Version

# Table of Contents

# Jogo de Fazenda :pig2: :baby\_chick: :seedling: :corn:

Trabalho de C++ do curso de Programação e Desenvolvimento de Software II - UFMG

Professor Luigi Domenico

## Sobre o projeto

No "nome do jogo", você deve realizar ações para receber moedas e suir de nível!

Transite entre as diferentes funcionalidades de um fazendeiro e obtenha os itens necessários para valorizar seus pertences.

Não se esqueça de fazer vendas na loja e aumentar sua produção!

## Funcionalidades

Interagir com as plantações

Ganhar pontos e subir de nível

Compre e venda itens na loja.

## Compilação e Execução

Para compilar e executar o programa, siga as instruções abaixo:

1. Clone o repositório ou faça o download dos arquivos.

2. Abra um terminal e navegue até o diretório do programa.

3. Compile o programa com o seguinte comando:

colocar o comando

## Integrantes do grupo:

Emanuelle King Amaral

Jean Pierry Alves Lopes

Mateus Souza Barreto

Naually Pereira Sejimo de Avila

Raissa Lauar Gonçalves Navarro da Silva

# Namespace Index

## Namespace List

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

**color (Biblioteca para adição de cores no terminal )**

# Hierarchical Index

## Class Hierarchy

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

doctest::Approx

doctest::AssertData

doctest::detail::ResultBuilder

std::basic\_istream< charT, traits >

std::basic\_ostream< charT, traits >

std::char\_traits< charT >

doctest::Contains

doctest::Context

doctest::ContextOptions

doctest::CurrentTestCaseStats

doctest::detail::types::enable\_if< COND, T >

doctest::detail::types::enable\_if< true, T >

doctest::detail::Expression\_lhs< L >

doctest::detail::ExpressionDecomposer

doctest::detail::types::false\_type

doctest::detail::deferred\_false< T >

doctest::detail::has\_insertion\_operator< T, typename >

doctest::detail::types::is\_array< T >

doctest::detail::types::is\_pointer< T >

doctest::detail::types::is\_rvalue\_reference< T >

Farmer

doctest::detail::filldata< T >

doctest::detail::filldata< const char[N]>

doctest::detail::filldata< const void \* >

doctest::detail::filldata< T \* >

doctest::detail::filldata< T[N]>

Fisherman

doctest::IContextScope

doctest::detail::ContextScopeBase

doctest::detail::ContextScope< L >

doctest::detail::IExceptionTranslator

doctest::detail::ExceptionTranslator< T >

Inventory

doctest::IReporter

doctest::detail::types::is\_enum< T >

doctest::IsNaN< F >

Item

Livestocker

LojaDeItens

Menu

doctest::MessageData

doctest::detail::MessageBuilder

Miner

Player

doctest::QueryData

doctest::detail::RelationalComparator< int, L, R >

doctest::detail::types::remove\_const< T >

doctest::detail::types::remove\_const< const T >

doctest::detail::types::remove\_reference< T >

doctest::detail::types::remove\_reference< T & >

doctest::detail::types::remove\_reference< T && >

doctest::detail::Result

doctest::detail::should\_stringify\_as\_underlying\_type< T >

Sistema

StorageManager< T >

StorageManager< Item >

ItemsStorage

doctest::String

doctest::AssertData::StringContains

doctest::detail::StringMakerBase< C >

doctest::detail::StringMakerBase< detail::has\_insertion\_operator< T >::value||detail::types::is\_pointer< T >::value||detail::types::is\_array< T >::value >

doctest::StringMaker< T >

doctest::detail::StringMakerBase< true >

doctest::detail::Subcase

doctest::SubcaseSignature

doctest::TestCaseData

doctest::detail::TestCase

doctest::TestCaseException

doctest::detail::TestFailureException

doctest::TestRunStats

doctest::detail::TestSuite

doctest::detail::types::true\_type

doctest::detail::has\_insertion\_operator< T, decltype(operator<<(declval< std::ostream & >(), declval< const T & >()), void())>

doctest::detail::types::is\_array< T[SIZE]>

doctest::detail::types::is\_pointer< T \* >

doctest::detail::types::is\_rvalue\_reference< T && >

std::tuple< Types >

doctest::detail::types::underlying\_type< T >

# Class Index

## Class List

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

**doctest::Approx**

**doctest::AssertData**

**std::basic\_istream< charT, traits >**

**std::basic\_ostream< charT, traits >**

**std::char\_traits< charT >**

**doctest::Contains**

**doctest::Context**

**doctest::ContextOptions (OCLINT too many fields )**

**doctest::detail::ContextScope< L >**

**doctest::detail::ContextScopeBase**

**doctest::CurrentTestCaseStats**

**doctest::detail::deferred\_false< T >**

**doctest::detail::types::enable\_if< COND, T >**

**doctest::detail::types::enable\_if< true, T >**

**doctest::detail::ExceptionTranslator< T > (OCLINT destructor of virtual class )**

**doctest::detail::Expression\_lhs< L >**

**doctest::detail::ExpressionDecomposer**

**doctest::detail::types::false\_type**

**Farmer (Representa um agricultor no jogo )**

**doctest::detail::filldata< T >**

**doctest::detail::filldata< const char[N]>**

**doctest::detail::filldata< const void \* >**

**doctest::detail::filldata< T \* >**

**doctest::detail::filldata< T[N]>**

**Fisherman (Representa um pescador no jogo )**

**doctest::detail::has\_insertion\_operator< T, typename >**

**doctest::detail::has\_insertion\_operator< T, decltype(operator<<(declval< std::ostream & >(), declval< const T & >()), void())>**

**doctest::IContextScope**

**doctest::detail::IExceptionTranslator**

**Inventory (Representa o inventário de um jogador )**

**doctest::IReporter**

**doctest::detail::types::is\_array< T >**

**doctest::detail::types::is\_array< T[SIZE]>**

**doctest::detail::types::is\_enum< T >**

**doctest::detail::types::is\_pointer< T >**

**doctest::detail::types::is\_pointer< T \* >**

**doctest::detail::types::is\_rvalue\_reference< T >**

**doctest::detail::types::is\_rvalue\_reference< T && >**

**doctest::IsNaN< F >**

**Item (Representa um item no sistema )**

**ItemsStorage (Gerencia o armazenamento de itens )**

**Livestocker (Representa um criador de animais no jogo )**

**LojaDeItens (Classe que representa uma loja de itens no jogo )**

**Menu (Classe que representa um menu )**

**doctest::detail::MessageBuilder**

**doctest::MessageData**

**Miner**

**Player (Classe que representa um jogador )**

**doctest::QueryData**

**doctest::detail::RelationalComparator< int, L, R >**

**doctest::detail::types::remove\_const< T >**

**doctest::detail::types::remove\_const< const T >**

**doctest::detail::types::remove\_reference< T >**

**doctest::detail::types::remove\_reference< T & >**

**doctest::detail::types::remove\_reference< T && >**

**doctest::detail::Result**

**doctest::detail::ResultBuilder**

**doctest::detail::should\_stringify\_as\_underlying\_type< T >**

**Sistema (Classe que representa o sistema do jogo )**

**StorageManager< T > (Classe modelo para gerenciamento de armazenamento )**

**doctest::String**

**doctest::AssertData::StringContains**

**doctest::StringMaker< T >**

**doctest::detail::StringMakerBase< C >**

**doctest::detail::StringMakerBase< true >**

**doctest::detail::Subcase**

**doctest::SubcaseSignature**

**doctest::detail::TestCase**

**doctest::TestCaseData**

**doctest::TestCaseException**

**doctest::detail::TestFailureException**

**doctest::TestRunStats**

**doctest::detail::TestSuite**

**doctest::detail::types::true\_type**

**std::tuple< Types >**

**doctest::detail::types::underlying\_type< T >**

# File Index

## File List

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

**doctest.h**

**Farmer.cpp**

**Farmer.hpp (Contém a definição da classe Farmer )**

**Fisherman.cpp**

**Fisherman.hpp (Contém a definição da classe Fisherman )**

**Inventory.cpp**

**Inventory.hpp (Contém a definição da classe Inventory )**

**Item.cpp**

**Item.hpp (Contém a definição da classe Item )**

**ItemsStorage.hpp (Contém a definição da classe ItemsStorage, uma subclasse de StorageManager )**

**Livestocker.cpp**

**Livestocker.hpp (Contém a definição da classe Livestocker )**

**LojaDeItens.cpp**

**LojaDeItens.hpp**

**Menu.cpp**

**Menu.hpp**

**Miner.cpp**

**MIner.hpp**

**Player.cpp**

**Player.hpp**

**Sistema.cpp**

**Sistema.hpp**

**StorageManager.hpp (Contém a definição da classe template StorageManager )**

**TerminalPalette.hpp**

# Namespace Documentation

## color Namespace Reference

biblioteca para adição de cores no terminal

### Variables

const std::string **gray** = "\e[30;10m"

const std::string **grayn** = "\e[30;1m"

const std::string **grayf** = "\e[30;2m"

const std::string **grayi** = "\e[30;3m"

const std::string **grays** = "\e[30;4m"

const std::string **grayp** = "\e[30;5m"

const std::string **grayb** = "\e[30;7m"

const std::string **grayc** = "\e[30;9m"

const std::string **red** = "\e[31;10m"

const std::string **redn** = "\e[31;1m"

const std::string **redf** = "\e[31;2m"

const std::string **redi** = "\e[31;3m"

const std::string **reds** = "\e[31;4m"

const std::string **redp** = "\e[31;5m"

const std::string **redb** = "\e[31;7m"

const std::string **redc** = "\e[31;9m"

const std::string **green** = "\e[32;10m"

const std::string **greenn** = "\e[32;1m"

const std::string **greenf** = "\e[32;2m"

const std::string **greeni** = "\e[32;3m"

const std::string **greens** = "\e[32;4m"

const std::string **greenp** = "\e[32;5m"

const std::string **greenb** = "\e[32;7m"

const std::string **greenc** = "\e[32;9m"

const std::string **yellow** = "\e[33;10m"

const std::string **yellown** = "\e[33;1m"

const std::string **yellowf** = "\e[33;2m"

const std::string **yellowi** = "\e[33;3m"

const std::string **yellows** = "\e[33;4m"

const std::string **yellowp** = "\e[33;5m"

const std::string **yellowb** = "\e[33;7m"

const std::string **yellowc** = "\e[33;9m"

const std::string **blue** = "\e[34;10m"

const std::string **bluen** = "\e[34;1m"

const std::string **bluef** = "\e[34;2m"

const std::string **bluei** = "\e[34;3m"

const std::string **blues** = "\e[34;4m"

const std::string **bluep** = "\e[34;5m"

const std::string **blueb** = "\e[34;7m"

const std::string **bluec** = "\e[34;9m"

const std::string **purple** = "\e[35;10m"

const std::string **purplen** = "\e[35;1m"

const std::string **purplef** = "\e[35;2m"

const std::string **purplei** = "\e[35;3m"

const std::string **purples** = "\e[35;4m"

const std::string **purplep** = "\e[35;5m"

const std::string **purpleb** = "\e[35;7m"

const std::string **purplec** = "\e[35;9m"

const std::string **cyan** = "\e[36;10m"

const std::string **cyann** = "\e[36;1m"

const std::string **cyanf** = "\e[36;2m"

const std::string **cyani** = "\e[36;3m"

const std::string **cyans** = "\e[36;4m"

const std::string **cyanp** = "\e[36;5m"

const std::string **cyanb** = "\e[36;7m"

const std::string **cyanc** = "\e[36;9m"

const std::string **white** = "\e[38;10m"

const std::string **whiten** = "\e[38;1m"

const std::string **whitef** = "\e[38;2m"

const std::string **whitei** = "\e[38;3m"

const std::string **whites** = "\e[38;4m"

const std::string **whitep** = "\e[38;5m"

const std::string **whiteb** = "\e[38;7m"

const std::string **whitec** = "\e[38;9m"

const std::string **off** = "\e[m"

### Detailed Description

biblioteca para adição de cores no terminal

# Class Documentation

## doctest::Approx Struct Reference

### Public Member Functions

**Approx** (double value)

**Approx** **operator()** (double value) const

**Approx** & **epsilon** (double newEpsilon)

**Approx** & **scale** (double newScale)

### Public Attributes

double **m\_epsilon**

double **m\_scale**

double **m\_value**

### Friends

DOCTEST\_INTERFACE friend bool **operator==** (double lhs, const **Approx** &rhs)

DOCTEST\_INTERFACE friend bool **operator==** (const **Approx** &lhs, double rhs)

DOCTEST\_INTERFACE friend bool **operator!=** (double lhs, const **Approx** &rhs)

DOCTEST\_INTERFACE friend bool **operator!=** (const **Approx** &lhs, double rhs)

DOCTEST\_INTERFACE friend bool **operator<=** (double lhs, const **Approx** &rhs)

DOCTEST\_INTERFACE friend bool **operator<=** (const **Approx** &lhs, double rhs)

DOCTEST\_INTERFACE friend bool **operator>=** (double lhs, const **Approx** &rhs)

DOCTEST\_INTERFACE friend bool **operator>=** (const **Approx** &lhs, double rhs)

DOCTEST\_INTERFACE friend bool **operator<** (double lhs, const **Approx** &rhs)

DOCTEST\_INTERFACE friend bool **operator<** (const **Approx** &lhs, double rhs)

DOCTEST\_INTERFACE friend bool **operator>** (double lhs, const **Approx** &rhs)

DOCTEST\_INTERFACE friend bool **operator>** (const **Approx** &lhs, double rhs)

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

doctest.h

## doctest::AssertData Struct Reference

Inheritance diagram for doctest::AssertData:

### Classes

### class StringContainsPublic Member Functions

**AssertData** (assertType::Enum at, const char \*file, int line, const char \*expr, const char \*exception\_type, const **StringContains** &exception\_string)

### Public Attributes

const **TestCaseData** \* **m\_test\_case**

assertType::Enum **m\_at**

const char \* **m\_file**

int **m\_line**

const char \* **m\_expr**

bool **m\_failed**

bool **m\_threw**

**String** **m\_exception**

**String** **m\_decomp**

bool **m\_threw\_as**

const char \* **m\_exception\_type**

class DOCTEST\_INTERFACE **doctest::AssertData::StringContains** **m\_exception\_string**

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

doctest.h

## std::basic\_istream< charT, traits > Class Template Reference

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

doctest.h

## std::basic\_ostream< charT, traits > Class Template Reference

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

doctest.h

## std::char\_traits< charT > Struct Template Reference

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

doctest.h

## doctest::Contains Class Reference

### Public Member Functions

**Contains** (const **String** &string)

bool **checkWith** (const **String** &other) const

### Public Attributes

**String** **string**

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

doctest.h

## doctest::Context Class Reference

### Public Member Functions

**Context** (int argc=0, const char \*const \*argv=nullptr)

**Context** (const **Context** &)=delete

**Context** (**Context** &&)=delete

**Context** & **operator=** (const **Context** &)=delete

**Context** & **operator=** (**Context** &&)=delete

void **applyCommandLine** (int argc, const char \*const \*argv)

void **addFilter** (const char \*filter, const char \*value)

void **clearFilters** ()

void **setOption** (const char \*option, bool value)

void **setOption** (const char \*option, int value)

void **setOption** (const char \*option, const char \*value)

bool **shouldExit** ()

void **setAsDefaultForAssertsOutOfTestCases** ()

void **setAssertHandler** (detail::assert\_handler ah)

void **setCout** (**std::ostream** \*out)

int **run** ()

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

doctest.h

## doctest::ContextOptions Struct Reference

OCLINT too many fields.

#include <doctest.h>

### Public Attributes

**std::ostream** \* **cout** = nullptr

**String** **binary\_name**

const **detail::TestCase** \* **currentTest** = nullptr

**String** **out**

**String** **order\_by**

unsigned **rand\_seed**

unsigned **first**

unsigned **last**

int **abort\_after**

int **subcase\_filter\_levels**

bool **success**

bool **case\_sensitive**

bool **exit**

bool **duration**

bool **minimal**

bool **quiet**

bool **no\_throw**

bool **no\_exitcode**

bool **no\_run**

bool **no\_intro**

bool **no\_version**

bool **no\_colors**

bool **force\_colors**

bool **no\_breaks**

bool **no\_skip**

bool **gnu\_file\_line**

bool **no\_path\_in\_filenames**

bool **no\_line\_numbers**

bool **no\_debug\_output**

bool **no\_skipped\_summary**

bool **no\_time\_in\_output**

bool **help**

bool **version**

bool **count**

bool **list\_test\_cases**

bool **list\_test\_suites**

bool **list\_reporters**

### Detailed Description

OCLINT too many fields.

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

doctest.h

## doctest::detail::ContextScope< L > Class Template Reference

Inheritance diagram for doctest::detail::ContextScope< L >:

### Public Member Functions

**ContextScope** (const L &lambda)

**ContextScope** (L &&lambda)

**ContextScope** (const **ContextScope** &)=delete

**ContextScope** (**ContextScope** &&) noexcept=default

**ContextScope** & **operator=** (const **ContextScope** &)=delete

**ContextScope** & **operator=** (**ContextScope** &&)=delete

void **stringify** (**std::ostream** \*s) const override

#### Public Member Functions inherited from doctest::detail::ContextScopeBase

**ContextScopeBase** (const **ContextScopeBase** &)=delete

**ContextScopeBase** & **operator=** (const **ContextScopeBase** &)=delete

**ContextScopeBase** & **operator=** (**ContextScopeBase** &&)=delete

### Additional Inherited Members

#### Protected Member Functions inherited from doctest::detail::ContextScopeBase

**ContextScopeBase** (**ContextScopeBase** &&other) noexcept

void **destroy** ()

#### Protected Attributes inherited from doctest::detail::ContextScopeBase

bool **need\_to\_destroy** {true}

### Member Function Documentation

#### template<typename L > void doctest::detail::ContextScope< L >::stringify (std::ostream \* *s*) const[inline], [override], [virtual]

Implements **doctest::IContextScope** (*p.*).

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

doctest.h

## doctest::detail::ContextScopeBase Struct Reference

Inheritance diagram for doctest::detail::ContextScopeBase:

### Public Member Functions

**ContextScopeBase** (const **ContextScopeBase** &)=delete

**ContextScopeBase** & **operator=** (const **ContextScopeBase** &)=delete

**ContextScopeBase** & **operator=** (**ContextScopeBase** &&)=delete

#### Public Member Functions inherited from doctest::IContextScope

virtual void **stringify** (**std::ostream** \*) const =0

### Protected Member Functions

**ContextScopeBase** (**ContextScopeBase** &&other) noexcept

void **destroy** ()

### Protected Attributes

bool **need\_to\_destroy** {true}

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

doctest.h

## doctest::CurrentTestCaseStats Struct Reference

### Public Attributes

int **numAssertsCurrentTest**

int **numAssertsFailedCurrentTest**

double **seconds**

int **failure\_flags**

bool **testCaseSuccess**

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

doctest.h

## doctest::detail::deferred\_false< T > Struct Template Reference

Inheritance diagram for doctest::detail::deferred\_false< T >:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::false\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **false**

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

doctest.h

## doctest::detail::types::enable\_if< COND, T > Struct Template Reference

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

doctest.h

## doctest::detail::types::enable\_if< true, T > Struct Template Reference

### Public Types

**using** **type** = **T**

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

doctest.h

## doctest::detail::ExceptionTranslator< T > Class Template Reference

OCLINT destructor of virtual class.

#include <doctest.h>

Inheritance diagram for doctest::detail::ExceptionTranslator< T >:

### Public Member Functions

**ExceptionTranslator** (**String**(\*translateFunction)(T))

bool **translate** (**String** &res) const override

### Detailed Description

#### template<typename T>

#### class doctest::detail::ExceptionTranslator< T >

OCLINT destructor of virtual class.

### Member Function Documentation

#### template<typename T > bool doctest::detail::ExceptionTranslator< T >::translate (String & *res*) const[inline], [override], [virtual]

Implements **doctest::detail::IExceptionTranslator** (*p.*).

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

doctest.h

## doctest::detail::Expression\_lhs< L > Struct Template Reference

### Public Member Functions

**Expression\_lhs** (L &&in, assertType::Enum at)

DOCTEST\_NOINLINE **operator Result** ()

**operator L** () const

### Public Attributes

L **lhs**

assertType::Enum **m\_at**

### Member Function Documentation

#### template<typename L > DOCTEST\_NOINLINE doctest::detail::Expression\_lhs< L >::operator Result ()[inline]

OCLINT bitwise operator in conditional

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

doctest.h

## doctest::detail::ExpressionDecomposer Struct Reference

### Public Member Functions

**ExpressionDecomposer** (assertType::Enum at)

template<typename L > **Expression\_lhs**< L > **operator<<** (L &&operand)

template<typename L , typename **types::enable\_if**<!**doctest::detail::types::is\_rvalue\_reference**< L >::value, void >::type \* = nullptr> **Expression\_lhs**< const L & > **operator<<** (const L &operand)

### Public Attributes

assertType::Enum **m\_at**

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

doctest.h

## doctest::detail::types::false\_type Struct Reference

Inheritance diagram for doctest::detail::types::false\_type:

### Static Public Attributes

**static** DOCTEST\_CONSTEXPR **bool** **value** = **false**

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

doctest.h

## Farmer Class Reference

Representa um agricultor no jogo.

#include <Farmer.hpp>

### Public Member Functions

void **getWheat** (int quantidade, **Player** \*player, **ItemsStorage** \*storage)

*Substitui o item semente por trigo.*

void **getCarrot** (int quantidade, **Player** \*player, **ItemsStorage** \*storage)

*Substitui o item semente por cenoura.*

### Detailed Description

Representa um agricultor no jogo.

### Member Function Documentation

#### void Farmer::getCarrot (int *quantidade*, Player \* *player*, ItemsStorage \* *storage*)

Substitui o item semente por cenoura.

##### Parameters

|  |  |
| --- | --- |
| *quantidade* | Número de cenouras a serem obtidas. |
| *player* | Ponteiro para o jogador. |
| *storage* | Ponteiro para o armazenamento de itens. |

#### void Farmer::getWheat (int *quantidade*, Player \* *player*, ItemsStorage \* *storage*)

Substitui o item semente por trigo.

##### Parameters

|  |  |
| --- | --- |
| *quantidade* | Número de trigos a serem obtidos. |
| *player* | Ponteiro para o jogador. |
| *storage* | Ponteiro para o armazenamento de itens. |

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

**Farmer.hpp**Farmer.cpp

## doctest::detail::filldata< T > Struct Template Reference

### Static Public Member Functions

static void **fill** (**std::ostream** \*stream, const T &in)

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

doctest.h

## doctest::detail::filldata< const char[N]> Struct Template Reference

### Static Public Member Functions

static void **fill** (**std::ostream** \*stream, const char(&in)[N])

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

doctest.h

## doctest::detail::filldata< const void \* > Struct Reference

### Static Public Member Functions

static void **fill** (**std::ostream** \*stream, const void \*in)

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

doctest.h

## doctest::detail::filldata< T \* > Struct Template Reference

### Static Public Member Functions

static void **fill** (**std::ostream** \*stream, const T \*in)

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

doctest.h

## doctest::detail::filldata< T[N]> Struct Template Reference

### Static Public Member Functions

static void **fill** (**std::ostream** \*stream, const T(&in)[N])

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

doctest.h

## Fisherman Class Reference

Representa um pescador no jogo.

#include <Fisherman.hpp>

### Public Member Functions

void **fish** (**Player** \*player, **ItemsStorage** \*storage)

*Gera peixes aleatórios para o jogador.*

void **cleanFish** (**Player** \*player)

*Aumenta o valor de venda dos peixes em 10%.*

### Detailed Description

Representa um pescador no jogo.

### Member Function Documentation

#### void Fisherman::cleanFish (Player \* *player*)

Aumenta o valor de venda dos peixes em 10%.

##### Parameters

|  |  |
| --- | --- |
| *player* | Ponteiro para o jogador. |

#### void Fisherman::fish (Player \* *player*, ItemsStorage \* *storage*)

Gera peixes aleatórios para o jogador.

##### Parameters

|  |  |
| --- | --- |
| *player* | Ponteiro para o jogador. |
| *storage* | Ponteiro para o armazenamento de itens. |

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

**Fisherman.hpp**Fisherman.cpp

## doctest::detail::has\_insertion\_operator< T, typename > Struct Template Reference

Inheritance diagram for doctest::detail::has\_insertion\_operator< T, typename >:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::false\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **false**

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

doctest.h

## doctest::detail::has\_insertion\_operator< T, decltype(operator<<(declval< std::ostream & >(), declval< const T & >()), void())> Struct Template Reference

Inheritance diagram for doctest::detail::has\_insertion\_operator< T, decltype(operator<<(declval< std::ostream & >(), declval< const T & >()), void())>:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::true\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **true**

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

doctest.h

## doctest::IContextScope Struct Reference

Inheritance diagram for doctest::IContextScope:

### Public Member Functions

virtual void **stringify** (**std::ostream** \*) const =0

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

doctest.h

## doctest::detail::IExceptionTranslator Struct Reference

Inheritance diagram for doctest::detail::IExceptionTranslator:

### Public Member Functions

virtual bool **translate** (**String** &) const =0

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

doctest.h

## Inventory Class Reference

Representa o inventário de um jogador.

#include <Inventory.hpp>

### Public Member Functions

**Inventory** (unsigned capacity)

*Construtor da classe* ***Inventory****.*

bool **hasItem** (int id, unsigned quantidade)

*Verifica se o inventário possui uma quantidade específica de um item.*

bool **insert** (int id, **Item** item)

*Insere um item no inventário do jogador.*

bool **remove** (int id, unsigned quantidade)

*Remove uma quantidade específica de um item do inventário.*

void **printInventory** ()

*Imprime todos os itens do inventário.*

std::map< int, std::pair< **Item**, unsigned > > **list** ()

*Lista todos os itens do inventário com suas quantidades.*

void **updateItemPrice** (int id, double novoPreco)

*Atualiza o preço de um item no inventário.*

void **updateItemStatus** (int id)

*Atualiza o status de um item no inventário.*

### Detailed Description

Representa o inventário de um jogador.

### Constructor & Destructor Documentation

#### Inventory::Inventory (unsigned *capacity*)

Construtor da classe **Inventory**.

##### Parameters

|  |  |
| --- | --- |
| *capacity* | Quantidade máxima de elementos no inventário. |

### Member Function Documentation

#### bool Inventory::hasItem (int *id*, unsigned *quantidade*)

Verifica se o inventário possui uma quantidade específica de um item.

##### Parameters

|  |  |
| --- | --- |
| *id* | Identificador único do item. |
| *quantidade* | Quantidade desejada do item. |

##### Returns

Verdadeiro se o inventário possuir a quantidade desejada do item, falso caso contrário.

#### bool Inventory::insert (int *id*, Item *item*)

Insere um item no inventário do jogador.

##### Parameters

|  |  |
| --- | --- |
| *id* | Identificador único do item. |
| *item* | **Item** a ser inserido no inventário. |

##### Returns

Verdadeiro se o elemento foi inserido, falso se não foi.

#### std::map< int, std::pair< Item, unsigned > > Inventory::list ()

Lista todos os itens do inventário com suas quantidades.

##### Returns

Mapa com os itens e suas quantidades.

#### bool Inventory::remove (int *id*, unsigned *quantidade*)

Remove uma quantidade específica de um item do inventário.

##### Parameters

|  |  |
| --- | --- |
| *id* | Identificador único do item. |
| *quantidade* | Quantidade do item a ser removida. |

##### Returns

Verdadeiro se o elemento foi removido, falso se não foi encontrado.

#### void Inventory::updateItemPrice (int *id*, double *novoPreco*)

Atualiza o preço de um item no inventário.

##### Parameters

|  |  |
| --- | --- |
| *id* | Identificador único do item. |
| *novoPreco* | Novo preço a ser atribuído ao item. |

#### void Inventory::updateItemStatus (int *id*)

Atualiza o status de um item no inventário.

##### Parameters

|  |  |
| --- | --- |
| *id* | Identificador único do item. |

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

**Inventory.hpp**Inventory.cpp

## doctest::IReporter Struct Reference

### Public Member Functions

virtual void **report\_query** (const **QueryData** &)=0

virtual void **test\_run\_start** ()=0

virtual void **test\_run\_end** (const **TestRunStats** &)=0

virtual void **test\_case\_start** (const **TestCaseData** &)=0

virtual void **test\_case\_reenter** (const **TestCaseData** &)=0

virtual void **test\_case\_end** (const **CurrentTestCaseStats** &)=0

virtual void **test\_case\_exception** (const **TestCaseException** &)=0

virtual void **subcase\_start** (const **SubcaseSignature** &)=0

virtual void **subcase\_end** ()=0

virtual void **log\_assert** (const **AssertData** &)=0

virtual void **log\_message** (const **MessageData** &)=0

virtual void **test\_case\_skipped** (const **TestCaseData** &)=0

### Static Public Member Functions

static int **get\_num\_active\_contexts** ()

static const **IContextScope** \*const \* **get\_active\_contexts** ()

static int **get\_num\_stringified\_contexts** ()

static const **String** \* **get\_stringified\_contexts** ()

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

doctest.h

## doctest::detail::types::is\_array< T > Struct Template Reference

Inheritance diagram for doctest::detail::types::is\_array< T >:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::false\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **false**

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

doctest.h

## doctest::detail::types::is\_array< T[SIZE]> Struct Template Reference

Inheritance diagram for doctest::detail::types::is\_array< T[SIZE]>:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::true\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **true**

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

doctest.h

## doctest::detail::types::is\_enum< T > Struct Template Reference

### Static Public Attributes

**static** DOCTEST\_CONSTEXPR **bool** **value** = **\_\_is\_enum**(**T**)

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

doctest.h

## doctest::detail::types::is\_pointer< T > Struct Template Reference

Inheritance diagram for doctest::detail::types::is\_pointer< T >:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::false\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **false**

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

doctest.h

## doctest::detail::types::is\_pointer< T \* > Struct Template Reference

Inheritance diagram for doctest::detail::types::is\_pointer< T \* >:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::true\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **true**

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

doctest.h

## doctest::detail::types::is\_rvalue\_reference< T > Struct Template Reference

Inheritance diagram for doctest::detail::types::is\_rvalue\_reference< T >:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::false\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **false**

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

doctest.h

## doctest::detail::types::is\_rvalue\_reference< T && > Struct Template Reference

Inheritance diagram for doctest::detail::types::is\_rvalue\_reference< T && >:

### Additional Inherited Members

#### Static Public Attributes inherited from doctest::detail::types::true\_type

**static** DOCTEST\_CONSTEXPR **bool** **value** = **true**

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

doctest.h

## doctest::IsNaN< F > Struct Template Reference

### Public Member Functions

**IsNaN** (F f, bool flip=false)

**IsNaN**< F > **operator!** () const

**operator bool** () const

### Public Attributes

F **value**

bool **flipped**

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

doctest.h

## Item Class Reference

Representa um item no sistema.

#include <Item.hpp>

### Public Member Functions

**Item** (int id, std::string name, double price, bool status)

*Construtor da classe* ***Item****.*

std::string **getName** ()

*Obtém o nome do item.*

double **getUnitaryPrice** ()

*Obtém o preço unitário do item.*

int **getId** ()

*Obtém o identificador único do item.*

bool **getStatus** ()

*Obtém o status do item.*

void **changePrice** (double novo)

*Altera o preço do item.*

void **changeStatus** ()

*Altera o status do item.*

### Detailed Description

Representa um item no sistema.

### Constructor & Destructor Documentation

#### Item::Item (int *id*, std::string *name*, double *price*, bool *status*)

Construtor da classe **Item**.

##### Parameters

|  |  |
| --- | --- |
| *id* | Identificador único do item. |
| *name* | Nome do item. |
| *price* | Preço do item. |
| *status* | Status do item. |

### Member Function Documentation

#### void Item::changePrice (double *novo*)

Altera o preço do item.

##### Parameters

|  |  |
| --- | --- |
| *novo* | Novo preço do item. |

#### int Item::getId ()

Obtém o identificador único do item.

##### Returns

Identificador único do item.

#### std::string Item::getName ()

Obtém o nome do item.

##### Returns

Nome do item.

#### bool Item::getStatus ()

Obtém o status do item.

##### Returns

Status do item.

#### double Item::getUnitaryPrice ()

Obtém o preço unitário do item.

##### Returns

Preço unitário do item.

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

**Item.hpp**Item.cpp

## ItemsStorage Class Reference

Gerencia o armazenamento de itens.

#include <ItemsStorage.hpp>

Inheritance diagram for ItemsStorage:

### Public Member Functions

**ItemsStorage** (char const \*file\_path)

*Construtor da classe* ***ItemsStorage****.*

**~ItemsStorage** ()

*Destrutor da classe* ***ItemsStorage****.*

void **write** (const **Item** &item) override

*Escreve um item no armazenamento.*

void **read** (int totalRegistros) override

*Lê dados do armazenamento.*

**Item** **findRandom** (const int tipo)

*Encontra um item no armazenamento.*

**Item** **findByName** (const std::string name)

#### Public Member Functions inherited from StorageManager< Item >

**StorageManager** (char const \*file\_path)

*Construtor da classe* ***StorageManager****.*

### Additional Inherited Members

#### Protected Attributes inherited from StorageManager< Item >

std::ifstream **\_file**

*Busca um registro aleatoriamente no arquivo.*

### Detailed Description

Gerencia o armazenamento de itens.

### Constructor & Destructor Documentation

#### ItemsStorage::ItemsStorage (char const \* *file\_path*)

Construtor da classe **ItemsStorage**.

##### Parameters

|  |  |
| --- | --- |
| *file\_path* | Caminho do arquivo de armazenamento. |

### Member Function Documentation

#### Item ItemsStorage::findRandom (const int *tipo*)

Encontra um item no armazenamento.

##### Parameters

|  |  |
| --- | --- |
| *tipo* | Tipo de item ramdômico que deve ser buscado. 1- Peixes 2- Minérios |

#### void ItemsStorage::read (int *totalRegistros*)[override], [virtual]

Lê dados do armazenamento.

##### Parameters

|  |  |
| --- | --- |
| *data* | Os dados a serem lidos. |

Reimplemented from **StorageManager< Item >** (*p.*).

#### void ItemsStorage::write (const Item & *item*)[override], [virtual]

Escreve um item no armazenamento.

##### Parameters

|  |  |
| --- | --- |
| *item* | O item a ser escrito. |

Reimplemented from **StorageManager< Item >** (*p.*).

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

**ItemsStorage.hpp**ItemsStorage.cpp

## Livestocker Class Reference

Representa um criador de animais no jogo.

#include <Livestocker.hpp>

### Public Member Functions

void **getBacon** (int quantidade, **Player** \*player, **ItemsStorage** \*storage)

*Substitui o item cenoura por bacon.*

void **getEgg** (int quantidade, **Player** \*player, **ItemsStorage** \*storage)

*Substitui o item trigo semente por ovo.*

### Detailed Description

Representa um criador de animais no jogo.

