VaException

17.01.09

Generated by Doxygen 1.8.13

Contents

Chapter 1

VaException Documentation

A little more convinient tool for throwing exceptions.

Version

17.01.09

Copyright

(C) Vladislav Aleinik (`valeinik00@gmail.com`)

Date

2017-01-09 22:13:37 +0400

What is VaException?

VaException (Vladik Aleinik Exception) is just like std::exception, but it gives more information on the error (it allows you to pass line, function and file, in which the exception was thrown). To learn more watch VaExc::Exception docs.

Warning

VaException is currently (and forever) in alpha state, so bugs can occur.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 VaExc Namespace Reference

Namespace with the whole code.

Namespaces

- _control

    Control over program behavior.
- _errorInfo

    Namespace for ErrorInfo struct to store information conviniently (not for user).
- _literals

    User-defined literals for convinient wrapping.
- _wrappers

    Wrappers to be used in Exception constructor (Exception()), mostly not for user use.

Classes

- class Exception

    The Exception.

Typedefs

- using **ArgMsg** = _wrappers::ArgMsg

5.1.1 Detailed Description

Namespace with the whole code.

This is for convinience and recognition, just like namespace std.

```
VaExc::Exception(...);
```

5.2 VaExc::_control Namespace Reference

Control over program behavior.

Variables

- const size_t MAX_INFO_SIZE = 400

  The maximumum amount of bytes an Exception can store.
- const size_t MAX_MSG_SIZE = 200

  The maximumum amount of bytes an explanation message in Exception can store.
- const size_t MAX_EXC_COUNT = 4

  The maximumum amount of Exception instances in "caused-by" chains.
- const size_t NOT_ENOUGH_SPACE_SIZE = 4

  The size of NOT_ENOUGH_SPACE array.
- const char NOT_ENOUGH_SPACE [NOT_ENOUGH_SPACE_SIZE+1] = "\n..."

  The string, which is printed out in case there's not enough space to print something.

5.2.1 Detailed Description

Control over program behavior.

Contains variables, which tell the program how much information (in bytes) an Exception can store.

5.3 VaExc::_errorInfo Namespace Reference

Namespace for ErrorInfo struct to store information conviniently (not for user).

Classes

- class ErrorInfo

  Class, which is only useful in Exception implementation (not for user).

5.3.1 Detailed Description

Namespace for ErrorInfo struct to store information conviniently (not for user).

5.4 VaExc::_literals Namespace Reference

User-defined literals for convinient wrapping.

Functions

- constexpr _wrappers::ArgMsgConstexpr operator""_msg (const char ∗str, size_t) noexcept

  Exception message.
- constexpr _wrappers::ArgFilename operator""_file (const char ∗str, size_t) noexcept

  Filename of the file in which Exception is created.
- constexpr _wrappers::ArgFunction operator""_func (const char ∗str, size_t) noexcept

  Function name in which Exception is created.
- constexpr _wrappers::ArgLine operator""_line (unsigned long long int line) noexcept

  Call line number in which Exception is created.

5.4.1 Detailed Description

User-defined literals for convinient wrapping.

5.4.2 Function Documentation

5.4.2.1 operator""""_file()

```
constexpr _wrappers::ArgFilename VaExc::_literals::operator""_file (
              const char * str,
              size_t  )  [noexcept]
```

Filename of the file in which Exception is created.

See also

Exception, VAEXC_POS

Examples

```
Exception("src/File.hpp"_file);

Exception(VAEXC_POS); // sometimes better option
```

5.4.2.2 operator""""_func()

```
constexpr _wrappers::ArgFunction VaExc::_literals::operator""_func (
              const char * str,
              size_t  )  [noexcept]
```

Function name in which Exception is created.

See also

Exception, VAEXC_POS

Examples

```
void foo()
{
    throw Exception("void foo()"_func);

    throw Exception(VAEXC_POS);  // a little better option
}
```

5.4.2.3 operator""""_line()

```
constexpr _wrappers::ArgLine VaExc::_literals::operator""_line (
              unsigned long long int line )  [noexcept]
```

Call line number in which Exception is created.

See also

Exception, VAEXC_POS

Examples

```
throw Exception(228_line);
throw Exception(VAEXC_POS); // ALWAYS MUCH BETTER
```

5.4.2.4 operator""""_msg()

```
constexpr _wrappers::ArgMsgConstexpr VaExc::_literals::operator""_msg (
              const char * str,
              size_t  )  [noexcept]
```

Exception message.

See also

Exception, VAEXC_POS

Examples

```
Exception("Oh no! Exception happened!"_msg);
```

5.5 VaExc::_wrappers Namespace Reference

Wrappers to be used in Exception constructor (Exception()), mostly not for user use.

Classes

- struct ArgFilename
  Wrapper to store filename of the file in which Exception is created.
- struct ArgFunction
  Wrapper to store function name in which Exception is created.
- struct ArgLine
  Wrapper to store call line number in which Exception is created.
- struct ArgMsg
  Wrapper that stores exception message.
- struct ArgMsgConstexpr
  Just like ArgMsg, but for c-style strings (not for user).

5.5.1 Detailed Description

Wrappers to be used in Exception constructor (Exception()), mostly not for user use.

Chapter 6

Class Documentation

6.1 VaExc::_wrappers::ArgFilename Struct Reference

Wrapper to store filename of the file in which Exception is created.

```
#include <VaException.hpp>
```

Public Attributes

- const char ∗ **file**

6.1.1 Detailed Description

Wrapper to store filename of the file in which Exception is created.

