

OmniPointer

Generated by Doxygen 1.9.8



|  |           |
|--|-----------|
| <b>1 Class Index</b>                         | <b>1</b>  |
| 1.1 Class List                               | 1         |
| <b>2 File Index</b>                          | <b>3</b>  |
| 2.1 File List                                | 3         |
| <b>3 Class Documentation</b>                 | <b>5</b>  |
| 3.1 OmniPointer Class Reference              | 5         |
| 3.1.1 Detailed Description                   | 6         |
| 3.1.2 Constructor & Destructor Documentation | 6         |
| 3.1.2.1 OmniPointer() [1/5]                  | 6         |
| 3.1.2.2 OmniPointer() [2/5]                  | 6         |
| 3.1.2.3 OmniPointer() [3/5]                  | 6         |
| 3.1.2.4 OmniPointer() [4/5]                  | 6         |
| 3.1.2.5 OmniPointer() [5/5]                  | 7         |
| 3.1.2.6 ~OmniPointer()                       | 7         |
| 3.1.3 Member Function Documentation          | 7         |
| 3.1.3.1 Get()                                | 7         |
| 3.1.3.2 operator T*()                        | 8         |
| 3.1.3.3 operator!=(())                       | 8         |
| 3.1.3.4 operator=() [1/3]                    | 8         |
| 3.1.3.5 operator=() [2/3]                    | 8         |
| 3.1.3.6 operator=() [3/3]                    | 9         |
| 3.1.3.7 operator==(())                       | 9         |
| 3.1.3.8 Release()                            | 9         |
| 3.1.3.9 Reset() [1/2]                        | 9         |
| 3.1.3.10 Reset() [2/2]                       | 10        |
| 3.1.3.11 Swap()                              | 11        |
| <b>4 File Documentation</b>                  | <b>13</b> |
| 4.1 OmniPointer.cpp File Reference           | 13        |
| 4.2 OmniPointer.hpp File Reference           | 13        |
| <b>Index</b>                                 | <b>15</b> |



# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

#### OmniPointer

A pointer that can point to basically any type of object. Each [OmniPointer](#) shall uniquely own an object, if any . . . . .

5



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

|                                 |       |    |
|---------------------------------|-------|----|
| <a href="#">OmniPointer.cpp</a> | ..... | 13 |
| <a href="#">OmniPointer.hpp</a> | ..... | 13 |





## Chapter 3

# Class Documentation

### 3.1 OmniPointer Class Reference

A pointer that can point to basically any type of object. Each `OmniPointer` shall uniquely own an object, if any.

```
#include <OmniPointer.hpp>
```

#### Public Member Functions

- `constexpr OmniPointer () noexcept`  
*Construct an empty `OmniPointer`.*
- `constexpr OmniPointer (std::nullptr_t) noexcept`  
*Has the same effect as the default constructor.*
- `OmniPointer (const OmniPointer &)=delete`  
*Deleted. A copy mechanism of `OmniPointer` shall not be allowed.*
- `OmniPointer (OmniPointer &&) noexcept`  
*Constructs an `OmniPointer` and moves in every information contained by the specified `OmniPointer`.*
- `template<class T >`  
`OmniPointer (T *pvar)`  
*Constructs an `OmniPointer` from the given pointer.*
- `template<class T >`  
`T * Get () const`  
*Gets the address of the object the `OmniPointer` is pointing to.*
- `template<class T >`  
`T * Release ()`  
*Releases the ownership of object, if any.*
- `template<class T >`  
`void Reset (T *tptr=(T *) nullptr)`  
*Resets the `OmniPointer` and changes the ownership of object if `tptr` is not null.*
- `void Reset (std::nullptr_t) noexcept`  
*Resets the `OmniPointer`.*
- `void Swap (OmniPointer &other) noexcept`  
*Swaps the ownerships between the `OmniPointer` and the other.*
- `OmniPointer & operator= (OmniPointer &&var) noexcept`  
*Resets the `OmniPointer` and moves in every information contained by the other `OmniPointer`.*
- `OmniPointer & operator= (const OmniPointer &)=delete`

