 Template Struct is_forward_iterator

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*

- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::ForwardIterator, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
`struct is_forward_iterator` : **`public`** camp::concepts::*requires_*<camp::concepts::ForwardIterator, *Args*...>

3.2.119 Template Struct is_forward_range

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::ForwardRange, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_forward_range : **public** camp::concepts::*requires_*<camp::concepts::ForwardRange, *Args*...>

3.2.120 Template Struct is_integral

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::Integral, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_integral : **public** camp::concepts::*requires_*<camp::concepts::Integral, *Args*...>

3.2.121 Template Struct is_iterator

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*

- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::Iterator, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
`struct is_iterator` : **`public`** camp::concepts::*requires_*<camp::concepts::Iterator, *Args*...>

3.2.122 Template Struct is_random_access_iterator

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::RandomAccessIterator, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_random_access_iterator : **public** camp::concepts::*requires_*<camp::concepts::RandomAccessIterator, *Args*...>

3.2.123 Template Struct is_random_access_range

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::RandomAccessRange, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_random_access_range : **public** camp::concepts::*requires_*<camp::concepts::RandomAccessRange, *Args*...>

3.2.124 Template Struct is_range

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*

- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::Range, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
`struct is_range` : **`public`** camp::concepts::*requires_*<camp::concepts::Range, *Args*...>

3.2.125 Template Struct is_signed

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::Signed, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_signed : **public** camp::concepts::*requires_*<camp::concepts::Signed, *Args*...>

3.2.126 Template Struct is_unsigned

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::concepts::requires_< camp::concepts::Unsigned, Args... >` (*Template Struct requires_*)

Template Parameter Order

1. `typename... Args`

Struct Documentation

template<typename ...**Args**>
struct is_unsigned : **public** camp::concepts::*requires_*<camp::concepts::Unsigned, *Args*...>

3.2.127 Template Struct SpecializationOf

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*

- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public camp::integral_constant< false >` (*Template Struct integral_constant*)

Template Parameter Order

1. `template< class... > class`
2. `typename T`

Struct Documentation

template<template<class...> class, typename **T**>
`struct SpecializationOf` : **`public`** camp::integral_constant<false>

3.2.128 Template Struct SpecializationOf< Expected, Actual< Args... > >

- Defined in file_camp_concepts.hpp

Page Contents

- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Struct Documentation*

Inheritance Relationships

Base Type

- `public detail::SpecializationOf< Expected, Actual, IsSpecialized< Expected, Args... >::value, Args... >`

Template Parameter Order

1. `template< class... > class Expected`
2. `template< class... > class Actual`
3. `class... Args`

Struct Documentation

template<template<class...> class **Expected**, template<class...> class **Actual**, class ...**Args**>
struct SpecializationOf<*Expected*, *Actual*<*Args*...>> : **public** detail::SpecializationOf<*Expected*, *Actual*, *IsSpecialized*<*Ex*

3.2.129 Template Struct value

- Defined in file_camp_value.hpp

Page Contents

- *Template Parameter Order*
- *Struct Documentation*

Template Parameter Order

1. `typename val`

Struct Documentation

template<typename **val**>
struct value

Public Types

template<>
using type = *val*

3.2.130 Class Event

- Defined in file_camp_resource_event.hpp

Page Contents

- *Nested Relationships*
  - *Nested Types*
- *Class Documentation*

Nested Relationships

Nested Types

- *Class Event::EventInterface*
- *Template Class Event::EventModel*

Class Documentation

class Event

Public Functions

Event()

template<typename **T**>
Event(*T* &&*value*)

bool **check**() **const**

void **wait**() **const**

template<typename **T**>
T ***try_get**()

template<typename **T**>
T **get**()

3.2.131 Class Event::EventInterface

- Defined in file_camp_resource_event.hpp

Page Contents

- *Nested Relationships*
- *Class Documentation*

Nested Relationships

This class is a nested type of *Class Event*.

Class Documentation

class EventInterface

Public Functions

virtual ~EventInterface()

virtual bool **check**() **const** = 0

virtual void **wait**() **const** = 0

3.2.132 Template Class Event::EventModel

- Defined in file_camp_resource_event.hpp

Page Contents

- *Nested Relationships*
- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Class Documentation*

Nested Relationships

This class is a nested type of *Class Event*.

Inheritance Relationships

Base Type

- `public camp::resources::v1::Event::EventInterface`

Template Parameter Order

1. `typename T`

Class Documentation

template<typename **T**>
class EventModel : **public** camp::resources::v1::*Event*::*EventInterface*

Public Functions

EventModel(T **const** &*modelVal*)

bool **check**() **const**

void **wait**() **const**

T ***get**()

3.2.133 Class Host

- Defined in file_camp_resource_host.hpp

Page Contents

- *Class Documentation*

Class Documentation

class Host

Public Functions

Host(int = -1)

Platform **get_platform**()

HostEvent **get_event**()

Event **get_event_erased**()

void **wait**()

void **wait_for**(*Event* **e*)

template<typename **T**>
T ***allocate**(size_t *n*)

void ***calloc**(size_t *size*)

void **deallocate**(void **p*)

void **memcpy**(void **dst*, **const** void **src*, size_t *size*)

void **memset**(void **p*, int *val*, size_t *size*)

Public Static Functions

static *Host* &**get_default**()

3.2.134 Class HostEvent

- Defined in file_camp_resource_host.hpp

Page Contents

- *Class Documentation*

Class Documentation

class HostEvent

Public Functions

HostEvent()

bool **check**() **const**

void **wait**() **const**

3.2.135 Class Resource

- Defined in file_camp_resource.hpp

Page Contents

- *Nested Relationships*
  - *Nested Types*
- *Class Documentation*

Nested Relationships

Nested Types

- *Class Resource::ContextInterface*
- *Template Class Resource::ContextModel*

Class Documentation

class Resource

Public Functions

Resource(*Resource*&&)

Resource(*Resource* **const**&)

Resource &**operator=**(*Resource*&&)

Resource &**operator=**(*Resource* **const**&)

template<typename **T**, typename = **typename** std::enable_if<!std::is_same<**typename** std::decay<*T*>::type, *Resource*>::value
Resource(*T* &&*value*)

template<typename **T**>
T ***try_get**()

template<typename **T**>
T **get**()

Platform **get_platform**()

template<typename **T**>
T ***allocate**(size_t *size*)

void ***calloc**(size_t *size*)

void **deallocate**(void **p*)

void **memcpy**(void **dst*, **const** void **src*, size_t *size*)

void **memset**(void **p*, int *val*, size_t *size*)

Event **get_event**()

void **wait_for**(*Event* **e*)

3.2.136 Class Resource::ContextInterface

- Defined in file_camp_resource.hpp

Page Contents

- *Nested Relationships*
- *Class Documentation*

Nested Relationships

This class is a nested type of *Class Resource*.