### Member Function Documentation

#### void Livestocker::getBacon (int *quantidade*, Player \* *player*, ItemsStorage \* *storage*)

Substitui o item cenoura por bacon.

##### Parameters

|  |  |
| --- | --- |
| *quantidade* | Número de bacons a serem obtidos. |
| *player* | Ponteiro para o jogador. |
| *storage* | Ponteiro para o gerenciador de items. |

#### void Livestocker::getEgg (int *quantidade*, Player \* *player*, ItemsStorage \* *storage*)

Substitui o item trigo semente por ovo.

##### Parameters

|  |  |
| --- | --- |
| *quantidade* | Número de ovos a serem obtidos. |
| *player* | Ponteiro para o jogador. |
| *storage* | Ponteiro para o gerenciador de items. |

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

**Livestocker.hpp**Livestocker.cpp

## LojaDeItens Class Reference

Classe que representa uma loja de itens no jogo.

#include <LojaDeItens.hpp>

### Public Member Functions

**LojaDeItens** (**Player** \*player, **Inventory** \*inventory, **Item** \*item)

*Construtor da loja de itens.*

void **exibirItens** ()

*Exibe os itens disponíveis na loja e permite ao jogador comprar ou vender.*

void **comprarItem** ()

*Realiza a compra dos itens na loja.*

void **venderItem** ()

*Vende os itens do inventário do jogador.*

### Detailed Description

Classe que representa uma loja de itens no jogo.

### Constructor & Destructor Documentation

#### LojaDeItens::LojaDeItens (Player \* *player*, Inventory \* *inventory*, Item \* *item*)

Construtor da loja de itens.

##### Parameters

|  |  |
| --- | --- |
| *player* | Ponteiro para o jogador que irá interagir com a loja |
| *inventory* | Ponteiro para o inventário do jogador |

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

LojaDeItens.hpp

LojaDeItens.cpp

## Menu Class Reference

Classe que representa um menu.

#include <Menu.hpp>

### Public Member Functions

**Menu** ()

*Construtor da classe* ***Menu****.*

void **listarAtividades** (**Player** \*p, **ItemsStorage** \*storage, **Sistema** \*sistema)

*Lista as atividades disponíveis para um jogador.*

### Detailed Description

Classe que representa um menu.

### Member Function Documentation

#### void Menu::listarAtividades (Player \* *p*, ItemsStorage \* *storage*, Sistema \* *sistema*)

Lista as atividades disponíveis para um jogador.

##### Parameters

|  |  |
| --- | --- |
| *p* | Ponteiro para o jogador. |

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

Menu.hpp

Menu.cpp

## doctest::detail::MessageBuilder Struct Reference

Inheritance diagram for doctest::detail::MessageBuilder:

### Public Member Functions

**MessageBuilder** (const char \*file, int line, assertType::Enum severity)

**MessageBuilder** (const **MessageBuilder** &)=delete

**MessageBuilder** (**MessageBuilder** &&)=delete

**MessageBuilder** & **operator=** (const **MessageBuilder** &)=delete

**MessageBuilder** & **operator=** (**MessageBuilder** &&)=delete

template<typename T > **MessageBuilder** & **operator,** (const T &in)

template<typename T > DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP **MessageBuilder** & **operator<<** (const T &in)

template<typename T > **MessageBuilder** & **operator\*** (const T &in)

bool **log** ()

void **react** ()

### Public Attributes

**std::ostream** \* **m\_stream**

bool **logged** = false

#### Public Attributes inherited from doctest::MessageData

**String** **m\_string**

const char \* **m\_file**

int **m\_line**

assertType::Enum **m\_severity**

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

doctest.h

## doctest::MessageData Struct Reference

Inheritance diagram for doctest::MessageData:

### Public Attributes

**String** **m\_string**

const char \* **m\_file**

int **m\_line**

assertType::Enum **m\_severity**

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

doctest.h

## Miner Class Reference

### Public Member Functions

void **ore** (**Player** \*p, **ItemsStorage** \*storage)

*Gera peixes aletórios para o usuário.*

void **refine** (**Player** \*p)

*Aumenta o valor de venda dos peixes em 10%.*

### Member Function Documentation

#### void Miner::ore (Player \* *p*, ItemsStorage \* *storage*)

Gera peixes aletórios para o usuário.

##### Parameters

|  |  |
| --- | --- |
| *inventario* | invetário do jogador com seus itens |
| *storage* | itens disponíves para serem adiconados ao invetário do player |

##### Returns

o inventário com seus valores alterados

#### void Miner::refine (Player \* *p*)

Aumenta o valor de venda dos peixes em 10%.

##### Parameters

|  |  |
| --- | --- |
| *inventario* | invetário do jogador com seus itens |

##### Returns

o inventário com seus valores alterados

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

MIner.hpp

Miner.cpp

## Player Class Reference

Classe que representa um jogador.

#include <Player.hpp>

### Public Member Functions

**Player** ()

*Construtor padrão da classe* ***Player****.*

**Player** (const std::string name)

*Construtor da classe* ***Player*** *com nome especificado.*

void **functeste** ()

*Função de teste para o jogador.*

void **printInventory** ()

*Imprime o inventário do jogador.*

void **addXp** (float newXp)

*Adiciona experiência ao jogador.*

void **displayProfile** ()

*Exibe o perfil do jogador.*

float **getLevel** ()

*Obtém o nível do jogador.*

**Inventory** \* **getInventory** ()

*Obtém o inventário do jogador.*

### Detailed Description

Classe que representa um jogador.

### Constructor & Destructor Documentation

#### Player::Player (const std::string *name*)

Construtor da classe **Player** com nome especificado.

##### Parameters

|  |  |
| --- | --- |
| *name* | O nome do jogador. |

### Member Function Documentation

#### void Player::addXp (float *newXp*)

Adiciona experiência ao jogador.

##### Parameters

|  |  |
| --- | --- |
| *newXp* | A quantidade de experiência a ser adicionada. |

#### Inventory \* Player::getInventory ()

Obtém o inventário do jogador.

##### Returns

Ponteiro para o inventário do jogador.

#### float Player::getLevel ()

Obtém o nível do jogador.

##### Returns

O nível do jogador.

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

Player.hpp

Player.cpp

## doctest::QueryData Struct Reference

### Public Attributes

const **TestRunStats** \* **run\_stats** = nullptr

const **TestCaseData** \*\* **data** = nullptr

unsigned **num\_data** = 0

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

doctest.h

## doctest::detail::RelationalComparator< int, L, R > Struct Template Reference

### Public Member Functions

bool **operator()** (const DOCTEST\_REF\_WRAP(L), const DOCTEST\_REF\_WRAP(R)) const

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

doctest.h

## doctest::detail::types::remove\_const< T > Struct Template Reference

### Public Types

**using** **type** = **T**

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

doctest.h

## doctest::detail::types::remove\_const< const T > Struct Template Reference

### Public Types

**using** **type** = **T**

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

doctest.h

## doctest::detail::types::remove\_reference< T > Struct Template Reference

### Public Types

**using** **type** = **T**

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

doctest.h

## doctest::detail::types::remove\_reference< T & > Struct Template Reference

### Public Types

**using** **type** = **T**

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

doctest.h

## doctest::detail::types::remove\_reference< T && > Struct Template Reference

### Public Types

**using** **type** = **T**

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

doctest.h

## doctest::detail::Result Struct Reference

### Public Member Functions

**Result** (bool passed, const **String** &decomposition=**String**())

### Public Attributes

bool **m\_passed**

**String** **m\_decomp**

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

doctest.h

## doctest::detail::ResultBuilder Struct Reference

Inheritance diagram for doctest::detail::ResultBuilder:

### Public Member Functions

**ResultBuilder** (assertType::Enum at, const char \*file, int line, const char \*expr, const char \*exception\_type="", const **String** &exception\_string="")

**ResultBuilder** (assertType::Enum at, const char \*file, int line, const char \*expr, const char \*exception\_type, const **Contains** &exception\_string)

void **setResult** (const **Result** &res)

template<int comparison, typename L , typename R > DOCTEST\_NOINLINE bool **binary\_assert** (const DOCTEST\_REF\_WRAP(L) lhs, const DOCTEST\_REF\_WRAP(R) rhs)

template<typename L > DOCTEST\_NOINLINE bool **unary\_assert** (const DOCTEST\_REF\_WRAP(L) val)

void **translateException** ()

bool **log** ()

void **react** () const

#### Public Member Functions inherited from doctest::AssertData

**AssertData** (assertType::Enum at, const char \*file, int line, const char \*expr, const char \*exception\_type, const **StringContains** &exception\_string)

### Additional Inherited Members

#### Public Attributes inherited from doctest::AssertData

const **TestCaseData** \* **m\_test\_case**

assertType::Enum **m\_at**

const char \* **m\_file**

int **m\_line**

const char \* **m\_expr**

bool **m\_failed**

bool **m\_threw**

**String** **m\_exception**

**String** **m\_decomp**

bool **m\_threw\_as**

const char \* **m\_exception\_type**

class DOCTEST\_INTERFACE **doctest::AssertData::StringContains** **m\_exception\_string**

### Member Function Documentation

#### template<typename L > DOCTEST\_NOINLINE bool doctest::detail::ResultBuilder::unary\_assert (const DOCTEST\_REF\_WRAP(L) *val*)[inline]

OCLINT bitwise operator in conditional

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

doctest.h

## doctest::detail::should\_stringify\_as\_underlying\_type< T > Struct Template Reference

### Static Public Attributes

static DOCTEST\_CONSTEXPR bool **value** = **detail::types::is\_enum**<T>::value && !**doctest::detail::has\_insertion\_operator**<T>::value

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

doctest.h

## Sistema Class Reference

Classe que representa o sistema do jogo.

#include <Sistema.hpp>

### Public Member Functions

**Sistema** ()

*Construtor da classe* ***Sistema****.*

void **criarPersonagem** ()

*Cria um personagem no sistema.*

void **printText** (const std::string texto)

*Imprime um texto na saída padrão.*

### Detailed Description

Classe que representa o sistema do jogo.

### Member Function Documentation

#### void Sistema::printText (const std::string *texto*)

Imprime um texto na saída padrão.

##### Parameters

|  |  |
| --- | --- |
| *texto* | O texto a ser impresso. |

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

Sistema.hpp

Sistema.cpp

## StorageManager< T > Class Template Reference

Classe modelo para gerenciamento de armazenamento.

#include <StorageManager.hpp>

### Public Member Functions

**StorageManager** (char const \*file\_path)

*Construtor da classe* ***StorageManager****.*

virtual void **write** (const T &data)

*Escreve dados no arquivo de armazenamento.*

virtual void **read** (int totalRegistros)

*Lê dados do arquivo de armazenamento.*

### Protected Attributes

std::ifstream **\_file**

*Busca um registro aleatoriamente no arquivo.*

### Detailed Description

#### template<typename T>

#### class StorageManager< T >

Classe modelo para gerenciamento de armazenamento.

##### Template Parameters

|  |  |
| --- | --- |
| *T* | Tipo de dados a ser armazenado. |

### Constructor & Destructor Documentation

#### template<typename T > StorageManager< T >::StorageManager (char const \* *file\_path*)

Construtor da classe **StorageManager**.

##### Parameters

|  |  |
| --- | --- |
| *file\_path* | Caminho do arquivo de armazenamento. |

### Member Function Documentation

#### template<typename T > virtual void StorageManager< T >::read (int *totalRegistros*)[virtual]

Lê dados do arquivo de armazenamento.

##### Parameters

|  |  |
| --- | --- |
| *data* | Os dados a serem lidos. |
| *totalRegistros* | O número total de registros a serem lidos. |

Reimplemented in **ItemsStorage** (*p.*).

#### template<typename T > virtual void StorageManager< T >::write (const T & *data*)[virtual]

Escreve dados no arquivo de armazenamento.

##### Parameters

|  |  |
| --- | --- |
| *data* | Os dados a serem escritos. |

Reimplemented in **ItemsStorage** (*p.*).

### Member Data Documentation

#### template<typename T > std::ifstream StorageManager< T >::\_file[protected]

Busca um registro aleatoriamente no arquivo.

##### Returns

registro escolhido. Fluxo de arquivo para leitura.

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

#### StorageManager.hpp

## doctest::String Class Reference

### Public Types

using **size\_type** = DOCTEST\_CONFIG\_STRING\_SIZE\_TYPE

### Public Member Functions

**String** (const char \*in)

**String** (const char \*in, size\_type in\_size)

**String** (**std::istream** &in, size\_type in\_size)

**String** (const **String** &other)

**String** & **operator=** (const **String** &other)

**String** & **operator+=** (const **String** &other)

**String** (**String** &&other) noexcept

**String** & **operator=** (**String** &&other) noexcept

char **operator[]** (size\_type i) const

char & **operator[]** (size\_type i)

const char \* **c\_str** () const

char \* **c\_str** ()

size\_type **size** () const

size\_type **capacity** () const

**String** **substr** (size\_type pos, size\_type cnt=npos) &&

**String** **substr** (size\_type pos, size\_type cnt=npos) const &

size\_type **find** (char ch, size\_type pos=0) const

size\_type **rfind** (char ch, size\_type pos=npos) const

int **compare** (const char \*other, bool no\_case=false) const

int **compare** (const **String** &other, bool no\_case=false) const

### Static Public Attributes

static DOCTEST\_CONSTEXPR size\_type **npos** = static\_cast<size\_type>(-1)

### Friends

DOCTEST\_INTERFACE **std::ostream** & **operator<<** (**std::ostream** &s, const **String** &in)

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

doctest.h

## doctest::AssertData::StringContains Class Reference

### Public Member Functions

**StringContains** (const **String** &str)

**StringContains** (**Contains** cntn)

bool **check** (const **String** &str)

**operator const String &** () const

const char \* **c\_str** () const

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

doctest.h

## doctest::StringMaker< T > Struct Template Reference

Inheritance diagram for doctest::StringMaker< T >:

### Additional Inherited Members

#### Static Public Member Functions inherited from doctest::detail::StringMakerBase< detail::has\_insertion\_operator< T >::value||detail::types::is\_pointer< T >::value||detail::types::is\_array< T >::value >

static **String** **convert** (const DOCTEST\_REF\_WRAP(T))

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

doctest.h

## doctest::detail::StringMakerBase< C > Struct Template Reference

### Static Public Member Functions

template<typename T > static **String** **convert** (const DOCTEST\_REF\_WRAP(T))

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

doctest.h

## doctest::detail::StringMakerBase< true > Struct Reference

### Static Public Member Functions

template<typename T > static **String** **convert** (const DOCTEST\_REF\_WRAP(T) in)

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

doctest.h

## doctest::detail::Subcase Struct Reference

### Public Member Functions

**Subcase** (const **String** &name, const char \*file, int line)

**Subcase** (const **Subcase** &)=delete

**Subcase** (**Subcase** &&)=delete

**Subcase** & **operator=** (const **Subcase** &)=delete

**Subcase** & **operator=** (**Subcase** &&)=delete

**operator bool** () const

### Public Attributes

**SubcaseSignature** **m\_signature**

bool **m\_entered** = false

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

doctest.h

## doctest::SubcaseSignature Struct Reference

### Public Member Functions

bool **operator==** (const **SubcaseSignature** &other) const

bool **operator<** (const **SubcaseSignature** &other) const

### Public Attributes

**String** **m\_name**

const char \* **m\_file**

int **m\_line**

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

doctest.h

## doctest::detail::TestCase Struct Reference

Inheritance diagram for doctest::detail::TestCase:

### Public Member Functions

**TestCase** (funcType test, const char \*file, unsigned line, const **TestSuite** &test\_suite, const **String** &type=**String**(), int template\_id=-1)

**TestCase** (const **TestCase** &other)

**TestCase** (**TestCase** &&)=delete

**TestCase** & **operator=** (const **TestCase** &other)

DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP **TestCase** & **operator=** (**TestCase** &&)=delete

**TestCase** & **operator\*** (const char \*in)

template<typename T > **TestCase** & **operator\*** (const T &in)

bool **operator<** (const **TestCase** &other) const

### Public Attributes

funcType **m\_test**

**String** **m\_type**

int **m\_template\_id**

**String** **m\_full\_name**

#### Public Attributes inherited from doctest::TestCaseData

**String** **m\_file**

unsigned **m\_line**

const char \* **m\_name**

const char \* **m\_test\_suite**

const char \* **m\_description**

bool **m\_skip**

bool **m\_no\_breaks**

bool **m\_no\_output**

bool **m\_may\_fail**

bool **m\_should\_fail**

int **m\_expected\_failures**

double **m\_timeout**

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

doctest.h

## doctest::TestCaseData Struct Reference

Inheritance diagram for doctest::TestCaseData:

### Public Attributes

**String** **m\_file**

unsigned **m\_line**

const char \* **m\_name**

const char \* **m\_test\_suite**

const char \* **m\_description**

bool **m\_skip**

bool **m\_no\_breaks**

bool **m\_no\_output**

bool **m\_may\_fail**

bool **m\_should\_fail**

int **m\_expected\_failures**

double **m\_timeout**

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

doctest.h

## doctest::TestCaseException Struct Reference

### Public Attributes

**String** **error\_string**

bool **is\_crash**

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

doctest.h

## doctest::detail::TestFailureException Struct Reference

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

doctest.h

## doctest::TestRunStats Struct Reference

### Public Attributes

unsigned **numTestCases**

unsigned **numTestCasesPassingFilters**

unsigned **numTestSuitesPassingFilters**

unsigned **numTestCasesFailed**

int **numAsserts**

int **numAssertsFailed**

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

doctest.h

## doctest::detail::TestSuite Struct Reference

### Public Member Functions

**TestSuite** & **operator\*** (const char \*in)

template<typename T > **TestSuite** & **operator\*** (const T &in)

### Public Attributes

const char \* **m\_test\_suite** = nullptr

const char \* **m\_description** = nullptr

bool **m\_skip** = false

bool **m\_no\_breaks** = false

bool **m\_no\_output** = false

bool **m\_may\_fail** = false

bool **m\_should\_fail** = false

int **m\_expected\_failures** = 0

double **m\_timeout** = 0

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

doctest.h

## doctest::detail::types::true\_type Struct Reference

Inheritance diagram for doctest::detail::types::true\_type:

### Static Public Attributes

**static** DOCTEST\_CONSTEXPR **bool** **value** = **true**

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

doctest.h

## std::tuple< Types > Class Template Reference

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

doctest.h

## doctest::detail::types::underlying\_type< T > Struct Template Reference

### Public Types

**using** **type** = **\_\_underlying\_type**(**T**)

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

doctest.h

# File Documentation

## doctest.h

1 // ====================================================================== lgtm [cpp/missing-header-guard]

2 // == DO NOT MODIFY THIS FILE BY HAND - IT IS AUTO GENERATED BY CMAKE! ==

3 // ======================================================================

4 //

5 // doctest.h - the lightest feature-rich C++ single-header testing framework for unit tests and TDD

6 //

7 // Copyright (c) 2016-2023 Viktor Kirilov

8 //

9 // Distributed under the MIT Software License

10 // See accompanying file LICENSE.txt or copy at

11 // https://opensource.org/licenses/MIT

12 //

13 // The documentation can be found at the library's page:

14 // https://github.com/doctest/doctest/blob/master/doc/markdown/readme.md

15 //

16 // =================================================================================================

17 // =================================================================================================

18 // =================================================================================================

19 //

20 // The library is heavily influenced by Catch - https://github.com/catchorg/Catch2

21 // which uses the Boost Software License - Version 1.0

22 // see here - https://github.com/catchorg/Catch2/blob/master/LICENSE.txt

23 //

24 // The concept of subcases (sections in Catch) and expression decomposition are from there.

25 // Some parts of the code are taken directly:

26 // - stringification - the detection of "ostream& operator<<(ostream&, const T&)" and StringMaker<>

27 // - the Approx() helper class for floating point comparison

28 // - colors in the console

29 // - breaking into a debugger

30 // - signal / SEH handling

31 // - timer

32 // - XmlWriter class - thanks to Phil Nash for allowing the direct reuse (AKA copy/paste)

33 //

34 // The expression decomposing templates are taken from lest - https://github.com/martinmoene/lest

35 // which uses the Boost Software License - Version 1.0

36 // see here - https://github.com/martinmoene/lest/blob/master/LICENSE.txt

37 //

38 // =================================================================================================

39 // =================================================================================================

40 // =================================================================================================

41

42 #ifndef DOCTEST\_LIBRARY\_INCLUDED

43 #define DOCTEST\_LIBRARY\_INCLUDED

44

45 // =================================================================================================

46 // == VERSION ======================================================================================

47 // =================================================================================================

48

49 #define DOCTEST\_VERSION\_MAJOR 2

50 #define DOCTEST\_VERSION\_MINOR 4

51 #define DOCTEST\_VERSION\_PATCH 11

52

53 // util we need here

54 #define DOCTEST\_TOSTR\_IMPL(x) #x

55 #define DOCTEST\_TOSTR(x) DOCTEST\_TOSTR\_IMPL(x)

56

57 #define DOCTEST\_VERSION\_STR \

58 DOCTEST\_TOSTR(DOCTEST\_VERSION\_MAJOR) "." \

59 DOCTEST\_TOSTR(DOCTEST\_VERSION\_MINOR) "." \

60 DOCTEST\_TOSTR(DOCTEST\_VERSION\_PATCH)

61

62 #define DOCTEST\_VERSION \

63 (DOCTEST\_VERSION\_MAJOR \* 10000 + DOCTEST\_VERSION\_MINOR \* 100 + DOCTEST\_VERSION\_PATCH)

64

65 // =================================================================================================

66 // == COMPILER VERSION =============================================================================

67 // =================================================================================================

68

69 // ideas for the version stuff are taken from here: https://github.com/cxxstuff/cxx\_detect

70

71 #ifdef \_MSC\_VER

72 #define DOCTEST\_CPLUSPLUS \_MSVC\_LANG

73 #else

74 #define DOCTEST\_CPLUSPLUS \_\_cplusplus

75 #endif

76

77 #define DOCTEST\_COMPILER(MAJOR, MINOR, PATCH) ((MAJOR)\*10000000 + (MINOR)\*100000 + (PATCH))

78

79 // GCC/Clang and GCC/MSVC are mutually exclusive, but Clang/MSVC are not because of clang-cl...

80 #if defined(\_MSC\_VER) && defined(\_MSC\_FULL\_VER)

81 #if \_MSC\_VER == \_MSC\_FULL\_VER / 10000

82 #define DOCTEST\_MSVC DOCTEST\_COMPILER(\_MSC\_VER / 100, \_MSC\_VER % 100, \_MSC\_FULL\_VER % 10000)

83 #else // MSVC

84 #define DOCTEST\_MSVC \

85 DOCTEST\_COMPILER(\_MSC\_VER / 100, (\_MSC\_FULL\_VER / 100000) % 100, \_MSC\_FULL\_VER % 100000)

86 #endif // MSVC

87 #endif // MSVC

88 #if defined(\_\_clang\_\_) && defined(\_\_clang\_minor\_\_) && defined(\_\_clang\_patchlevel\_\_)

89 #define DOCTEST\_CLANG DOCTEST\_COMPILER(\_\_clang\_major\_\_, \_\_clang\_minor\_\_, \_\_clang\_patchlevel\_\_)

90 #elif defined(\_\_GNUC\_\_) && defined(\_\_GNUC\_MINOR\_\_) && defined(\_\_GNUC\_PATCHLEVEL\_\_) && \

91 !defined(\_\_INTEL\_COMPILER)

92 #define DOCTEST\_GCC DOCTEST\_COMPILER(\_\_GNUC\_\_, \_\_GNUC\_MINOR\_\_, \_\_GNUC\_PATCHLEVEL\_\_)

93 #endif // GCC

94 #if defined(\_\_INTEL\_COMPILER)

95 #define DOCTEST\_ICC DOCTEST\_COMPILER(\_\_INTEL\_COMPILER / 100, \_\_INTEL\_COMPILER % 100, 0)

96 #endif // ICC

97

98 #ifndef DOCTEST\_MSVC

99 #define DOCTEST\_MSVC 0

100 #endif // DOCTEST\_MSVC

101 #ifndef DOCTEST\_CLANG

102 #define DOCTEST\_CLANG 0

103 #endif // DOCTEST\_CLANG

104 #ifndef DOCTEST\_GCC

105 #define DOCTEST\_GCC 0

106 #endif // DOCTEST\_GCC

107 #ifndef DOCTEST\_ICC

108 #define DOCTEST\_ICC 0

109 #endif // DOCTEST\_ICC

110

111 // =================================================================================================

112 // == COMPILER WARNINGS HELPERS ====================================================================

113 // =================================================================================================

114

115 #if DOCTEST\_CLANG && !DOCTEST\_ICC

116 #define DOCTEST\_PRAGMA\_TO\_STR(x) \_Pragma(#x)

117 #define DOCTEST\_CLANG\_SUPPRESS\_WARNING\_PUSH \_Pragma("clang diagnostic push")

118 #define DOCTEST\_CLANG\_SUPPRESS\_WARNING(w) DOCTEST\_PRAGMA\_TO\_STR(clang diagnostic ignored w)

119 #define DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP \_Pragma("clang diagnostic pop")

120 #define DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH(w) \

121 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_PUSH DOCTEST\_CLANG\_SUPPRESS\_WARNING(w)

122 #else // DOCTEST\_CLANG

123 #define DOCTEST\_CLANG\_SUPPRESS\_WARNING\_PUSH

124 #define DOCTEST\_CLANG\_SUPPRESS\_WARNING(w)

125 #define DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

126 #define DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH(w)

127 #endif // DOCTEST\_CLANG

128

129 #if DOCTEST\_GCC

130 #define DOCTEST\_PRAGMA\_TO\_STR(x) \_Pragma(#x)

131 #define DOCTEST\_GCC\_SUPPRESS\_WARNING\_PUSH \_Pragma("GCC diagnostic push")

132 #define DOCTEST\_GCC\_SUPPRESS\_WARNING(w) DOCTEST\_PRAGMA\_TO\_STR(GCC diagnostic ignored w)

133 #define DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP \_Pragma("GCC diagnostic pop")

134 #define DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH(w) \

135 DOCTEST\_GCC\_SUPPRESS\_WARNING\_PUSH DOCTEST\_GCC\_SUPPRESS\_WARNING(w)

136 #else // DOCTEST\_GCC

137 #define DOCTEST\_GCC\_SUPPRESS\_WARNING\_PUSH

138 #define DOCTEST\_GCC\_SUPPRESS\_WARNING(w)

139 #define DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

140 #define DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH(w)

141 #endif // DOCTEST\_GCC

142

143 #if DOCTEST\_MSVC

144 #define DOCTEST\_MSVC\_SUPPRESS\_WARNING\_PUSH \_\_pragma(warning(push))

145 #define DOCTEST\_MSVC\_SUPPRESS\_WARNING(w) \_\_pragma(warning(disable : w))

146 #define DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP \_\_pragma(warning(pop))

147 #define DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(w) \

148 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_PUSH DOCTEST\_MSVC\_SUPPRESS\_WARNING(w)

149 #else // DOCTEST\_MSVC

150 #define DOCTEST\_MSVC\_SUPPRESS\_WARNING\_PUSH

151 #define DOCTEST\_MSVC\_SUPPRESS\_WARNING(w)

152 #define DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

153 #define DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(w)

154 #endif // DOCTEST\_MSVC

155

156 // =================================================================================================

157 // == COMPILER WARNINGS ============================================================================

158 // =================================================================================================

159

160 // both the header and the implementation suppress all of these,

161 // so it only makes sense to aggregate them like so

162 #define DOCTEST\_SUPPRESS\_COMMON\_WARNINGS\_PUSH \

163 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_PUSH \

164 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wunknown-pragmas") \

165 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wweak-vtables") \

166 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wpadded") \

167 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wmissing-prototypes") \

168 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wc++98-compat") \

169 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wc++98-compat-pedantic") \

170 \

171 DOCTEST\_GCC\_SUPPRESS\_WARNING\_PUSH \

172 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wunknown-pragmas") \

173 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wpragmas") \

174 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Weffc++") \

175 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wstrict-overflow") \

176 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wstrict-aliasing") \

177 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wmissing-declarations") \

178 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wuseless-cast") \

179 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wnoexcept") \

180 \

181 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_PUSH \

182 /\* these 4 also disabled globally via cmake: \*/ \

183 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4514) /\* unreferenced inline function has been removed \*/ \

184 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4571) /\* SEH related \*/ \

185 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4710) /\* function not inlined \*/ \

186 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4711) /\* function selected for inline expansion\*/ \

187 /\* common ones \*/ \

188 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4616) /\* invalid compiler warning \*/ \

189 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4619) /\* invalid compiler warning \*/ \

190 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4996) /\* The compiler encountered a deprecated declaration \*/ \

191 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4706) /\* assignment within conditional expression \*/ \

192 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4512) /\* 'class' : assignment operator could not be generated \*/ \

193 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4127) /\* conditional expression is constant \*/ \

194 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4820) /\* padding \*/ \

195 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4625) /\* copy constructor was implicitly deleted \*/ \

196 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4626) /\* assignment operator was implicitly deleted \*/ \

197 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5027) /\* move assignment operator implicitly deleted \*/ \

198 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5026) /\* move constructor was implicitly deleted \*/ \

199 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4640) /\* construction of local static object not thread-safe \*/ \

200 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5045) /\* Spectre mitigation for memory load \*/ \

201 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5264) /\* 'variable-name': 'const' variable is not used \*/ \

202 /\* static analysis \*/ \

203 DOCTEST\_MSVC\_SUPPRESS\_WARNING(26439) /\* Function may not throw. Declare it 'noexcept' \*/ \

204 DOCTEST\_MSVC\_SUPPRESS\_WARNING(26495) /\* Always initialize a member variable \*/ \

205 DOCTEST\_MSVC\_SUPPRESS\_WARNING(26451) /\* Arithmetic overflow ... \*/ \

206 DOCTEST\_MSVC\_SUPPRESS\_WARNING(26444) /\* Avoid unnamed objects with custom ctor and dtor... \*/ \

207 DOCTEST\_MSVC\_SUPPRESS\_WARNING(26812) /\* Prefer 'enum class' over 'enum' \*/

208

209 #define DOCTEST\_SUPPRESS\_COMMON\_WARNINGS\_POP \

210 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP \

211 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP \

212 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

213

214 DOCTEST\_SUPPRESS\_COMMON\_WARNINGS\_PUSH

215

216 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_PUSH

217 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wnon-virtual-dtor")

218 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wdeprecated")

219

220 DOCTEST\_GCC\_SUPPRESS\_WARNING\_PUSH

221 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wctor-dtor-privacy")

222 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wnon-virtual-dtor")

223 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wsign-promo")

224

225 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_PUSH

226 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4623) // default constructor was implicitly defined as deleted

227

228 #define DOCTEST\_MAKE\_STD\_HEADERS\_CLEAN\_FROM\_WARNINGS\_ON\_WALL\_BEGIN \

229 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_PUSH \

230 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4548) /\* before comma no effect; expected side - effect \*/ \

231 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4265) /\* virtual functions, but destructor is not virtual \*/ \

232 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4986) /\* exception specification does not match previous \*/ \

233 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4350) /\* 'member1' called instead of 'member2' \*/ \

234 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4668) /\* not defined as a preprocessor macro \*/ \

235 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4365) /\* signed/unsigned mismatch \*/ \

236 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4774) /\* format string not a string literal \*/ \

237 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4820) /\* padding \*/ \

238 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4625) /\* copy constructor was implicitly deleted \*/ \

239 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4626) /\* assignment operator was implicitly deleted \*/ \

240 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5027) /\* move assignment operator implicitly deleted \*/ \

241 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5026) /\* move constructor was implicitly deleted \*/ \

242 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4623) /\* default constructor was implicitly deleted \*/ \

243 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5039) /\* pointer to pot. throwing function passed to extern C \*/ \

244 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5045) /\* Spectre mitigation for memory load \*/ \

245 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5105) /\* macro producing 'defined' has undefined behavior \*/ \

246 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4738) /\* storing float result in memory, loss of performance \*/ \

247 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5262) /\* implicit fall-through \*/

248

249 #define DOCTEST\_MAKE\_STD\_HEADERS\_CLEAN\_FROM\_WARNINGS\_ON\_WALL\_END DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

250

251 // =================================================================================================

252 // == FEATURE DETECTION ============================================================================

253 // =================================================================================================

254

255 // general compiler feature support table: https://en.cppreference.com/w/cpp/compiler\_support

256 // MSVC C++11 feature support table: https://msdn.microsoft.com/en-us/library/hh567368.aspx

257 // GCC C++11 feature support table: https://gcc.gnu.org/projects/cxx-status.html

258 // MSVC version table:

259 // https://en.wikipedia.org/wiki/Microsoft\_Visual\_C%2B%2B#Internal\_version\_numbering

260 // MSVC++ 14.3 (17) \_MSC\_VER == 1930 (Visual Studio 2022)

261 // MSVC++ 14.2 (16) \_MSC\_VER == 1920 (Visual Studio 2019)

262 // MSVC++ 14.1 (15) \_MSC\_VER == 1910 (Visual Studio 2017)

263 // MSVC++ 14.0 \_MSC\_VER == 1900 (Visual Studio 2015)

264 // MSVC++ 12.0 \_MSC\_VER == 1800 (Visual Studio 2013)

265 // MSVC++ 11.0 \_MSC\_VER == 1700 (Visual Studio 2012)

266 // MSVC++ 10.0 \_MSC\_VER == 1600 (Visual Studio 2010)

267 // MSVC++ 9.0 \_MSC\_VER == 1500 (Visual Studio 2008)

268 // MSVC++ 8.0 \_MSC\_VER == 1400 (Visual Studio 2005)

269

270 // Universal Windows Platform support

271 #if defined(WINAPI\_FAMILY) && (WINAPI\_FAMILY == WINAPI\_FAMILY\_APP)

272 #define DOCTEST\_CONFIG\_NO\_WINDOWS\_SEH

273 #endif // WINAPI\_FAMILY

274 #if DOCTEST\_MSVC && !defined(DOCTEST\_CONFIG\_WINDOWS\_SEH)

275 #define DOCTEST\_CONFIG\_WINDOWS\_SEH

276 #endif // MSVC

277 #if defined(DOCTEST\_CONFIG\_NO\_WINDOWS\_SEH) && defined(DOCTEST\_CONFIG\_WINDOWS\_SEH)

278 #undef DOCTEST\_CONFIG\_WINDOWS\_SEH

279 #endif // DOCTEST\_CONFIG\_NO\_WINDOWS\_SEH

280

281 #if !defined(\_WIN32) && !defined(\_\_QNX\_\_) && !defined(DOCTEST\_CONFIG\_POSIX\_SIGNALS) && \

282 !defined(\_\_EMSCRIPTEN\_\_) && !defined(\_\_wasi\_\_)