- Deleted. A copy mechanism of `OmniPointer` shall not be allowed.*
- `OmniPointer & operator= (std::nullptr_t) noexcept`  
*Has the same effect as `OmniPointer::Reset (std::nullptr_t)`.*
- `bool operator== (const OmniPointer &other) const noexcept`  
*Determines if two `OmniPointers` are pointing at the same address.*
- `bool operator!= (const OmniPointer &other) const noexcept`  
*Determines if two `OmniPointers` are pointing at different addresses.*
- `template<class T > operator T* () const`  
*The same as `OmniPointer::Get<T> ()`.*
- `~OmniPointer () noexcept`  
*Destroys the `OmniPointer` and the stored object therein, if any.*

### 3.1.1 Detailed Description

A pointer that can point to basically any type of object. Each `OmniPointer` shall uniquely own an object, if any.

#### Author

Jean-Valentin Auguste

#### Date

December 2023

#### Warning

Cannot be used for pointers of dynamically-allocated array.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 `OmniPointer()` [1/5]

```
constexpr OmniPointer::OmniPointer ( ) [inline], [constexpr], [noexcept]
```

Construct an empty `OmniPointer`.

#### 3.1.2.2 `OmniPointer()` [2/5]

```
constexpr OmniPointer::OmniPointer (
    std::nullptr_t ) [inline], [constexpr], [noexcept]
```

Has the same effect as the default constructor.

#### 3.1.2.3 `OmniPointer()` [3/5]

```
OmniPointer::OmniPointer (
    const OmniPointer & ) [delete]
```

Deleted. A copy mechanism of `OmniPointer` shall not be allowed.

#### 3.1.2.4 `OmniPointer()` [4/5]

```
OmniPointer::OmniPointer (
    OmniPointer && var ) [noexcept]
```

Constructs an `OmniPointer` and moves in every information contained by the specified `OmniPointer`.

## Parameters

|            |   |
|------------|---|
| <i>var</i> | The other <a href="#">OmniPointer</a> . |
|------------|---|

**3.1.2.5 OmniPointer()** [5/5]

```
template<class T >
OmniPointer::OmniPointer (
    T * pvar ) [inline]
```

Constructs an [OmniPointer](#) from the given pointer.

## Template Parameters

|          |  |
|----------|--|
| <i>T</i> | The type of the object the given pointer is pointing at. |
|----------|--|

## Parameters

|             |                    |
|-------------|--------------------|
| <i>pvar</i> | The given pointer. |
|-------------|--------------------|

## Warning

Assigning a local raw pointer is not recommended for it may cause dangling pointers in case two [OmniPointers](#) are constructed from the same raw pointer or it is itself a dangling or wild pointer. Assigning a pointer of a dynamic array is also not recommended for the deletion mechanism will most likely be invalid.

**3.1.2.6 ~OmniPointer()**

```
OmniPointer::~~OmniPointer ( ) [noexcept]
```

Destroys the [OmniPointer](#) and the stored object therein, if any.

**3.1.3 Member Function Documentation****3.1.3.1 Get()**

```
template<class T >
T * OmniPointer::Get ( ) const [inline]
```

Gets the address of the object the [OmniPointer](#) is pointing to.

## Template Parameters

|          |                  |
|----------|------------------|
| <i>T</i> | The desired type |
|----------|------------------|

**Returns**

A `T` pointer containing the address the `OmniPointer` is pointing at.

**Exceptions**

|                               |  |
|-------------------------------|--|
| <code>std::logic_error</code> | when <code>T`</code> does not match the information stored in the <code>`OmniPointer`</code> unless <code>T`</code> is <code>`void`</code> . |
|-------------------------------|--|

**3.1.3.2 operator T\*()**

```
template<class T >
OmniPointer::operator T* ( ) const [inline]
```

The same as `OmniPointer::Get<T>()`.

**Template Parameters**

|                |                  |
|----------------|------------------|
| <code>T</code> | The desired type |
|----------------|------------------|

**Exceptions**

|                               |  |
|-------------------------------|--|
| <code>std::logic_error</code> | when <code>T`</code> does not match the information stored in the <code>`OmniPointer`</code> . |
|-------------------------------|--|

**3.1.3.3 operator"!=(())**

```
bool OmniPointer::operator!= (
    const OmniPointer & other ) const [noexcept]
```

Determines if two `OmniPointers` are pointing at different addresses.