Class Documentation

class ContextInterface

Public Functions

virtual ~ContextInterface()

virtual *Platform* **get_platform**() = 0

virtual void ***calloc**(size_t *size*) = 0

virtual void **deallocate**(void **p*) = 0

virtual void **memcpy**(void **dst*, **const** void **src*, size_t *size*) = 0

virtual void **memset**(void **p*, int *val*, size_t *size*) = 0

virtual *Event* **get_event**() = 0

virtual void **wait_for**(*Event* **e*) = 0

3.2.137 Template Class Resource::ContextModel

- Defined in file_camp_resource.hpp

Page Contents

- *Nested Relationships*
- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Class Documentation*

Nested Relationships

This class is a nested type of *Class Resource*.

Inheritance Relationships

Base Type

- `public camp::resources::v1::Resource::ContextInterface`

Template Parameter Order

1. `typename T`

Class Documentation

template<typename **T**>
class ContextModel : **public** camp::resources::v1::*Resource*::*ContextInterface*

Public Functions

ContextModel(T **const** &*modelVal*)

Platform **get_platform**()

void ***calloc**(size_t *size*)

void **deallocate**(void **p*)

void **memcpy**(void **dst*, **const** void **src*, size_t *size*)

void **memset**(void **p*, int *val*, size_t *size*)

Event **get_event**()

void **wait_for**(*Event* **e*)

T ***get**()

3.2.138 Template Class tagged_tuple

- Defined in file_camp_tuple.hpp

Page Contents

- *Nested Relationships*
  - *Nested Types*
- *Inheritance Relationships*
  - *Base Type*
- *Template Parameter Order*
- *Class Documentation*

Nested Relationships

Nested Types

- *Template Struct tagged_tuple::is_pack_this_tuple*
- *Template Struct tagged_tuple::is_pack_this_tuple< That >*

Inheritance Relationships

Base Type

- `public camp::tuple< Elements... >` (*Template Struct tuple*)

Template Parameter Order

1. `typename TagList`
2. `typename... Elements`

Class Documentation

template<typename **TagList**, typename ...**Elements**>
class tagged_tuple : **public** camp::tuple<*Elements*...>

Public Types

template<>
using TList = camp::*list*<Elements...>

template<>
using TMap = **typename** internal::tag_map<TagList, camp::*make_idx_seq_t*<sizeof...(Elements)>>::type

template<>
using type = tagged_tuple

Public Functions

template<bool B = concepts::metalib::all_of< std::is_default_constructi

constexpr tagged_tuple(*tagged_tuple* **const** &*o*)

constexpr tagged_tuple(*tagged_tuple* &&*o*)

tagged_tuple &**operator=**(tagged_tuple **const** &*rhs*)

tagged_tuple &**operator=**(tagged_tuple &&*rhs*)

template<typename... Args, typename std::enable_if< !is_pack_this_tuple< Args... >::va

template<template<typename...> class **T**, typename ...**RTypes**>
constexpr Self &**operator=**(**const** *T*<*RTypes*...> &*rhs*)

3.2.139 Template Class tuple<>

- Defined in file_camp_tuple.hpp

Page Contents

- *Class Documentation*

Class Documentation

template<>
class tuple<>

Public Types

template<>
using TList = camp::*list*<>

template<>
using TMap = *TList*

template<>
using type = tuple

3.3 Enums

3.3.1 Enum Platform

- Defined in file_camp_resource_platform.hpp

Enum Documentation

enum camp::resources::v1::**Platform**
Values:

undefined = 0

host = 1

cuda = 2

omp_target = 4

hip = 8

sycl = 16

3.4 Functions

3.4.1 Template Function ___valid_expr___

- Defined in file_camp_concepts.hpp

Function Documentation

template<typename ...**T**>
camp::*true_type* **___valid_expr___**(*T*&&...)

3.4.2 Function camp::CAMP_MAKE_L(accumulate)

- Defined in file_camp_camp.hpp

Function Documentation

`camp::`**`CAMP_MAKE_L`**(*accumulate*)

3.4.3 Function camp::CAMP_MAKE_L(filter)

- Defined in file_camp_camp.hpp

Function Documentation

`camp::`**`CAMP_MAKE_L`**(*filter*)

3.4.4 Function camp::CAMP_MAKE_L(bind_front)

- Defined in file_camp_lambda.hpp

Function Documentation

`camp::`**`CAMP_MAKE_L`**(*bind_front*)

3.4.5 Function camp::CAMP_MAKE_L(find_if)

- Defined in file_camp_list_find_if.hpp

Function Documentation

camp::**CAMP_MAKE_L**(*find_if*)

3.4.6 Template Function camp::concepts::convertible_to

- Defined in file_camp_concepts.hpp

Function Documentation

template<typename **T**, typename **U**>
constexpr auto camp::concepts::**convertible_to**(*U &&u*)
metafunction for use within decltype expression to validate return type is convertible to given type

3.4.7 Template Function camp::concepts::has_type

- Defined in file_camp_concepts.hpp

Function Documentation

template<typename **T**, typename **U**>
constexpr auto camp::concepts::**has_type**(*U&&*)
metafunction for use within decltype expression to validate type of expression