283 #define DOCTEST\_CONFIG\_POSIX\_SIGNALS

284 #endif // \_WIN32

285 #if defined(DOCTEST\_CONFIG\_NO\_POSIX\_SIGNALS) && defined(DOCTEST\_CONFIG\_POSIX\_SIGNALS)

286 #undef DOCTEST\_CONFIG\_POSIX\_SIGNALS

287 #endif // DOCTEST\_CONFIG\_NO\_POSIX\_SIGNALS

288

289 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

290 #if !defined(\_\_cpp\_exceptions) && !defined(\_\_EXCEPTIONS) && !defined(\_CPPUNWIND) \

291 || defined(\_\_wasi\_\_)

292 #define DOCTEST\_CONFIG\_NO\_EXCEPTIONS

293 #endif // no exceptions

294 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

295

296 #ifdef DOCTEST\_CONFIG\_NO\_EXCEPTIONS\_BUT\_WITH\_ALL\_ASSERTS

297 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

298 #define DOCTEST\_CONFIG\_NO\_EXCEPTIONS

299 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

300 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS\_BUT\_WITH\_ALL\_ASSERTS

301

302 #if defined(DOCTEST\_CONFIG\_NO\_EXCEPTIONS) && !defined(DOCTEST\_CONFIG\_NO\_TRY\_CATCH\_IN\_ASSERTS)

303 #define DOCTEST\_CONFIG\_NO\_TRY\_CATCH\_IN\_ASSERTS

304 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS && !DOCTEST\_CONFIG\_NO\_TRY\_CATCH\_IN\_ASSERTS

305

306 #ifdef \_\_wasi\_\_

307 #define DOCTEST\_CONFIG\_NO\_MULTITHREADING

308 #endif

309

310 #if defined(DOCTEST\_CONFIG\_IMPLEMENT\_WITH\_MAIN) && !defined(DOCTEST\_CONFIG\_IMPLEMENT)

311 #define DOCTEST\_CONFIG\_IMPLEMENT

312 #endif // DOCTEST\_CONFIG\_IMPLEMENT\_WITH\_MAIN

313

314 #if defined(\_WIN32) || defined(\_\_CYGWIN\_\_)

315 #if DOCTEST\_MSVC

316 #define DOCTEST\_SYMBOL\_EXPORT \_\_declspec(dllexport)

317 #define DOCTEST\_SYMBOL\_IMPORT \_\_declspec(dllimport)

318 #else // MSVC

319 #define DOCTEST\_SYMBOL\_EXPORT \_\_attribute\_\_((dllexport))

320 #define DOCTEST\_SYMBOL\_IMPORT \_\_attribute\_\_((dllimport))

321 #endif // MSVC

322 #else // \_WIN32

323 #define DOCTEST\_SYMBOL\_EXPORT \_\_attribute\_\_((visibility("default")))

324 #define DOCTEST\_SYMBOL\_IMPORT

325 #endif // \_WIN32

326

327 #ifdef DOCTEST\_CONFIG\_IMPLEMENTATION\_IN\_DLL

328 #ifdef DOCTEST\_CONFIG\_IMPLEMENT

329 #define DOCTEST\_INTERFACE DOCTEST\_SYMBOL\_EXPORT

330 #else // DOCTEST\_CONFIG\_IMPLEMENT

331 #define DOCTEST\_INTERFACE DOCTEST\_SYMBOL\_IMPORT

332 #endif // DOCTEST\_CONFIG\_IMPLEMENT

333 #else // DOCTEST\_CONFIG\_IMPLEMENTATION\_IN\_DLL

334 #define DOCTEST\_INTERFACE

335 #endif // DOCTEST\_CONFIG\_IMPLEMENTATION\_IN\_DLL

336

337 // needed for extern template instantiations

338 // see https://github.com/fmtlib/fmt/issues/2228

339 #if DOCTEST\_MSVC

340 #define DOCTEST\_INTERFACE\_DECL

341 #define DOCTEST\_INTERFACE\_DEF DOCTEST\_INTERFACE

342 #else // DOCTEST\_MSVC

343 #define DOCTEST\_INTERFACE\_DECL DOCTEST\_INTERFACE

344 #define DOCTEST\_INTERFACE\_DEF

345 #endif // DOCTEST\_MSVC

346

347 #define DOCTEST\_EMPTY

348

349 #if DOCTEST\_MSVC

350 #define DOCTEST\_NOINLINE \_\_declspec(noinline)

351 #define DOCTEST\_UNUSED

352 #define DOCTEST\_ALIGNMENT(x)

353 #elif DOCTEST\_CLANG && DOCTEST\_CLANG < DOCTEST\_COMPILER(3, 5, 0)

354 #define DOCTEST\_NOINLINE

355 #define DOCTEST\_UNUSED

356 #define DOCTEST\_ALIGNMENT(x)

357 #else

358 #define DOCTEST\_NOINLINE \_\_attribute\_\_((noinline))

359 #define DOCTEST\_UNUSED \_\_attribute\_\_((unused))

360 #define DOCTEST\_ALIGNMENT(x) \_\_attribute\_\_((aligned(x)))

361 #endif

362

363 #ifdef DOCTEST\_CONFIG\_NO\_CONTRADICTING\_INLINE

364 #define DOCTEST\_INLINE\_NOINLINE inline

365 #else

366 #define DOCTEST\_INLINE\_NOINLINE inline DOCTEST\_NOINLINE

367 #endif

368

369 #ifndef DOCTEST\_NORETURN

370 #if DOCTEST\_MSVC && (DOCTEST\_MSVC < DOCTEST\_COMPILER(19, 0, 0))

371 #define DOCTEST\_NORETURN

372 #else // DOCTEST\_MSVC

373 #define DOCTEST\_NORETURN [[noreturn]]

374 #endif // DOCTEST\_MSVC

375 #endif // DOCTEST\_NORETURN

376

377 #ifndef DOCTEST\_NOEXCEPT

378 #if DOCTEST\_MSVC && (DOCTEST\_MSVC < DOCTEST\_COMPILER(19, 0, 0))

379 #define DOCTEST\_NOEXCEPT

380 #else // DOCTEST\_MSVC

381 #define DOCTEST\_NOEXCEPT noexcept

382 #endif // DOCTEST\_MSVC

383 #endif // DOCTEST\_NOEXCEPT

384

385 #ifndef DOCTEST\_CONSTEXPR

386 #if DOCTEST\_MSVC && (DOCTEST\_MSVC < DOCTEST\_COMPILER(19, 0, 0))

387 #define DOCTEST\_CONSTEXPR const

388 #define DOCTEST\_CONSTEXPR\_FUNC inline

389 #else // DOCTEST\_MSVC

390 #define DOCTEST\_CONSTEXPR constexpr

391 #define DOCTEST\_CONSTEXPR\_FUNC constexpr

392 #endif // DOCTEST\_MSVC

393 #endif // DOCTEST\_CONSTEXPR

394

395 #ifndef DOCTEST\_NO\_SANITIZE\_INTEGER

396 #if DOCTEST\_CLANG >= DOCTEST\_COMPILER(3, 7, 0)

397 #define DOCTEST\_NO\_SANITIZE\_INTEGER \_\_attribute\_\_((no\_sanitize("integer")))

398 #else

399 #define DOCTEST\_NO\_SANITIZE\_INTEGER

400 #endif

401 #endif // DOCTEST\_NO\_SANITIZE\_INTEGER

402

403 // =================================================================================================

404 // == FEATURE DETECTION END ========================================================================

405 // =================================================================================================

406

407 #define DOCTEST\_DECLARE\_INTERFACE(name) \

408 virtual ~name(); \

409 name() = default; \

410 name(const name&) = delete; \

411 name(name&&) = delete; \

412 name& operator=(const name&) = delete; \

413 name& operator=(name&&) = delete;

414

415 #define DOCTEST\_DEFINE\_INTERFACE(name) \

416 name::~name() = default;

417

418 // internal macros for string concatenation and anonymous variable name generation

419 #define DOCTEST\_CAT\_IMPL(s1, s2) s1##s2

420 #define DOCTEST\_CAT(s1, s2) DOCTEST\_CAT\_IMPL(s1, s2)

421 #ifdef \_\_COUNTER\_\_ // not standard and may be missing for some compilers

422 #define DOCTEST\_ANONYMOUS(x) DOCTEST\_CAT(x, \_\_COUNTER\_\_)

423 #else // \_\_COUNTER\_\_

424 #define DOCTEST\_ANONYMOUS(x) DOCTEST\_CAT(x, \_\_LINE\_\_)

425 #endif // \_\_COUNTER\_\_

426

427 #ifndef DOCTEST\_CONFIG\_ASSERTION\_PARAMETERS\_BY\_VALUE

428 #define DOCTEST\_REF\_WRAP(x) x&

429 #else // DOCTEST\_CONFIG\_ASSERTION\_PARAMETERS\_BY\_VALUE

430 #define DOCTEST\_REF\_WRAP(x) x

431 #endif // DOCTEST\_CONFIG\_ASSERTION\_PARAMETERS\_BY\_VALUE

432

433 // not using \_\_APPLE\_\_ because... this is how Catch does it

434 #ifdef \_\_MAC\_OS\_X\_VERSION\_MIN\_REQUIRED

435 #define DOCTEST\_PLATFORM\_MAC

436 #elif defined(\_\_IPHONE\_OS\_VERSION\_MIN\_REQUIRED)

437 #define DOCTEST\_PLATFORM\_IPHONE

438 #elif defined(\_WIN32)

439 #define DOCTEST\_PLATFORM\_WINDOWS

440 #elif defined(\_\_wasi\_\_)

441 #define DOCTEST\_PLATFORM\_WASI

442 #else // DOCTEST\_PLATFORM

443 #define DOCTEST\_PLATFORM\_LINUX

444 #endif // DOCTEST\_PLATFORM

445

446 namespace doctest { namespace detail {

447 static DOCTEST\_CONSTEXPR int consume(const int\*, int) noexcept { return 0; }

448 }}

449

450 #define DOCTEST\_GLOBAL\_NO\_WARNINGS(var, ...) \

451 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wglobal-constructors") \

452 static const int var = doctest::detail::consume(&var, \_\_VA\_ARGS\_\_); \

453 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

454

455 #ifndef DOCTEST\_BREAK\_INTO\_DEBUGGER

456 // should probably take a look at https://github.com/scottt/debugbreak

457 #ifdef DOCTEST\_PLATFORM\_LINUX

458 #if defined(\_\_GNUC\_\_) && (defined(\_\_i386) || defined(\_\_x86\_64))

459 // Break at the location of the failing check if possible

460 #define DOCTEST\_BREAK\_INTO\_DEBUGGER() \_\_asm\_\_("int $3\n" : :) // NOLINT(hicpp-no-assembler)

461 #else

462 #include <signal.h>

463 #define DOCTEST\_BREAK\_INTO\_DEBUGGER() raise(SIGTRAP)

464 #endif

465 #elif defined(DOCTEST\_PLATFORM\_MAC)

466 #if defined(\_\_x86\_64) || defined(\_\_x86\_64\_\_) || defined(\_\_amd64\_\_) || defined(\_\_i386)

467 #define DOCTEST\_BREAK\_INTO\_DEBUGGER() \_\_asm\_\_("int $3\n" : :) // NOLINT(hicpp-no-assembler)

468 #elif defined(\_\_ppc\_\_) || defined(\_\_ppc64\_\_)

469 // https://www.cocoawithlove.com/2008/03/break-into-debugger.html

470 #define DOCTEST\_BREAK\_INTO\_DEBUGGER() \_\_asm\_\_("li r0, 20\nsc\nnop\nli r0, 37\nli r4, 2\nsc\nnop\n": : : "memory","r0","r3","r4") // NOLINT(hicpp-no-assembler)

471 #else

472 #define DOCTEST\_BREAK\_INTO\_DEBUGGER() \_\_asm\_\_("brk #0"); // NOLINT(hicpp-no-assembler)

473 #endif

474 #elif DOCTEST\_MSVC

475 #define DOCTEST\_BREAK\_INTO\_DEBUGGER() \_\_debugbreak()

476 #elif defined(\_\_MINGW32\_\_)

477 DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wredundant-decls")

478 extern "C" \_\_declspec(dllimport) void \_\_stdcall DebugBreak();

479 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

480 #define DOCTEST\_BREAK\_INTO\_DEBUGGER() ::DebugBreak()

481 #else // linux

482 #define DOCTEST\_BREAK\_INTO\_DEBUGGER() (static\_cast<void>(0))

483 #endif // linux

484 #endif // DOCTEST\_BREAK\_INTO\_DEBUGGER

485

486 // this is kept here for backwards compatibility since the config option was changed

487 #ifdef DOCTEST\_CONFIG\_USE\_IOSFWD

488 #ifndef DOCTEST\_CONFIG\_USE\_STD\_HEADERS

489 #define DOCTEST\_CONFIG\_USE\_STD\_HEADERS

490 #endif

491 #endif // DOCTEST\_CONFIG\_USE\_IOSFWD

492

493 // for clang - always include ciso646 (which drags some std stuff) because

494 // we want to check if we are using libc++ with the \_LIBCPP\_VERSION macro in

495 // which case we don't want to forward declare stuff from std - for reference:

496 // https://github.com/doctest/doctest/issues/126

497 // https://github.com/doctest/doctest/issues/356

498 #if DOCTEST\_CLANG

499 #include <ciso646>

500 #endif // clang

501

502 #ifdef \_LIBCPP\_VERSION

503 #ifndef DOCTEST\_CONFIG\_USE\_STD\_HEADERS

504 #define DOCTEST\_CONFIG\_USE\_STD\_HEADERS

505 #endif

506 #endif // \_LIBCPP\_VERSION

507

508 #ifdef DOCTEST\_CONFIG\_USE\_STD\_HEADERS

509 #ifndef DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

510 #define DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

511 #endif // DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

512 DOCTEST\_MAKE\_STD\_HEADERS\_CLEAN\_FROM\_WARNINGS\_ON\_WALL\_BEGIN

513 #include <cstddef>

514 #include <ostream>

515 #include <istream>

516 DOCTEST\_MAKE\_STD\_HEADERS\_CLEAN\_FROM\_WARNINGS\_ON\_WALL\_END

517 #else // DOCTEST\_CONFIG\_USE\_STD\_HEADERS

518

519 // Forward declaring 'X' in namespace std is not permitted by the C++ Standard.

520 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4643)

521

522 namespace std { // NOLINT(cert-dcl58-cpp)

523 typedef decltype(nullptr) nullptr\_t; // NOLINT(modernize-use-using)

524 typedef decltype(sizeof(void\*)) size\_t; // NOLINT(modernize-use-using)

525 template <class charT>

526 struct char\_traits;

527 template <>

528 struct char\_traits<char>;

529 template <class charT, class traits>

530 class basic\_ostream; // NOLINT(fuchsia-virtual-inheritance)

531 typedef basic\_ostream<char, char\_traits<char>> ostream; // NOLINT(modernize-use-using)

532 template<class traits>

533 // NOLINTNEXTLINE

534 basic\_ostream<char, traits>& operator<<(basic\_ostream<char, traits>&, const char\*);

535 template <class charT, class traits>

536 class basic\_istream;

537 typedef basic\_istream<char, char\_traits<char>> istream; // NOLINT(modernize-use-using)

538 template <class... Types>

539 class tuple;

540 #if DOCTEST\_MSVC >= DOCTEST\_COMPILER(19, 20, 0)

541 // see this issue on why this is needed: https://github.com/doctest/doctest/issues/183

542 template <class Ty>

543 class allocator;

544 template <class Elem, class Traits, class Alloc>

545 class basic\_string;

546 using string = basic\_string<char, char\_traits<char>, allocator<char>>;

547 #endif // VS 2019

548 } // namespace std

549

550 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

551

552 #endif // DOCTEST\_CONFIG\_USE\_STD\_HEADERS

553

554 #ifdef DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

555 #include <type\_traits>

556 #endif // DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

557

558 namespace doctest {

559

560 using std::size\_t;

561

562 DOCTEST\_INTERFACE extern bool is\_running\_in\_test;

563

564 #ifndef DOCTEST\_CONFIG\_STRING\_SIZE\_TYPE

565 #define DOCTEST\_CONFIG\_STRING\_SIZE\_TYPE unsigned

566 #endif

567

568 // A 24 byte string class (can be as small as 17 for x64 and 13 for x86) that can hold strings with length

569 // of up to 23 chars on the stack before going on the heap - the last byte of the buffer is used for:

570 // - "is small" bit - the highest bit - if "0" then it is small - otherwise its "1" (128)

571 // - if small - capacity left before going on the heap - using the lowest 5 bits

572 // - if small - 2 bits are left unused - the second and third highest ones

573 // - if small - acts as a null terminator if strlen() is 23 (24 including the null terminator)

574 // and the "is small" bit remains "0" ("as well as the capacity left") so its OK

575 // Idea taken from this lecture about the string implementation of facebook/folly - fbstring

576 // https://www.youtube.com/watch?v=kPR8h4-qZdk

577 // TODO:

578 // - optimizations - like not deleting memory unnecessarily in operator= and etc.

579 // - resize/reserve/clear

580 // - replace

581 // - back/front

582 // - iterator stuff

583 // - find & friends

584 // - push\_back/pop\_back

585 // - assign/insert/erase

586 // - relational operators as free functions - taking const char\* as one of the params

587 class DOCTEST\_INTERFACE String

588 {

589 public:

590 using size\_type = DOCTEST\_CONFIG\_STRING\_SIZE\_TYPE;

591

592 private:

593 static DOCTEST\_CONSTEXPR size\_type len = 24;

594 static DOCTEST\_CONSTEXPR size\_type last = len - 1;

595

596 struct view // len should be more than sizeof(view) - because of the final byte for flags

597 {

598 char\* ptr;

599 size\_type size;

600 size\_type capacity;

601 };

602

603 union

604 {

605 char buf[len]; // NOLINT(\*-avoid-c-arrays)

606 view data;

607 };

608

609 char\* allocate(size\_type sz);

610

611 bool isOnStack() const noexcept { return (buf[last] & 128) == 0; }

612 void setOnHeap() noexcept;

613 void setLast(size\_type in = last) noexcept;

614 void setSize(size\_type sz) noexcept;

615

616 void copy(const String& other);

617

618 public:

619 static DOCTEST\_CONSTEXPR size\_type npos = static\_cast<size\_type>(-1);

620

621 String() noexcept;

622 ~String();

623

624 // cppcheck-suppress noExplicitConstructor

625 String(const char\* in);

626 String(const char\* in, size\_type in\_size);

627

628 String(std::istream& in, size\_type in\_size);

629

630 String(const String& other);

631 String& operator=(const String& other);

632

633 String& operator+=(const String& other);

634

635 String(String&& other) noexcept;

636 String& operator=(String&& other) noexcept;

637

638 char operator[](size\_type i) const;

639 char& operator[](size\_type i);

640

641 // the only functions I'm willing to leave in the interface - available for inlining

642 const char\* c\_str() const { return const\_cast<String\*>(this)->c\_str(); } // NOLINT

643 char\* c\_str() {

644 if (isOnStack()) {

645 return reinterpret\_cast<char\*>(buf);

646 }

647 return data.ptr;

648 }

649

650 size\_type size() const;

651 size\_type capacity() const;

652

653 String substr(size\_type pos, size\_type cnt = npos) &&;

654 String substr(size\_type pos, size\_type cnt = npos) const &;

655

656 size\_type find(char ch, size\_type pos = 0) const;

657 size\_type rfind(char ch, size\_type pos = npos) const;

658

659 int compare(const char\* other, bool no\_case = false) const;

660 int compare(const String& other, bool no\_case = false) const;

661

662 friend DOCTEST\_INTERFACE std::ostream& operator<<(std::ostream& s, const String& in);

663 };

664

665 DOCTEST\_INTERFACE String operator+(const String& lhs, const String& rhs);

666

667 DOCTEST\_INTERFACE bool operator==(const String& lhs, const String& rhs);

668 DOCTEST\_INTERFACE bool operator!=(const String& lhs, const String& rhs);

669 DOCTEST\_INTERFACE bool operator<(const String& lhs, const String& rhs);

670 DOCTEST\_INTERFACE bool operator>(const String& lhs, const String& rhs);

671 DOCTEST\_INTERFACE bool operator<=(const String& lhs, const String& rhs);

672 DOCTEST\_INTERFACE bool operator>=(const String& lhs, const String& rhs);

673

674 class DOCTEST\_INTERFACE Contains {

675 public:

676 explicit Contains(const String& string);

677

678 bool checkWith(const String& other) const;

679

680 String string;

681 };

682

683 DOCTEST\_INTERFACE String toString(const Contains& in);

684

685 DOCTEST\_INTERFACE bool operator==(const String& lhs, const Contains& rhs);

686 DOCTEST\_INTERFACE bool operator==(const Contains& lhs, const String& rhs);

687 DOCTEST\_INTERFACE bool operator!=(const String& lhs, const Contains& rhs);

688 DOCTEST\_INTERFACE bool operator!=(const Contains& lhs, const String& rhs);

689

690 namespace Color {

691 enum Enum

692 {

693 None = 0,

694 White,

695 Red,

696 Green,

697 Blue,

698 Cyan,

699 Yellow,

700 Grey,

701

702 Bright = 0x10,

703

704 BrightRed = Bright | Red,

705 BrightGreen = Bright | Green,

706 LightGrey = Bright | Grey,

707 BrightWhite = Bright | White

708 };

709

710 DOCTEST\_INTERFACE std::ostream& operator<<(std::ostream& s, Color::Enum code);

711 } // namespace Color

712

713 namespace assertType {

714 enum Enum

715 {

716 // macro traits

717

718 is\_warn = 1,

719 is\_check = 2 \* is\_warn,

720 is\_require = 2 \* is\_check,

721

722 is\_normal = 2 \* is\_require,

723 is\_throws = 2 \* is\_normal,

724 is\_throws\_as = 2 \* is\_throws,

725 is\_throws\_with = 2 \* is\_throws\_as,

726 is\_nothrow = 2 \* is\_throws\_with,

727

728 is\_false = 2 \* is\_nothrow,

729 is\_unary = 2 \* is\_false, // not checked anywhere - used just to distinguish the types

730

731 is\_eq = 2 \* is\_unary,

732 is\_ne = 2 \* is\_eq,

733

734 is\_lt = 2 \* is\_ne,

735 is\_gt = 2 \* is\_lt,

736

737 is\_ge = 2 \* is\_gt,

738 is\_le = 2 \* is\_ge,

739

740 // macro types

741

742 DT\_WARN = is\_normal | is\_warn,

743 DT\_CHECK = is\_normal | is\_check,

744 DT\_REQUIRE = is\_normal | is\_require,

745

746 DT\_WARN\_FALSE = is\_normal | is\_false | is\_warn,

747 DT\_CHECK\_FALSE = is\_normal | is\_false | is\_check,

748 DT\_REQUIRE\_FALSE = is\_normal | is\_false | is\_require,

749

750 DT\_WARN\_THROWS = is\_throws | is\_warn,

751 DT\_CHECK\_THROWS = is\_throws | is\_check,

752 DT\_REQUIRE\_THROWS = is\_throws | is\_require,

753

754 DT\_WARN\_THROWS\_AS = is\_throws\_as | is\_warn,

755 DT\_CHECK\_THROWS\_AS = is\_throws\_as | is\_check,

756 DT\_REQUIRE\_THROWS\_AS = is\_throws\_as | is\_require,

757

758 DT\_WARN\_THROWS\_WITH = is\_throws\_with | is\_warn,

759 DT\_CHECK\_THROWS\_WITH = is\_throws\_with | is\_check,

760 DT\_REQUIRE\_THROWS\_WITH = is\_throws\_with | is\_require,

761

762 DT\_WARN\_THROWS\_WITH\_AS = is\_throws\_with | is\_throws\_as | is\_warn,

763 DT\_CHECK\_THROWS\_WITH\_AS = is\_throws\_with | is\_throws\_as | is\_check,

764 DT\_REQUIRE\_THROWS\_WITH\_AS = is\_throws\_with | is\_throws\_as | is\_require,

765

766 DT\_WARN\_NOTHROW = is\_nothrow | is\_warn,

767 DT\_CHECK\_NOTHROW = is\_nothrow | is\_check,

768 DT\_REQUIRE\_NOTHROW = is\_nothrow | is\_require,

769

770 DT\_WARN\_EQ = is\_normal | is\_eq | is\_warn,

771 DT\_CHECK\_EQ = is\_normal | is\_eq | is\_check,

772 DT\_REQUIRE\_EQ = is\_normal | is\_eq | is\_require,

773

774 DT\_WARN\_NE = is\_normal | is\_ne | is\_warn,

775 DT\_CHECK\_NE = is\_normal | is\_ne | is\_check,

776 DT\_REQUIRE\_NE = is\_normal | is\_ne | is\_require,

777

778 DT\_WARN\_GT = is\_normal | is\_gt | is\_warn,

779 DT\_CHECK\_GT = is\_normal | is\_gt | is\_check,

780 DT\_REQUIRE\_GT = is\_normal | is\_gt | is\_require,

781

782 DT\_WARN\_LT = is\_normal | is\_lt | is\_warn,

783 DT\_CHECK\_LT = is\_normal | is\_lt | is\_check,

784 DT\_REQUIRE\_LT = is\_normal | is\_lt | is\_require,

785

786 DT\_WARN\_GE = is\_normal | is\_ge | is\_warn,

787 DT\_CHECK\_GE = is\_normal | is\_ge | is\_check,

788 DT\_REQUIRE\_GE = is\_normal | is\_ge | is\_require,

789

790 DT\_WARN\_LE = is\_normal | is\_le | is\_warn,

791 DT\_CHECK\_LE = is\_normal | is\_le | is\_check,

792 DT\_REQUIRE\_LE = is\_normal | is\_le | is\_require,

793

794 DT\_WARN\_UNARY = is\_normal | is\_unary | is\_warn,

795 DT\_CHECK\_UNARY = is\_normal | is\_unary | is\_check,

796 DT\_REQUIRE\_UNARY = is\_normal | is\_unary | is\_require,

797

798 DT\_WARN\_UNARY\_FALSE = is\_normal | is\_false | is\_unary | is\_warn,

799 DT\_CHECK\_UNARY\_FALSE = is\_normal | is\_false | is\_unary | is\_check,

800 DT\_REQUIRE\_UNARY\_FALSE = is\_normal | is\_false | is\_unary | is\_require,

801 };

802 } // namespace assertType

803

804 DOCTEST\_INTERFACE const char\* assertString(assertType::Enum at);

805 DOCTEST\_INTERFACE const char\* failureString(assertType::Enum at);

806 DOCTEST\_INTERFACE const char\* skipPathFromFilename(const char\* file);

807

808 struct DOCTEST\_INTERFACE TestCaseData

809 {

810 String m\_file; // the file in which the test was registered (using String - see #350)

811 unsigned m\_line; // the line where the test was registered

812 const char\* m\_name; // name of the test case

813 const char\* m\_test\_suite; // the test suite in which the test was added

814 const char\* m\_description;

815 bool m\_skip;

816 bool m\_no\_breaks;

817 bool m\_no\_output;

818 bool m\_may\_fail;

819 bool m\_should\_fail;

820 int m\_expected\_failures;

821 double m\_timeout;

822 };

823

824 struct DOCTEST\_INTERFACE AssertData

825 {

826 // common - for all asserts

827 const TestCaseData\* m\_test\_case;

828 assertType::Enum m\_at;

829 const char\* m\_file;

830 int m\_line;

831 const char\* m\_expr;

832 bool m\_failed;

833

834 // exception-related - for all asserts

835 bool m\_threw;

836 String m\_exception;

837

838 // for normal asserts

839 String m\_decomp;

840

841 // for specific exception-related asserts

842 bool m\_threw\_as;

843 const char\* m\_exception\_type;

844

845 class DOCTEST\_INTERFACE StringContains {

846 private:

847 Contains content;

848 bool isContains;

849

850 public:

851 StringContains(const String& str) : content(str), isContains(false) { }

852 StringContains(Contains cntn) : content(static\_cast<Contains&&>(cntn)), isContains(true) { }

853

854 bool check(const String& str) { return isContains ? (content == str) : (content.string == str); }

855

856 operator const String&() const { return content.string; }

857

858 const char\* c\_str() const { return content.string.c\_str(); }

859 } m\_exception\_string;

860

861 AssertData(assertType::Enum at, const char\* file, int line, const char\* expr,

862 const char\* exception\_type, const StringContains& exception\_string);

863 };

864

865 struct DOCTEST\_INTERFACE MessageData

866 {

867 String m\_string;

868 const char\* m\_file;

869 int m\_line;

870 assertType::Enum m\_severity;

871 };

872

873 struct DOCTEST\_INTERFACE SubcaseSignature

874 {

875 String m\_name;

876 const char\* m\_file;

877 int m\_line;

878

879 bool operator==(const SubcaseSignature& other) const;

880 bool operator<(const SubcaseSignature& other) const;

881 };

882

883 struct DOCTEST\_INTERFACE IContextScope

884 {

885 DOCTEST\_DECLARE\_INTERFACE(IContextScope)

886 virtual void stringify(std::ostream\*) const = 0;

887 };

888

889 namespace detail {

890 struct DOCTEST\_INTERFACE TestCase;

891 } // namespace detail

892

893 struct ContextOptions

894 {

895 std::ostream\* cout = nullptr; // stdout stream

896 String binary\_name; // the test binary name

897

898 const detail::TestCase\* currentTest = nullptr;

899

900 // == parameters from the command line

901 String out; // output filename

902 String order\_by; // how tests should be ordered

903 unsigned rand\_seed; // the seed for rand ordering

904

905 unsigned first; // the first (matching) test to be executed

906 unsigned last; // the last (matching) test to be executed

907

908 int abort\_after; // stop tests after this many failed assertions

909 int subcase\_filter\_levels; // apply the subcase filters for the first N levels

910

911 bool success; // include successful assertions in output

912 bool case\_sensitive; // if filtering should be case sensitive

913 bool exit; // if the program should be exited after the tests are ran/whatever

914 bool duration; // print the time duration of each test case

915 bool minimal; // minimal console output (only test failures)

916 bool quiet; // no console output

917 bool no\_throw; // to skip exceptions-related assertion macros

918 bool no\_exitcode; // if the framework should return 0 as the exitcode

919 bool no\_run; // to not run the tests at all (can be done with an "\*" exclude)

920 bool no\_intro; // to not print the intro of the framework

921 bool no\_version; // to not print the version of the framework

922 bool no\_colors; // if output to the console should be colorized

923 bool force\_colors; // forces the use of colors even when a tty cannot be detected

924 bool no\_breaks; // to not break into the debugger

925 bool no\_skip; // don't skip test cases which are marked to be skipped

926 bool gnu\_file\_line; // if line numbers should be surrounded with :x: and not (x):

927 bool no\_path\_in\_filenames; // if the path to files should be removed from the output

928 bool no\_line\_numbers; // if source code line numbers should be omitted from the output

929 bool no\_debug\_output; // no output in the debug console when a debugger is attached

930 bool no\_skipped\_summary; // don't print "skipped" in the summary !!! UNDOCUMENTED !!!

931 bool no\_time\_in\_output; // omit any time/timestamps from output !!! UNDOCUMENTED !!!

932

933 bool help; // to print the help

934 bool version; // to print the version

935 bool count; // if only the count of matching tests is to be retrieved

936 bool list\_test\_cases; // to list all tests matching the filters

937 bool list\_test\_suites; // to list all suites matching the filters

938 bool list\_reporters; // lists all registered reporters

939 };

940

941 namespace detail {

942 namespace types {

943 #ifdef DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

944 using namespace std;

945 #else

946 template <bool COND, typename T = void>

947 struct enable\_if { };

948

949 template <typename T>

950 struct enable\_if<true, T> { using type = T; };

951

952 struct true\_type { static DOCTEST\_CONSTEXPR bool value = true; };

953 struct false\_type { static DOCTEST\_CONSTEXPR bool value = false; };

954

955 template <typename T> struct remove\_reference { using type = T; };

956 template <typename T> struct remove\_reference<T&> { using type = T; };

957 template <typename T> struct remove\_reference<T&&> { using type = T; };

958

959 template <typename T> struct is\_rvalue\_reference : false\_type { };

960 template <typename T> struct is\_rvalue\_reference<T&&> : true\_type { };

961

962 template<typename T> struct remove\_const { using type = T; };

963 template <typename T> struct remove\_const<const T> { using type = T; };

964

965 // Compiler intrinsics

966 template <typename T> struct is\_enum { static DOCTEST\_CONSTEXPR bool value = \_\_is\_enum(T); };

967 template <typename T> struct underlying\_type { using type = \_\_underlying\_type(T); };

968

969 template <typename T> struct is\_pointer : false\_type { };

970 template <typename T> struct is\_pointer<T\*> : true\_type { };

971

972 template <typename T> struct is\_array : false\_type { };

973 // NOLINTNEXTLINE(\*-avoid-c-arrays)

974 template <typename T, size\_t SIZE> struct is\_array<T[SIZE]> : true\_type { };

975 #endif

976 }

977

978 // <utility>

979 template <typename T>

980 T&& declval();

981

982 template <class T>

983 DOCTEST\_CONSTEXPR\_FUNC T&& forward(typename types::remove\_reference<T>::type& t) DOCTEST\_NOEXCEPT {

984 return static\_cast<T&&>(t);

985 }

986

987 template <class T>

988 DOCTEST\_CONSTEXPR\_FUNC T&& forward(typename types::remove\_reference<T>::type&& t) DOCTEST\_NOEXCEPT {

989 return static\_cast<T&&>(t);

990 }

991

992 template <typename T>

993 struct deferred\_false : types::false\_type { };

994

995 // MSVS 2015 :(

996 #if !DOCTEST\_CLANG && defined(\_MSC\_VER) && \_MSC\_VER <= 1900

997 template <typename T, typename = void>

998 struct has\_global\_insertion\_operator : types::false\_type { };

999

1000 template <typename T>

1001 struct has\_global\_insertion\_operator<T, decltype(::operator<<(declval<std::ostream&>(), declval<const T&>()), void())> : types::true\_type { };

1002

1003 template <typename T, typename = void>

1004 struct has\_insertion\_operator { static DOCTEST\_CONSTEXPR bool value = has\_global\_insertion\_operator<T>::value; };

1005

1006 template <typename T, bool global>

1007 struct insert\_hack;

1008

1009 template <typename T>

1010 struct insert\_hack<T, true> {

1011 static void insert(std::ostream& os, const T& t) { ::operator<<(os, t); }

1012 };

1013

1014 template <typename T>

1015 struct insert\_hack<T, false> {

1016 static void insert(std::ostream& os, const T& t) { operator<<(os, t); }

1017 };

1018

1019 template <typename T>

1020 using insert\_hack\_t = insert\_hack<T, has\_global\_insertion\_operator<T>::value>;

1021 #else

1022 template <typename T, typename = void>

1023 struct has\_insertion\_operator : types::false\_type { };

1024 #endif

1025

1026 template <typename T>

1027 struct has\_insertion\_operator<T, decltype(operator<<(declval<std::ostream&>(), declval<const T&>()), void())> : types::true\_type { };

1028

1029 template <typename T>

1030 struct should\_stringify\_as\_underlying\_type {

1031 static DOCTEST\_CONSTEXPR bool value = detail::types::is\_enum<T>::value && !doctest::detail::has\_insertion\_operator<T>::value;

1032 };

1033

1034 DOCTEST\_INTERFACE std::ostream\* tlssPush();

1035 DOCTEST\_INTERFACE String tlssPop();

1036

1037 template <bool C>

1038 struct StringMakerBase {

1039 template <typename T>

1040 static String convert(const DOCTEST\_REF\_WRAP(T)) {

1041 #ifdef DOCTEST\_CONFIG\_REQUIRE\_STRINGIFICATION\_FOR\_ALL\_USED\_TYPES

1042 static\_assert(deferred\_false<T>::value, "No stringification detected for type T. See string conversion manual");

1043 #endif

1044 return "{?}";

1045 }

1046 };

1047

1048 template <typename T>

1049 struct filldata;

1050

1051 template <typename T>

1052 void filloss(std::ostream\* stream, const T& in) {

1053 filldata<T>::fill(stream, in);

1054 }

1055

1056 template <typename T, size\_t N>

1057 void filloss(std::ostream\* stream, const T (&in)[N]) { // NOLINT(\*-avoid-c-arrays)

1058 // T[N], T(&)[N], T(&&)[N] have same behaviour.