**3.1.3.4 operator=() [1/3]**

```
OmniPointer & OmniPointer::operator= (
    const OmniPointer & ) [delete]
```

Deleted. A copy mechanism of `OmniPointer` shall not be allowed.

**3.1.3.5 operator=() [2/3]**

```
OmniPointer & OmniPointer::operator= (
    OmniPointer && var ) [noexcept]
```

Resets the `OmniPointer` and moves in every information contained by the other `OmniPointer`.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>var</code> | The other <code>OmniPointer</code> . |
|------------------|--------------------------------------|

### 3.1.3.6 operator=() [3/3]

```
OmniPointer & OmniPointer::operator= (
    std::nullptr_t ) [noexcept]
```

Has the same effect as `OmniPointer::Reset (std::nullptr_t)`.

### 3.1.3.7 operator==(

```
bool OmniPointer::operator== (
    const OmniPointer & other ) const [noexcept]
```

Determines if two `OmniPointers` are pointing at the same address.

### 3.1.3.8 Release()

```
template<class T >
T * OmniPointer::Release ( ) [inline]
```

Releases the ownership of object, if any.

#### Template Parameters

|          |                  |
|----------|------------------|
| <i>T</i> | The desired type |
|----------|------------------|

#### Returns

A `T` pointer containing the address the `OmniPointer` was previously pointing at.

#### Exceptions

|                               |  |
|-------------------------------|--|
| <code>std::logic_error</code> | when <code>`T`</code> does not match the information stored in the <code>`OmniPointer`</code> unless <code>`T`</code> is <code>`void`</code> . |
|-------------------------------|--|

#### Note

A null pointer may be returned if the `OmniPointer` previously had no stored object.

### 3.1.3.9 Reset() [1/2]

```
void OmniPointer::Reset (
    std::nullptr_t ) [noexcept]
```

Resets the `OmniPointer`.

### 3.1.3.10 Reset() [2/2]

```
template<class T >
void OmniPointer::Reset (
    T * tptr = (T*) nullptr ) [inline]
```

Resets the [OmniPointer](#) and changes the ownership of object if `tptr` is not null.

## Template Parameters

|          |                  |
|----------|------------------|
| <i>T</i> | The desired type |
|----------|------------------|

## Parameters

|             |   |
|-------------|---|
| <i>tptr</i> | The pointer to the object to be acquired ownership of |
|-------------|---|

**3.1.3.11 Swap()**

```
void OmniPointer::Swap (
    OmniPointer & other ) [noexcept]
```

Swaps the ownerships between the `OmniPointer` and the other.

## Parameters

|              |  |
|--------------|--|
| <i>other</i> | The other <code>OmniPointer</code> to swap ownerships with |
|--------------|--|

The documentation for this class was generated from the following files:

- [OmniPointer.hpp](#)
- [OmniPointer.cpp](#)





# Chapter 4

## File Documentation

### 4.1 OmniPointer.cpp File Reference

```
#include "OmniPointer.hpp"
```

### 4.2 OmniPointer.hpp File Reference

```
#include <cstdlib>
#include <stdexcept>
#include <type_traits>
#include <typeinfo>
```

#### Classes

- class [OmniPointer](#)

*A pointer that can point to basically any type of object. Each [OmniPointer](#) shall uniquely own an object, if any.*



# Index

- ~OmniPointer
  - OmniPointer, [7](#)
- Get
  - OmniPointer, [7](#)
- OmniPointer, [5](#)
  - ~OmniPointer, [7](#)
  - Get, [7](#)
  - OmniPointer, [6](#), [7](#)
  - operator T\*, [8](#)
  - operator!=, [8](#)
  - operator=, [8](#), [9](#)
  - operator==, [9](#)
  - Release, [9](#)
  - Reset, [9](#)
  - Swap, [11](#)
- OmniPointer.cpp, [13](#)
- OmniPointer.hpp, [13](#)
- operator T\*
  - OmniPointer, [8](#)
- operator!=
  - OmniPointer, [8](#)
- operator=
  - OmniPointer, [8](#), [9](#)
- operator==
  - OmniPointer, [9](#)
- Release
  - OmniPointer, [9](#)
- Reset
  - OmniPointer, [9](#)
- Swap
  - OmniPointer, [11](#)