3.4.8 Template Function camp::concepts::is

- Defined in file_camp_concepts.hpp

Function Documentation

template<typename **BoolLike**>
constexpr auto camp::concepts::**is**(*BoolLike*)

3.4.9 Template Function camp::concepts::is_not

- Defined in file_camp_concepts.hpp

Function Documentation

template<typename **BoolLike**>
constexpr auto camp::concepts::**is_not**(*BoolLike*)

3.4.10 Template Function camp::cval

- Defined in file_camp_helpers.hpp

Function Documentation

template<typename **T**>
auto camp::**cval**()
metafunction to get instance of const type

3.4.11 Template Function camp::declptr

- Defined in file_camp_helpers.hpp

Function Documentation

template<typename **T**>
T *camp::**declptr**()
metafunction to get instance of pointer type

3.4.12 Template Function camp::forward(type::ref::rem<T>&)

- Defined in file_camp_helpers.hpp

Function Documentation

template<class **T**>
constexpr *T* &&camp::**forward**(type::ref::*rem*<*T*> &*t*)

3.4.13 Template Function camp::forward(type::ref::rem<T>&&)

- Defined in file_camp_helpers.hpp

Function Documentation

template<class **T**>
constexpr *T* &&camp::**forward**(type::ref::*rem*<*T*> &&*t*)

3.4.14 Template Function camp::forward_as_tuple

- Defined in file_camp_tuple.hpp

Function Documentation

template<typename ...**Args**>
constexpr auto camp::**forward_as_tuple**(*Args*&&... *args*)

3.4.15 Template Function camp::get(const Tuple&)

- Defined in file_camp_tuple.hpp

Function Documentation

template<camp::*idx_t* **index**, class **Tuple**>
constexpr auto camp::**get**(**const** *Tuple* &*t*)

3.4.16 Template Function camp::get(Tuple&)

- Defined in file_camp_tuple.hpp

Function Documentation

template<camp::*idx_t* **index**, class **Tuple**>
constexpr auto camp::**get**(*Tuple* &*t*)

3.4.17 Template Function camp::make_tagged_tuple

- Defined in file_camp_tuple.hpp

Function Documentation

template<typename **TagList**, typename ...**Args**>
constexpr auto camp::**make_tagged_tuple**(*Args*&&... *args*)

3.4.18 Template Function camp::make_tuple

- Defined in file_camp_tuple.hpp

Function Documentation

template<typename ...**Args**>
constexpr auto camp::**make_tuple**(*Args*&&... *args*)

3.4.19 Template Function camp::make_unique

- Defined in file_camp_make_unique.hpp

Function Documentation

template<typename **T**, typename ...**Args**>
constexpr std::unique_ptr<*T*> camp**::make_unique** (*Args*&&... *args*)

3.4.20 Template Function camp::move

- Defined in file_camp_helpers.hpp

Function Documentation

template<typename **T**>
constexpr type::ref::*rem*<*T*> &&camp**::move** (*T* &&*t*)

3.4.21 Template Function camp::safe_swap(T&, T&)

- Defined in file_camp_helpers.hpp

Function Documentation

template<typename **T**>
void camp**::safe_swap** (*T* &*t1*, *T* &*t2*)

3.4.22 Template Function camp::safe_swap(T&, T&)

- Defined in file_camp_helpers.hpp

Function Documentation

template<typename **T**>
void camp**::safe_swap** (*T* &*t1*, *T* &*t2*)

3.4.23 Template Function camp::sink

- Defined in file_camp_helpers.hpp

Function Documentation

template<typename ...**Ts**>
void `camp::sink`(*Ts* **const**&...)
metafunction to expand a parameter pack and ignore result

3.4.24 Template Function camp::tie

- Defined in file_camp_tuple.hpp

Function Documentation

template<class ...**Types**>
constexpr tuple<*Types*&...> `camp::tie`(*Types*&... *args*)

3.4.25 Template Function camp::tuple_cat_pair(tuple<Lelem...> const&, camp::idx_seq<Lidx...>, tuple<Relem...> const&, camp::idx_seq<Ridx...>)

- Defined in file_camp_tuple.hpp

Function Documentation

template<typename ...**Lelem**, typename ...**Relem**, camp::*idx_t*... **Lidx**, camp::*idx_t*... **Ridx**>
constexpr auto `camp::tuple_cat_pair`(tuple<*Lelem*...> **const** &*l*, camp::*idx_seq*<*Lidx*...>, tuple<*Relem*...> **const** &*r*, camp::*idx_seq*<*Ridx*...>)

3.4.26 Template Function camp::tuple_cat_pair(L const&, R const&)

- Defined in file_camp_tuple.hpp

Function Documentation

template<typename **L**, typename **R**>
constexpr auto `camp::tuple_cat_pair`(*L* **const** &*l*, *R* **const** &*r*)

3.4.27 Template Function camp::val

- Defined in file_camp_helpers.hpp

Function Documentation

template<typename **T**>
auto `camp::val`()
metafunction to get instance of value type

3.4.28 Template Function operator<<

- Defined in file_camp_tuple.hpp

Function Documentation

template<class ...**Args**>
auto **operator<<**(std::ostream &*os*, camp::tuple<*Args*...> **const** &*t*)

3.5 Variables

3.5.1 Variable camp::r

- Defined in file_camp_tuple.hpp

Variable Documentation

camp::r

3.6 Defines

3.6.1 Define CAMP_ALLOW_UNUSED_LOCAL

- Defined in file_camp_defines.hpp

Define Documentation

CAMP_ALLOW_UNUSED_LOCAL(*X*)

3.6.2 Define CAMP_CONSTEXPR14

- Defined in file_camp_defines.hpp

Define Documentation

CAMP_CONSTEXPR14

3.6.3 Define CAMP_DEVICE

- Defined in file_camp_defines.hpp

Define Documentation

CAMP_DEVICE

3.6.4 Define CAMP_HIP_HOST_DEVICE

- Defined in file_camp_defines.hpp

Define Documentation

CAMP_HIP_HOST_DEVICE

3.6.5 Define CAMP_HOST_DEVICE

- Defined in file_camp_defines.hpp

Define Documentation

CAMP_HOST_DEVICE

3.6.6 Define CAMP_MAKE_L

- Defined in file_camp_defines.hpp

Define Documentation

CAMP_MAKE_L(*X*)

3.6.7 Define CAMP_SUPPRESS_HD_WARN

- Defined in file_camp_defines.hpp

Define Documentation

CAMP_SUPPRESS_HD_WARN

3.6.8 Define DefineConcept

- Defined in file_camp_concepts.hpp

Define Documentation

DefineConcept(...)