1059 // Hence remove reference.

1060 filloss<typename types::remove\_reference<decltype(in)>::type>(stream, in);

1061 }

1062

1063 template <typename T>

1064 String toStream(const T& in) {

1065 std::ostream\* stream = tlssPush();

1066 filloss(stream, in);

1067 return tlssPop();

1068 }

1069

1070 template <>

1071 struct StringMakerBase<true> {

1072 template <typename T>

1073 static String convert(const DOCTEST\_REF\_WRAP(T) in) {

1074 return toStream(in);

1075 }

1076 };

1077 } // namespace detail

1078

1079 template <typename T>

1080 struct StringMaker : public detail::StringMakerBase<

1081 detail::has\_insertion\_operator<T>::value || detail::types::is\_pointer<T>::value || detail::types::is\_array<T>::value>

1082 {};

1083

1084 #ifndef DOCTEST\_STRINGIFY

1085 #ifdef DOCTEST\_CONFIG\_DOUBLE\_STRINGIFY

1086 #define DOCTEST\_STRINGIFY(...) toString(toString(\_\_VA\_ARGS\_\_))

1087 #else

1088 #define DOCTEST\_STRINGIFY(...) toString(\_\_VA\_ARGS\_\_)

1089 #endif

1090 #endif

1091

1092 template <typename T>

1093 String toString() {

1094 #if DOCTEST\_CLANG == 0 && DOCTEST\_GCC == 0 && DOCTEST\_ICC == 0

1095 String ret = \_\_FUNCSIG\_\_; // class doctest::String \_\_cdecl doctest::toString<TYPE>(void)

1096 String::size\_type beginPos = ret.find('<');

1097 return ret.substr(beginPos + 1, ret.size() - beginPos - static\_cast<String::size\_type>(sizeof(">(void)")));

1098 #else

1099 String ret = \_\_PRETTY\_FUNCTION\_\_; // doctest::String toString() [with T = TYPE]

1100 String::size\_type begin = ret.find('=') + 2;

1101 return ret.substr(begin, ret.size() - begin - 1);

1102 #endif

1103 }

1104

1105 template <typename T, typename detail::types::enable\_if<!detail::should\_stringify\_as\_underlying\_type<T>::value, bool>::type = true>

1106 String toString(const DOCTEST\_REF\_WRAP(T) value) {

1107 return StringMaker<T>::convert(value);

1108 }

1109

1110 #ifdef DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1111 DOCTEST\_INTERFACE String toString(const char\* in);

1112 #endif // DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1113

1114 #if DOCTEST\_MSVC >= DOCTEST\_COMPILER(19, 20, 0)

1115 // see this issue on why this is needed: https://github.com/doctest/doctest/issues/183

1116 DOCTEST\_INTERFACE String toString(const std::string& in);

1117 #endif // VS 2019

1118

1119 DOCTEST\_INTERFACE String toString(String in);

1120

1121 DOCTEST\_INTERFACE String toString(std::nullptr\_t);

1122

1123 DOCTEST\_INTERFACE String toString(bool in);

1124

1125 DOCTEST\_INTERFACE String toString(float in);

1126 DOCTEST\_INTERFACE String toString(double in);

1127 DOCTEST\_INTERFACE String toString(double long in);

1128

1129 DOCTEST\_INTERFACE String toString(char in);

1130 DOCTEST\_INTERFACE String toString(char signed in);

1131 DOCTEST\_INTERFACE String toString(char unsigned in);

1132 DOCTEST\_INTERFACE String toString(short in);

1133 DOCTEST\_INTERFACE String toString(short unsigned in);

1134 DOCTEST\_INTERFACE String toString(signed in);

1135 DOCTEST\_INTERFACE String toString(unsigned in);

1136 DOCTEST\_INTERFACE String toString(long in);

1137 DOCTEST\_INTERFACE String toString(long unsigned in);

1138 DOCTEST\_INTERFACE String toString(long long in);

1139 DOCTEST\_INTERFACE String toString(long long unsigned in);

1140

1141 template <typename T, typename detail::types::enable\_if<detail::should\_stringify\_as\_underlying\_type<T>::value, bool>::type = true>

1142 String toString(const DOCTEST\_REF\_WRAP(T) value) {

1143 using UT = typename detail::types::underlying\_type<T>::type;

1144 return (DOCTEST\_STRINGIFY(static\_cast<UT>(value)));

1145 }

1146

1147 namespace detail {

1148 template <typename T>

1149 struct filldata

1150 {

1151 static void fill(std::ostream\* stream, const T& in) {

1152 #if defined(\_MSC\_VER) && \_MSC\_VER <= 1900

1153 insert\_hack\_t<T>::insert(\*stream, in);

1154 #else

1155 operator<<(\*stream, in);

1156 #endif

1157 }

1158 };

1159

1160 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4866)

1161 // NOLINTBEGIN(\*-avoid-c-arrays)

1162 template <typename T, size\_t N>

1163 struct filldata<T[N]> {

1164 static void fill(std::ostream\* stream, const T(&in)[N]) {

1165 \*stream << "[";

1166 for (size\_t i = 0; i < N; i++) {

1167 if (i != 0) { \*stream << ", "; }

1168 \*stream << (DOCTEST\_STRINGIFY(in[i]));

1169 }

1170 \*stream << "]";

1171 }

1172 };

1173 // NOLINTEND(\*-avoid-c-arrays)

1174 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

1175

1176 // Specialized since we don't want the terminating null byte!

1177 // NOLINTBEGIN(\*-avoid-c-arrays)

1178 template <size\_t N>

1179 struct filldata<const char[N]> {

1180 static void fill(std::ostream\* stream, const char (&in)[N]) {

1181 \*stream << String(in, in[N - 1] ? N : N - 1);

1182 } // NOLINT(clang-analyzer-cplusplus.NewDeleteLeaks)

1183 };

1184 // NOLINTEND(\*-avoid-c-arrays)

1185

1186 template <>

1187 struct filldata<const void\*> {

1188 static void fill(std::ostream\* stream, const void\* in);

1189 };

1190

1191 template <typename T>

1192 struct filldata<T\*> {

1193 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4180)

1194 static void fill(std::ostream\* stream, const T\* in) {

1195 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

1196 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wmicrosoft-cast")

1197 filldata<const void\*>::fill(stream,

1198 #if DOCTEST\_GCC == 0 || DOCTEST\_GCC >= DOCTEST\_COMPILER(4, 9, 0)

1199 reinterpret\_cast<const void\*>(in)

1200 #else

1201 \*reinterpret\_cast<const void\* const\*>(&in)

1202 #endif

1203 );

1204 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

1205 }

1206 };

1207 }

1208

1209 struct DOCTEST\_INTERFACE Approx

1210 {

1211 Approx(double value);

1212

1213 Approx operator()(double value) const;

1214

1215 #ifdef DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

1216 template <typename T>

1217 explicit Approx(const T& value,

1218 typename detail::types::enable\_if<std::is\_constructible<double, T>::value>::type\* =

1219 static\_cast<T\*>(nullptr)) {

1220 \*this = static\_cast<double>(value);

1221 }

1222 #endif // DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

1223

1224 Approx& epsilon(double newEpsilon);

1225

1226 #ifdef DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

1227 template <typename T>

1228 typename std::enable\_if<std::is\_constructible<double, T>::value, Approx&>::type epsilon(

1229 const T& newEpsilon) {

1230 m\_epsilon = static\_cast<double>(newEpsilon);

1231 return \*this;

1232 }

1233 #endif // DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

1234

1235 Approx& scale(double newScale);

1236

1237 #ifdef DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

1238 template <typename T>

1239 typename std::enable\_if<std::is\_constructible<double, T>::value, Approx&>::type scale(

1240 const T& newScale) {

1241 m\_scale = static\_cast<double>(newScale);

1242 return \*this;

1243 }

1244 #endif // DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

1245

1246 // clang-format off

1247 DOCTEST\_INTERFACE friend bool operator==(double lhs, const Approx & rhs);

1248 DOCTEST\_INTERFACE friend bool operator==(const Approx & lhs, double rhs);

1249 DOCTEST\_INTERFACE friend bool operator!=(double lhs, const Approx & rhs);

1250 DOCTEST\_INTERFACE friend bool operator!=(const Approx & lhs, double rhs);

1251 DOCTEST\_INTERFACE friend bool operator<=(double lhs, const Approx & rhs);

1252 DOCTEST\_INTERFACE friend bool operator<=(const Approx & lhs, double rhs);

1253 DOCTEST\_INTERFACE friend bool operator>=(double lhs, const Approx & rhs);

1254 DOCTEST\_INTERFACE friend bool operator>=(const Approx & lhs, double rhs);

1255 DOCTEST\_INTERFACE friend bool operator< (double lhs, const Approx & rhs);

1256 DOCTEST\_INTERFACE friend bool operator< (const Approx & lhs, double rhs);

1257 DOCTEST\_INTERFACE friend bool operator> (double lhs, const Approx & rhs);

1258 DOCTEST\_INTERFACE friend bool operator> (const Approx & lhs, double rhs);

1259

1260 #ifdef DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

1261 #define DOCTEST\_APPROX\_PREFIX \

1262 template <typename T> friend typename std::enable\_if<std::is\_constructible<double, T>::value, bool>::type

1263

1264 DOCTEST\_APPROX\_PREFIX operator==(const T& lhs, const Approx& rhs) { return operator==(static\_cast<double>(lhs), rhs); }

1265 DOCTEST\_APPROX\_PREFIX operator==(const Approx& lhs, const T& rhs) { return operator==(rhs, lhs); }

1266 DOCTEST\_APPROX\_PREFIX operator!=(const T& lhs, const Approx& rhs) { return !operator==(lhs, rhs); }

1267 DOCTEST\_APPROX\_PREFIX operator!=(const Approx& lhs, const T& rhs) { return !operator==(rhs, lhs); }

1268 DOCTEST\_APPROX\_PREFIX operator<=(const T& lhs, const Approx& rhs) { return static\_cast<double>(lhs) < rhs.m\_value || lhs == rhs; }

1269 DOCTEST\_APPROX\_PREFIX operator<=(const Approx& lhs, const T& rhs) { return lhs.m\_value < static\_cast<double>(rhs) || lhs == rhs; }

1270 DOCTEST\_APPROX\_PREFIX operator>=(const T& lhs, const Approx& rhs) { return static\_cast<double>(lhs) > rhs.m\_value || lhs == rhs; }

1271 DOCTEST\_APPROX\_PREFIX operator>=(const Approx& lhs, const T& rhs) { return lhs.m\_value > static\_cast<double>(rhs) || lhs == rhs; }

1272 DOCTEST\_APPROX\_PREFIX operator< (const T& lhs, const Approx& rhs) { return static\_cast<double>(lhs) < rhs.m\_value && lhs != rhs; }

1273 DOCTEST\_APPROX\_PREFIX operator< (const Approx& lhs, const T& rhs) { return lhs.m\_value < static\_cast<double>(rhs) && lhs != rhs; }

1274 DOCTEST\_APPROX\_PREFIX operator> (const T& lhs, const Approx& rhs) { return static\_cast<double>(lhs) > rhs.m\_value && lhs != rhs; }

1275 DOCTEST\_APPROX\_PREFIX operator> (const Approx& lhs, const T& rhs) { return lhs.m\_value > static\_cast<double>(rhs) && lhs != rhs; }

1276 #undef DOCTEST\_APPROX\_PREFIX

1277 #endif // DOCTEST\_CONFIG\_INCLUDE\_TYPE\_TRAITS

1278

1279 // clang-format on

1280

1281 double m\_epsilon;

1282 double m\_scale;

1283 double m\_value;

1284 };

1285

1286 DOCTEST\_INTERFACE String toString(const Approx& in);

1287

1288 DOCTEST\_INTERFACE const ContextOptions\* getContextOptions();

1289

1290 template <typename F>

1291 struct DOCTEST\_INTERFACE\_DECL IsNaN

1292 {

1293 F value; bool flipped;

1294 IsNaN(F f, bool flip = false) : value(f), flipped(flip) { }

1295 IsNaN<F> operator!() const { return { value, !flipped }; }

1296 operator bool() const;

1297 };

1298 #ifndef \_\_MINGW32\_\_

1299 extern template struct DOCTEST\_INTERFACE\_DECL IsNaN<float>;

1300 extern template struct DOCTEST\_INTERFACE\_DECL IsNaN<double>;

1301 extern template struct DOCTEST\_INTERFACE\_DECL IsNaN<long double>;

1302 #endif

1303 DOCTEST\_INTERFACE String toString(IsNaN<float> in);

1304 DOCTEST\_INTERFACE String toString(IsNaN<double> in);

1305 DOCTEST\_INTERFACE String toString(IsNaN<double long> in);

1306

1307 #ifndef DOCTEST\_CONFIG\_DISABLE

1308

1309 namespace detail {

1310 // clang-format off

1311 #ifdef DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1312 template<class T> struct decay\_array { using type = T; };

1313 template<class T, unsigned N> struct decay\_array<T[N]> { using type = T\*; };

1314 template<class T> struct decay\_array<T[]> { using type = T\*; };

1315

1316 template<class T> struct not\_char\_pointer { static DOCTEST\_CONSTEXPR int value = 1; };

1317 template<> struct not\_char\_pointer<char\*> { static DOCTEST\_CONSTEXPR int value = 0; };

1318 template<> struct not\_char\_pointer<const char\*> { static DOCTEST\_CONSTEXPR int value = 0; };

1319

1320 template<class T> struct can\_use\_op : public not\_char\_pointer<typename decay\_array<T>::type> {};

1321 #endif // DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1322 // clang-format on

1323

1324 struct DOCTEST\_INTERFACE TestFailureException

1325 {

1326 };

1327

1328 DOCTEST\_INTERFACE bool checkIfShouldThrow(assertType::Enum at);

1329

1330 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

1331 DOCTEST\_NORETURN

1332 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

1333 DOCTEST\_INTERFACE void throwException();

1334

1335 struct DOCTEST\_INTERFACE Subcase

1336 {

1337 SubcaseSignature m\_signature;

1338 bool m\_entered = false;

1339

1340 Subcase(const String& name, const char\* file, int line);

1341 Subcase(const Subcase&) = delete;

1342 Subcase(Subcase&&) = delete;

1343 Subcase& operator=(const Subcase&) = delete;

1344 Subcase& operator=(Subcase&&) = delete;

1345 ~Subcase();

1346

1347 operator bool() const;

1348

1349 private:

1350 bool checkFilters();

1351 };

1352

1353 template <typename L, typename R>

1354 String stringifyBinaryExpr(const DOCTEST\_REF\_WRAP(L) lhs, const char\* op,

1355 const DOCTEST\_REF\_WRAP(R) rhs) {

1356 return (DOCTEST\_STRINGIFY(lhs)) + op + (DOCTEST\_STRINGIFY(rhs));

1357 }

1358

1359 #if DOCTEST\_CLANG && DOCTEST\_CLANG < DOCTEST\_COMPILER(3, 6, 0)

1360 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wunused-comparison")

1361 #endif

1362

1363 // This will check if there is any way it could find a operator like member or friend and uses it.

1364 // If not it doesn't find the operator or if the operator at global scope is defined after

1365 // this template, the template won't be instantiated due to SFINAE. Once the template is not

1366 // instantiated it can look for global operator using normal conversions.

1367 #ifdef \_\_NVCC\_\_

1368 #define SFINAE\_OP(ret,op) ret

1369 #else

1370 #define SFINAE\_OP(ret,op) decltype((void)(doctest::detail::declval<L>() op doctest::detail::declval<R>()),ret{})

1371 #endif

1372

1373 #define DOCTEST\_DO\_BINARY\_EXPRESSION\_COMPARISON(op, op\_str, op\_macro) \

1374 template <typename R> \

1375 DOCTEST\_NOINLINE SFINAE\_OP(Result,op) operator op(R&& rhs) { \

1376 bool res = op\_macro(doctest::detail::forward<const L>(lhs), doctest::detail::forward<R>(rhs)); \

1377 if(m\_at & assertType::is\_false) \

1378 res = !res; \

1379 if(!res || doctest::getContextOptions()->success) \

1380 return Result(res, stringifyBinaryExpr(lhs, op\_str, rhs)); \

1381 return Result(res); \

1382 }

1383

1384 // more checks could be added - like in Catch:

1385 // https://github.com/catchorg/Catch2/pull/1480/files

1386 // https://github.com/catchorg/Catch2/pull/1481/files

1387 #define DOCTEST\_FORBIT\_EXPRESSION(rt, op) \

1388 template <typename R> \

1389 rt& operator op(const R&) { \

1390 static\_assert(deferred\_false<R>::value, \

1391 "Expression Too Complex Please Rewrite As Binary Comparison!"); \

1392 return \*this; \

1393 }

1394

1395 struct DOCTEST\_INTERFACE Result // NOLINT(\*-member-init)

1396 {

1397 bool m\_passed;

1398 String m\_decomp;

1399

1400 Result() = default; // TODO: Why do we need this? (To remove NOLINT)

1401 Result(bool passed, const String& decomposition = String());

1402

1403 // forbidding some expressions based on this table: https://en.cppreference.com/w/cpp/language/operator\_precedence

1404 DOCTEST\_FORBIT\_EXPRESSION(Result, &)

1405 DOCTEST\_FORBIT\_EXPRESSION(Result, ^)

1406 DOCTEST\_FORBIT\_EXPRESSION(Result, |)

1407 DOCTEST\_FORBIT\_EXPRESSION(Result, &&)

1408 DOCTEST\_FORBIT\_EXPRESSION(Result, ||)

1409 DOCTEST\_FORBIT\_EXPRESSION(Result, ==)

1410 DOCTEST\_FORBIT\_EXPRESSION(Result, !=)

1411 DOCTEST\_FORBIT\_EXPRESSION(Result, <)

1412 DOCTEST\_FORBIT\_EXPRESSION(Result, >)

1413 DOCTEST\_FORBIT\_EXPRESSION(Result, <=)

1414 DOCTEST\_FORBIT\_EXPRESSION(Result, >=)

1415 DOCTEST\_FORBIT\_EXPRESSION(Result, =)

1416 DOCTEST\_FORBIT\_EXPRESSION(Result, +=)

1417 DOCTEST\_FORBIT\_EXPRESSION(Result, -=)

1418 DOCTEST\_FORBIT\_EXPRESSION(Result, \*=)

1419 DOCTEST\_FORBIT\_EXPRESSION(Result, /=)

1420 DOCTEST\_FORBIT\_EXPRESSION(Result, %=)

1421 DOCTEST\_FORBIT\_EXPRESSION(Result, <<=)

1422 DOCTEST\_FORBIT\_EXPRESSION(Result, >>=)

1423 DOCTEST\_FORBIT\_EXPRESSION(Result, &=)

1424 DOCTEST\_FORBIT\_EXPRESSION(Result, ^=)

1425 DOCTEST\_FORBIT\_EXPRESSION(Result, |=)

1426 };

1427

1428 #ifndef DOCTEST\_CONFIG\_NO\_COMPARISON\_WARNING\_SUPPRESSION

1429

1430 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_PUSH

1431 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wsign-conversion")

1432 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wsign-compare")

1433 //DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wdouble-promotion")

1434 //DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wconversion")

1435 //DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wfloat-equal")

1436

1437 DOCTEST\_GCC\_SUPPRESS\_WARNING\_PUSH

1438 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wsign-conversion")

1439 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wsign-compare")

1440 //DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wdouble-promotion")

1441 //DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wconversion")

1442 //DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wfloat-equal")

1443

1444 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_PUSH

1445 // https://stackoverflow.com/questions/39479163 what's the difference between 4018 and 4389

1446 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4388) // signed/unsigned mismatch

1447 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4389) // 'operator' : signed/unsigned mismatch

1448 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4018) // 'expression' : signed/unsigned mismatch

1449 //DOCTEST\_MSVC\_SUPPRESS\_WARNING(4805) // 'operation' : unsafe mix of type 'type' and type 'type' in operation

1450

1451 #endif // DOCTEST\_CONFIG\_NO\_COMPARISON\_WARNING\_SUPPRESSION

1452

1453 // clang-format off

1454 #ifndef DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1455 #define DOCTEST\_COMPARISON\_RETURN\_TYPE bool

1456 #else // DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1457 #define DOCTEST\_COMPARISON\_RETURN\_TYPE typename types::enable\_if<can\_use\_op<L>::value || can\_use\_op<R>::value, bool>::type

1458 inline bool eq(const char\* lhs, const char\* rhs) { return String(lhs) == String(rhs); }

1459 inline bool ne(const char\* lhs, const char\* rhs) { return String(lhs) != String(rhs); }

1460 inline bool lt(const char\* lhs, const char\* rhs) { return String(lhs) < String(rhs); }

1461 inline bool gt(const char\* lhs, const char\* rhs) { return String(lhs) > String(rhs); }

1462 inline bool le(const char\* lhs, const char\* rhs) { return String(lhs) <= String(rhs); }

1463 inline bool ge(const char\* lhs, const char\* rhs) { return String(lhs) >= String(rhs); }

1464 #endif // DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1465 // clang-format on

1466

1467 #define DOCTEST\_RELATIONAL\_OP(name, op) \

1468 template <typename L, typename R> \

1469 DOCTEST\_COMPARISON\_RETURN\_TYPE name(const DOCTEST\_REF\_WRAP(L) lhs, \

1470 const DOCTEST\_REF\_WRAP(R) rhs) { \

1471 return lhs op rhs; \

1472 }

1473

1474 DOCTEST\_RELATIONAL\_OP(eq, ==)

1475 DOCTEST\_RELATIONAL\_OP(ne, !=)

1476 DOCTEST\_RELATIONAL\_OP(lt, <)

1477 DOCTEST\_RELATIONAL\_OP(gt, >)

1478 DOCTEST\_RELATIONAL\_OP(le, <=)

1479 DOCTEST\_RELATIONAL\_OP(ge, >=)

1480

1481 #ifndef DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1482 #define DOCTEST\_CMP\_EQ(l, r) l == r

1483 #define DOCTEST\_CMP\_NE(l, r) l != r

1484 #define DOCTEST\_CMP\_GT(l, r) l > r

1485 #define DOCTEST\_CMP\_LT(l, r) l < r

1486 #define DOCTEST\_CMP\_GE(l, r) l >= r

1487 #define DOCTEST\_CMP\_LE(l, r) l <= r

1488 #else // DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1489 #define DOCTEST\_CMP\_EQ(l, r) eq(l, r)

1490 #define DOCTEST\_CMP\_NE(l, r) ne(l, r)

1491 #define DOCTEST\_CMP\_GT(l, r) gt(l, r)

1492 #define DOCTEST\_CMP\_LT(l, r) lt(l, r)

1493 #define DOCTEST\_CMP\_GE(l, r) ge(l, r)

1494 #define DOCTEST\_CMP\_LE(l, r) le(l, r)

1495 #endif // DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

1496

1497 template <typename L>

1498 // cppcheck-suppress copyCtorAndEqOperator

1499 struct Expression\_lhs

1500 {

1501 L lhs;

1502 assertType::Enum m\_at;

1503

1504 explicit Expression\_lhs(L&& in, assertType::Enum at)

1505 : lhs(static\_cast<L&&>(in))

1506 , m\_at(at) {}

1507

1508 DOCTEST\_NOINLINE operator Result() {

1509 // this is needed only for MSVC 2015

1510 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4800) // 'int': forcing value to bool

1511 bool res = static\_cast<bool>(lhs);

1512 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

1513 if(m\_at & assertType::is\_false) {

1514 res = !res;

1515 }

1516

1517 if(!res || getContextOptions()->success) {

1518 return { res, (DOCTEST\_STRINGIFY(lhs)) };

1519 }

1520 return { res };

1521 }

1522

1523 /\* This is required for user-defined conversions from Expression\_lhs to L \*/

1524 operator L() const { return lhs; }

1525

1526 // clang-format off

1527 DOCTEST\_DO\_BINARY\_EXPRESSION\_COMPARISON(==, " == ", DOCTEST\_CMP\_EQ)

1528 DOCTEST\_DO\_BINARY\_EXPRESSION\_COMPARISON(!=, " != ", DOCTEST\_CMP\_NE)

1529 DOCTEST\_DO\_BINARY\_EXPRESSION\_COMPARISON(>, " > ", DOCTEST\_CMP\_GT)

1530 DOCTEST\_DO\_BINARY\_EXPRESSION\_COMPARISON(<, " < ", DOCTEST\_CMP\_LT)

1531 DOCTEST\_DO\_BINARY\_EXPRESSION\_COMPARISON(>=, " >= ", DOCTEST\_CMP\_GE)

1532 DOCTEST\_DO\_BINARY\_EXPRESSION\_COMPARISON(<=, " <= ", DOCTEST\_CMP\_LE)

1533 // clang-format on

1534

1535 // forbidding some expressions based on this table: https://en.cppreference.com/w/cpp/language/operator\_precedence

1536 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, &)

1537 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, ^)

1538 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, |)

1539 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, &&)

1540 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, ||)

1541 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, =)

1542 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, +=)

1543 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, -=)

1544 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, \*=)

1545 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, /=)

1546 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, %=)

1547 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, <<=)

1548 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, >>=)

1549 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, &=)

1550 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, ^=)

1551 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, |=)

1552 // these 2 are unfortunate because they should be allowed - they have higher precedence over the comparisons, but the

1553 // ExpressionDecomposer class uses the left shift operator to capture the left operand of the binary expression...

1554 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, <<)

1555 DOCTEST\_FORBIT\_EXPRESSION(Expression\_lhs, >>)

1556 };

1557

1558 #ifndef DOCTEST\_CONFIG\_NO\_COMPARISON\_WARNING\_SUPPRESSION

1559

1560 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

1561 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

1562 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

1563

1564 #endif // DOCTEST\_CONFIG\_NO\_COMPARISON\_WARNING\_SUPPRESSION

1565

1566 #if DOCTEST\_CLANG && DOCTEST\_CLANG < DOCTEST\_COMPILER(3, 6, 0)

1567 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

1568 #endif

1569

1570 struct DOCTEST\_INTERFACE ExpressionDecomposer

1571 {

1572 assertType::Enum m\_at;

1573

1574 ExpressionDecomposer(assertType::Enum at);

1575

1576 // The right operator for capturing expressions is "<=" instead of "<<" (based on the operator precedence table)

1577 // but then there will be warnings from GCC about "-Wparentheses" and since "\_Pragma()" is problematic this will stay for now...

1578 // https://github.com/catchorg/Catch2/issues/870

1579 // https://github.com/catchorg/Catch2/issues/565

1580 template <typename L>

1581 Expression\_lhs<L> operator<<(L&& operand) {

1582 return Expression\_lhs<L>(static\_cast<L&&>(operand), m\_at);

1583 }

1584

1585 template <typename L,typename types::enable\_if<!doctest::detail::types::is\_rvalue\_reference<L>::value,void >::type\* = nullptr>

1586 Expression\_lhs<const L&> operator<<(const L &operand) {

1587 return Expression\_lhs<const L&>(operand, m\_at);

1588 }

1589 };

1590

1591 struct DOCTEST\_INTERFACE TestSuite

1592 {

1593 const char\* m\_test\_suite = nullptr;

1594 const char\* m\_description = nullptr;

1595 bool m\_skip = false;

1596 bool m\_no\_breaks = false;

1597 bool m\_no\_output = false;

1598 bool m\_may\_fail = false;

1599 bool m\_should\_fail = false;

1600 int m\_expected\_failures = 0;

1601 double m\_timeout = 0;

1602

1603 TestSuite& operator\*(const char\* in);

1604

1605 template <typename T>

1606 TestSuite& operator\*(const T& in) {

1607 in.fill(\*this);

1608 return \*this;

1609 }

1610 };

1611

1612 using funcType = void (\*)();

1613

1614 struct DOCTEST\_INTERFACE TestCase : public TestCaseData

1615 {

1616 funcType m\_test; // a function pointer to the test case

1617

1618 String m\_type; // for templated test cases - gets appended to the real name

1619 int m\_template\_id; // an ID used to distinguish between the different versions of a templated test case

1620 String m\_full\_name; // contains the name (only for templated test cases!) + the template type

1621

1622 TestCase(funcType test, const char\* file, unsigned line, const TestSuite& test\_suite,

1623 const String& type = String(), int template\_id = -1);

1624

1625 TestCase(const TestCase& other);

1626 TestCase(TestCase&&) = delete;

1627

1628 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(26434) // hides a non-virtual function

1629 TestCase& operator=(const TestCase& other);

1630 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

1631

1632 TestCase& operator=(TestCase&&) = delete;

1633

1634 TestCase& operator\*(const char\* in);

1635

1636 template <typename T>

1637 TestCase& operator\*(const T& in) {

1638 in.fill(\*this);

1639 return \*this;

1640 }

1641

1642 bool operator<(const TestCase& other) const;

1643

1644 ~TestCase() = default;

1645 };

1646

1647 // forward declarations of functions used by the macros

1648 DOCTEST\_INTERFACE int regTest(const TestCase& tc);

1649 DOCTEST\_INTERFACE int setTestSuite(const TestSuite& ts);

1650 DOCTEST\_INTERFACE bool isDebuggerActive();

1651

1652 template<typename T>

1653 int instantiationHelper(const T&) { return 0; }

1654

1655 namespace binaryAssertComparison {

1656 enum Enum

1657 {

1658 eq = 0,

1659 ne,

1660 gt,

1661 lt,

1662 ge,

1663 le

1664 };

1665 } // namespace binaryAssertComparison

1666

1667 // clang-format off

1668 template <int, class L, class R> struct RelationalComparator { bool operator()(const DOCTEST\_REF\_WRAP(L), const DOCTEST\_REF\_WRAP(R) ) const { return false; } };

1669

1670 #define DOCTEST\_BINARY\_RELATIONAL\_OP(n, op) \

1671 template <class L, class R> struct RelationalComparator<n, L, R> { bool operator()(const DOCTEST\_REF\_WRAP(L) lhs, const DOCTEST\_REF\_WRAP(R) rhs) const { return op(lhs, rhs); } };

1672 // clang-format on

1673

1674 DOCTEST\_BINARY\_RELATIONAL\_OP(0, doctest::detail::eq)

1675 DOCTEST\_BINARY\_RELATIONAL\_OP(1, doctest::detail::ne)

1676 DOCTEST\_BINARY\_RELATIONAL\_OP(2, doctest::detail::gt)

1677 DOCTEST\_BINARY\_RELATIONAL\_OP(3, doctest::detail::lt)

1678 DOCTEST\_BINARY\_RELATIONAL\_OP(4, doctest::detail::ge)

1679 DOCTEST\_BINARY\_RELATIONAL\_OP(5, doctest::detail::le)

1680

1681 struct DOCTEST\_INTERFACE ResultBuilder : public AssertData

1682 {

1683 ResultBuilder(assertType::Enum at, const char\* file, int line, const char\* expr,

1684 const char\* exception\_type = "", const String& exception\_string = "");

1685

1686 ResultBuilder(assertType::Enum at, const char\* file, int line, const char\* expr,

1687 const char\* exception\_type, const Contains& exception\_string);

1688

1689 void setResult(const Result& res);

1690

1691 template <int comparison, typename L, typename R>

1692 DOCTEST\_NOINLINE bool binary\_assert(const DOCTEST\_REF\_WRAP(L) lhs,

1693 const DOCTEST\_REF\_WRAP(R) rhs) {

1694 m\_failed = !RelationalComparator<comparison, L, R>()(lhs, rhs);

1695 if (m\_failed || getContextOptions()->success) {

1696 m\_decomp = stringifyBinaryExpr(lhs, ", ", rhs);

1697 }

1698 return !m\_failed;

1699 }

1700

1701 template <typename L>

1702 DOCTEST\_NOINLINE bool unary\_assert(const DOCTEST\_REF\_WRAP(L) val) {

1703 m\_failed = !val;

1704

1705 if (m\_at & assertType::is\_false) {

1706 m\_failed = !m\_failed;

1707 }

1708

1709 if (m\_failed || getContextOptions()->success) {

1710 m\_decomp = (DOCTEST\_STRINGIFY(val));

1711 }

1712

1713 return !m\_failed;

1714 }

1715

1716 void translateException();

1717

1718 bool log();

1719 void react() const;

1720 };

1721

1722 namespace assertAction {

1723 enum Enum

1724 {

1725 nothing = 0,

1726 dbgbreak = 1,

1727 shouldthrow = 2

1728 };

1729 } // namespace assertAction

1730

1731 DOCTEST\_INTERFACE void failed\_out\_of\_a\_testing\_context(const AssertData& ad);

1732

1733 DOCTEST\_INTERFACE bool decomp\_assert(assertType::Enum at, const char\* file, int line,

1734 const char\* expr, const Result& result);

1735

1736 #define DOCTEST\_ASSERT\_OUT\_OF\_TESTS(decomp) \

1737 do { \

1738 if(!is\_running\_in\_test) { \

1739 if(failed) { \

1740 ResultBuilder rb(at, file, line, expr); \

1741 rb.m\_failed = failed; \

1742 rb.m\_decomp = decomp; \

1743 failed\_out\_of\_a\_testing\_context(rb); \

1744 if(isDebuggerActive() && !getContextOptions()->no\_breaks) \

1745 DOCTEST\_BREAK\_INTO\_DEBUGGER(); \

1746 if(checkIfShouldThrow(at)) \

1747 throwException(); \

1748 } \

1749 return !failed; \

1750 } \

1751 } while(false)

1752

1753 #define DOCTEST\_ASSERT\_IN\_TESTS(decomp) \

1754 ResultBuilder rb(at, file, line, expr); \

1755 rb.m\_failed = failed; \

1756 if(rb.m\_failed || getContextOptions()->success) \

1757 rb.m\_decomp = decomp; \

1758 if(rb.log()) \

1759 DOCTEST\_BREAK\_INTO\_DEBUGGER(); \

1760 if(rb.m\_failed && checkIfShouldThrow(at)) \

1761 throwException()

1762

1763 template <int comparison, typename L, typename R>

1764 DOCTEST\_NOINLINE bool binary\_assert(assertType::Enum at, const char\* file, int line,

1765 const char\* expr, const DOCTEST\_REF\_WRAP(L) lhs,

1766 const DOCTEST\_REF\_WRAP(R) rhs) {

1767 bool failed = !RelationalComparator<comparison, L, R>()(lhs, rhs);

1768

1769 // ###################################################################################

1770 // IF THE DEBUGGER BREAKS HERE - GO 1 LEVEL UP IN THE CALLSTACK FOR THE FAILING ASSERT

1771 // THIS IS THE EFFECT OF HAVING 'DOCTEST\_CONFIG\_SUPER\_FAST\_ASSERTS' DEFINED

1772 // ###################################################################################

1773 DOCTEST\_ASSERT\_OUT\_OF\_TESTS(stringifyBinaryExpr(lhs, ", ", rhs));

1774 DOCTEST\_ASSERT\_IN\_TESTS(stringifyBinaryExpr(lhs, ", ", rhs));

1775 return !failed;

1776 }

1777

1778 template <typename L>

1779 DOCTEST\_NOINLINE bool unary\_assert(assertType::Enum at, const char\* file, int line,

1780 const char\* expr, const DOCTEST\_REF\_WRAP(L) val) {

1781 bool failed = !val;

1782

1783 if(at & assertType::is\_false)

1784 failed = !failed;

1785

1786 // ###################################################################################

1787 // IF THE DEBUGGER BREAKS HERE - GO 1 LEVEL UP IN THE CALLSTACK FOR THE FAILING ASSERT

1788 // THIS IS THE EFFECT OF HAVING 'DOCTEST\_CONFIG\_SUPER\_FAST\_ASSERTS' DEFINED

1789 // ###################################################################################

1790 DOCTEST\_ASSERT\_OUT\_OF\_TESTS((DOCTEST\_STRINGIFY(val)));

1791 DOCTEST\_ASSERT\_IN\_TESTS((DOCTEST\_STRINGIFY(val)));

1792 return !failed;

1793 }

1794

1795 struct DOCTEST\_INTERFACE IExceptionTranslator

1796 {

1797 DOCTEST\_DECLARE\_INTERFACE(IExceptionTranslator)

1798 virtual bool translate(String&) const = 0;

1799 };

1800

1801 template <typename T>

1802 class ExceptionTranslator : public IExceptionTranslator

1803 {

1804 public:

1805 explicit ExceptionTranslator(String (\*translateFunction)(T))

1806 : m\_translateFunction(translateFunction) {}

1807

1808 bool translate(String& res) const override {

1809 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

1810 try {

1811 throw; // lgtm [cpp/rethrow-no-exception]

1812 // cppcheck-suppress catchExceptionByValue

1813 } catch(const T& ex) {

1814 res = m\_translateFunction(ex);

1815 return true;

1816 } catch(...) {}

1817 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

1818 static\_cast<void>(res); // to silence -Wunused-parameter

1819 return false;

1820 }

1821

1822 private:

1823 String (\*m\_translateFunction)(T);

1824 };

1825

1826 DOCTEST\_INTERFACE void registerExceptionTranslatorImpl(const IExceptionTranslator\* et);

1827

1828 // ContextScope base class used to allow implementing methods of ContextScope

1829 // that don't depend on the template parameter in doctest.cpp.