See also

Exception, operator""_file()

The documentation for this struct was generated from the following file:

- /Users/vladislav_aleinik/Dropbox/Programming/cpp/2016-2017/my_exception_improved/src/VaException.↩
hpp

6.2 VaExc::_wrappers::ArgFunction Struct Reference

Wrapper to store function name in which Exception is created.

```
#include <VaException.hpp>
```

Public Attributes

- const char ∗ **func**

6.2.1 Detailed Description

Wrapper to store function name in which Exception is created.

See also

Exception, operator""_func()

The documentation for this struct was generated from the following file:

- /Users/vladislav_aleinik/Dropbox/Programming/cpp/2016-2017/my_exception_improved/src/VaException.↩
hpp

6.3 VaExc::_wrappers::ArgLine Struct Reference

Wrapper to store call line number in which Exception is created.

```
#include <VaException.hpp>
```

Public Attributes

- size_t **line**

6.3.1 Detailed Description

Wrapper to store call line number in which Exception is created.

See also

Exception, operator""_line()

The documentation for this struct was generated from the following file:

- /Users/vladislav_aleinik/Dropbox/Programming/cpp/2016-2017/my_exception_improved/src/VaException.↩
hpp

6.4 VaExc::_wrappers::ArgMsg Struct Reference

Wrapper that stores exception message.

```
#include <VaException.hpp>
```

Public Member Functions

- template<typename... Args>
  ArgMsg (const char ∗format, Args &&... args) noexcept

  ArgMsg Constructor.

Public Attributes

- char **msg** [_control::MAX_MSG_SIZE+1]

6.4.1 Detailed Description

Wrapper that stores exception message.

See also

Exception

```
throw Exception(..., ArgMsg("Too many kitten explosions per second. %ull total", kitExpPerSec), ...);
```

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ArgMsg()

```
template<typename... Args>
VaExc::_wrappers::ArgMsg::ArgMsg (
            const char * format,
            Args &&... args ) [inline], [noexcept]
```

ArgMsg Constructor.

See also

ArgMsg

The documentation for this struct was generated from the following file:

- /Users/vladislav_aleinik/Dropbox/Programming/cpp/2016-2017/my_exception_improved/src/VaException.↩
  hpp

6.5 VaExc::_wrappers::ArgMsgConstexpr Struct Reference

Just like ArgMsg, but for c-style strings (not for user).

```
#include <VaException.hpp>
```

Public Attributes

- const char ∗ **msg**

6.5.1 Detailed Description

Just like ArgMsg, but for c-style strings (not for user).

See also

Exception, operator""_msg()

The documentation for this struct was generated from the following file:

- /Users/vladislav_aleinik/Dropbox/Programming/cpp/2016-2017/my_exception_improved/src/VaException.↩ hpp

6.6 VaExc::_errorInfo::ErrorInfo Class Reference

Class, which is only useful in Exception implementation (not for user).

```
#include <VaException.hpp>
```

Public Member Functions

- **ErrorInfo** (const ErrorInfo &that)=default
- ErrorInfo & **operator=** (const ErrorInfo &that)=default
- bool **writeInfoWithCaption** (const char ∗caption, const char ∗text) noexcept
- const char ∗ **info** () const noexcept

6.6.1 Detailed Description

Class, which is only useful in Exception implementation (not for user).

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

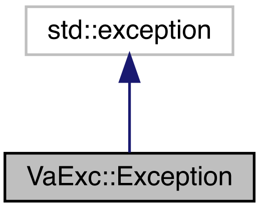
- /Users/vladislav_aleinik/Dropbox/Programming/cpp/2016-2017/my_exception_improved/src/VaException.↩ hpp

6.7 VaExc::Exception Class Reference

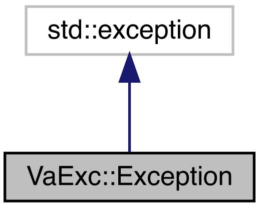
The Exception.

```
#include <VaException.hpp>
```

Inheritance diagram for VaExc::Exception:



Collaboration diagram for VaExc::Exception:



Public Member Functions

- template<class... Args>
  Exception (Args &&... args) noexcept
- virtual const char $*$ **what** () const noexcept override

6.7.1 Detailed Description

The Exception.

Exception is just like std::exception, but it gives more information on the error (func, line, file), Exception::what() can print arguments in any order ([msg, line, func] or [func, msg, line]), Exception is inherited from std::exception (thus, it is a std::exception), Exception supports formatted strings, just like std::printf.

See also

operator""_msg(), operator""_file(), operator""_func(), operator""_line(), _wrappers::ArgMsg::ArgMsg(), VA↩ EXC_POS

Examples

```
try
{
    using namespace VaExc;

    // ArgMsg is implemented via std::sprintf, so formatting is the same.
    throw Exception(ArgMsg("Error code: %d", error_code), VAEXC_POS);

    ...
}
catch (Exception& exc)
{
    throw Exception("Exception occured!!!", exc);
}

...

try
{
    using namespace VaExc;

    throw Exception("Aaaa!"_msg, "Fuuuunc!11!1"_func, 1111_line);

    ...
}
catch (std::exception& exc) // Also catches Exception
{
    throw Exception("Exception occured!!!", VAEXC_POS, exc);
}
```

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

- /Users/vladislav_aleinik/Dropbox/Programming/cpp/2016-2017/my_exception_improved/src/VaException.↩ hpp