3.6.9 Define DefineTypeTraitFromConcept

- Defined in file_camp_concepts.hpp

Define Documentation

DefineTypeTraitFromConcept(*TTName*, *ConceptName*)

3.7 Typedefs

3.7.1 Typedef camp::_1

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_1** = *arg*<1>

3.7.2 Typedef camp::_2

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_2** = *arg*<2>

3.7.3 Typedef camp::_3

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_3** = *arg*<3>

3.7.4 Typedef camp::_4

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_4** = *arg*<4>

3.7.5 Typedef camp::_5

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_5** = *arg*<5>

3.7.6 Typedef camp::_6

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_6** = *arg*<6>

3.7.7 Typedef camp::_7

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_7** = *arg*<7>

3.7.8 Typedef camp::_8

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_8** = *arg*<8>

3.7.9 Typedef camp::_9

- Defined in file_camp_lambda.hpp

Typedef Documentation

using camp::**_9** = *arg*<9>

3.7.10 Typedef camp::as_list

- Defined in file_camp_list_list.hpp

Typedef Documentation

using camp::**as_list** = **typename** *as_list_s*<T>::type

3.7.11 Typedef camp::at_key

- Defined in file_camp_map.hpp

Typedef Documentation

using camp::**at_key** = **typename** *at_key_s*<Map, Key>::type
Get value at Key from Map.

Template Parameters

- Map: The map, or associative list, to index
- Key: The key to find

3.7.12 Typedef camp::at_t

- Defined in file_camp_list_at.hpp

Typedef Documentation

using camp::**at_t** = **typename** *at*<T, U>::type

3.7.13 Typedef camp::at_v

- Defined in file_camp_list_at.hpp

Typedef Documentation

using camp::**at_v** = **typename** *at*<T, *num*<Idx>>::type

3.7.14 Typedef camp::cartesian_product

- Defined in file_camp_camp.hpp

Typedef Documentation

using camp::**cartesian_product** = **typename** *accumulate*<detail::product, *list*<*list*<>>, *list*<Seqs...>>::type

3.7.15 Typedef camp::concepts::enable_if

- Defined in file_camp_concepts.hpp

Typedef Documentation

using camp::concepts::**enable_if** = **typename** std::enable_if<*all_of*<Args...>::value, void>::type
SFINAE multiple type traits.

3.7.16 Typedef camp::concepts::negate

- Defined in file_camp_concepts.hpp

Typedef Documentation

using camp::concepts::**negate** = metalib::*negate_t*<T>

3.7.17 Typedef camp::decay

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::**decay** = type::cv::*rem*<type::ref::*rem*<T>>

3.7.18 Typedef camp::diff_between

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::**diff_between** = decltype(*val*<*plain*<T>>() - *val*<*plain*<U>>())

3.7.19 Typedef camp::diff_from

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::**diff_from** = decltype(*val*<*plain*<T>>() - *val*<*plain*<T>>())

3.7.20 Typedef camp::eval

- Defined in file_camp_value_eval.hpp

Typedef Documentation

using camp::**eval** = **typename** Val::type

3.7.21 Typedef camp::false_type

- Defined in file_camp_number_number.hpp

Typedef Documentation

using camp::**false_type** = *num*<false>

3.7.22 Typedef camp::first

- Defined in file_camp_list_at.hpp

Typedef Documentation

using camp::**first** = **typename** *at*<T, *num*<0>>::type

3.7.23 Typedef camp::idx_seq

- Defined in file_camp_number.hpp

Typedef Documentation

using camp::**idx_seq** = *int_seq*<*idx_t*, vs...>
Index list, use for indexing into parameter packs and lists.

3.7.24 Typedef camp::idx_seq_for_t

- Defined in file_camp_number.hpp

Typedef Documentation

using camp::**idx_seq_for_t** = **typename** *make_idx_seq*<sizeof...(Ts)>::type

3.7.25 Typedef camp::idx_seq_from_t

- Defined in file_camp_number.hpp

Typedef Documentation

using camp::**idx_seq_from_t** = **typename** *idx_seq_from*<camp::*decay*<T>>::type

3.7.26 Typedef camp::idx_t

- Defined in file_camp_defines.hpp

Typedef Documentation

using camp::**idx_t** = std::ptrdiff_t

3.7.27 Typedef camp::if_

- Defined in file_camp_number_if.hpp

Typedef Documentation

using camp::**if_** = **typename** *if_s*<Ts...>::type

3.7.28 Typedef camp::if_c

- Defined in file_camp_number_if.hpp

Typedef Documentation

using camp::**if_c** = **typename** if_cs<Cond, Then, Else>::*type*

3.7.29 Typedef camp::is_same

- Defined in file_camp_type_traits_is_same.hpp

Typedef Documentation

using camp::**is_same** = **typename** *is_same_s*<T, U>::type

3.7.30 Typedef camp::is_same_t

- Defined in file_camp_type_traits_is_same.hpp

Typedef Documentation

using camp::**is_same_t** = *is_same*<T, U>

3.7.31 Typedef camp::is_tuple

- Defined in file_camp_tuple.hpp

Typedef Documentation

using camp::**is_tuple** = **typename** std::is_base_of<*tuple*<>, Tup<>>::type

3.7.32 Typedef camp::is_value

- Defined in file_camp_value.hpp

Typedef Documentation

using camp::**is_value** = **typename** is_value_s<Val>::*type*
Test whether a type is a valid camp value.