1830 struct DOCTEST\_INTERFACE ContextScopeBase : public IContextScope {

1831 ContextScopeBase(const ContextScopeBase&) = delete;

1832

1833 ContextScopeBase& operator=(const ContextScopeBase&) = delete;

1834 ContextScopeBase& operator=(ContextScopeBase&&) = delete;

1835

1836 ~ContextScopeBase() override = default;

1837

1838 protected:

1839 ContextScopeBase();

1840 ContextScopeBase(ContextScopeBase&& other) noexcept;

1841

1842 void destroy();

1843 bool need\_to\_destroy{true};

1844 };

1845

1846 template <typename L> class ContextScope : public ContextScopeBase

1847 {

1848 L lambda\_;

1849

1850 public:

1851 explicit ContextScope(const L &lambda) : lambda\_(lambda) {}

1852 explicit ContextScope(L&& lambda) : lambda\_(static\_cast<L&&>(lambda)) { }

1853

1854 ContextScope(const ContextScope&) = delete;

1855 ContextScope(ContextScope&&) noexcept = default;

1856

1857 ContextScope& operator=(const ContextScope&) = delete;

1858 ContextScope& operator=(ContextScope&&) = delete;

1859

1860 void stringify(std::ostream\* s) const override { lambda\_(s); }

1861

1862 ~ContextScope() override {

1863 if (need\_to\_destroy) {

1864 destroy();

1865 }

1866 }

1867 };

1868

1869 struct DOCTEST\_INTERFACE MessageBuilder : public MessageData

1870 {

1871 std::ostream\* m\_stream;

1872 bool logged = false;

1873

1874 MessageBuilder(const char\* file, int line, assertType::Enum severity);

1875

1876 MessageBuilder(const MessageBuilder&) = delete;

1877 MessageBuilder(MessageBuilder&&) = delete;

1878

1879 MessageBuilder& operator=(const MessageBuilder&) = delete;

1880 MessageBuilder& operator=(MessageBuilder&&) = delete;

1881

1882 ~MessageBuilder();

1883

1884 // the preferred way of chaining parameters for stringification

1885 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4866)

1886 template <typename T>

1887 MessageBuilder& operator,(const T& in) {

1888 \*m\_stream << (DOCTEST\_STRINGIFY(in));

1889 return \*this;

1890 }

1891 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

1892

1893 // kept here just for backwards-compatibility - the comma operator should be preferred now

1894 template <typename T>

1895 MessageBuilder& operator<<(const T& in) { return this->operator,(in); }

1896

1897 // the `,` operator has the lowest operator precedence - if `<<` is used by the user then

1898 // the `,` operator will be called last which is not what we want and thus the `\*` operator

1899 // is used first (has higher operator precedence compared to `<<`) so that we guarantee that

1900 // an operator of the MessageBuilder class is called first before the rest of the parameters

1901 template <typename T>

1902 MessageBuilder& operator\*(const T& in) { return this->operator,(in); }

1903

1904 bool log();

1905 void react();

1906 };

1907

1908 template <typename L>

1909 ContextScope<L> MakeContextScope(const L &lambda) {

1910 return ContextScope<L>(lambda);

1911 }

1912 } // namespace detail

1913

1914 #define DOCTEST\_DEFINE\_DECORATOR(name, type, def) \

1915 struct name \

1916 { \

1917 type data; \

1918 name(type in = def) \

1919 : data(in) {} \

1920 void fill(detail::TestCase& state) const { state.DOCTEST\_CAT(m\_, name) = data; } \

1921 void fill(detail::TestSuite& state) const { state.DOCTEST\_CAT(m\_, name) = data; } \

1922 }

1923

1924 DOCTEST\_DEFINE\_DECORATOR(test\_suite, const char\*, "");

1925 DOCTEST\_DEFINE\_DECORATOR(description, const char\*, "");

1926 DOCTEST\_DEFINE\_DECORATOR(skip, bool, true);

1927 DOCTEST\_DEFINE\_DECORATOR(no\_breaks, bool, true);

1928 DOCTEST\_DEFINE\_DECORATOR(no\_output, bool, true);

1929 DOCTEST\_DEFINE\_DECORATOR(timeout, double, 0);

1930 DOCTEST\_DEFINE\_DECORATOR(may\_fail, bool, true);

1931 DOCTEST\_DEFINE\_DECORATOR(should\_fail, bool, true);

1932 DOCTEST\_DEFINE\_DECORATOR(expected\_failures, int, 0);

1933

1934 template <typename T>

1935 int registerExceptionTranslator(String (\*translateFunction)(T)) {

1936 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wexit-time-destructors")

1937 static detail::ExceptionTranslator<T> exceptionTranslator(translateFunction);

1938 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

1939 detail::registerExceptionTranslatorImpl(&exceptionTranslator);

1940 return 0;

1941 }

1942

1943 } // namespace doctest

1944

1945 // in a separate namespace outside of doctest because the DOCTEST\_TEST\_SUITE macro

1946 // introduces an anonymous namespace in which getCurrentTestSuite gets overridden

1947 namespace doctest\_detail\_test\_suite\_ns {

1948 DOCTEST\_INTERFACE doctest::detail::TestSuite& getCurrentTestSuite();

1949 } // namespace doctest\_detail\_test\_suite\_ns

1950

1951 namespace doctest {

1952 #else // DOCTEST\_CONFIG\_DISABLE

1953 template <typename T>

1954 int registerExceptionTranslator(String (\*)(T)) {

1955 return 0;

1956 }

1957 #endif // DOCTEST\_CONFIG\_DISABLE

1958

1959 namespace detail {

1960 using assert\_handler = void (\*)(const AssertData&);

1961 struct ContextState;

1962 } // namespace detail

1963

1964 class DOCTEST\_INTERFACE Context

1965 {

1966 detail::ContextState\* p;

1967

1968 void parseArgs(int argc, const char\* const\* argv, bool withDefaults = false);

1969

1970 public:

1971 explicit Context(int argc = 0, const char\* const\* argv = nullptr);

1972

1973 Context(const Context&) = delete;

1974 Context(Context&&) = delete;

1975

1976 Context& operator=(const Context&) = delete;

1977 Context& operator=(Context&&) = delete;

1978

1979 ~Context(); // NOLINT(performance-trivially-destructible)

1980

1981 void applyCommandLine(int argc, const char\* const\* argv);

1982

1983 void addFilter(const char\* filter, const char\* value);

1984 void clearFilters();

1985 void setOption(const char\* option, bool value);

1986 void setOption(const char\* option, int value);

1987 void setOption(const char\* option, const char\* value);

1988

1989 bool shouldExit();

1990

1991 void setAsDefaultForAssertsOutOfTestCases();

1992

1993 void setAssertHandler(detail::assert\_handler ah);

1994

1995 void setCout(std::ostream\* out);

1996

1997 int run();

1998 };

1999

2000 namespace TestCaseFailureReason {

2001 enum Enum

2002 {

2003 None = 0,

2004 AssertFailure = 1, // an assertion has failed in the test case

2005 Exception = 2, // test case threw an exception

2006 Crash = 4, // a crash...

2007 TooManyFailedAsserts = 8, // the abort-after option

2008 Timeout = 16, // see the timeout decorator

2009 ShouldHaveFailedButDidnt = 32, // see the should\_fail decorator

2010 ShouldHaveFailedAndDid = 64, // see the should\_fail decorator

2011 DidntFailExactlyNumTimes = 128, // see the expected\_failures decorator

2012 FailedExactlyNumTimes = 256, // see the expected\_failures decorator

2013 CouldHaveFailedAndDid = 512 // see the may\_fail decorator

2014 };

2015 } // namespace TestCaseFailureReason

2016

2017 struct DOCTEST\_INTERFACE CurrentTestCaseStats

2018 {

2019 int numAssertsCurrentTest;

2020 int numAssertsFailedCurrentTest;

2021 double seconds;

2022 int failure\_flags; // use TestCaseFailureReason::Enum

2023 bool testCaseSuccess;

2024 };

2025

2026 struct DOCTEST\_INTERFACE TestCaseException

2027 {

2028 String error\_string;

2029 bool is\_crash;

2030 };

2031

2032 struct DOCTEST\_INTERFACE TestRunStats

2033 {

2034 unsigned numTestCases;

2035 unsigned numTestCasesPassingFilters;

2036 unsigned numTestSuitesPassingFilters;

2037 unsigned numTestCasesFailed;

2038 int numAsserts;

2039 int numAssertsFailed;

2040 };

2041

2042 struct QueryData

2043 {

2044 const TestRunStats\* run\_stats = nullptr;

2045 const TestCaseData\*\* data = nullptr;

2046 unsigned num\_data = 0;

2047 };

2048

2049 struct DOCTEST\_INTERFACE IReporter

2050 {

2051 // The constructor has to accept "const ContextOptions&" as a single argument

2052 // which has most of the options for the run + a pointer to the stdout stream

2053 // Reporter(const ContextOptions& in)

2054

2055 // called when a query should be reported (listing test cases, printing the version, etc.)

2056 virtual void report\_query(const QueryData&) = 0;

2057

2058 // called when the whole test run starts

2059 virtual void test\_run\_start() = 0;

2060 // called when the whole test run ends (caching a pointer to the input doesn't make sense here)

2061 virtual void test\_run\_end(const TestRunStats&) = 0;

2062

2063 // called when a test case is started (safe to cache a pointer to the input)

2064 virtual void test\_case\_start(const TestCaseData&) = 0;

2065 // called when a test case is reentered because of unfinished subcases (safe to cache a pointer to the input)

2066 virtual void test\_case\_reenter(const TestCaseData&) = 0;

2067 // called when a test case has ended

2068 virtual void test\_case\_end(const CurrentTestCaseStats&) = 0;

2069

2070 // called when an exception is thrown from the test case (or it crashes)

2071 virtual void test\_case\_exception(const TestCaseException&) = 0;

2072

2073 // called whenever a subcase is entered (don't cache pointers to the input)

2074 virtual void subcase\_start(const SubcaseSignature&) = 0;

2075 // called whenever a subcase is exited (don't cache pointers to the input)

2076 virtual void subcase\_end() = 0;

2077

2078 // called for each assert (don't cache pointers to the input)

2079 virtual void log\_assert(const AssertData&) = 0;

2080 // called for each message (don't cache pointers to the input)

2081 virtual void log\_message(const MessageData&) = 0;

2082

2083 // called when a test case is skipped either because it doesn't pass the filters, has a skip decorator

2084 // or isn't in the execution range (between first and last) (safe to cache a pointer to the input)

2085 virtual void test\_case\_skipped(const TestCaseData&) = 0;

2086

2087 DOCTEST\_DECLARE\_INTERFACE(IReporter)

2088

2089 // can obtain all currently active contexts and stringify them if one wishes to do so

2090 static int get\_num\_active\_contexts();

2091 static const IContextScope\* const\* get\_active\_contexts();

2092

2093 // can iterate through contexts which have been stringified automatically in their destructors when an exception has been thrown

2094 static int get\_num\_stringified\_contexts();

2095 static const String\* get\_stringified\_contexts();

2096 };

2097

2098 namespace detail {

2099 using reporterCreatorFunc = IReporter\* (\*)(const ContextOptions&);

2100

2101 DOCTEST\_INTERFACE void registerReporterImpl(const char\* name, int prio, reporterCreatorFunc c, bool isReporter);

2102

2103 template <typename Reporter>

2104 IReporter\* reporterCreator(const ContextOptions& o) {

2105 return new Reporter(o);

2106 }

2107 } // namespace detail

2108

2109 template <typename Reporter>

2110 int registerReporter(const char\* name, int priority, bool isReporter) {

2111 detail::registerReporterImpl(name, priority, detail::reporterCreator<Reporter>, isReporter);

2112 return 0;

2113 }

2114 } // namespace doctest

2115

2116 #ifdef DOCTEST\_CONFIG\_ASSERTS\_RETURN\_VALUES

2117 #define DOCTEST\_FUNC\_EMPTY [] { return false; }()

2118 #else

2119 #define DOCTEST\_FUNC\_EMPTY (void)0

2120 #endif

2121

2122 // if registering is not disabled

2123 #ifndef DOCTEST\_CONFIG\_DISABLE

2124

2125 #ifdef DOCTEST\_CONFIG\_ASSERTS\_RETURN\_VALUES

2126 #define DOCTEST\_FUNC\_SCOPE\_BEGIN [&]

2127 #define DOCTEST\_FUNC\_SCOPE\_END ()

2128 #define DOCTEST\_FUNC\_SCOPE\_RET(v) return v

2129 #else

2130 #define DOCTEST\_FUNC\_SCOPE\_BEGIN do

2131 #define DOCTEST\_FUNC\_SCOPE\_END while(false)

2132 #define DOCTEST\_FUNC\_SCOPE\_RET(v) (void)0

2133 #endif

2134

2135 // common code in asserts - for convenience

2136 #define DOCTEST\_ASSERT\_LOG\_REACT\_RETURN(b) \

2137 if(b.log()) DOCTEST\_BREAK\_INTO\_DEBUGGER(); \

2138 b.react(); \

2139 DOCTEST\_FUNC\_SCOPE\_RET(!b.m\_failed)

2140

2141 #ifdef DOCTEST\_CONFIG\_NO\_TRY\_CATCH\_IN\_ASSERTS

2142 #define DOCTEST\_WRAP\_IN\_TRY(x) x;

2143 #else // DOCTEST\_CONFIG\_NO\_TRY\_CATCH\_IN\_ASSERTS

2144 #define DOCTEST\_WRAP\_IN\_TRY(x) \

2145 try { \

2146 x; \

2147 } catch(...) { DOCTEST\_RB.translateException(); }

2148 #endif // DOCTEST\_CONFIG\_NO\_TRY\_CATCH\_IN\_ASSERTS

2149

2150 #ifdef DOCTEST\_CONFIG\_VOID\_CAST\_EXPRESSIONS

2151 #define DOCTEST\_CAST\_TO\_VOID(...) \

2152 DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wuseless-cast") \

2153 static\_cast<void>(\_\_VA\_ARGS\_\_); \

2154 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

2155 #else // DOCTEST\_CONFIG\_VOID\_CAST\_EXPRESSIONS

2156 #define DOCTEST\_CAST\_TO\_VOID(...) \_\_VA\_ARGS\_\_;

2157 #endif // DOCTEST\_CONFIG\_VOID\_CAST\_EXPRESSIONS

2158

2159 // registers the test by initializing a dummy var with a function

2160 #define DOCTEST\_REGISTER\_FUNCTION(global\_prefix, f, decorators) \

2161 global\_prefix DOCTEST\_GLOBAL\_NO\_WARNINGS(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_VAR\_), /\* NOLINT \*/ \

2162 doctest::detail::regTest( \

2163 doctest::detail::TestCase( \

2164 f, \_\_FILE\_\_, \_\_LINE\_\_, \

2165 doctest\_detail\_test\_suite\_ns::getCurrentTestSuite()) \* \

2166 decorators))

2167

2168 #define DOCTEST\_IMPLEMENT\_FIXTURE(der, base, func, decorators) \

2169 namespace { /\* NOLINT \*/ \

2170 struct der : public base \

2171 { \

2172 void f(); \

2173 }; \

2174 static DOCTEST\_INLINE\_NOINLINE void func() { \

2175 der v; \

2176 v.f(); \

2177 } \

2178 DOCTEST\_REGISTER\_FUNCTION(DOCTEST\_EMPTY, func, decorators) \

2179 } \

2180 DOCTEST\_INLINE\_NOINLINE void der::f() // NOLINT(misc-definitions-in-headers)

2181

2182 #define DOCTEST\_CREATE\_AND\_REGISTER\_FUNCTION(f, decorators) \

2183 static void f(); \

2184 DOCTEST\_REGISTER\_FUNCTION(DOCTEST\_EMPTY, f, decorators) \

2185 static void f()

2186

2187 #define DOCTEST\_CREATE\_AND\_REGISTER\_FUNCTION\_IN\_CLASS(f, proxy, decorators) \

2188 static doctest::detail::funcType proxy() { return f; } \

2189 DOCTEST\_REGISTER\_FUNCTION(inline, proxy(), decorators) \

2190 static void f()

2191

2192 // for registering tests

2193 #define DOCTEST\_TEST\_CASE(decorators) \

2194 DOCTEST\_CREATE\_AND\_REGISTER\_FUNCTION(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_FUNC\_), decorators)

2195

2196 // for registering tests in classes - requires C++17 for inline variables!

2197 #if DOCTEST\_CPLUSPLUS >= 201703L

2198 #define DOCTEST\_TEST\_CASE\_CLASS(decorators) \

2199 DOCTEST\_CREATE\_AND\_REGISTER\_FUNCTION\_IN\_CLASS(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_FUNC\_), \

2200 DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_PROXY\_), \

2201 decorators)

2202 #else // DOCTEST\_TEST\_CASE\_CLASS

2203 #define DOCTEST\_TEST\_CASE\_CLASS(...) \

2204 TEST\_CASES\_CAN\_BE\_REGISTERED\_IN\_CLASSES\_ONLY\_IN\_CPP17\_MODE\_OR\_WITH\_VS\_2017\_OR\_NEWER

2205 #endif // DOCTEST\_TEST\_CASE\_CLASS

2206

2207 // for registering tests with a fixture

2208 #define DOCTEST\_TEST\_CASE\_FIXTURE(c, decorators) \

2209 DOCTEST\_IMPLEMENT\_FIXTURE(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_CLASS\_), c, \

2210 DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_FUNC\_), decorators)

2211

2212 // for converting types to strings without the <typeinfo> header and demangling

2213 #define DOCTEST\_TYPE\_TO\_STRING\_AS(str, ...) \

2214 namespace doctest { \

2215 template <> \

2216 inline String toString<\_\_VA\_ARGS\_\_>() { \

2217 return str; \

2218 } \

2219 } \

2220 static\_assert(true, "")

2221

2222 #define DOCTEST\_TYPE\_TO\_STRING(...) DOCTEST\_TYPE\_TO\_STRING\_AS(#\_\_VA\_ARGS\_\_, \_\_VA\_ARGS\_\_)

2223

2224 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_DEFINE\_IMPL(dec, T, iter, func) \

2225 template <typename T> \

2226 static void func(); \

2227 namespace { /\* NOLINT \*/ \

2228 template <typename Tuple> \

2229 struct iter; \

2230 template <typename Type, typename... Rest> \

2231 struct iter<std::tuple<Type, Rest...>> \

2232 { \

2233 iter(const char\* file, unsigned line, int index) { \

2234 doctest::detail::regTest(doctest::detail::TestCase(func<Type>, file, line, \

2235 doctest\_detail\_test\_suite\_ns::getCurrentTestSuite(), \

2236 doctest::toString<Type>(), \

2237 int(line) \* 1000 + index) \

2238 \* dec); \

2239 iter<std::tuple<Rest...>>(file, line, index + 1); \

2240 } \

2241 }; \

2242 template <> \

2243 struct iter<std::tuple<>> \

2244 { \

2245 iter(const char\*, unsigned, int) {} \

2246 }; \

2247 } \

2248 template <typename T> \

2249 static void func()

2250

2251 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_DEFINE(dec, T, id) \

2252 DOCTEST\_TEST\_CASE\_TEMPLATE\_DEFINE\_IMPL(dec, T, DOCTEST\_CAT(id, ITERATOR), \

2253 DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TMP\_))

2254

2255 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_INSTANTIATE\_IMPL(id, anon, ...) \

2256 DOCTEST\_GLOBAL\_NO\_WARNINGS(DOCTEST\_CAT(anon, DUMMY), /\* NOLINT(cert-err58-cpp, fuchsia-statically-constructed-objects) \*/ \

2257 doctest::detail::instantiationHelper( \

2258 DOCTEST\_CAT(id, ITERATOR)<\_\_VA\_ARGS\_\_>(\_\_FILE\_\_, \_\_LINE\_\_, 0)))

2259

2260 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_INVOKE(id, ...) \

2261 DOCTEST\_TEST\_CASE\_TEMPLATE\_INSTANTIATE\_IMPL(id, DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TMP\_), std::tuple<\_\_VA\_ARGS\_\_>) \

2262 static\_assert(true, "")

2263

2264 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_APPLY(id, ...) \

2265 DOCTEST\_TEST\_CASE\_TEMPLATE\_INSTANTIATE\_IMPL(id, DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TMP\_), \_\_VA\_ARGS\_\_) \

2266 static\_assert(true, "")

2267

2268 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_IMPL(dec, T, anon, ...) \

2269 DOCTEST\_TEST\_CASE\_TEMPLATE\_DEFINE\_IMPL(dec, T, DOCTEST\_CAT(anon, ITERATOR), anon); \

2270 DOCTEST\_TEST\_CASE\_TEMPLATE\_INSTANTIATE\_IMPL(anon, anon, std::tuple<\_\_VA\_ARGS\_\_>) \

2271 template <typename T> \

2272 static void anon()

2273

2274 #define DOCTEST\_TEST\_CASE\_TEMPLATE(dec, T, ...) \

2275 DOCTEST\_TEST\_CASE\_TEMPLATE\_IMPL(dec, T, DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TMP\_), \_\_VA\_ARGS\_\_)

2276

2277 // for subcases

2278 #define DOCTEST\_SUBCASE(name) \

2279 if(const doctest::detail::Subcase & DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_SUBCASE\_) DOCTEST\_UNUSED = \

2280 doctest::detail::Subcase(name, \_\_FILE\_\_, \_\_LINE\_\_))

2281

2282 // for grouping tests in test suites by using code blocks

2283 #define DOCTEST\_TEST\_SUITE\_IMPL(decorators, ns\_name) \

2284 namespace ns\_name { namespace doctest\_detail\_test\_suite\_ns { \

2285 static DOCTEST\_NOINLINE doctest::detail::TestSuite& getCurrentTestSuite() noexcept { \

2286 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4640) \

2287 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wexit-time-destructors") \

2288 DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wmissing-field-initializers") \

2289 static doctest::detail::TestSuite data{}; \

2290 static bool inited = false; \

2291 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP \

2292 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP \

2293 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP \

2294 if(!inited) { \

2295 data\* decorators; \

2296 inited = true; \

2297 } \

2298 return data; \

2299 } \

2300 } \

2301 } \

2302 namespace ns\_name

2303

2304 #define DOCTEST\_TEST\_SUITE(decorators) \

2305 DOCTEST\_TEST\_SUITE\_IMPL(decorators, DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_SUITE\_))

2306

2307 // for starting a testsuite block

2308 #define DOCTEST\_TEST\_SUITE\_BEGIN(decorators) \

2309 DOCTEST\_GLOBAL\_NO\_WARNINGS(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_VAR\_), /\* NOLINT(cert-err58-cpp) \*/ \

2310 doctest::detail::setTestSuite(doctest::detail::TestSuite() \* decorators)) \

2311 static\_assert(true, "")

2312

2313 // for ending a testsuite block

2314 #define DOCTEST\_TEST\_SUITE\_END \

2315 DOCTEST\_GLOBAL\_NO\_WARNINGS(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_VAR\_), /\* NOLINT(cert-err58-cpp) \*/ \

2316 doctest::detail::setTestSuite(doctest::detail::TestSuite() \* "")) \

2317 using DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_FOR\_SEMICOLON\_) = int

2318

2319 // for registering exception translators

2320 #define DOCTEST\_REGISTER\_EXCEPTION\_TRANSLATOR\_IMPL(translatorName, signature) \

2321 inline doctest::String translatorName(signature); \

2322 DOCTEST\_GLOBAL\_NO\_WARNINGS(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TRANSLATOR\_), /\* NOLINT(cert-err58-cpp) \*/ \

2323 doctest::registerExceptionTranslator(translatorName)) \

2324 doctest::String translatorName(signature)

2325

2326 #define DOCTEST\_REGISTER\_EXCEPTION\_TRANSLATOR(signature) \

2327 DOCTEST\_REGISTER\_EXCEPTION\_TRANSLATOR\_IMPL(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TRANSLATOR\_), \

2328 signature)

2329

2330 // for registering reporters

2331 #define DOCTEST\_REGISTER\_REPORTER(name, priority, reporter) \

2332 DOCTEST\_GLOBAL\_NO\_WARNINGS(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_REPORTER\_), /\* NOLINT(cert-err58-cpp) \*/ \

2333 doctest::registerReporter<reporter>(name, priority, true)) \

2334 static\_assert(true, "")

2335

2336 // for registering listeners

2337 #define DOCTEST\_REGISTER\_LISTENER(name, priority, reporter) \

2338 DOCTEST\_GLOBAL\_NO\_WARNINGS(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_REPORTER\_), /\* NOLINT(cert-err58-cpp) \*/ \

2339 doctest::registerReporter<reporter>(name, priority, false)) \

2340 static\_assert(true, "")

2341

2342 // clang-format off

2343 // for logging - disabling formatting because it's important to have these on 2 separate lines - see PR #557

2344 #define DOCTEST\_INFO(...) \

2345 DOCTEST\_INFO\_IMPL(DOCTEST\_ANONYMOUS(DOCTEST\_CAPTURE\_), \

2346 DOCTEST\_ANONYMOUS(DOCTEST\_CAPTURE\_OTHER\_), \

2347 \_\_VA\_ARGS\_\_)

2348 // clang-format on

2349

2350 #define DOCTEST\_INFO\_IMPL(mb\_name, s\_name, ...) \

2351 auto DOCTEST\_ANONYMOUS(DOCTEST\_CAPTURE\_) = doctest::detail::MakeContextScope( \

2352 [&](std::ostream\* s\_name) { \

2353 doctest::detail::MessageBuilder mb\_name(\_\_FILE\_\_, \_\_LINE\_\_, doctest::assertType::is\_warn); \

2354 mb\_name.m\_stream = s\_name; \

2355 mb\_name \* \_\_VA\_ARGS\_\_; \

2356 })

2357

2358 #define DOCTEST\_CAPTURE(x) DOCTEST\_INFO(#x " := ", x)

2359

2360 #define DOCTEST\_ADD\_AT\_IMPL(type, file, line, mb, ...) \

2361 DOCTEST\_FUNC\_SCOPE\_BEGIN { \

2362 doctest::detail::MessageBuilder mb(file, line, doctest::assertType::type); \

2363 mb \* \_\_VA\_ARGS\_\_; \

2364 if(mb.log()) \

2365 DOCTEST\_BREAK\_INTO\_DEBUGGER(); \

2366 mb.react(); \

2367 } DOCTEST\_FUNC\_SCOPE\_END

2368

2369 // clang-format off

2370 #define DOCTEST\_ADD\_MESSAGE\_AT(file, line, ...) DOCTEST\_ADD\_AT\_IMPL(is\_warn, file, line, DOCTEST\_ANONYMOUS(DOCTEST\_MESSAGE\_), \_\_VA\_ARGS\_\_)

2371 #define DOCTEST\_ADD\_FAIL\_CHECK\_AT(file, line, ...) DOCTEST\_ADD\_AT\_IMPL(is\_check, file, line, DOCTEST\_ANONYMOUS(DOCTEST\_MESSAGE\_), \_\_VA\_ARGS\_\_)

2372 #define DOCTEST\_ADD\_FAIL\_AT(file, line, ...) DOCTEST\_ADD\_AT\_IMPL(is\_require, file, line, DOCTEST\_ANONYMOUS(DOCTEST\_MESSAGE\_), \_\_VA\_ARGS\_\_)

2373 // clang-format on

2374

2375 #define DOCTEST\_MESSAGE(...) DOCTEST\_ADD\_MESSAGE\_AT(\_\_FILE\_\_, \_\_LINE\_\_, \_\_VA\_ARGS\_\_)

2376 #define DOCTEST\_FAIL\_CHECK(...) DOCTEST\_ADD\_FAIL\_CHECK\_AT(\_\_FILE\_\_, \_\_LINE\_\_, \_\_VA\_ARGS\_\_)

2377 #define DOCTEST\_FAIL(...) DOCTEST\_ADD\_FAIL\_AT(\_\_FILE\_\_, \_\_LINE\_\_, \_\_VA\_ARGS\_\_)

2378

2379 #define DOCTEST\_TO\_LVALUE(...) \_\_VA\_ARGS\_\_ // Not removed to keep backwards compatibility.

