

Git Ref: master
Date: 2020-03-03
Revises: master
Reply at: <https://github.com/johelegp/jegp/issues>

JEGP Library

Contents

1	Scope	1
2	References	2
3	Introduction	3
3.1	General	3
3.2	Library-wide requirements	3
4	General utilities library	4
4.1	General	4
4.2	Utility components	4
	Cross references	6
	Index	7
	Index of library headers	8
	Index of library names	9

1 Scope

[scope]

¹ This document describes the contents of the *JEGP library*.

2 References

[refs]

¹ The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document.

(1.1) — ISO/IEC 14882:2020, *Programming Languages — C++*

² ISO/IEC 14882 is herein called the *C++ Standard*.

3 Introduction

[intro]

3.1 General

[intro.general]

- ¹ The library specification subsumes the C++ Standard's [library], assumingly amended to the context of this library. [*Example*:
- (1.1) — Per C++ Standard's [namespace.future], `::jpeg2` is reserved.
 - (1.2) — Per C++ Standard's [contents]#3, a name `x` means `::jpeg::x`.
- *end example*] The following subclauses describe additions to it.

Table 1: Library categories [tab:library.categories]

Clause	Category
Clause 4	General utilities library

3.2 Library-wide requirements

[requirements]

3.2.1 Library contents

[contents]

- ¹ Whenever a name is qualified with `X::`, `::X::` is meant. [*Example*: When `std::Y` is mentioned, `::std::Y` is meant. — *end example*]

3.2.2 Reserved names

[reserved.names]

- ¹ The JPEG library reserves macro names starting with `JEGP_`.

4 General utilities library

[utilities]

4.1 General

[utilities.general]

- ¹ This clause describes generally useful utilities. These utilities are summarized in [Table 2](#).

Table 2: General utilities library summary [tab:utilities.summary]

Subclause	Header
4.2 Utility components	<jegp/utility.hpp>

4.2 Utility components

[utility]

4.2.1 Header <jegp/utility.hpp> synopsis

[utility.syn]

- ¹ The header <jegp/utility.hpp> contains some basic constructs.

```

namespace jegp
{
    // 4.2.2, underlying
    template <class Enum>
    constexpr std::underlying_type_t<Enum> underlying(Enum e) noexcept;

    // 4.2.3, static_downcast
    template <class DerivedRef, class Base>
    constexpr DerivedRef static_downcast(Base&& base) noexcept;

    // 4.2.4, hash_combine
    template <class... Args>
    constexpr std::size_t hash_combine(const Args&... args) noexcept(see below);
} // namespace jegp

```

4.2.2 underlying

[utility.underlying]

```

template <class Enum>
constexpr std::underlying_type_t<Enum> underlying(Enum e) noexcept;

```

- ¹ *Constraints:* `std::is_enum_v<Enum>` is true.
- ² *Returns:* `static_cast<std::underlying_type_t<Enum>>(e)`.

4.2.3 static_downcast

[static.downcast]

- ¹ A `static_cast` that performs a downcast.

```

template <class DerivedRef, class Base>
constexpr DerivedRef static_downcast(Base&& base) noexcept;

2 Let derived-ref be static_cast<DerivedRef>(std::forward<Base>(base)).

```

- ³ *Constraints:*

- (3.1) — `DerivedRef` is a reference type,
- (3.2) — `std::remove_cvref_t<DerivedRef>` is derived from `std::remove_cvref_t<Base>`, and
- (3.3) — *derived-ref* is well-formed.

- ⁴ *Preconditions:* *derived-ref* has well-defined behavior.

- ⁵ *Returns:* *derived-ref*.

4.2.4 hash_combine**[hash.combine]**

- ¹ Inspired by Boost.ContainerHash. Useful in the specializations of `std::hash` whose Key's salient parts consist of two or more objects.

```
template <class... Args>
constexpr std::size_t hash_combine(const Args&... args) noexcept(see below);
```

² *Constraints:*

- (2.1) — `sizeof...(Args) ≥ 2` and
- (2.2) — `std::hash<T>` is enabled (C++ Standard's [unord.hash]) for all T in Args.

³ *Effects:* Equivalent to:

```
std::size_t seed{0};
return (... , (seed ^= std::hash<Args>{}(args) + (seed << 6) + (seed >> 2)));
```

⁴ *Remarks:* The expression inside `noexcept` is equivalent to

```
(noexcept(std::hash<Args>{}(args)) && ...)
```

Cross references

This annex lists each clause or subclause label and the corresponding clause or subclause number and page number, in alphabetical order by label.

contents ([3.2.1](#)) 3

hash.combine ([4.2.4](#)) 5

intro ([Clause 3](#)) 3

intro.general ([3.1](#)) 3

refs ([Clause 2](#)) 2

requirements ([3.2](#)) 3

reserved.names ([3.2.2](#)) 3

scope ([Clause 1](#)) 1

static.downcast ([4.2.3](#)) 4

utilities ([Clause 4](#)) 4

utilities.general ([4.1](#)) 4

utility ([4.2](#)) 4

utility.syn ([4.2.1](#)) 4

utility.underlying ([4.2.2](#)) 4

Index

C

C++ Standard, [2](#)

J

JPEG library, [1](#)

<jpeg/utility.hpp>, [4](#)

Index of library headers

The bold page number for each entry refers to the page where the synopsis of the header is shown.

<jpeg/utility.hpp>, [4](#)

Index of library names

H

hash_combine, [5](#)

S

static_downcast, [4](#)

U

underlying, [4](#)