3.7.33 Typedef camp::iterator_from

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::**iterator_from** = decltype(begin(*val*<*plain*<T>>()))

3.7.34 Typedef camp::make_idx_seq_t

- Defined in file_camp_number.hpp

Typedef Documentation

using camp::**make_idx_seq_t** = **typename** make_idx_seq<Upper>::*type*

3.7.35 Typedef camp::make_int_seq_t

- Defined in file_camp_number.hpp

Typedef Documentation

using camp::**make_int_seq_t** = **typename** *make_int_seq*<T, Upper>::type

3.7.36 Typedef camp::nil

- Defined in file_camp_value.hpp

Typedef Documentation

using camp::**nil** = *value*<>
A non-value, in truth tests evaluates to false.

3.7.37 Typedef camp::nullptr_t

- Defined in file_camp_defines.hpp

Typedef Documentation

using camp::**nullptr_t** = decltype(nullptr)

3.7.38 Typedef camp::num

- Defined in file_camp_number_number.hpp

Typedef Documentation

using camp::**num** = integral_constant<*idx_t*, N>
Short-form for a whole number.

Template Parameters

- N: The integral value

3.7.39 Typedef camp::plain

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::**plain** = type::ref::*rem*<T>

3.7.40 Typedef camp::second

- Defined in file_camp_list_at.hpp

Typedef Documentation

using camp::**second** = **typename** *at*<T, *num*<1>>::type

3.7.41 Typedef camp::t

- Defined in file_camp_number_number.hpp

Typedef Documentation

using camp::**t** = *num*<true>

3.7.42 Typedef camp::true_type

- Defined in file_camp_number_number.hpp

Typedef Documentation

using camp::**true_type** = *num*<true>

3.7.43 Typedef camp::tuple_ebt_t

- Defined in file_camp_tuple.hpp

Typedef Documentation

using camp::**tuple_ebt_t** = **typename** *tuple_element*<camp::*at_key*<**typename** Tuple::TMap, T>::value, Tuple>::type

3.7.44 Typedef camp::tuple_element_t

- Defined in file_camp_tuple.hpp

Typedef Documentation

using camp::**tuple_element_t** = **typename** tuple_element<i, T>::*type*

3.7.45 Typedef camp::type::c::add

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::c::**add** = **const** T
add const qualifier to T

3.7.46 Typedef camp::type::c::rem

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::c::**rem** = **typename** rem_s<T>::*type*
remove const qualifier from T

3.7.47 Typedef camp::type::cv::add

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::cv::**add** = **volatile const** T
add const and volatile qualifiers to T

3.7.48 Typedef camp::type::cv::rem

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::cv::**rem** = **typename** rem_s<T>::*type*
remove const and volatile qualifiers from T

3.7.49 Typedef camp::type::ref::add

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::ref::**add** = T&
add remove reference to T

3.7.50 Typedef camp::type::ref::rem

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::ref::**rem** = **typename** rem_s<T>::*type*
remove reference from T

3.7.51 Typedef camp::type::rvref::add

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::rvref::**add** = T&&
add rvalue reference to T

3.7.52 Typedef camp::type::v::add

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::v::**add** = **volatile** T
add volatile qualifier to T

3.7.53 Typedef camp::type::v::rem

- Defined in file_camp_helpers.hpp

Typedef Documentation

using camp::type::v::**rem** = **typename** rem_s<T>::*type*
remove volatile qualifier from T

3.7.54 Typedef camp::type_traits::IsSpecialized

- Defined in file_camp_concepts.hpp

Typedef Documentation

using camp::type_traits::**IsSpecialized** = detail::IsSpecialized<void, Outer, Args...>

3.7.55 Typedef camp::type_traits::IterableValue

- Defined in file_camp_concepts.hpp

Typedef Documentation

using camp::type_traits::**IterableValue** = decltype(*std::begin(camp::*val*<T>()))

3.7.56 Typedef camp::type_traits::IteratorValue

- Defined in file_camp_concepts.hpp

Typedef Documentation

using camp::type_traits::**IteratorValue** = decltype(*camp::*val*<T>())

INDEX

Symbols

___valid_expr___ (*C++ function*), 108

C

camp::_1 (*C++ type*), 116
camp::_2 (*C++ type*), 116
camp::_3 (*C++ type*), 117
camp::_4 (*C++ type*), 117
camp::_5 (*C++ type*), 117
camp::_6 (*C++ type*), 117
camp::_7 (*C++ type*), 117
camp::_8 (*C++ type*), 118
camp::_9 (*C++ type*), 118
camp::accumulate (*C++ struct*), 18
camp::accumulate<Op, Initial, list<Elements...>> (*C++ struct*), 19
camp::accumulate<Op, Initial, list<Elements...>>::type (*C++ type*), 19
camp::append (*C++ struct*), 19
camp::append<list<Elements...>, T> (*C++ struct*), 20
camp::append<list<Elements...>, T>::type (*C++ type*), 20
camp::apply_l (*C++ struct*), 20
camp::apply_l<Lambda, list<Args...>> (*C++ struct*), 21
camp::apply_l<Lambda, list<Args...>>::type (*C++ type*), 21
camp::arg (*C++ struct*), 21
camp::arg<n>::expr (*C++ type*), 21
camp::as_list (*C++ type*), 118
camp::as_list_s (*C++ struct*), 22
camp::as_list_s<tagged_tuple<camp::list<Tags...>, Args...>> (*C++ struct*), 23
camp::as_list_s<tagged_tuple<camp::list<Tags...>, Args...>>::type (*C++ type*), 23
camp::at (*C++ struct*), 23
camp::at_key (*C++ type*), 118
camp::at_key_s (*C++ struct*), 24
camp::at_key_s<Seq, Key>::type (*C++ type*), 24
camp::at_t (*C++ type*), 119
camp::at_v (*C++ type*), 119
camp::at<T, num<Val>> (*C++ struct*), 24
camp::at<T, num<Val>>::type (*C++ type*), 24
camp::bind (*C++ struct*), 25
camp::bind_front (*C++ struct*), 26
camp::bind_front<Expr, BoundArgs>::expr (*C++ type*), 26
camp::bind_front<Expr, BoundArgs>::type (*C++ type*), 26
camp::bind<Expr, ArgBindings>::bindings (*C++ type*), 25
camp::bind<Expr, ArgBindings>::expr (*C++ type*), 25
camp::bind<Expr, ArgBindings>::type (*C++ type*), 25
camp::CAMP_MAKE_L (*C++ function*), 108, 109
camp::cartesian_product (*C++ type*), 119
camp::concepts::all_of (*C++ struct*), 27
camp::concepts::any_of (*C++ struct*), 27
camp::concepts::Comparable (*C++ struct*), 30
camp::concepts::convertible_to (*C++ function*), 109
camp::concepts::enable_if (*C++ type*), 119
camp::concepts::has_type (*C++ function*), 109
camp::concepts::is (*C++ function*), 109
camp::concepts::is_not (*C++ function*), 109
camp::concepts::metalib::all_of (*C++ struct*), 38
camp::concepts::metalib::all_of_t (*C++ struct*), 38
camp::concepts::metalib::any_of (*C++ struct*), 39
camp::concepts::metalib::any_of_t (*C++ struct*), 40
camp::concepts::metalib::negate_t (*C++ struct*), 40
camp::concepts::metalib::none_of (*C++ struct*), 41
camp::concepts::metalib::none_of_t (*C++