2380

2381 #ifndef DOCTEST\_CONFIG\_SUPER\_FAST\_ASSERTS

2382

2383 #define DOCTEST\_ASSERT\_IMPLEMENT\_2(assert\_type, ...) \

2384 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Woverloaded-shift-op-parentheses") \

2385 /\* NOLINTNEXTLINE(clang-analyzer-cplusplus.NewDeleteLeaks) \*/ \

2386 doctest::detail::ResultBuilder DOCTEST\_RB(doctest::assertType::assert\_type, \_\_FILE\_\_, \

2387 \_\_LINE\_\_, #\_\_VA\_ARGS\_\_); \

2388 DOCTEST\_WRAP\_IN\_TRY(DOCTEST\_RB.setResult( \

2389 doctest::detail::ExpressionDecomposer(doctest::assertType::assert\_type) \

2390 << \_\_VA\_ARGS\_\_)) /\* NOLINTNEXTLINE(clang-analyzer-cplusplus.NewDeleteLeaks) \*/ \

2391 DOCTEST\_ASSERT\_LOG\_REACT\_RETURN(DOCTEST\_RB) \

2392 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

2393

2394 #define DOCTEST\_ASSERT\_IMPLEMENT\_1(assert\_type, ...) \

2395 DOCTEST\_FUNC\_SCOPE\_BEGIN { \

2396 DOCTEST\_ASSERT\_IMPLEMENT\_2(assert\_type, \_\_VA\_ARGS\_\_); \

2397 } DOCTEST\_FUNC\_SCOPE\_END // NOLINT(clang-analyzer-cplusplus.NewDeleteLeaks)

2398

2399 #define DOCTEST\_BINARY\_ASSERT(assert\_type, comp, ...) \

2400 DOCTEST\_FUNC\_SCOPE\_BEGIN { \

2401 doctest::detail::ResultBuilder DOCTEST\_RB(doctest::assertType::assert\_type, \_\_FILE\_\_, \

2402 \_\_LINE\_\_, #\_\_VA\_ARGS\_\_); \

2403 DOCTEST\_WRAP\_IN\_TRY( \

2404 DOCTEST\_RB.binary\_assert<doctest::detail::binaryAssertComparison::comp>( \

2405 \_\_VA\_ARGS\_\_)) \

2406 DOCTEST\_ASSERT\_LOG\_REACT\_RETURN(DOCTEST\_RB); \

2407 } DOCTEST\_FUNC\_SCOPE\_END

2408

2409 #define DOCTEST\_UNARY\_ASSERT(assert\_type, ...) \

2410 DOCTEST\_FUNC\_SCOPE\_BEGIN { \

2411 doctest::detail::ResultBuilder DOCTEST\_RB(doctest::assertType::assert\_type, \_\_FILE\_\_, \

2412 \_\_LINE\_\_, #\_\_VA\_ARGS\_\_); \

2413 DOCTEST\_WRAP\_IN\_TRY(DOCTEST\_RB.unary\_assert(\_\_VA\_ARGS\_\_)) \

2414 DOCTEST\_ASSERT\_LOG\_REACT\_RETURN(DOCTEST\_RB); \

2415 } DOCTEST\_FUNC\_SCOPE\_END

2416

2417 #else // DOCTEST\_CONFIG\_SUPER\_FAST\_ASSERTS

2418

2419 // necessary for <ASSERT>\_MESSAGE

2420 #define DOCTEST\_ASSERT\_IMPLEMENT\_2 DOCTEST\_ASSERT\_IMPLEMENT\_1

2421

2422 #define DOCTEST\_ASSERT\_IMPLEMENT\_1(assert\_type, ...) \

2423 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Woverloaded-shift-op-parentheses") \

2424 doctest::detail::decomp\_assert( \

2425 doctest::assertType::assert\_type, \_\_FILE\_\_, \_\_LINE\_\_, #\_\_VA\_ARGS\_\_, \

2426 doctest::detail::ExpressionDecomposer(doctest::assertType::assert\_type) \

2427 << \_\_VA\_ARGS\_\_) DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

2428

2429 #define DOCTEST\_BINARY\_ASSERT(assert\_type, comparison, ...) \

2430 doctest::detail::binary\_assert<doctest::detail::binaryAssertComparison::comparison>( \

2431 doctest::assertType::assert\_type, \_\_FILE\_\_, \_\_LINE\_\_, #\_\_VA\_ARGS\_\_, \_\_VA\_ARGS\_\_)

2432

2433 #define DOCTEST\_UNARY\_ASSERT(assert\_type, ...) \

2434 doctest::detail::unary\_assert(doctest::assertType::assert\_type, \_\_FILE\_\_, \_\_LINE\_\_, \

2435 #\_\_VA\_ARGS\_\_, \_\_VA\_ARGS\_\_)

2436

2437 #endif // DOCTEST\_CONFIG\_SUPER\_FAST\_ASSERTS

2438

2439 #define DOCTEST\_WARN(...) DOCTEST\_ASSERT\_IMPLEMENT\_1(DT\_WARN, \_\_VA\_ARGS\_\_)

2440 #define DOCTEST\_CHECK(...) DOCTEST\_ASSERT\_IMPLEMENT\_1(DT\_CHECK, \_\_VA\_ARGS\_\_)

2441 #define DOCTEST\_REQUIRE(...) DOCTEST\_ASSERT\_IMPLEMENT\_1(DT\_REQUIRE, \_\_VA\_ARGS\_\_)

2442 #define DOCTEST\_WARN\_FALSE(...) DOCTEST\_ASSERT\_IMPLEMENT\_1(DT\_WARN\_FALSE, \_\_VA\_ARGS\_\_)

2443 #define DOCTEST\_CHECK\_FALSE(...) DOCTEST\_ASSERT\_IMPLEMENT\_1(DT\_CHECK\_FALSE, \_\_VA\_ARGS\_\_)

2444 #define DOCTEST\_REQUIRE\_FALSE(...) DOCTEST\_ASSERT\_IMPLEMENT\_1(DT\_REQUIRE\_FALSE, \_\_VA\_ARGS\_\_)

2445

2446 // clang-format off

2447 #define DOCTEST\_WARN\_MESSAGE(cond, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_ASSERT\_IMPLEMENT\_2(DT\_WARN, cond); } DOCTEST\_FUNC\_SCOPE\_END

2448 #define DOCTEST\_CHECK\_MESSAGE(cond, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_ASSERT\_IMPLEMENT\_2(DT\_CHECK, cond); } DOCTEST\_FUNC\_SCOPE\_END

2449 #define DOCTEST\_REQUIRE\_MESSAGE(cond, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_ASSERT\_IMPLEMENT\_2(DT\_REQUIRE, cond); } DOCTEST\_FUNC\_SCOPE\_END

2450 #define DOCTEST\_WARN\_FALSE\_MESSAGE(cond, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_ASSERT\_IMPLEMENT\_2(DT\_WARN\_FALSE, cond); } DOCTEST\_FUNC\_SCOPE\_END

2451 #define DOCTEST\_CHECK\_FALSE\_MESSAGE(cond, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_ASSERT\_IMPLEMENT\_2(DT\_CHECK\_FALSE, cond); } DOCTEST\_FUNC\_SCOPE\_END

2452 #define DOCTEST\_REQUIRE\_FALSE\_MESSAGE(cond, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_ASSERT\_IMPLEMENT\_2(DT\_REQUIRE\_FALSE, cond); } DOCTEST\_FUNC\_SCOPE\_END

2453 // clang-format on

2454

2455 #define DOCTEST\_WARN\_EQ(...) DOCTEST\_BINARY\_ASSERT(DT\_WARN\_EQ, eq, \_\_VA\_ARGS\_\_)

2456 #define DOCTEST\_CHECK\_EQ(...) DOCTEST\_BINARY\_ASSERT(DT\_CHECK\_EQ, eq, \_\_VA\_ARGS\_\_)

2457 #define DOCTEST\_REQUIRE\_EQ(...) DOCTEST\_BINARY\_ASSERT(DT\_REQUIRE\_EQ, eq, \_\_VA\_ARGS\_\_)

2458 #define DOCTEST\_WARN\_NE(...) DOCTEST\_BINARY\_ASSERT(DT\_WARN\_NE, ne, \_\_VA\_ARGS\_\_)

2459 #define DOCTEST\_CHECK\_NE(...) DOCTEST\_BINARY\_ASSERT(DT\_CHECK\_NE, ne, \_\_VA\_ARGS\_\_)

2460 #define DOCTEST\_REQUIRE\_NE(...) DOCTEST\_BINARY\_ASSERT(DT\_REQUIRE\_NE, ne, \_\_VA\_ARGS\_\_)

2461 #define DOCTEST\_WARN\_GT(...) DOCTEST\_BINARY\_ASSERT(DT\_WARN\_GT, gt, \_\_VA\_ARGS\_\_)

2462 #define DOCTEST\_CHECK\_GT(...) DOCTEST\_BINARY\_ASSERT(DT\_CHECK\_GT, gt, \_\_VA\_ARGS\_\_)

2463 #define DOCTEST\_REQUIRE\_GT(...) DOCTEST\_BINARY\_ASSERT(DT\_REQUIRE\_GT, gt, \_\_VA\_ARGS\_\_)

2464 #define DOCTEST\_WARN\_LT(...) DOCTEST\_BINARY\_ASSERT(DT\_WARN\_LT, lt, \_\_VA\_ARGS\_\_)

2465 #define DOCTEST\_CHECK\_LT(...) DOCTEST\_BINARY\_ASSERT(DT\_CHECK\_LT, lt, \_\_VA\_ARGS\_\_)

2466 #define DOCTEST\_REQUIRE\_LT(...) DOCTEST\_BINARY\_ASSERT(DT\_REQUIRE\_LT, lt, \_\_VA\_ARGS\_\_)

2467 #define DOCTEST\_WARN\_GE(...) DOCTEST\_BINARY\_ASSERT(DT\_WARN\_GE, ge, \_\_VA\_ARGS\_\_)

2468 #define DOCTEST\_CHECK\_GE(...) DOCTEST\_BINARY\_ASSERT(DT\_CHECK\_GE, ge, \_\_VA\_ARGS\_\_)

2469 #define DOCTEST\_REQUIRE\_GE(...) DOCTEST\_BINARY\_ASSERT(DT\_REQUIRE\_GE, ge, \_\_VA\_ARGS\_\_)

2470 #define DOCTEST\_WARN\_LE(...) DOCTEST\_BINARY\_ASSERT(DT\_WARN\_LE, le, \_\_VA\_ARGS\_\_)

2471 #define DOCTEST\_CHECK\_LE(...) DOCTEST\_BINARY\_ASSERT(DT\_CHECK\_LE, le, \_\_VA\_ARGS\_\_)

2472 #define DOCTEST\_REQUIRE\_LE(...) DOCTEST\_BINARY\_ASSERT(DT\_REQUIRE\_LE, le, \_\_VA\_ARGS\_\_)

2473

2474 #define DOCTEST\_WARN\_UNARY(...) DOCTEST\_UNARY\_ASSERT(DT\_WARN\_UNARY, \_\_VA\_ARGS\_\_)

2475 #define DOCTEST\_CHECK\_UNARY(...) DOCTEST\_UNARY\_ASSERT(DT\_CHECK\_UNARY, \_\_VA\_ARGS\_\_)

2476 #define DOCTEST\_REQUIRE\_UNARY(...) DOCTEST\_UNARY\_ASSERT(DT\_REQUIRE\_UNARY, \_\_VA\_ARGS\_\_)

2477 #define DOCTEST\_WARN\_UNARY\_FALSE(...) DOCTEST\_UNARY\_ASSERT(DT\_WARN\_UNARY\_FALSE, \_\_VA\_ARGS\_\_)

2478 #define DOCTEST\_CHECK\_UNARY\_FALSE(...) DOCTEST\_UNARY\_ASSERT(DT\_CHECK\_UNARY\_FALSE, \_\_VA\_ARGS\_\_)

2479 #define DOCTEST\_REQUIRE\_UNARY\_FALSE(...) DOCTEST\_UNARY\_ASSERT(DT\_REQUIRE\_UNARY\_FALSE, \_\_VA\_ARGS\_\_)

2480

2481 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

2482

2483 #define DOCTEST\_ASSERT\_THROWS\_AS(expr, assert\_type, message, ...) \

2484 DOCTEST\_FUNC\_SCOPE\_BEGIN { \

2485 if(!doctest::getContextOptions()->no\_throw) { \

2486 doctest::detail::ResultBuilder DOCTEST\_RB(doctest::assertType::assert\_type, \_\_FILE\_\_, \

2487 \_\_LINE\_\_, #expr, #\_\_VA\_ARGS\_\_, message); \

2488 try { \

2489 DOCTEST\_CAST\_TO\_VOID(expr) \

2490 } catch(const typename doctest::detail::types::remove\_const< \

2491 typename doctest::detail::types::remove\_reference<\_\_VA\_ARGS\_\_>::type>::type&) {\

2492 DOCTEST\_RB.translateException(); \

2493 DOCTEST\_RB.m\_threw\_as = true; \

2494 } catch(...) { DOCTEST\_RB.translateException(); } \

2495 DOCTEST\_ASSERT\_LOG\_REACT\_RETURN(DOCTEST\_RB); \

2496 } else { /\* NOLINT(\*-else-after-return) \*/ \

2497 DOCTEST\_FUNC\_SCOPE\_RET(false); \

2498 } \

2499 } DOCTEST\_FUNC\_SCOPE\_END

2500

2501 #define DOCTEST\_ASSERT\_THROWS\_WITH(expr, expr\_str, assert\_type, ...) \

2502 DOCTEST\_FUNC\_SCOPE\_BEGIN { \

2503 if(!doctest::getContextOptions()->no\_throw) { \

2504 doctest::detail::ResultBuilder DOCTEST\_RB(doctest::assertType::assert\_type, \_\_FILE\_\_, \

2505 \_\_LINE\_\_, expr\_str, "", \_\_VA\_ARGS\_\_); \

2506 try { \

2507 DOCTEST\_CAST\_TO\_VOID(expr) \

2508 } catch(...) { DOCTEST\_RB.translateException(); } \

2509 DOCTEST\_ASSERT\_LOG\_REACT\_RETURN(DOCTEST\_RB); \

2510 } else { /\* NOLINT(\*-else-after-return) \*/ \

2511 DOCTEST\_FUNC\_SCOPE\_RET(false); \

2512 } \

2513 } DOCTEST\_FUNC\_SCOPE\_END

2514

2515 #define DOCTEST\_ASSERT\_NOTHROW(assert\_type, ...) \

2516 DOCTEST\_FUNC\_SCOPE\_BEGIN { \

2517 doctest::detail::ResultBuilder DOCTEST\_RB(doctest::assertType::assert\_type, \_\_FILE\_\_, \

2518 \_\_LINE\_\_, #\_\_VA\_ARGS\_\_); \

2519 try { \

2520 DOCTEST\_CAST\_TO\_VOID(\_\_VA\_ARGS\_\_) \

2521 } catch(...) { DOCTEST\_RB.translateException(); } \

2522 DOCTEST\_ASSERT\_LOG\_REACT\_RETURN(DOCTEST\_RB); \

2523 } DOCTEST\_FUNC\_SCOPE\_END

2524

2525 // clang-format off

2526 #define DOCTEST\_WARN\_THROWS(...) DOCTEST\_ASSERT\_THROWS\_WITH((\_\_VA\_ARGS\_\_), #\_\_VA\_ARGS\_\_, DT\_WARN\_THROWS, "")

2527 #define DOCTEST\_CHECK\_THROWS(...) DOCTEST\_ASSERT\_THROWS\_WITH((\_\_VA\_ARGS\_\_), #\_\_VA\_ARGS\_\_, DT\_CHECK\_THROWS, "")

2528 #define DOCTEST\_REQUIRE\_THROWS(...) DOCTEST\_ASSERT\_THROWS\_WITH((\_\_VA\_ARGS\_\_), #\_\_VA\_ARGS\_\_, DT\_REQUIRE\_THROWS, "")

2529

2530 #define DOCTEST\_WARN\_THROWS\_AS(expr, ...) DOCTEST\_ASSERT\_THROWS\_AS(expr, DT\_WARN\_THROWS\_AS, "", \_\_VA\_ARGS\_\_)

2531 #define DOCTEST\_CHECK\_THROWS\_AS(expr, ...) DOCTEST\_ASSERT\_THROWS\_AS(expr, DT\_CHECK\_THROWS\_AS, "", \_\_VA\_ARGS\_\_)

2532 #define DOCTEST\_REQUIRE\_THROWS\_AS(expr, ...) DOCTEST\_ASSERT\_THROWS\_AS(expr, DT\_REQUIRE\_THROWS\_AS, "", \_\_VA\_ARGS\_\_)

2533

2534 #define DOCTEST\_WARN\_THROWS\_WITH(expr, ...) DOCTEST\_ASSERT\_THROWS\_WITH(expr, #expr, DT\_WARN\_THROWS\_WITH, \_\_VA\_ARGS\_\_)

2535 #define DOCTEST\_CHECK\_THROWS\_WITH(expr, ...) DOCTEST\_ASSERT\_THROWS\_WITH(expr, #expr, DT\_CHECK\_THROWS\_WITH, \_\_VA\_ARGS\_\_)

2536 #define DOCTEST\_REQUIRE\_THROWS\_WITH(expr, ...) DOCTEST\_ASSERT\_THROWS\_WITH(expr, #expr, DT\_REQUIRE\_THROWS\_WITH, \_\_VA\_ARGS\_\_)

2537

2538 #define DOCTEST\_WARN\_THROWS\_WITH\_AS(expr, message, ...) DOCTEST\_ASSERT\_THROWS\_AS(expr, DT\_WARN\_THROWS\_WITH\_AS, message, \_\_VA\_ARGS\_\_)

2539 #define DOCTEST\_CHECK\_THROWS\_WITH\_AS(expr, message, ...) DOCTEST\_ASSERT\_THROWS\_AS(expr, DT\_CHECK\_THROWS\_WITH\_AS, message, \_\_VA\_ARGS\_\_)

2540 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_AS(expr, message, ...) DOCTEST\_ASSERT\_THROWS\_AS(expr, DT\_REQUIRE\_THROWS\_WITH\_AS, message, \_\_VA\_ARGS\_\_)

2541

2542 #define DOCTEST\_WARN\_NOTHROW(...) DOCTEST\_ASSERT\_NOTHROW(DT\_WARN\_NOTHROW, \_\_VA\_ARGS\_\_)

2543 #define DOCTEST\_CHECK\_NOTHROW(...) DOCTEST\_ASSERT\_NOTHROW(DT\_CHECK\_NOTHROW, \_\_VA\_ARGS\_\_)

2544 #define DOCTEST\_REQUIRE\_NOTHROW(...) DOCTEST\_ASSERT\_NOTHROW(DT\_REQUIRE\_NOTHROW, \_\_VA\_ARGS\_\_)

2545

2546 #define DOCTEST\_WARN\_THROWS\_MESSAGE(expr, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_WARN\_THROWS(expr); } DOCTEST\_FUNC\_SCOPE\_END

2547 #define DOCTEST\_CHECK\_THROWS\_MESSAGE(expr, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_CHECK\_THROWS(expr); } DOCTEST\_FUNC\_SCOPE\_END

2548 #define DOCTEST\_REQUIRE\_THROWS\_MESSAGE(expr, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_REQUIRE\_THROWS(expr); } DOCTEST\_FUNC\_SCOPE\_END

2549 #define DOCTEST\_WARN\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_WARN\_THROWS\_AS(expr, ex); } DOCTEST\_FUNC\_SCOPE\_END

2550 #define DOCTEST\_CHECK\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_CHECK\_THROWS\_AS(expr, ex); } DOCTEST\_FUNC\_SCOPE\_END

2551 #define DOCTEST\_REQUIRE\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_REQUIRE\_THROWS\_AS(expr, ex); } DOCTEST\_FUNC\_SCOPE\_END

2552 #define DOCTEST\_WARN\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_WARN\_THROWS\_WITH(expr, with); } DOCTEST\_FUNC\_SCOPE\_END

2553 #define DOCTEST\_CHECK\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_CHECK\_THROWS\_WITH(expr, with); } DOCTEST\_FUNC\_SCOPE\_END

2554 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_REQUIRE\_THROWS\_WITH(expr, with); } DOCTEST\_FUNC\_SCOPE\_END

2555 #define DOCTEST\_WARN\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_WARN\_THROWS\_WITH\_AS(expr, with, ex); } DOCTEST\_FUNC\_SCOPE\_END

2556 #define DOCTEST\_CHECK\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_CHECK\_THROWS\_WITH\_AS(expr, with, ex); } DOCTEST\_FUNC\_SCOPE\_END

2557 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_REQUIRE\_THROWS\_WITH\_AS(expr, with, ex); } DOCTEST\_FUNC\_SCOPE\_END

2558 #define DOCTEST\_WARN\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_WARN\_NOTHROW(expr); } DOCTEST\_FUNC\_SCOPE\_END

2559 #define DOCTEST\_CHECK\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_CHECK\_NOTHROW(expr); } DOCTEST\_FUNC\_SCOPE\_END

2560 #define DOCTEST\_REQUIRE\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_FUNC\_SCOPE\_BEGIN { DOCTEST\_INFO(\_\_VA\_ARGS\_\_); DOCTEST\_REQUIRE\_NOTHROW(expr); } DOCTEST\_FUNC\_SCOPE\_END

2561 // clang-format on

2562

2563 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

2564

2565 // =================================================================================================

2566 // == WHAT FOLLOWS IS VERSIONS OF THE MACROS THAT DO NOT DO ANY REGISTERING! ==

2567 // == THIS CAN BE ENABLED BY DEFINING DOCTEST\_CONFIG\_DISABLE GLOBALLY! ==

2568 // =================================================================================================

2569 #else // DOCTEST\_CONFIG\_DISABLE

2570

2571 #define DOCTEST\_IMPLEMENT\_FIXTURE(der, base, func, name) \

2572 namespace /\* NOLINT \*/ { \

2573 template <typename DOCTEST\_UNUSED\_TEMPLATE\_TYPE> \

2574 struct der : public base \

2575 { void f(); }; \

2576 } \

2577 template <typename DOCTEST\_UNUSED\_TEMPLATE\_TYPE> \

2578 inline void der<DOCTEST\_UNUSED\_TEMPLATE\_TYPE>::f()

2579

2580 #define DOCTEST\_CREATE\_AND\_REGISTER\_FUNCTION(f, name) \

2581 template <typename DOCTEST\_UNUSED\_TEMPLATE\_TYPE> \

2582 static inline void f()

2583

2584 // for registering tests

2585 #define DOCTEST\_TEST\_CASE(name) \

2586 DOCTEST\_CREATE\_AND\_REGISTER\_FUNCTION(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_FUNC\_), name)

2587

2588 // for registering tests in classes

2589 #define DOCTEST\_TEST\_CASE\_CLASS(name) \

2590 DOCTEST\_CREATE\_AND\_REGISTER\_FUNCTION(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_FUNC\_), name)

2591

2592 // for registering tests with a fixture

2593 #define DOCTEST\_TEST\_CASE\_FIXTURE(x, name) \

2594 DOCTEST\_IMPLEMENT\_FIXTURE(DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_CLASS\_), x, \

2595 DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_FUNC\_), name)

2596

2597 // for converting types to strings without the <typeinfo> header and demangling

2598 #define DOCTEST\_TYPE\_TO\_STRING\_AS(str, ...) static\_assert(true, "")

2599 #define DOCTEST\_TYPE\_TO\_STRING(...) static\_assert(true, "")

2600

2601 // for typed tests

2602 #define DOCTEST\_TEST\_CASE\_TEMPLATE(name, type, ...) \

2603 template <typename type> \

2604 inline void DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TMP\_)()

2605

2606 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_DEFINE(name, type, id) \

2607 template <typename type> \

2608 inline void DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TMP\_)()

2609

2610 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_INVOKE(id, ...) static\_assert(true, "")

2611 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_APPLY(id, ...) static\_assert(true, "")

2612

2613 // for subcases

2614 #define DOCTEST\_SUBCASE(name)

2615

2616 // for a testsuite block

2617 #define DOCTEST\_TEST\_SUITE(name) namespace // NOLINT

2618

2619 // for starting a testsuite block

2620 #define DOCTEST\_TEST\_SUITE\_BEGIN(name) static\_assert(true, "")

2621

2622 // for ending a testsuite block

2623 #define DOCTEST\_TEST\_SUITE\_END using DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_FOR\_SEMICOLON\_) = int

2624

2625 #define DOCTEST\_REGISTER\_EXCEPTION\_TRANSLATOR(signature) \

2626 template <typename DOCTEST\_UNUSED\_TEMPLATE\_TYPE> \

2627 static inline doctest::String DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_TRANSLATOR\_)(signature)

2628

2629 #define DOCTEST\_REGISTER\_REPORTER(name, priority, reporter)

2630 #define DOCTEST\_REGISTER\_LISTENER(name, priority, reporter)

2631

2632 #define DOCTEST\_INFO(...) (static\_cast<void>(0))

2633 #define DOCTEST\_CAPTURE(x) (static\_cast<void>(0))

2634 #define DOCTEST\_ADD\_MESSAGE\_AT(file, line, ...) (static\_cast<void>(0))

2635 #define DOCTEST\_ADD\_FAIL\_CHECK\_AT(file, line, ...) (static\_cast<void>(0))

2636 #define DOCTEST\_ADD\_FAIL\_AT(file, line, ...) (static\_cast<void>(0))

2637 #define DOCTEST\_MESSAGE(...) (static\_cast<void>(0))

2638 #define DOCTEST\_FAIL\_CHECK(...) (static\_cast<void>(0))

2639 #define DOCTEST\_FAIL(...) (static\_cast<void>(0))

2640

2641 #if defined(DOCTEST\_CONFIG\_EVALUATE\_ASSERTS\_EVEN\_WHEN\_DISABLED) \

2642 && defined(DOCTEST\_CONFIG\_ASSERTS\_RETURN\_VALUES)

2643

2644 #define DOCTEST\_WARN(...) [&] { return \_\_VA\_ARGS\_\_; }()

2645 #define DOCTEST\_CHECK(...) [&] { return \_\_VA\_ARGS\_\_; }()

2646 #define DOCTEST\_REQUIRE(...) [&] { return \_\_VA\_ARGS\_\_; }()

2647 #define DOCTEST\_WARN\_FALSE(...) [&] { return !(\_\_VA\_ARGS\_\_); }()

2648 #define DOCTEST\_CHECK\_FALSE(...) [&] { return !(\_\_VA\_ARGS\_\_); }()

2649 #define DOCTEST\_REQUIRE\_FALSE(...) [&] { return !(\_\_VA\_ARGS\_\_); }()

2650

2651 #define DOCTEST\_WARN\_MESSAGE(cond, ...) [&] { return cond; }()

2652 #define DOCTEST\_CHECK\_MESSAGE(cond, ...) [&] { return cond; }()

2653 #define DOCTEST\_REQUIRE\_MESSAGE(cond, ...) [&] { return cond; }()

2654 #define DOCTEST\_WARN\_FALSE\_MESSAGE(cond, ...) [&] { return !(cond); }()

2655 #define DOCTEST\_CHECK\_FALSE\_MESSAGE(cond, ...) [&] { return !(cond); }()

2656 #define DOCTEST\_REQUIRE\_FALSE\_MESSAGE(cond, ...) [&] { return !(cond); }()

2657

2658 namespace doctest {

2659 namespace detail {

2660 #define DOCTEST\_RELATIONAL\_OP(name, op) \

2661 template <typename L, typename R> \

2662 bool name(const DOCTEST\_REF\_WRAP(L) lhs, const DOCTEST\_REF\_WRAP(R) rhs) { return lhs op rhs; }

2663

2664 DOCTEST\_RELATIONAL\_OP(eq, ==)

2665 DOCTEST\_RELATIONAL\_OP(ne, !=)

2666 DOCTEST\_RELATIONAL\_OP(lt, <)

2667 DOCTEST\_RELATIONAL\_OP(gt, >)

2668 DOCTEST\_RELATIONAL\_OP(le, <=)

2669 DOCTEST\_RELATIONAL\_OP(ge, >=)

2670 } // namespace detail

2671 } // namespace doctest

2672

2673 #define DOCTEST\_WARN\_EQ(...) [&] { return doctest::detail::eq(\_\_VA\_ARGS\_\_); }()

2674 #define DOCTEST\_CHECK\_EQ(...) [&] { return doctest::detail::eq(\_\_VA\_ARGS\_\_); }()

2675 #define DOCTEST\_REQUIRE\_EQ(...) [&] { return doctest::detail::eq(\_\_VA\_ARGS\_\_); }()

2676 #define DOCTEST\_WARN\_NE(...) [&] { return doctest::detail::ne(\_\_VA\_ARGS\_\_); }()

2677 #define DOCTEST\_CHECK\_NE(...) [&] { return doctest::detail::ne(\_\_VA\_ARGS\_\_); }()

2678 #define DOCTEST\_REQUIRE\_NE(...) [&] { return doctest::detail::ne(\_\_VA\_ARGS\_\_); }()

2679 #define DOCTEST\_WARN\_LT(...) [&] { return doctest::detail::lt(\_\_VA\_ARGS\_\_); }()

2680 #define DOCTEST\_CHECK\_LT(...) [&] { return doctest::detail::lt(\_\_VA\_ARGS\_\_); }()

2681 #define DOCTEST\_REQUIRE\_LT(...) [&] { return doctest::detail::lt(\_\_VA\_ARGS\_\_); }()

2682 #define DOCTEST\_WARN\_GT(...) [&] { return doctest::detail::gt(\_\_VA\_ARGS\_\_); }()

2683 #define DOCTEST\_CHECK\_GT(...) [&] { return doctest::detail::gt(\_\_VA\_ARGS\_\_); }()

2684 #define DOCTEST\_REQUIRE\_GT(...) [&] { return doctest::detail::gt(\_\_VA\_ARGS\_\_); }()

2685 #define DOCTEST\_WARN\_LE(...) [&] { return doctest::detail::le(\_\_VA\_ARGS\_\_); }()

2686 #define DOCTEST\_CHECK\_LE(...) [&] { return doctest::detail::le(\_\_VA\_ARGS\_\_); }()

2687 #define DOCTEST\_REQUIRE\_LE(...) [&] { return doctest::detail::le(\_\_VA\_ARGS\_\_); }()

2688 #define DOCTEST\_WARN\_GE(...) [&] { return doctest::detail::ge(\_\_VA\_ARGS\_\_); }()

2689 #define DOCTEST\_CHECK\_GE(...) [&] { return doctest::detail::ge(\_\_VA\_ARGS\_\_); }()

2690 #define DOCTEST\_REQUIRE\_GE(...) [&] { return doctest::detail::ge(\_\_VA\_ARGS\_\_); }()

2691 #define DOCTEST\_WARN\_UNARY(...) [&] { return \_\_VA\_ARGS\_\_; }()

2692 #define DOCTEST\_CHECK\_UNARY(...) [&] { return \_\_VA\_ARGS\_\_; }()

2693 #define DOCTEST\_REQUIRE\_UNARY(...) [&] { return \_\_VA\_ARGS\_\_; }()

2694 #define DOCTEST\_WARN\_UNARY\_FALSE(...) [&] { return !(\_\_VA\_ARGS\_\_); }()

2695 #define DOCTEST\_CHECK\_UNARY\_FALSE(...) [&] { return !(\_\_VA\_ARGS\_\_); }()

2696 #define DOCTEST\_REQUIRE\_UNARY\_FALSE(...) [&] { return !(\_\_VA\_ARGS\_\_); }()

2697

2698 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

2699

2700 #define DOCTEST\_WARN\_THROWS\_WITH(expr, with, ...) [] { static\_assert(false, "Exception translation is not available when doctest is disabled."); return false; }()

2701 #define DOCTEST\_CHECK\_THROWS\_WITH(expr, with, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2702 #define DOCTEST\_REQUIRE\_THROWS\_WITH(expr, with, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2703 #define DOCTEST\_WARN\_THROWS\_WITH\_AS(expr, with, ex, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2704 #define DOCTEST\_CHECK\_THROWS\_WITH\_AS(expr, with, ex, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2705 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_AS(expr, with, ex, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2706

2707 #define DOCTEST\_WARN\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2708 #define DOCTEST\_CHECK\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2709 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2710 #define DOCTEST\_WARN\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2711 #define DOCTEST\_CHECK\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2712 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_WARN\_THROWS\_WITH(,,)

2713

2714 #define DOCTEST\_WARN\_THROWS(...) [&] { try { \_\_VA\_ARGS\_\_; return false; } catch (...) { return true; } }()

2715 #define DOCTEST\_CHECK\_THROWS(...) [&] { try { \_\_VA\_ARGS\_\_; return false; } catch (...) { return true; } }()

2716 #define DOCTEST\_REQUIRE\_THROWS(...) [&] { try { \_\_VA\_ARGS\_\_; return false; } catch (...) { return true; } }()

2717 #define DOCTEST\_WARN\_THROWS\_AS(expr, ...) [&] { try { expr; } catch (\_\_VA\_ARGS\_\_) { return true; } catch (...) { } return false; }()

2718 #define DOCTEST\_CHECK\_THROWS\_AS(expr, ...) [&] { try { expr; } catch (\_\_VA\_ARGS\_\_) { return true; } catch (...) { } return false; }()

2719 #define DOCTEST\_REQUIRE\_THROWS\_AS(expr, ...) [&] { try { expr; } catch (\_\_VA\_ARGS\_\_) { return true; } catch (...) { } return false; }()

2720 #define DOCTEST\_WARN\_NOTHROW(...) [&] { try { \_\_VA\_ARGS\_\_; return true; } catch (...) { return false; } }()

2721 #define DOCTEST\_CHECK\_NOTHROW(...) [&] { try { \_\_VA\_ARGS\_\_; return true; } catch (...) { return false; } }()

2722 #define DOCTEST\_REQUIRE\_NOTHROW(...) [&] { try { \_\_VA\_ARGS\_\_; return true; } catch (...) { return false; } }()

2723

2724 #define DOCTEST\_WARN\_THROWS\_MESSAGE(expr, ...) [&] { try { \_\_VA\_ARGS\_\_; return false; } catch (...) { return true; } }()

2725 #define DOCTEST\_CHECK\_THROWS\_MESSAGE(expr, ...) [&] { try { \_\_VA\_ARGS\_\_; return false; } catch (...) { return true; } }()

2726 #define DOCTEST\_REQUIRE\_THROWS\_MESSAGE(expr, ...) [&] { try { \_\_VA\_ARGS\_\_; return false; } catch (...) { return true; } }()

2727 #define DOCTEST\_WARN\_THROWS\_AS\_MESSAGE(expr, ex, ...) [&] { try { expr; } catch (\_\_VA\_ARGS\_\_) { return true; } catch (...) { } return false; }()

2728 #define DOCTEST\_CHECK\_THROWS\_AS\_MESSAGE(expr, ex, ...) [&] { try { expr; } catch (\_\_VA\_ARGS\_\_) { return true; } catch (...) { } return false; }()

2729 #define DOCTEST\_REQUIRE\_THROWS\_AS\_MESSAGE(expr, ex, ...) [&] { try { expr; } catch (\_\_VA\_ARGS\_\_) { return true; } catch (...) { } return false; }()

2730 #define DOCTEST\_WARN\_NOTHROW\_MESSAGE(expr, ...) [&] { try { \_\_VA\_ARGS\_\_; return true; } catch (...) { return false; } }()

2731 #define DOCTEST\_CHECK\_NOTHROW\_MESSAGE(expr, ...) [&] { try { \_\_VA\_ARGS\_\_; return true; } catch (...) { return false; } }()

2732 #define DOCTEST\_REQUIRE\_NOTHROW\_MESSAGE(expr, ...) [&] { try { \_\_VA\_ARGS\_\_; return true; } catch (...) { return false; } }()