struct), 42
camp::concepts::negate (*C++ type*), 119
camp::concepts::none_of (*C++ struct*), 42
camp::concepts::requires_ (*C++ struct*), 45
camp::cval (*C++ function*), 110
camp::decay (*C++ type*), 120
camp::declptr (*C++ function*), 110
camp::diff_between (*C++ type*), 120
camp::diff_from (*C++ type*), 120
camp::eval (*C++ type*), 120
camp::extend (*C++ struct*), 47
camp::extend<list<Elements...>, list<NewElements...>> (*C++ struct*), 48
camp::extend<list<Elements...>, list<NewElements...>>::type (*C++ type*), 48
camp::false_type (*C++ type*), 120
camp::filter (*C++ struct*), 49
camp::filter<Op, list<Elements...>> (*C++ struct*), 49
camp::filter<Op, list<Elements...>>::append_if (*C++ type*), 49
camp::filter<Op, list<Elements...>>::type (*C++ type*), 49
camp::find_if (*C++ struct*), 50
camp::find_if<Cond, list<Elements...>> (*C++ struct*), 50
camp::find_if<Cond, list<Elements...>>::type (*C++ type*), 50
camp::first (*C++ type*), 121
camp::flatten (*C++ struct*), 51
camp::flatten<list<Elements...>> (*C++ struct*), 51
camp::forward (*C++ function*), 110
camp::forward_as_tuple (*C++ function*), 111
camp::get (*C++ function*), 111
camp::idx_seq (*C++ type*), 121
camp::idx_seq_for_t (*C++ type*), 121
camp::idx_seq_from (*C++ struct*), 52
camp::idx_seq_from_t (*C++ type*), 121
camp::idx_seq_from<int_seq<T, Args...>> (*C++ struct*), 52
camp::idx_seq_from<T<Args...>> (*C++ struct*), 53
camp::idx_t (*C++ type*), 121
camp::if_ (*C++ type*), 122
camp::if_c (*C++ type*), 122
camp::if_cs (*C++ struct*), 53
camp::if_cs<Cond, Then, Else>::type (*C++ type*), 54
camp::if_cs<false, Then, Else> (*C++ struct*), 54
camp::if_cs<false, Then, Else>::type (*C++ type*), 54
camp::if_s (*C++ struct*), 55
camp::if_s<nil, Then, Else> (*C++ struct*), 56
camp::index_of (*C++ struct*), 56
camp::index_of<T, list<Elements...>> (*C++ struct*), 57
camp::index_of<T, list<Elements...>>::inc_until (*C++ type*), 57
camp::index_of<T, list<Elements...>>::indices (*C++ type*), 57
camp::index_of<T, list<Elements...>>::type (*C++ type*), 57
camp::int_seq (*C++ struct*), 57
camp::int_seq<T, vs>::type (*C++ type*), 58
camp::integral_constant (*C++ struct*), 58
camp::integral_constant::operator value_type (*C++ function*), 59
camp::integral_constant::operator() (*C++ function*), 59
camp::integral_constant::value (*C++ member*), 59
camp::integral_constant<NumT, v>::type (*C++ type*), 59
camp::integral_constant<NumT, v>::value_type (*C++ type*), 59
camp::invoke_l (*C++ struct*), 59
camp::invoke_l<Lambda, Args>::type (*C++ type*), 59
camp::is_same (*C++ type*), 122
camp::is_same_s (*C++ struct*), 60
camp::is_same_s<T, T> (*C++ struct*), 61
camp::is_same_t (*C++ type*), 122
camp::is_tuple (*C++ type*), 122
camp::is_value (*C++ type*), 123
camp::is_value_s (*C++ struct*), 61
camp::is_value_s<Val>::type (*C++ type*), 61
camp::iterator_from (*C++ type*), 123
camp::join (*C++ struct*), 62
camp::join<> (*C++ struct*), 63
camp::join<>::type (*C++ type*), 64
camp::join<Seq1, Seq2, Rest...> (*C++ struct*), 63
camp::join<Seq1, Seq2, Rest...>::type (*C++ type*), 63
camp::join<Seq1> (*C++ struct*), 62
camp::join<Seq1>::type (*C++ type*), 62
camp::lambda (*C++ struct*), 64
camp::lambda<Expr>::expr (*C++ type*), 64
camp::list (*C++ struct*), 65

camp::list<Ts>::type (C++ type), 65
camp::make_idx_seq (C++ struct), 65
camp::make_idx_seq_t (C++ type), 123
camp::make_idx_seq<Upper>::type (C++ type), 65
camp::make_int_seq (C++ struct), 66
camp::make_int_seq_t (C++ type), 123
camp::make_tagged_tuple (C++ function), 111
camp::make_tuple (C++ function), 111
camp::make_unique (C++ function), 112
camp::move (C++ function), 112
camp::nil (C++ type), 123
camp::not_ (C++ struct), 66
camp::not_<T>::type (C++ type), 67
camp::nullptr_t (C++ type), 124
camp::num (C++ type), 124
camp::plain (C++ type), 124
camp::prepend (C++ struct), 67
camp::prepend<list<Elements...>, T> (C++ struct), 68
camp::prepend<list<Elements...>, T>::type (C++ type), 68
camp::resources::v1::cuda (C++ enumerator), 107
camp::resources::v1::Event (C++ class), 99
camp::resources::v1::Event::check (C++ function), 99
camp::resources::v1::Event::Event (C++ function), 99
camp::resources::v1::Event::EventInterface (C++ class), 100
camp::resources::v1::Event::EventInterface::~EventInterface (C++ function), 100
camp::resources::v1::Event::EventInterface::check (C++ function), 100
camp::resources::v1::Event::EventInterface::wait (C++ function), 100
camp::resources::v1::Event::EventModel (C++ class), 101
camp::resources::v1::Event::EventModel::check (C++ function), 101
camp::resources::v1::Event::EventModel::EventModel (C++ function), 101
camp::resources::v1::Event::EventModel::get (C++ function), 101
camp::resources::v1::Event::EventModel::wait (C++ function), 101
camp::resources::v1::Event::get (C++ function), 99
camp::resources::v1::Event::try_get (C++ function), 99
camp::resources::v1::Event::wait (C++ function), 99
camp::resources::v1::hip (C++ enumerator), 107
camp::resources::v1::Host (C++ class), 101
camp::resources::v1::host (C++ enumerator), 107
camp::resources::v1::Host::allocate (C++ function), 101
camp::resources::v1::Host::calloc (C++ function), 101
camp::resources::v1::Host::deallocate (C++ function), 101
camp::resources::v1::Host::get_default (C++ function), 102
camp::resources::v1::Host::get_event (C++ function), 101
camp::resources::v1::Host::get_event_erased (C++ function), 101
camp::resources::v1::Host::get_platform (C++ function), 101
camp::resources::v1::Host::Host (C++ function), 101
camp::resources::v1::Host::memcpy (C++ function), 101
camp::resources::v1::Host::memset (C++ function), 101
camp::resources::v1::Host::wait (C++ function), 101
camp::resources::v1::Host::wait_for (C++ function), 101
camp::resources::v1::HostEvent (C++ class), 102
camp::resources::v1::HostEvent::check (C++ function), 102
camp::resources::v1::HostEvent::HostEvent (C++ function), 102
camp::resources::v1::HostEvent::wait (C++ function), 102
camp::resources::v1::omp_target (C++ enumerator), 107
camp::resources::v1::Platform (C++ enum), 107
camp::resources::v1::Resource (C++ class), 103
camp::resources::v1::Resource::allocate (C++ function), 103
camp::resources::v1::Resource::calloc (C++ function), 103
camp::resources::v1::Resource::ContextInterface (C++ class), 104
camp::resources::v1::Resource::ContextInterface::~C (C++ function), 104
camp::resources::v1::Resource::ContextInterface::ca (C++ function), 104
camp::resources::v1::Resource::ContextInterface::de (C++ function), 104

camp::resources::v1::Resource::ContextInterface::get_event (C++ function), 104
camp::resources::v1::Resource::ContextInterface::get_platform (C++ function), 104
camp::resources::v1::Resource::ContextInterface::memcpy (C++ function), 104
camp::resources::v1::Resource::ContextInterface::memset (C++ function), 104
camp::resources::v1::Resource::ContextInterface::wait (C++ function), 104
camp::resources::v1::Resource::ContextModel (C++ class), 105
camp::resources::v1::Resource::ContextModel::allocate (C++ function), 105
camp::resources::v1::Resource::ContextModel::construct (C++ function), 105
camp::resources::v1::Resource::ContextModel::deallocate (C++ function), 105
camp::resources::v1::Resource::ContextModel::get (C++ function), 105
camp::resources::v1::Resource::ContextModel::get_event (C++ function), 105
camp::resources::v1::Resource::ContextModel::get_platform (C++ function), 105
camp::resources::v1::Resource::ContextModel::memcpy (C++ function), 105
camp::resources::v1::Resource::ContextModel::memset (C++ function), 105
camp::resources::v1::Resource::ContextModel::wait (C++ function), 105
camp::resources::v1::Resource::deallocate (C++ function), 103
camp::resources::v1::Resource::get (C++ function), 103
camp::resources::v1::Resource::get_event (C++ function), 103
camp::resources::v1::Resource::get_platform (C++ function), 103
camp::resources::v1::Resource::memcpy (C++ function), 103
camp::resources::v1::Resource::memset (C++ function), 103
camp::resources::v1::Resource::operator= (C++ function), 103
camp::resources::v1::Resource::Resource (C++ function), 103
camp::resources::v1::Resource::try_get (C++ function), 103
camp::resources::v1::Resource::wait_for (C++ function), 103
camp::resources::v1::resource_from_platform (C++ struct), 68
camp::resources::v1::resource_from_platform<Platform> (C++ struct), 68
camp::resources::v1::resource_from_platform<Platform::undefined> (C++ type), 69
camp::resources::v1::resource_from_platform<Platform::sycl> (C++ enumerator), 107
camp::resources::v1::undefined (C++ enumerator), 107
camp::safe_mod (C++ function), 112
camp::second (C++ type), 124
camp::seq_at (C++ struct), 69
camp::seq_at::value (C++ member), 70
camp::seq_at<0, camp::int_seq<T, Idx0, IdxRest...>> (C++ struct), 70
camp::seq_at<N, camp::int_seq<T, Idx0, IdxRest...>> (C++ struct), 70
camp::sink (C++ function), 113
camp::size (C++ struct), 71
camp::size::value (C++ member), 71, 72
camp::size<int_seq<T, Args...>> (C++ struct), 71
camp::size<int_seq<T, Args...>>::type (C++ type), 71
camp::size<list<Args...>> (C++ struct), 72
camp::size<list<Args...>>::type (C++ type), 72
camp::sum (C++ type), 125
camp::tagged_tuple (C++ class), 106
camp::tagged_tuple::is_pack_this_tuple (C++ struct), 73
camp::tagged_tuple::is_pack_this_tuple<That> (C++ struct), 74
camp::tagged_tuple::operator= (C++ function), 106
camp::tagged_tuple::tagged_tuple (C++ function), 106
camp::tagged_tuple<TagList, Elements>::TList (C++ type), 106
camp::tagged_tuple<TagList, Elements>::TMap (C++ type), 106
camp::tagged_tuple<TagList, Elements>::type (C++ type), 106
camp::tie (C++ function), 113
camp::transform (C++ struct), 74
camp::transform<Op, list<Elements...>> (C++ struct), 75
camp::transform<Op, list<Elements...>>::type (C++ type), 75
camp::true_type (C++ type), 125
camp::tuple (C++ struct), 76
camp::tuple::is_pack_this_tuple (C++ struct), 77
camp::tuple::is_pack_this_tuple<That> (C++ struct), 78
camp::tuple::operator= (C++ function), 76