2733

2734 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

2735

2736 #else // DOCTEST\_CONFIG\_EVALUATE\_ASSERTS\_EVEN\_WHEN\_DISABLED

2737

2738 #define DOCTEST\_WARN(...) DOCTEST\_FUNC\_EMPTY

2739 #define DOCTEST\_CHECK(...) DOCTEST\_FUNC\_EMPTY

2740 #define DOCTEST\_REQUIRE(...) DOCTEST\_FUNC\_EMPTY

2741 #define DOCTEST\_WARN\_FALSE(...) DOCTEST\_FUNC\_EMPTY

2742 #define DOCTEST\_CHECK\_FALSE(...) DOCTEST\_FUNC\_EMPTY

2743 #define DOCTEST\_REQUIRE\_FALSE(...) DOCTEST\_FUNC\_EMPTY

2744

2745 #define DOCTEST\_WARN\_MESSAGE(cond, ...) DOCTEST\_FUNC\_EMPTY

2746 #define DOCTEST\_CHECK\_MESSAGE(cond, ...) DOCTEST\_FUNC\_EMPTY

2747 #define DOCTEST\_REQUIRE\_MESSAGE(cond, ...) DOCTEST\_FUNC\_EMPTY

2748 #define DOCTEST\_WARN\_FALSE\_MESSAGE(cond, ...) DOCTEST\_FUNC\_EMPTY

2749 #define DOCTEST\_CHECK\_FALSE\_MESSAGE(cond, ...) DOCTEST\_FUNC\_EMPTY

2750 #define DOCTEST\_REQUIRE\_FALSE\_MESSAGE(cond, ...) DOCTEST\_FUNC\_EMPTY

2751

2752 #define DOCTEST\_WARN\_EQ(...) DOCTEST\_FUNC\_EMPTY

2753 #define DOCTEST\_CHECK\_EQ(...) DOCTEST\_FUNC\_EMPTY

2754 #define DOCTEST\_REQUIRE\_EQ(...) DOCTEST\_FUNC\_EMPTY

2755 #define DOCTEST\_WARN\_NE(...) DOCTEST\_FUNC\_EMPTY

2756 #define DOCTEST\_CHECK\_NE(...) DOCTEST\_FUNC\_EMPTY

2757 #define DOCTEST\_REQUIRE\_NE(...) DOCTEST\_FUNC\_EMPTY

2758 #define DOCTEST\_WARN\_GT(...) DOCTEST\_FUNC\_EMPTY

2759 #define DOCTEST\_CHECK\_GT(...) DOCTEST\_FUNC\_EMPTY

2760 #define DOCTEST\_REQUIRE\_GT(...) DOCTEST\_FUNC\_EMPTY

2761 #define DOCTEST\_WARN\_LT(...) DOCTEST\_FUNC\_EMPTY

2762 #define DOCTEST\_CHECK\_LT(...) DOCTEST\_FUNC\_EMPTY

2763 #define DOCTEST\_REQUIRE\_LT(...) DOCTEST\_FUNC\_EMPTY

2764 #define DOCTEST\_WARN\_GE(...) DOCTEST\_FUNC\_EMPTY

2765 #define DOCTEST\_CHECK\_GE(...) DOCTEST\_FUNC\_EMPTY

2766 #define DOCTEST\_REQUIRE\_GE(...) DOCTEST\_FUNC\_EMPTY

2767 #define DOCTEST\_WARN\_LE(...) DOCTEST\_FUNC\_EMPTY

2768 #define DOCTEST\_CHECK\_LE(...) DOCTEST\_FUNC\_EMPTY

2769 #define DOCTEST\_REQUIRE\_LE(...) DOCTEST\_FUNC\_EMPTY

2770

2771 #define DOCTEST\_WARN\_UNARY(...) DOCTEST\_FUNC\_EMPTY

2772 #define DOCTEST\_CHECK\_UNARY(...) DOCTEST\_FUNC\_EMPTY

2773 #define DOCTEST\_REQUIRE\_UNARY(...) DOCTEST\_FUNC\_EMPTY

2774 #define DOCTEST\_WARN\_UNARY\_FALSE(...) DOCTEST\_FUNC\_EMPTY

2775 #define DOCTEST\_CHECK\_UNARY\_FALSE(...) DOCTEST\_FUNC\_EMPTY

2776 #define DOCTEST\_REQUIRE\_UNARY\_FALSE(...) DOCTEST\_FUNC\_EMPTY

2777

2778 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

2779

2780 #define DOCTEST\_WARN\_THROWS(...) DOCTEST\_FUNC\_EMPTY

2781 #define DOCTEST\_CHECK\_THROWS(...) DOCTEST\_FUNC\_EMPTY

2782 #define DOCTEST\_REQUIRE\_THROWS(...) DOCTEST\_FUNC\_EMPTY

2783 #define DOCTEST\_WARN\_THROWS\_AS(expr, ...) DOCTEST\_FUNC\_EMPTY

2784 #define DOCTEST\_CHECK\_THROWS\_AS(expr, ...) DOCTEST\_FUNC\_EMPTY

2785 #define DOCTEST\_REQUIRE\_THROWS\_AS(expr, ...) DOCTEST\_FUNC\_EMPTY

2786 #define DOCTEST\_WARN\_THROWS\_WITH(expr, ...) DOCTEST\_FUNC\_EMPTY

2787 #define DOCTEST\_CHECK\_THROWS\_WITH(expr, ...) DOCTEST\_FUNC\_EMPTY

2788 #define DOCTEST\_REQUIRE\_THROWS\_WITH(expr, ...) DOCTEST\_FUNC\_EMPTY

2789 #define DOCTEST\_WARN\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_FUNC\_EMPTY

2790 #define DOCTEST\_CHECK\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_FUNC\_EMPTY

2791 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_FUNC\_EMPTY

2792 #define DOCTEST\_WARN\_NOTHROW(...) DOCTEST\_FUNC\_EMPTY

2793 #define DOCTEST\_CHECK\_NOTHROW(...) DOCTEST\_FUNC\_EMPTY

2794 #define DOCTEST\_REQUIRE\_NOTHROW(...) DOCTEST\_FUNC\_EMPTY

2795

2796 #define DOCTEST\_WARN\_THROWS\_MESSAGE(expr, ...) DOCTEST\_FUNC\_EMPTY

2797 #define DOCTEST\_CHECK\_THROWS\_MESSAGE(expr, ...) DOCTEST\_FUNC\_EMPTY

2798 #define DOCTEST\_REQUIRE\_THROWS\_MESSAGE(expr, ...) DOCTEST\_FUNC\_EMPTY

2799 #define DOCTEST\_WARN\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_FUNC\_EMPTY

2800 #define DOCTEST\_CHECK\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_FUNC\_EMPTY

2801 #define DOCTEST\_REQUIRE\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_FUNC\_EMPTY

2802 #define DOCTEST\_WARN\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_FUNC\_EMPTY

2803 #define DOCTEST\_CHECK\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_FUNC\_EMPTY

2804 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_FUNC\_EMPTY

2805 #define DOCTEST\_WARN\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_FUNC\_EMPTY

2806 #define DOCTEST\_CHECK\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_FUNC\_EMPTY

2807 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_FUNC\_EMPTY

2808 #define DOCTEST\_WARN\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_FUNC\_EMPTY

2809 #define DOCTEST\_CHECK\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_FUNC\_EMPTY

2810 #define DOCTEST\_REQUIRE\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_FUNC\_EMPTY

2811

2812 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

2813

2814 #endif // DOCTEST\_CONFIG\_EVALUATE\_ASSERTS\_EVEN\_WHEN\_DISABLED

2815

2816 #endif // DOCTEST\_CONFIG\_DISABLE

2817

2818 #ifdef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

2819

2820 #ifdef DOCTEST\_CONFIG\_NO\_EXCEPTIONS\_BUT\_WITH\_ALL\_ASSERTS

2821 #define DOCTEST\_EXCEPTION\_EMPTY\_FUNC DOCTEST\_FUNC\_EMPTY

2822 #else // DOCTEST\_CONFIG\_NO\_EXCEPTIONS\_BUT\_WITH\_ALL\_ASSERTS

2823 #define DOCTEST\_EXCEPTION\_EMPTY\_FUNC [] { static\_assert(false, "Exceptions are disabled! " \

2824 "Use DOCTEST\_CONFIG\_NO\_EXCEPTIONS\_BUT\_WITH\_ALL\_ASSERTS if you want to compile with exceptions disabled."); return false; }()

2825

2826 #undef DOCTEST\_REQUIRE

2827 #undef DOCTEST\_REQUIRE\_FALSE

2828 #undef DOCTEST\_REQUIRE\_MESSAGE

2829 #undef DOCTEST\_REQUIRE\_FALSE\_MESSAGE

2830 #undef DOCTEST\_REQUIRE\_EQ

2831 #undef DOCTEST\_REQUIRE\_NE

2832 #undef DOCTEST\_REQUIRE\_GT

2833 #undef DOCTEST\_REQUIRE\_LT

2834 #undef DOCTEST\_REQUIRE\_GE

2835 #undef DOCTEST\_REQUIRE\_LE

2836 #undef DOCTEST\_REQUIRE\_UNARY

2837 #undef DOCTEST\_REQUIRE\_UNARY\_FALSE

2838

2839 #define DOCTEST\_REQUIRE DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2840 #define DOCTEST\_REQUIRE\_FALSE DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2841 #define DOCTEST\_REQUIRE\_MESSAGE DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2842 #define DOCTEST\_REQUIRE\_FALSE\_MESSAGE DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2843 #define DOCTEST\_REQUIRE\_EQ DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2844 #define DOCTEST\_REQUIRE\_NE DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2845 #define DOCTEST\_REQUIRE\_GT DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2846 #define DOCTEST\_REQUIRE\_LT DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2847 #define DOCTEST\_REQUIRE\_GE DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2848 #define DOCTEST\_REQUIRE\_LE DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2849 #define DOCTEST\_REQUIRE\_UNARY DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2850 #define DOCTEST\_REQUIRE\_UNARY\_FALSE DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2851

2852 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS\_BUT\_WITH\_ALL\_ASSERTS

2853

2854 #define DOCTEST\_WARN\_THROWS(...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2855 #define DOCTEST\_CHECK\_THROWS(...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2856 #define DOCTEST\_REQUIRE\_THROWS(...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2857 #define DOCTEST\_WARN\_THROWS\_AS(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2858 #define DOCTEST\_CHECK\_THROWS\_AS(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2859 #define DOCTEST\_REQUIRE\_THROWS\_AS(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2860 #define DOCTEST\_WARN\_THROWS\_WITH(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2861 #define DOCTEST\_CHECK\_THROWS\_WITH(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2862 #define DOCTEST\_REQUIRE\_THROWS\_WITH(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2863 #define DOCTEST\_WARN\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2864 #define DOCTEST\_CHECK\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2865 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2866 #define DOCTEST\_WARN\_NOTHROW(...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2867 #define DOCTEST\_CHECK\_NOTHROW(...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2868 #define DOCTEST\_REQUIRE\_NOTHROW(...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2869

2870 #define DOCTEST\_WARN\_THROWS\_MESSAGE(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2871 #define DOCTEST\_CHECK\_THROWS\_MESSAGE(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2872 #define DOCTEST\_REQUIRE\_THROWS\_MESSAGE(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2873 #define DOCTEST\_WARN\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2874 #define DOCTEST\_CHECK\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2875 #define DOCTEST\_REQUIRE\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2876 #define DOCTEST\_WARN\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2877 #define DOCTEST\_CHECK\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2878 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2879 #define DOCTEST\_WARN\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2880 #define DOCTEST\_CHECK\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2881 #define DOCTEST\_REQUIRE\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2882 #define DOCTEST\_WARN\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2883 #define DOCTEST\_CHECK\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2884 #define DOCTEST\_REQUIRE\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_EXCEPTION\_EMPTY\_FUNC

2885

2886 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

2887

2888 // clang-format off

2889 // KEPT FOR BACKWARDS COMPATIBILITY - FORWARDING TO THE RIGHT MACROS

2890 #define DOCTEST\_FAST\_WARN\_EQ DOCTEST\_WARN\_EQ

2891 #define DOCTEST\_FAST\_CHECK\_EQ DOCTEST\_CHECK\_EQ

2892 #define DOCTEST\_FAST\_REQUIRE\_EQ DOCTEST\_REQUIRE\_EQ

2893 #define DOCTEST\_FAST\_WARN\_NE DOCTEST\_WARN\_NE

2894 #define DOCTEST\_FAST\_CHECK\_NE DOCTEST\_CHECK\_NE

2895 #define DOCTEST\_FAST\_REQUIRE\_NE DOCTEST\_REQUIRE\_NE

2896 #define DOCTEST\_FAST\_WARN\_GT DOCTEST\_WARN\_GT

2897 #define DOCTEST\_FAST\_CHECK\_GT DOCTEST\_CHECK\_GT

2898 #define DOCTEST\_FAST\_REQUIRE\_GT DOCTEST\_REQUIRE\_GT

2899 #define DOCTEST\_FAST\_WARN\_LT DOCTEST\_WARN\_LT

2900 #define DOCTEST\_FAST\_CHECK\_LT DOCTEST\_CHECK\_LT

2901 #define DOCTEST\_FAST\_REQUIRE\_LT DOCTEST\_REQUIRE\_LT

2902 #define DOCTEST\_FAST\_WARN\_GE DOCTEST\_WARN\_GE

2903 #define DOCTEST\_FAST\_CHECK\_GE DOCTEST\_CHECK\_GE

2904 #define DOCTEST\_FAST\_REQUIRE\_GE DOCTEST\_REQUIRE\_GE

2905 #define DOCTEST\_FAST\_WARN\_LE DOCTEST\_WARN\_LE

2906 #define DOCTEST\_FAST\_CHECK\_LE DOCTEST\_CHECK\_LE

2907 #define DOCTEST\_FAST\_REQUIRE\_LE DOCTEST\_REQUIRE\_LE

2908

2909 #define DOCTEST\_FAST\_WARN\_UNARY DOCTEST\_WARN\_UNARY

2910 #define DOCTEST\_FAST\_CHECK\_UNARY DOCTEST\_CHECK\_UNARY

2911 #define DOCTEST\_FAST\_REQUIRE\_UNARY DOCTEST\_REQUIRE\_UNARY

2912 #define DOCTEST\_FAST\_WARN\_UNARY\_FALSE DOCTEST\_WARN\_UNARY\_FALSE

2913 #define DOCTEST\_FAST\_CHECK\_UNARY\_FALSE DOCTEST\_CHECK\_UNARY\_FALSE

2914 #define DOCTEST\_FAST\_REQUIRE\_UNARY\_FALSE DOCTEST\_REQUIRE\_UNARY\_FALSE

2915

2916 #define DOCTEST\_TEST\_CASE\_TEMPLATE\_INSTANTIATE(id, ...) DOCTEST\_TEST\_CASE\_TEMPLATE\_INVOKE(id,\_\_VA\_ARGS\_\_)

2917 // clang-format on

2918

2919 // BDD style macros

2920 // clang-format off

2921 #define DOCTEST\_SCENARIO(name) DOCTEST\_TEST\_CASE(" Scenario: " name)

2922 #define DOCTEST\_SCENARIO\_CLASS(name) DOCTEST\_TEST\_CASE\_CLASS(" Scenario: " name)

2923 #define DOCTEST\_SCENARIO\_TEMPLATE(name, T, ...) DOCTEST\_TEST\_CASE\_TEMPLATE(" Scenario: " name, T, \_\_VA\_ARGS\_\_)

2924 #define DOCTEST\_SCENARIO\_TEMPLATE\_DEFINE(name, T, id) DOCTEST\_TEST\_CASE\_TEMPLATE\_DEFINE(" Scenario: " name, T, id)

2925

2926 #define DOCTEST\_GIVEN(name) DOCTEST\_SUBCASE(" Given: " name)

2927 #define DOCTEST\_WHEN(name) DOCTEST\_SUBCASE(" When: " name)

2928 #define DOCTEST\_AND\_WHEN(name) DOCTEST\_SUBCASE("And when: " name)

2929 #define DOCTEST\_THEN(name) DOCTEST\_SUBCASE(" Then: " name)

2930 #define DOCTEST\_AND\_THEN(name) DOCTEST\_SUBCASE(" And: " name)

2931 // clang-format on

2932

2933 // == SHORT VERSIONS OF THE MACROS

2934 #ifndef DOCTEST\_CONFIG\_NO\_SHORT\_MACRO\_NAMES

2935

2936 #define TEST\_CASE(name) DOCTEST\_TEST\_CASE(name)

2937 #define TEST\_CASE\_CLASS(name) DOCTEST\_TEST\_CASE\_CLASS(name)

2938 #define TEST\_CASE\_FIXTURE(x, name) DOCTEST\_TEST\_CASE\_FIXTURE(x, name)

2939 #define TYPE\_TO\_STRING\_AS(str, ...) DOCTEST\_TYPE\_TO\_STRING\_AS(str, \_\_VA\_ARGS\_\_)

2940 #define TYPE\_TO\_STRING(...) DOCTEST\_TYPE\_TO\_STRING(\_\_VA\_ARGS\_\_)

2941 #define TEST\_CASE\_TEMPLATE(name, T, ...) DOCTEST\_TEST\_CASE\_TEMPLATE(name, T, \_\_VA\_ARGS\_\_)

2942 #define TEST\_CASE\_TEMPLATE\_DEFINE(name, T, id) DOCTEST\_TEST\_CASE\_TEMPLATE\_DEFINE(name, T, id)

2943 #define TEST\_CASE\_TEMPLATE\_INVOKE(id, ...) DOCTEST\_TEST\_CASE\_TEMPLATE\_INVOKE(id, \_\_VA\_ARGS\_\_)

2944 #define TEST\_CASE\_TEMPLATE\_APPLY(id, ...) DOCTEST\_TEST\_CASE\_TEMPLATE\_APPLY(id, \_\_VA\_ARGS\_\_)

2945 #define SUBCASE(name) DOCTEST\_SUBCASE(name)

2946 #define TEST\_SUITE(decorators) DOCTEST\_TEST\_SUITE(decorators)

2947 #define TEST\_SUITE\_BEGIN(name) DOCTEST\_TEST\_SUITE\_BEGIN(name)

2948 #define TEST\_SUITE\_END DOCTEST\_TEST\_SUITE\_END

2949 #define REGISTER\_EXCEPTION\_TRANSLATOR(signature) DOCTEST\_REGISTER\_EXCEPTION\_TRANSLATOR(signature)

2950 #define REGISTER\_REPORTER(name, priority, reporter) DOCTEST\_REGISTER\_REPORTER(name, priority, reporter)

2951 #define REGISTER\_LISTENER(name, priority, reporter) DOCTEST\_REGISTER\_LISTENER(name, priority, reporter)

2952 #define INFO(...) DOCTEST\_INFO(\_\_VA\_ARGS\_\_)

2953 #define CAPTURE(x) DOCTEST\_CAPTURE(x)

2954 #define ADD\_MESSAGE\_AT(file, line, ...) DOCTEST\_ADD\_MESSAGE\_AT(file, line, \_\_VA\_ARGS\_\_)

2955 #define ADD\_FAIL\_CHECK\_AT(file, line, ...) DOCTEST\_ADD\_FAIL\_CHECK\_AT(file, line, \_\_VA\_ARGS\_\_)

2956 #define ADD\_FAIL\_AT(file, line, ...) DOCTEST\_ADD\_FAIL\_AT(file, line, \_\_VA\_ARGS\_\_)

2957 #define MESSAGE(...) DOCTEST\_MESSAGE(\_\_VA\_ARGS\_\_)

2958 #define FAIL\_CHECK(...) DOCTEST\_FAIL\_CHECK(\_\_VA\_ARGS\_\_)

2959 #define FAIL(...) DOCTEST\_FAIL(\_\_VA\_ARGS\_\_)

2960 #define TO\_LVALUE(...) DOCTEST\_TO\_LVALUE(\_\_VA\_ARGS\_\_)

2961

2962 #define WARN(...) DOCTEST\_WARN(\_\_VA\_ARGS\_\_)

2963 #define WARN\_FALSE(...) DOCTEST\_WARN\_FALSE(\_\_VA\_ARGS\_\_)

2964 #define WARN\_THROWS(...) DOCTEST\_WARN\_THROWS(\_\_VA\_ARGS\_\_)

2965 #define WARN\_THROWS\_AS(expr, ...) DOCTEST\_WARN\_THROWS\_AS(expr, \_\_VA\_ARGS\_\_)

2966 #define WARN\_THROWS\_WITH(expr, ...) DOCTEST\_WARN\_THROWS\_WITH(expr, \_\_VA\_ARGS\_\_)

2967 #define WARN\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_WARN\_THROWS\_WITH\_AS(expr, with, \_\_VA\_ARGS\_\_)

2968 #define WARN\_NOTHROW(...) DOCTEST\_WARN\_NOTHROW(\_\_VA\_ARGS\_\_)

2969 #define CHECK(...) DOCTEST\_CHECK(\_\_VA\_ARGS\_\_)

2970 #define CHECK\_FALSE(...) DOCTEST\_CHECK\_FALSE(\_\_VA\_ARGS\_\_)

2971 #define CHECK\_THROWS(...) DOCTEST\_CHECK\_THROWS(\_\_VA\_ARGS\_\_)

2972 #define CHECK\_THROWS\_AS(expr, ...) DOCTEST\_CHECK\_THROWS\_AS(expr, \_\_VA\_ARGS\_\_)

2973 #define CHECK\_THROWS\_WITH(expr, ...) DOCTEST\_CHECK\_THROWS\_WITH(expr, \_\_VA\_ARGS\_\_)

2974 #define CHECK\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_CHECK\_THROWS\_WITH\_AS(expr, with, \_\_VA\_ARGS\_\_)

2975 #define CHECK\_NOTHROW(...) DOCTEST\_CHECK\_NOTHROW(\_\_VA\_ARGS\_\_)

2976 #define REQUIRE(...) DOCTEST\_REQUIRE(\_\_VA\_ARGS\_\_)

2977 #define REQUIRE\_FALSE(...) DOCTEST\_REQUIRE\_FALSE(\_\_VA\_ARGS\_\_)

2978 #define REQUIRE\_THROWS(...) DOCTEST\_REQUIRE\_THROWS(\_\_VA\_ARGS\_\_)

2979 #define REQUIRE\_THROWS\_AS(expr, ...) DOCTEST\_REQUIRE\_THROWS\_AS(expr, \_\_VA\_ARGS\_\_)

2980 #define REQUIRE\_THROWS\_WITH(expr, ...) DOCTEST\_REQUIRE\_THROWS\_WITH(expr, \_\_VA\_ARGS\_\_)

2981 #define REQUIRE\_THROWS\_WITH\_AS(expr, with, ...) DOCTEST\_REQUIRE\_THROWS\_WITH\_AS(expr, with, \_\_VA\_ARGS\_\_)

2982 #define REQUIRE\_NOTHROW(...) DOCTEST\_REQUIRE\_NOTHROW(\_\_VA\_ARGS\_\_)

2983

2984 #define WARN\_MESSAGE(cond, ...) DOCTEST\_WARN\_MESSAGE(cond, \_\_VA\_ARGS\_\_)

2985 #define WARN\_FALSE\_MESSAGE(cond, ...) DOCTEST\_WARN\_FALSE\_MESSAGE(cond, \_\_VA\_ARGS\_\_)

2986 #define WARN\_THROWS\_MESSAGE(expr, ...) DOCTEST\_WARN\_THROWS\_MESSAGE(expr, \_\_VA\_ARGS\_\_)

2987 #define WARN\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_WARN\_THROWS\_AS\_MESSAGE(expr, ex, \_\_VA\_ARGS\_\_)

2988 #define WARN\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_WARN\_THROWS\_WITH\_MESSAGE(expr, with, \_\_VA\_ARGS\_\_)

2989 #define WARN\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_WARN\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, \_\_VA\_ARGS\_\_)

2990 #define WARN\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_WARN\_NOTHROW\_MESSAGE(expr, \_\_VA\_ARGS\_\_)

2991 #define CHECK\_MESSAGE(cond, ...) DOCTEST\_CHECK\_MESSAGE(cond, \_\_VA\_ARGS\_\_)

2992 #define CHECK\_FALSE\_MESSAGE(cond, ...) DOCTEST\_CHECK\_FALSE\_MESSAGE(cond, \_\_VA\_ARGS\_\_)

2993 #define CHECK\_THROWS\_MESSAGE(expr, ...) DOCTEST\_CHECK\_THROWS\_MESSAGE(expr, \_\_VA\_ARGS\_\_)

2994 #define CHECK\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_CHECK\_THROWS\_AS\_MESSAGE(expr, ex, \_\_VA\_ARGS\_\_)

2995 #define CHECK\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_CHECK\_THROWS\_WITH\_MESSAGE(expr, with, \_\_VA\_ARGS\_\_)

2996 #define CHECK\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_CHECK\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, \_\_VA\_ARGS\_\_)

2997 #define CHECK\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_CHECK\_NOTHROW\_MESSAGE(expr, \_\_VA\_ARGS\_\_)

2998 #define REQUIRE\_MESSAGE(cond, ...) DOCTEST\_REQUIRE\_MESSAGE(cond, \_\_VA\_ARGS\_\_)

2999 #define REQUIRE\_FALSE\_MESSAGE(cond, ...) DOCTEST\_REQUIRE\_FALSE\_MESSAGE(cond, \_\_VA\_ARGS\_\_)

3000 #define REQUIRE\_THROWS\_MESSAGE(expr, ...) DOCTEST\_REQUIRE\_THROWS\_MESSAGE(expr, \_\_VA\_ARGS\_\_)

3001 #define REQUIRE\_THROWS\_AS\_MESSAGE(expr, ex, ...) DOCTEST\_REQUIRE\_THROWS\_AS\_MESSAGE(expr, ex, \_\_VA\_ARGS\_\_)

3002 #define REQUIRE\_THROWS\_WITH\_MESSAGE(expr, with, ...) DOCTEST\_REQUIRE\_THROWS\_WITH\_MESSAGE(expr, with, \_\_VA\_ARGS\_\_)

3003 #define REQUIRE\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, ...) DOCTEST\_REQUIRE\_THROWS\_WITH\_AS\_MESSAGE(expr, with, ex, \_\_VA\_ARGS\_\_)

3004 #define REQUIRE\_NOTHROW\_MESSAGE(expr, ...) DOCTEST\_REQUIRE\_NOTHROW\_MESSAGE(expr, \_\_VA\_ARGS\_\_)

3005

3006 #define SCENARIO(name) DOCTEST\_SCENARIO(name)

3007 #define SCENARIO\_CLASS(name) DOCTEST\_SCENARIO\_CLASS(name)

3008 #define SCENARIO\_TEMPLATE(name, T, ...) DOCTEST\_SCENARIO\_TEMPLATE(name, T, \_\_VA\_ARGS\_\_)

3009 #define SCENARIO\_TEMPLATE\_DEFINE(name, T, id) DOCTEST\_SCENARIO\_TEMPLATE\_DEFINE(name, T, id)

3010 #define GIVEN(name) DOCTEST\_GIVEN(name)

3011 #define WHEN(name) DOCTEST\_WHEN(name)

3012 #define AND\_WHEN(name) DOCTEST\_AND\_WHEN(name)

3013 #define THEN(name) DOCTEST\_THEN(name)

3014 #define AND\_THEN(name) DOCTEST\_AND\_THEN(name)

3015

3016 #define WARN\_EQ(...) DOCTEST\_WARN\_EQ(\_\_VA\_ARGS\_\_)

3017 #define CHECK\_EQ(...) DOCTEST\_CHECK\_EQ(\_\_VA\_ARGS\_\_)

3018 #define REQUIRE\_EQ(...) DOCTEST\_REQUIRE\_EQ(\_\_VA\_ARGS\_\_)

3019 #define WARN\_NE(...) DOCTEST\_WARN\_NE(\_\_VA\_ARGS\_\_)

3020 #define CHECK\_NE(...) DOCTEST\_CHECK\_NE(\_\_VA\_ARGS\_\_)

3021 #define REQUIRE\_NE(...) DOCTEST\_REQUIRE\_NE(\_\_VA\_ARGS\_\_)

3022 #define WARN\_GT(...) DOCTEST\_WARN\_GT(\_\_VA\_ARGS\_\_)

3023 #define CHECK\_GT(...) DOCTEST\_CHECK\_GT(\_\_VA\_ARGS\_\_)

3024 #define REQUIRE\_GT(...) DOCTEST\_REQUIRE\_GT(\_\_VA\_ARGS\_\_)

3025 #define WARN\_LT(...) DOCTEST\_WARN\_LT(\_\_VA\_ARGS\_\_)

3026 #define CHECK\_LT(...) DOCTEST\_CHECK\_LT(\_\_VA\_ARGS\_\_)

3027 #define REQUIRE\_LT(...) DOCTEST\_REQUIRE\_LT(\_\_VA\_ARGS\_\_)

3028 #define WARN\_GE(...) DOCTEST\_WARN\_GE(\_\_VA\_ARGS\_\_)

3029 #define CHECK\_GE(...) DOCTEST\_CHECK\_GE(\_\_VA\_ARGS\_\_)

3030 #define REQUIRE\_GE(...) DOCTEST\_REQUIRE\_GE(\_\_VA\_ARGS\_\_)

3031 #define WARN\_LE(...) DOCTEST\_WARN\_LE(\_\_VA\_ARGS\_\_)

3032 #define CHECK\_LE(...) DOCTEST\_CHECK\_LE(\_\_VA\_ARGS\_\_)

3033 #define REQUIRE\_LE(...) DOCTEST\_REQUIRE\_LE(\_\_VA\_ARGS\_\_)

3034 #define WARN\_UNARY(...) DOCTEST\_WARN\_UNARY(\_\_VA\_ARGS\_\_)

3035 #define CHECK\_UNARY(...) DOCTEST\_CHECK\_UNARY(\_\_VA\_ARGS\_\_)

3036 #define REQUIRE\_UNARY(...) DOCTEST\_REQUIRE\_UNARY(\_\_VA\_ARGS\_\_)

3037 #define WARN\_UNARY\_FALSE(...) DOCTEST\_WARN\_UNARY\_FALSE(\_\_VA\_ARGS\_\_)

3038 #define CHECK\_UNARY\_FALSE(...) DOCTEST\_CHECK\_UNARY\_FALSE(\_\_VA\_ARGS\_\_)

3039 #define REQUIRE\_UNARY\_FALSE(...) DOCTEST\_REQUIRE\_UNARY\_FALSE(\_\_VA\_ARGS\_\_)

3040

3041 // KEPT FOR BACKWARDS COMPATIBILITY

3042 #define FAST\_WARN\_EQ(...) DOCTEST\_FAST\_WARN\_EQ(\_\_VA\_ARGS\_\_)

3043 #define FAST\_CHECK\_EQ(...) DOCTEST\_FAST\_CHECK\_EQ(\_\_VA\_ARGS\_\_)

3044 #define FAST\_REQUIRE\_EQ(...) DOCTEST\_FAST\_REQUIRE\_EQ(\_\_VA\_ARGS\_\_)

3045 #define FAST\_WARN\_NE(...) DOCTEST\_FAST\_WARN\_NE(\_\_VA\_ARGS\_\_)

3046 #define FAST\_CHECK\_NE(...) DOCTEST\_FAST\_CHECK\_NE(\_\_VA\_ARGS\_\_)

3047 #define FAST\_REQUIRE\_NE(...) DOCTEST\_FAST\_REQUIRE\_NE(\_\_VA\_ARGS\_\_)

3048 #define FAST\_WARN\_GT(...) DOCTEST\_FAST\_WARN\_GT(\_\_VA\_ARGS\_\_)

3049 #define FAST\_CHECK\_GT(...) DOCTEST\_FAST\_CHECK\_GT(\_\_VA\_ARGS\_\_)

3050 #define FAST\_REQUIRE\_GT(...) DOCTEST\_FAST\_REQUIRE\_GT(\_\_VA\_ARGS\_\_)

3051 #define FAST\_WARN\_LT(...) DOCTEST\_FAST\_WARN\_LT(\_\_VA\_ARGS\_\_)

3052 #define FAST\_CHECK\_LT(...) DOCTEST\_FAST\_CHECK\_LT(\_\_VA\_ARGS\_\_)

3053 #define FAST\_REQUIRE\_LT(...) DOCTEST\_FAST\_REQUIRE\_LT(\_\_VA\_ARGS\_\_)

3054 #define FAST\_WARN\_GE(...) DOCTEST\_FAST\_WARN\_GE(\_\_VA\_ARGS\_\_)

3055 #define FAST\_CHECK\_GE(...) DOCTEST\_FAST\_CHECK\_GE(\_\_VA\_ARGS\_\_)

3056 #define FAST\_REQUIRE\_GE(...) DOCTEST\_FAST\_REQUIRE\_GE(\_\_VA\_ARGS\_\_)

3057 #define FAST\_WARN\_LE(...) DOCTEST\_FAST\_WARN\_LE(\_\_VA\_ARGS\_\_)

3058 #define FAST\_CHECK\_LE(...) DOCTEST\_FAST\_CHECK\_LE(\_\_VA\_ARGS\_\_)

3059 #define FAST\_REQUIRE\_LE(...) DOCTEST\_FAST\_REQUIRE\_LE(\_\_VA\_ARGS\_\_)

3060

3061 #define FAST\_WARN\_UNARY(...) DOCTEST\_FAST\_WARN\_UNARY(\_\_VA\_ARGS\_\_)

3062 #define FAST\_CHECK\_UNARY(...) DOCTEST\_FAST\_CHECK\_UNARY(\_\_VA\_ARGS\_\_)

3063 #define FAST\_REQUIRE\_UNARY(...) DOCTEST\_FAST\_REQUIRE\_UNARY(\_\_VA\_ARGS\_\_)

3064 #define FAST\_WARN\_UNARY\_FALSE(...) DOCTEST\_FAST\_WARN\_UNARY\_FALSE(\_\_VA\_ARGS\_\_)

3065 #define FAST\_CHECK\_UNARY\_FALSE(...) DOCTEST\_FAST\_CHECK\_UNARY\_FALSE(\_\_VA\_ARGS\_\_)

3066 #define FAST\_REQUIRE\_UNARY\_FALSE(...) DOCTEST\_FAST\_REQUIRE\_UNARY\_FALSE(\_\_VA\_ARGS\_\_)

3067

3068 #define TEST\_CASE\_TEMPLATE\_INSTANTIATE(id, ...) DOCTEST\_TEST\_CASE\_TEMPLATE\_INSTANTIATE(id, \_\_VA\_ARGS\_\_)

3069

3070 #endif // DOCTEST\_CONFIG\_NO\_SHORT\_MACRO\_NAMES

3071

3072 #ifndef DOCTEST\_CONFIG\_DISABLE

3073

3074 // this is here to clear the 'current test suite' for the current translation unit - at the top

3075 DOCTEST\_TEST\_SUITE\_END();

3076

3077 #endif // DOCTEST\_CONFIG\_DISABLE

3078

3079 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

3080 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

3081 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

3082

3083 DOCTEST\_SUPPRESS\_COMMON\_WARNINGS\_POP

3084

3085 #endif // DOCTEST\_LIBRARY\_INCLUDED

3086

3087 #ifndef DOCTEST\_SINGLE\_HEADER

3088 #define DOCTEST\_SINGLE\_HEADER

3089 #endif // DOCTEST\_SINGLE\_HEADER

3090

3091 #if defined(DOCTEST\_CONFIG\_IMPLEMENT) || !defined(DOCTEST\_SINGLE\_HEADER)

3092

3093 #ifndef DOCTEST\_SINGLE\_HEADER

3094 #include "doctest\_fwd.h"

3095 #endif // DOCTEST\_SINGLE\_HEADER

3096

3097 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wunused-macros")

3098

3099 #ifndef DOCTEST\_LIBRARY\_IMPLEMENTATION

3100 #define DOCTEST\_LIBRARY\_IMPLEMENTATION

3101

3102 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

3103

3104 DOCTEST\_SUPPRESS\_COMMON\_WARNINGS\_PUSH

3105

3106 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_PUSH

3107 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wglobal-constructors")

3108 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wexit-time-destructors")

3109 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wsign-conversion")

3110 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wshorten-64-to-32")

3111 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wmissing-variable-declarations")

3112 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wswitch")

3113 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wswitch-enum")

3114 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wcovered-switch-default")

3115 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wmissing-noreturn")

3116 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wdisabled-macro-expansion")

3117 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wmissing-braces")

3118 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wmissing-field-initializers")

3119 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wunused-member-function")

3120 DOCTEST\_CLANG\_SUPPRESS\_WARNING("-Wnonportable-system-include-path")

3121

3122 DOCTEST\_GCC\_SUPPRESS\_WARNING\_PUSH

3123 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wconversion")

3124 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wsign-conversion")

3125 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wmissing-field-initializers")

3126 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wmissing-braces")

3127 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wswitch")

3128 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wswitch-enum")

3129 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wswitch-default")

3130 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wunsafe-loop-optimizations")

3131 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wold-style-cast")

3132 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wunused-function")

3133 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wmultiple-inheritance")

3134 DOCTEST\_GCC\_SUPPRESS\_WARNING("-Wsuggest-attribute")

3135

3136 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_PUSH

3137 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4267) // 'var' : conversion from 'x' to 'y', possible loss of data

3138 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4530) // C++ exception handler used, but unwind semantics not enabled

3139 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4577) // 'noexcept' used with no exception handling mode specified

3140 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4774) // format string expected in argument is not a string literal

3141 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4365) // conversion from 'int' to 'unsigned', signed/unsigned mismatch

3142 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5039) // pointer to potentially throwing function passed to extern C

3143 DOCTEST\_MSVC\_SUPPRESS\_WARNING(4800) // forcing value to bool 'true' or 'false' (performance warning)

3144 DOCTEST\_MSVC\_SUPPRESS\_WARNING(5245) // unreferenced function with internal linkage has been removed

3145

3146 DOCTEST\_MAKE\_STD\_HEADERS\_CLEAN\_FROM\_WARNINGS\_ON\_WALL\_BEGIN

3147

3148 // required includes - will go only in one translation unit!