camp::tuple::tuple (C++ function), 76
camp::tuple_cat_pair (C++ function), 113
camp::tuple_ebt_t (C++ type), 125
camp::tuple_element (C++ struct), 78
camp::tuple_element_t (C++ type), 125
camp::tuple_element<i, T>::type (C++ type), 78
camp::tuple_size (C++ struct), 79
camp::tuple_size::value (C++ member), 79–81
camp::tuple_size<tagged_tuple<L, Args...>&> (C++ struct), 79
camp::tuple_size<tagged_tuple<L, Args...>> (C++ struct), 80
camp::tuple_size<tuple<Args...>&> (C++ struct), 80
camp::tuple_size<tuple<Args...>> (C++ struct), 81
camp::tuple<> (C++ class), 107
camp::tuple<>::TList (C++ type), 107
camp::tuple<>::TMap (C++ type), 107
camp::tuple<>::type (C++ type), 107
camp::tuple<Elements>::TList (C++ type), 76
camp::tuple<Elements>::TMap (C++ type), 76
camp::tuple<Elements>::type (C++ type), 76
camp::type::c::add (C++ type), 125
camp::type::c::rem (C++ type), 126
camp::type::c::rem_s (C++ struct), 81
camp::type::c::rem_s<const T> (C++ struct), 82
camp::type::c::rem_s<const T>::type (C++ type), 82
camp::type::c::rem_s<T>::type (C++ type), 81
camp::type::cv::add (C++ type), 126
camp::type::cv::rem (C++ type), 126
camp::type::cv::rem_s (C++ struct), 82
camp::type::cv::rem_s<const T> (C++ struct), 83
camp::type::cv::rem_s<const T>::type (C++ type), 83
camp::type::cv::rem_s<T>::type (C++ type), 82
camp::type::cv::rem_s<volatile const T> (C++ struct), 83
camp::type::cv::rem_s<volatile const T>::type (C++ type), 83
camp::type::cv::rem_s<volatile T> (C++ struct), 84
camp::type::cv::rem_s<volatile T>::type (C++ type), 84
camp::type::ref::add (C++ type), 126
camp::type::ref::rem (C++ type), 126
camp::type::ref::rem_s (C++ struct), 84
camp::type::ref::rem_s<T&&> (C++ struct), 85
camp::type::ref::rem_s<T&&>::type (C++ type), 85
camp::type::ref::rem_s<T&> (C++ struct), 85
camp::type::ref::rem_s<T&>::type (C++ type), 85
camp::type::ref::rem_s<T>::type (C++ type), 84
camp::type::rvref::add (C++ type), 127
camp::type::v::add (C++ type), 127
camp::type::v::rem (C++ type), 127
camp::type::v::rem_s (C++ struct), 86
camp::type::v::rem_s<T>::type (C++ type), 86
camp::type::v::rem_s<volatile T> (C++ struct), 86
camp::type::v::rem_s<volatile T>::type (C++ type), 86
camp::type_traits::is_arithmetic (C++ struct), 87
camp::type_traits::is_bidirectional_iterator (C++ struct), 88
camp::type_traits::is_bidirectional_range (C++ struct), 88
camp::type_traits::is_comparable (C++ struct), 89
camp::type_traits::is_comparable_to (C++ struct), 90
camp::type_traits::is_floating_point (C++ struct), 90
camp::type_traits::is_forward_iterator (C++ struct), 91
camp::type_traits::is_forward_range (C++ struct), 92
camp::type_traits::is_integral (C++ struct), 92
camp::type_traits::is_iterator (C++ struct), 93
camp::type_traits::is_random_access_iterator (C++ struct), 94
camp::type_traits::is_random_access_range (C++ struct), 94
camp::type_traits::is_range (C++ struct), 95
camp::type_traits::is_signed (C++ struct), 96
camp::type_traits::is_unsigned (C++ struct), 96
camp::type_traits::IsSpecialized (C++ type), 127
camp::type_traits::IterableValue (C++ type), 128
camp::type_traits::IteratorValue (C++

type), 128
camp::type_traits::SpecializationOf (*C++ struct*), 97
camp::type_traits::SpecializationOf<Expected, Actual<Args...>> (*C++ struct*), 98
camp::val (*C++ function*), 114
camp::value (*C++ struct*), 98
camp::value<val>::type (*C++ type*), 98
CAMP_ALLOW_UNUSED_LOCAL (*C macro*), 114
CAMP_CONSTEXPR14 (*C macro*), 115
CAMP_DEVICE (*C macro*), 115
CAMP_HIP_HOST_DEVICE (*C macro*), 115
CAMP_HOST_DEVICE (*C macro*), 115
CAMP_MAKE_L (*C macro*), 115
CAMP_SUPPRESS_HD_WARN (*C macro*), 116

D

DefineConcept (*C macro*), 116
DefineTypeTraitFromConcept (*C macro*), 116

O

operator<< (*C++ function*), 114