3149 #include <ctime>

3150 #include <cmath>

3151 #include <climits>

3152 // borland (Embarcadero) compiler requires math.h and not cmath - https://github.com/doctest/doctest/pull/37

3153 #ifdef \_\_BORLANDC\_\_

3154 #include <math.h>

3155 #endif // \_\_BORLANDC\_\_

3156 #include <new>

3157 #include <cstdio>

3158 #include <cstdlib>

3159 #include <cstring>

3160 #include <limits>

3161 #include <utility>

3162 #include <fstream>

3163 #include <sstream>

3164 #ifndef DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

3165 #include <iostream>

3166 #endif // DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

3167 #include <algorithm>

3168 #include <iomanip>

3169 #include <vector>

3170 #ifndef DOCTEST\_CONFIG\_NO\_MULTITHREADING

3171 #include <atomic>

3172 #include <mutex>

3173 #define DOCTEST\_DECLARE\_MUTEX(name) std::mutex name;

3174 #define DOCTEST\_DECLARE\_STATIC\_MUTEX(name) static DOCTEST\_DECLARE\_MUTEX(name)

3175 #define DOCTEST\_LOCK\_MUTEX(name) std::lock\_guard<std::mutex> DOCTEST\_ANONYMOUS(DOCTEST\_ANON\_LOCK\_)(name);

3176 #else // DOCTEST\_CONFIG\_NO\_MULTITHREADING

3177 #define DOCTEST\_DECLARE\_MUTEX(name)

3178 #define DOCTEST\_DECLARE\_STATIC\_MUTEX(name)

3179 #define DOCTEST\_LOCK\_MUTEX(name)

3180 #endif // DOCTEST\_CONFIG\_NO\_MULTITHREADING

3181 #include <set>

3182 #include <map>

3183 #include <unordered\_set>

3184 #include <exception>

3185 #include <stdexcept>

3186 #include <csignal>

3187 #include <cfloat>

3188 #include <cctype>

3189 #include <cstdint>

3190 #include <string>

3191

3192 #ifdef DOCTEST\_PLATFORM\_MAC

3193 #include <sys/types.h>

3194 #include <unistd.h>

3195 #include <sys/sysctl.h>

3196 #endif // DOCTEST\_PLATFORM\_MAC

3197

3198 #ifdef DOCTEST\_PLATFORM\_WINDOWS

3199

3200 // defines for a leaner windows.h

3201 #ifndef WIN32\_LEAN\_AND\_MEAN

3202 #define WIN32\_LEAN\_AND\_MEAN

3203 #define DOCTEST\_UNDEF\_WIN32\_LEAN\_AND\_MEAN

3204 #endif // WIN32\_LEAN\_AND\_MEAN

3205 #ifndef NOMINMAX

3206 #define NOMINMAX

3207 #define DOCTEST\_UNDEF\_NOMINMAX

3208 #endif // NOMINMAX

3209

3210 // not sure what AfxWin.h is for - here I do what Catch does

3211 #ifdef \_\_AFXDLL

3212 #include <AfxWin.h>

3213 #else

3214 #include <windows.h>

3215 #endif

3216 #include <io.h>

3217

3218 #else // DOCTEST\_PLATFORM\_WINDOWS

3219

3220 #include <sys/time.h>

3221 #include <unistd.h>

3222

3223 #endif // DOCTEST\_PLATFORM\_WINDOWS

3224

3225 // this is a fix for https://github.com/doctest/doctest/issues/348

3226 // https://mail.gnome.org/archives/xml/2012-January/msg00000.html

3227 #if !defined(HAVE\_UNISTD\_H) && !defined(STDOUT\_FILENO)

3228 #define STDOUT\_FILENO fileno(stdout)

3229 #endif // HAVE\_UNISTD\_H

3230

3231 DOCTEST\_MAKE\_STD\_HEADERS\_CLEAN\_FROM\_WARNINGS\_ON\_WALL\_END

3232

3233 // counts the number of elements in a C array

3234 #define DOCTEST\_COUNTOF(x) (sizeof(x) / sizeof(x[0]))

3235

3236 #ifdef DOCTEST\_CONFIG\_DISABLE

3237 #define DOCTEST\_BRANCH\_ON\_DISABLED(if\_disabled, if\_not\_disabled) if\_disabled

3238 #else // DOCTEST\_CONFIG\_DISABLE

3239 #define DOCTEST\_BRANCH\_ON\_DISABLED(if\_disabled, if\_not\_disabled) if\_not\_disabled

3240 #endif // DOCTEST\_CONFIG\_DISABLE

3241

3242 #ifndef DOCTEST\_CONFIG\_OPTIONS\_PREFIX

3243 #define DOCTEST\_CONFIG\_OPTIONS\_PREFIX "dt-"

3244 #endif

3245

3246 #ifndef DOCTEST\_THREAD\_LOCAL

3247 #if defined(DOCTEST\_CONFIG\_NO\_MULTITHREADING) || DOCTEST\_MSVC && (DOCTEST\_MSVC < DOCTEST\_COMPILER(19, 0, 0))

3248 #define DOCTEST\_THREAD\_LOCAL

3249 #else // DOCTEST\_MSVC

3250 #define DOCTEST\_THREAD\_LOCAL thread\_local

3251 #endif // DOCTEST\_MSVC

3252 #endif // DOCTEST\_THREAD\_LOCAL

3253

3254 #ifndef DOCTEST\_MULTI\_LANE\_ATOMICS\_THREAD\_LANES

3255 #define DOCTEST\_MULTI\_LANE\_ATOMICS\_THREAD\_LANES 32

3256 #endif

3257

3258 #ifndef DOCTEST\_MULTI\_LANE\_ATOMICS\_CACHE\_LINE\_SIZE

3259 #define DOCTEST\_MULTI\_LANE\_ATOMICS\_CACHE\_LINE\_SIZE 64

3260 #endif

3261

3262 #ifdef DOCTEST\_CONFIG\_NO\_UNPREFIXED\_OPTIONS

3263 #define DOCTEST\_OPTIONS\_PREFIX\_DISPLAY DOCTEST\_CONFIG\_OPTIONS\_PREFIX

3264 #else

3265 #define DOCTEST\_OPTIONS\_PREFIX\_DISPLAY ""

3266 #endif

3267

3268 #if defined(WINAPI\_FAMILY) && (WINAPI\_FAMILY == WINAPI\_FAMILY\_APP)

3269 #define DOCTEST\_CONFIG\_NO\_MULTI\_LANE\_ATOMICS

3270 #endif

3271

3272 #ifndef DOCTEST\_CDECL

3273 #define DOCTEST\_CDECL \_\_cdecl

3274 #endif

3275

3276 namespace doctest {

3277

3278 bool is\_running\_in\_test = false;

3279

3280 namespace {

3281 using namespace detail;

3282

3283 template <typename Ex>

3284 DOCTEST\_NORETURN void throw\_exception(Ex const& e) {

3285 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

3286 throw e;

3287 #else // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

3288 #ifdef DOCTEST\_CONFIG\_HANDLE\_EXCEPTION

3289 DOCTEST\_CONFIG\_HANDLE\_EXCEPTION(e);

3290 #else // DOCTEST\_CONFIG\_HANDLE\_EXCEPTION

3291 #ifndef DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

3292 std::cerr << "doctest will terminate because it needed to throw an exception.\n"

3293 << "The message was: " << e.what() << '\n';

3294 #endif // DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

3295 #endif // DOCTEST\_CONFIG\_HANDLE\_EXCEPTION

3296 std::terminate();

3297 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

3298 }

3299

3300 #ifndef DOCTEST\_INTERNAL\_ERROR

3301 #define DOCTEST\_INTERNAL\_ERROR(msg) \

3302 throw\_exception(std::logic\_error( \

3303 \_\_FILE\_\_ ":" DOCTEST\_TOSTR(\_\_LINE\_\_) ": Internal doctest error: " msg))

3304 #endif // DOCTEST\_INTERNAL\_ERROR

3305

3306 // case insensitive strcmp

3307 int stricmp(const char\* a, const char\* b) {

3308 for(;; a++, b++) {

3309 const int d = tolower(\*a) - tolower(\*b);

3310 if(d != 0 || !\*a)

3311 return d;

3312 }

3313 }

3314

3315 struct Endianness

3316 {

3317 enum Arch

3318 {

3319 Big,

3320 Little

3321 };

3322

3323 static Arch which() {

3324 int x = 1;

3325 // casting any data pointer to char\* is allowed

3326 auto ptr = reinterpret\_cast<char\*>(&x);

3327 if(\*ptr)

3328 return Little;

3329 return Big;

3330 }

3331 };

3332 } // namespace

3333

3334 namespace detail {

3335 DOCTEST\_THREAD\_LOCAL class

3336 {

3337 std::vector<std::streampos> stack;

3338 std::stringstream ss;

3339

3340 public:

3341 std::ostream\* push() {

3342 stack.push\_back(ss.tellp());

3343 return &ss;

3344 }

3345

3346 String pop() {

3347 if (stack.empty())

3348 DOCTEST\_INTERNAL\_ERROR("TLSS was empty when trying to pop!");

3349

3350 std::streampos pos = stack.back();

3351 stack.pop\_back();

3352 unsigned sz = static\_cast<unsigned>(ss.tellp() - pos);

3353 ss.rdbuf()->pubseekpos(pos, std::ios::in | std::ios::out);

3354 return String(ss, sz);

3355 }

3356 } g\_oss;

3357

3358 std::ostream\* tlssPush() {

3359 return g\_oss.push();

3360 }

3361

3362 String tlssPop() {

3363 return g\_oss.pop();

3364 }

3365

3366 #ifndef DOCTEST\_CONFIG\_DISABLE

3367

3368 namespace timer\_large\_integer

3369 {

3370

3371 #if defined(DOCTEST\_PLATFORM\_WINDOWS)

3372 using type = ULONGLONG;

3373 #else // DOCTEST\_PLATFORM\_WINDOWS

3374 using type = std::uint64\_t;

3375 #endif // DOCTEST\_PLATFORM\_WINDOWS

3376 }

3377

3378 using ticks\_t = timer\_large\_integer::type;

3379

3380 #ifdef DOCTEST\_CONFIG\_GETCURRENTTICKS

3381 ticks\_t getCurrentTicks() { return DOCTEST\_CONFIG\_GETCURRENTTICKS(); }

3382 #elif defined(DOCTEST\_PLATFORM\_WINDOWS)

3383 ticks\_t getCurrentTicks() {

3384 static LARGE\_INTEGER hz = { {0} }, hzo = { {0} };

3385 if(!hz.QuadPart) {

3386 QueryPerformanceFrequency(&hz);

3387 QueryPerformanceCounter(&hzo);

3388 }

3389 LARGE\_INTEGER t;

3390 QueryPerformanceCounter(&t);

3391 return ((t.QuadPart - hzo.QuadPart) \* LONGLONG(1000000)) / hz.QuadPart;

3392 }

3393 #else // DOCTEST\_PLATFORM\_WINDOWS

3394 ticks\_t getCurrentTicks() {

3395 timeval t;

3396 gettimeofday(&t, nullptr);

3397 return static\_cast<ticks\_t>(t.tv\_sec) \* 1000000 + static\_cast<ticks\_t>(t.tv\_usec);

3398 }

3399 #endif // DOCTEST\_PLATFORM\_WINDOWS

3400

3401 struct Timer

3402 {

3403 void start() { m\_ticks = getCurrentTicks(); }

3404 unsigned int getElapsedMicroseconds() const {

3405 return static\_cast<unsigned int>(getCurrentTicks() - m\_ticks);

3406 }

3407 //unsigned int getElapsedMilliseconds() const {

3408 // return static\_cast<unsigned int>(getElapsedMicroseconds() / 1000);

3409 //}

3410 double getElapsedSeconds() const { return static\_cast<double>(getCurrentTicks() - m\_ticks) / 1000000.0; }

3411

3412 private:

3413 ticks\_t m\_ticks = 0;

3414 };

3415

3416 #ifdef DOCTEST\_CONFIG\_NO\_MULTITHREADING

3417 template <typename T>

3418 using Atomic = T;

3419 #else // DOCTEST\_CONFIG\_NO\_MULTITHREADING

3420 template <typename T>

3421 using Atomic = std::atomic<T>;

3422 #endif // DOCTEST\_CONFIG\_NO\_MULTITHREADING

3423

3424 #if defined(DOCTEST\_CONFIG\_NO\_MULTI\_LANE\_ATOMICS) || defined(DOCTEST\_CONFIG\_NO\_MULTITHREADING)

3425 template <typename T>

3426 using MultiLaneAtomic = Atomic<T>;

3427 #else // DOCTEST\_CONFIG\_NO\_MULTI\_LANE\_ATOMICS

3428 // Provides a multilane implementation of an atomic variable that supports add, sub, load,

3429 // store. Instead of using a single atomic variable, this splits up into multiple ones,

3430 // each sitting on a separate cache line. The goal is to provide a speedup when most

3431 // operations are modifying. It achieves this with two properties:

3432 //

3433 // \* Multiple atomics are used, so chance of congestion from the same atomic is reduced.

3434 // \* Each atomic sits on a separate cache line, so false sharing is reduced.

3435 //

3436 // The disadvantage is that there is a small overhead due to the use of TLS, and load/store

3437 // is slower because all atomics have to be accessed.

3438 template <typename T>

3439 class MultiLaneAtomic

3440 {

3441 struct CacheLineAlignedAtomic

3442 {

3443 Atomic<T> atomic{};

3444 char padding[DOCTEST\_MULTI\_LANE\_ATOMICS\_CACHE\_LINE\_SIZE - sizeof(Atomic<T>)];

3445 };

3446 CacheLineAlignedAtomic m\_atomics[DOCTEST\_MULTI\_LANE\_ATOMICS\_THREAD\_LANES];

3447

3448 static\_assert(sizeof(CacheLineAlignedAtomic) == DOCTEST\_MULTI\_LANE\_ATOMICS\_CACHE\_LINE\_SIZE,

3449 "guarantee one atomic takes exactly one cache line");

3450

3451 public:

3452 T operator++() DOCTEST\_NOEXCEPT { return fetch\_add(1) + 1; }

3453

3454 T operator++(int) DOCTEST\_NOEXCEPT { return fetch\_add(1); }

3455

3456 T fetch\_add(T arg, std::memory\_order order = std::memory\_order\_seq\_cst) DOCTEST\_NOEXCEPT {

3457 return myAtomic().fetch\_add(arg, order);

3458 }

3459

3460 T fetch\_sub(T arg, std::memory\_order order = std::memory\_order\_seq\_cst) DOCTEST\_NOEXCEPT {

3461 return myAtomic().fetch\_sub(arg, order);

3462 }

3463

3464 operator T() const DOCTEST\_NOEXCEPT { return load(); }

3465

3466 T load(std::memory\_order order = std::memory\_order\_seq\_cst) const DOCTEST\_NOEXCEPT {

3467 auto result = T();

3468 for(auto const& c : m\_atomics) {

3469 result += c.atomic.load(order);

3470 }

3471 return result;

3472 }

3473

3474 T operator=(T desired) DOCTEST\_NOEXCEPT { // lgtm [cpp/assignment-does-not-return-this]

3475 store(desired);

3476 return desired;

3477 }

3478

3479 void store(T desired, std::memory\_order order = std::memory\_order\_seq\_cst) DOCTEST\_NOEXCEPT {

3480 // first value becomes desired", all others become 0.

3481 for(auto& c : m\_atomics) {

3482 c.atomic.store(desired, order);

3483 desired = {};

3484 }

3485 }

3486

3487 private:

3488 // Each thread has a different atomic that it operates on. If more than NumLanes threads

3489 // use this, some will use the same atomic. So performance will degrade a bit, but still

3490 // everything will work.

3491 //

3492 // The logic here is a bit tricky. The call should be as fast as possible, so that there

3493 // is minimal to no overhead in determining the correct atomic for the current thread.

3494 //

3495 // 1. A global static counter laneCounter counts continuously up.

3496 // 2. Each successive thread will use modulo operation of that counter so it gets an atomic

3497 // assigned in a round-robin fashion.

3498 // 3. This tlsLaneIdx is stored in the thread local data, so it is directly available with

3499 // little overhead.

3500 Atomic<T>& myAtomic() DOCTEST\_NOEXCEPT {

3501 static Atomic<size\_t> laneCounter;

3502 DOCTEST\_THREAD\_LOCAL size\_t tlsLaneIdx =

3503 laneCounter++ % DOCTEST\_MULTI\_LANE\_ATOMICS\_THREAD\_LANES;

3504

3505 return m\_atomics[tlsLaneIdx].atomic;

3506 }

3507 };

3508 #endif // DOCTEST\_CONFIG\_NO\_MULTI\_LANE\_ATOMICS

3509

3510 // this holds both parameters from the command line and runtime data for tests

3511 struct ContextState : ContextOptions, TestRunStats, CurrentTestCaseStats

3512 {

3513 MultiLaneAtomic<int> numAssertsCurrentTest\_atomic;

3514 MultiLaneAtomic<int> numAssertsFailedCurrentTest\_atomic;

3515

3516 std::vector<std::vector<String>> filters = decltype(filters)(9); // 9 different filters

3517

3518 std::vector<IReporter\*> reporters\_currently\_used;

3519

3520 assert\_handler ah = nullptr;

3521

3522 Timer timer;

3523

3524 std::vector<String> stringifiedContexts; // logging from INFO() due to an exception

3525

3526 // stuff for subcases

3527 bool reachedLeaf;

3528 std::vector<SubcaseSignature> subcaseStack;

3529 std::vector<SubcaseSignature> nextSubcaseStack;

3530 std::unordered\_set<unsigned long long> fullyTraversedSubcases;

3531 size\_t currentSubcaseDepth;

3532 Atomic<bool> shouldLogCurrentException;

3533

3534 void resetRunData() {

3535 numTestCases = 0;

3536 numTestCasesPassingFilters = 0;

3537 numTestSuitesPassingFilters = 0;

3538 numTestCasesFailed = 0;

3539 numAsserts = 0;

3540 numAssertsFailed = 0;

3541 numAssertsCurrentTest = 0;

3542 numAssertsFailedCurrentTest = 0;

3543 }

3544

3545 void finalizeTestCaseData() {

3546 seconds = timer.getElapsedSeconds();

3547

3548 // update the non-atomic counters

3549 numAsserts += numAssertsCurrentTest\_atomic;

3550 numAssertsFailed += numAssertsFailedCurrentTest\_atomic;

3551 numAssertsCurrentTest = numAssertsCurrentTest\_atomic;

3552 numAssertsFailedCurrentTest = numAssertsFailedCurrentTest\_atomic;

3553

3554 if(numAssertsFailedCurrentTest)

3555 failure\_flags |= TestCaseFailureReason::AssertFailure;

3556

3557 if(Approx(currentTest->m\_timeout).epsilon(DBL\_EPSILON) != 0 &&

3558 Approx(seconds).epsilon(DBL\_EPSILON) > currentTest->m\_timeout)

3559 failure\_flags |= TestCaseFailureReason::Timeout;

3560

3561 if(currentTest->m\_should\_fail) {

3562 if(failure\_flags) {

3563 failure\_flags |= TestCaseFailureReason::ShouldHaveFailedAndDid;

3564 } else {

3565 failure\_flags |= TestCaseFailureReason::ShouldHaveFailedButDidnt;

3566 }

3567 } else if(failure\_flags && currentTest->m\_may\_fail) {

3568 failure\_flags |= TestCaseFailureReason::CouldHaveFailedAndDid;

3569 } else if(currentTest->m\_expected\_failures > 0) {

3570 if(numAssertsFailedCurrentTest == currentTest->m\_expected\_failures) {

3571 failure\_flags |= TestCaseFailureReason::FailedExactlyNumTimes;

3572 } else {

3573 failure\_flags |= TestCaseFailureReason::DidntFailExactlyNumTimes;

3574 }

3575 }

3576

3577 bool ok\_to\_fail = (TestCaseFailureReason::ShouldHaveFailedAndDid & failure\_flags) ||

3578 (TestCaseFailureReason::CouldHaveFailedAndDid & failure\_flags) ||

3579 (TestCaseFailureReason::FailedExactlyNumTimes & failure\_flags);

3580

3581 // if any subcase has failed - the whole test case has failed

3582 testCaseSuccess = !(failure\_flags && !ok\_to\_fail);

3583 if(!testCaseSuccess)

3584 numTestCasesFailed++;

3585 }

3586 };

3587

3588 ContextState\* g\_cs = nullptr;

3589

3590 // used to avoid locks for the debug output

3591 // TODO: figure out if this is indeed necessary/correct - seems like either there still

3592 // could be a race or that there wouldn't be a race even if using the context directly

3593 DOCTEST\_THREAD\_LOCAL bool g\_no\_colors;

3594

3595 #endif // DOCTEST\_CONFIG\_DISABLE

3596 } // namespace detail

3597

3598 char\* String::allocate(size\_type sz) {

3599 if (sz <= last) {

3600 buf[sz] = '\0';

3601 setLast(last - sz);

3602 return buf;

3603 } else {

3604 setOnHeap();

3605 data.size = sz;

3606 data.capacity = data.size + 1;

3607 data.ptr = new char[data.capacity];

3608 data.ptr[sz] = '\0';

3609 return data.ptr;

3610 }

3611 }

3612

3613 void String::setOnHeap() noexcept { \*reinterpret\_cast<unsigned char\*>(&buf[last]) = 128; }

3614 void String::setLast(size\_type in) noexcept { buf[last] = char(in); }

3615 void String::setSize(size\_type sz) noexcept {

3616 if (isOnStack()) { buf[sz] = '\0'; setLast(last - sz); }

3617 else { data.ptr[sz] = '\0'; data.size = sz; }

3618 }

3619

3620 void String::copy(const String& other) {

3621 if(other.isOnStack()) {

3622 memcpy(buf, other.buf, len);

3623 } else {

3624 memcpy(allocate(other.data.size), other.data.ptr, other.data.size);

3625 }

3626 }

3627

3628 String::String() noexcept {

3629 buf[0] = '\0';

3630 setLast();

3631 }

3632

3633 String::~String() {

3634 if(!isOnStack())

3635 delete[] data.ptr;

3636 } // NOLINT(clang-analyzer-cplusplus.NewDeleteLeaks)

3637

3638 String::String(const char\* in)

3639 : String(in, strlen(in)) {}

3640

3641 String::String(const char\* in, size\_type in\_size) {

3642 memcpy(allocate(in\_size), in, in\_size);

3643 }

3644

3645 String::String(std::istream& in, size\_type in\_size) {

3646 in.read(allocate(in\_size), in\_size);

3647 }

3648

3649 String::String(const String& other) { copy(other); }

3650

3651 String& String::operator=(const String& other) {

3652 if(this != &other) {

3653 if(!isOnStack())

3654 delete[] data.ptr;

3655

3656 copy(other);

3657 }

3658

3659 return \*this;

3660 }

3661

3662 String& String::operator+=(const String& other) {

3663 const size\_type my\_old\_size = size();

3664 const size\_type other\_size = other.size();

3665 const size\_type total\_size = my\_old\_size + other\_size;

3666 if(isOnStack()) {

3667 if(total\_size < len) {

3668 // append to the current stack space

3669 memcpy(buf + my\_old\_size, other.c\_str(), other\_size + 1);

3670 // NOLINTNEXTLINE(clang-analyzer-cplusplus.NewDeleteLeaks)

3671 setLast(last - total\_size);

3672 } else {

3673 // alloc new chunk

3674 char\* temp = new char[total\_size + 1];

3675 // copy current data to new location before writing in the union

3676 memcpy(temp, buf, my\_old\_size); // skip the +1 ('\0') for speed

3677 // update data in union

3678 setOnHeap();

3679 data.size = total\_size;

3680 data.capacity = data.size + 1;

3681 data.ptr = temp;

3682 // transfer the rest of the data

3683 memcpy(data.ptr + my\_old\_size, other.c\_str(), other\_size + 1);

3684 }

3685 } else {

3686 if(data.capacity > total\_size) {

3687 // append to the current heap block

3688 data.size = total\_size;

3689 memcpy(data.ptr + my\_old\_size, other.c\_str(), other\_size + 1);

3690 } else {

3691 // resize

3692 data.capacity \*= 2;

3693 if(data.capacity <= total\_size)

3694 data.capacity = total\_size + 1;

3695 // alloc new chunk

3696 char\* temp = new char[data.capacity];

3697 // copy current data to new location before releasing it

3698 memcpy(temp, data.ptr, my\_old\_size); // skip the +1 ('\0') for speed

3699 // release old chunk

3700 delete[] data.ptr;

3701 // update the rest of the union members

3702 data.size = total\_size;

3703 data.ptr = temp;

3704 // transfer the rest of the data

3705 memcpy(data.ptr + my\_old\_size, other.c\_str(), other\_size + 1);

3706 }

3707 }

3708

3709 return \*this;

3710 }

3711

3712 String::String(String&& other) noexcept {

3713 memcpy(buf, other.buf, len);

3714 other.buf[0] = '\0';

3715 other.setLast();

3716 }

3717

3718 String& String::operator=(String&& other) noexcept {

3719 if(this != &other) {

3720 if(!isOnStack())

3721 delete[] data.ptr;

3722 memcpy(buf, other.buf, len);

3723 other.buf[0] = '\0';

3724 other.setLast();

3725 }

3726 return \*this;

3727 }

3728

3729 char String::operator[](size\_type i) const {

3730 return const\_cast<String\*>(this)->operator[](i);

3731 }

3732

3733 char& String::operator[](size\_type i) {

3734 if(isOnStack())

3735 return reinterpret\_cast<char\*>(buf)[i];

3736 return data.ptr[i];

3737 }

3738

3739 DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wmaybe-uninitialized")

3740 String::size\_type String::size() const {

3741 if(isOnStack())

3742 return last - (size\_type(buf[last]) & 31); // using "last" would work only if "len" is 32

3743 return data.size;

3744 }

3745 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

3746

3747 String::size\_type String::capacity() const {

3748 if(isOnStack())

3749 return len;

3750 return data.capacity;

3751 }

3752

3753 String String::substr(size\_type pos, size\_type cnt) && {

3754 cnt = std::min(cnt, size() - 1 - pos);

3755 char\* cptr = c\_str();

3756 memmove(cptr, cptr + pos, cnt);

3757 setSize(cnt);

3758 return std::move(\*this);

3759 }

3760

3761 String String::substr(size\_type pos, size\_type cnt) const & {

3762 cnt = std::min(cnt, size() - 1 - pos);

3763 return String{ c\_str() + pos, cnt };

3764 }

3765

3766 String::size\_type String::find(char ch, size\_type pos) const {

3767 const char\* begin = c\_str();

3768 const char\* end = begin + size();

3769 const char\* it = begin + pos;

3770 for (; it < end && \*it != ch; it++);

3771 if (it < end) { return static\_cast<size\_type>(it - begin); }

3772 else { return npos; }

3773 }

3774

3775 String::size\_type String::rfind(char ch, size\_type pos) const {

3776 const char\* begin = c\_str();

3777 const char\* it = begin + std::min(pos, size() - 1);

3778 for (; it >= begin && \*it != ch; it--);

3779 if (it >= begin) { return static\_cast<size\_type>(it - begin); }

3780 else { return npos; }

3781 }

3782

3783 int String::compare(const char\* other, bool no\_case) const {

3784 if(no\_case)

3785 return doctest::stricmp(c\_str(), other);

3786 return std::strcmp(c\_str(), other);

3787 }

3788

3789 int String::compare(const String& other, bool no\_case) const {

3790 return compare(other.c\_str(), no\_case);

3791 }

3792

3793 String operator+(const String& lhs, const String& rhs) { return String(lhs) += rhs; }

3794

3795 bool operator==(const String& lhs, const String& rhs) { return lhs.compare(rhs) == 0; }

3796 bool operator!=(const String& lhs, const String& rhs) { return lhs.compare(rhs) != 0; }

3797 bool operator< (const String& lhs, const String& rhs) { return lhs.compare(rhs) < 0; }

3798 bool operator> (const String& lhs, const String& rhs) { return lhs.compare(rhs) > 0; }

3799 bool operator<=(const String& lhs, const String& rhs) { return (lhs != rhs) ? lhs.compare(rhs) < 0 : true; }

3800 bool operator>=(const String& lhs, const String& rhs) { return (lhs != rhs) ? lhs.compare(rhs) > 0 : true; }

3801

3802 std::ostream& operator<<(std::ostream& s, const String& in) { return s << in.c\_str(); }

3803

3804 Contains::Contains(const String& str) : string(str) { }

3805

3806 bool Contains::checkWith(const String& other) const {

3807 return strstr(other.c\_str(), string.c\_str()) != nullptr;

3808 }

3809

3810 String toString(const Contains& in) {

3811 return "Contains( " + in.string + " )";

3812 }

3813

3814 bool operator==(const String& lhs, const Contains& rhs) { return rhs.checkWith(lhs); }

3815 bool operator==(const Contains& lhs, const String& rhs) { return lhs.checkWith(rhs); }

3816 bool operator!=(const String& lhs, const Contains& rhs) { return !rhs.checkWith(lhs); }

3817 bool operator!=(const Contains& lhs, const String& rhs) { return !lhs.checkWith(rhs); }

3818

3819 namespace {

3820 void color\_to\_stream(std::ostream&, Color::Enum) DOCTEST\_BRANCH\_ON\_DISABLED({}, ;)

3821 } // namespace

3822

3823 namespace Color {

3824 std::ostream& operator<<(std::ostream& s, Color::Enum code) {

3825 color\_to\_stream(s, code);

3826 return s;

3827 }

3828 } // namespace Color

3829

3830 // clang-format off

3831 const char\* assertString(assertType::Enum at) {

3832 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4061) // enum 'x' in switch of enum 'y' is not explicitly handled

3833 #define DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASE(assert\_type) case assertType::DT\_ ## assert\_type: return #assert\_type

3834 #define DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(assert\_type) \

3835 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASE(WARN\_ ## assert\_type); \

3836 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASE(CHECK\_ ## assert\_type); \

3837 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASE(REQUIRE\_ ## assert\_type)

3838 switch(at) {

3839 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASE(WARN);

3840 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASE(CHECK);

3841 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASE(REQUIRE);

3842

3843 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(FALSE);

3844

3845 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(THROWS);

3846

3847 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(THROWS\_AS);

3848

3849 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(THROWS\_WITH);

3850

3851 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(THROWS\_WITH\_AS);

3852

3853 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(NOTHROW);

3854

3855 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(EQ);

3856 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(NE);

3857 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(GT);

3858 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(LT);

3859 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(GE);

3860 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(LE);

3861

3862 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(UNARY);

3863 DOCTEST\_GENERATE\_ASSERT\_TYPE\_CASES(UNARY\_FALSE);

3864

3865 default: DOCTEST\_INTERNAL\_ERROR("Tried stringifying invalid assert type!");

3866 }

3867 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

3868 }

3869 // clang-format on

3870

3871 const char\* failureString(assertType::Enum at) {

3872 if(at & assertType::is\_warn)

3873 return "WARNING";

3874 if(at & assertType::is\_check)

3875 return "ERROR";

3876 if(at & assertType::is\_require)

3877 return "FATAL ERROR";

3878 return "";

3879 }

3880

3881 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wnull-dereference")

3882 DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wnull-dereference")

3883 // depending on the current options this will remove the path of filenames

3884 const char\* skipPathFromFilename(const char\* file) {

3885 #ifndef DOCTEST\_CONFIG\_DISABLE

3886 if(getContextOptions()->no\_path\_in\_filenames) {

3887 auto back = std::strrchr(file, '\\');

3888 auto forward = std::strrchr(file, '/');

3889 if(back || forward) {

3890 if(back > forward)

3891 forward = back;

3892 return forward + 1;

3893 }

3894 }

3895 #endif // DOCTEST\_CONFIG\_DISABLE

3896 return file;

3897 }

3898 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

3899 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

3900

3901 bool SubcaseSignature::operator==(const SubcaseSignature& other) const {

3902 return m\_line == other.m\_line

3903 && std::strcmp(m\_file, other.m\_file) == 0

3904 && m\_name == other.m\_name;

3905 }

3906

3907 bool SubcaseSignature::operator<(const SubcaseSignature& other) const {

3908 if(m\_line != other.m\_line)

3909 return m\_line < other.m\_line;

3910 if(std::strcmp(m\_file, other.m\_file) != 0)

3911 return std::strcmp(m\_file, other.m\_file) < 0;

3912 return m\_name.compare(other.m\_name) < 0;

3913 }

3914

3915 DOCTEST\_DEFINE\_INTERFACE(IContextScope)

3916

3917 namespace detail {

3918 void filldata<const void\*>::fill(std::ostream\* stream, const void\* in) {

3919 if (in) { \*stream << in; }

3920 else { \*stream << "nullptr"; }

3921 }

3922

3923 template <typename T>

3924 String toStreamLit(T t) {

3925 std::ostream\* os = tlssPush();

3926 os->operator<<(t);

3927 return tlssPop();

3928 }

3929 }

3930

3931 #ifdef DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

3932 String toString(const char\* in) { return String("\"") + (in ? in : "{null string}") + "\""; }

3933 #endif // DOCTEST\_CONFIG\_TREAT\_CHAR\_STAR\_AS\_STRING

3934

3935 #if DOCTEST\_MSVC >= DOCTEST\_COMPILER(19, 20, 0)

3936 // see this issue on why this is needed: https://github.com/doctest/doctest/issues/183

3937 String toString(const std::string& in) { return in.c\_str(); }

3938 #endif // VS 2019

3939

3940 String toString(String in) { return in; }

3941

3942 String toString(std::nullptr\_t) { return "nullptr"; }

3943

3944 String toString(bool in) { return in ? "true" : "false"; }

3945

3946 String toString(float in) { return toStreamLit(in); }

3947 String toString(double in) { return toStreamLit(in); }

3948 String toString(double long in) { return toStreamLit(in); }

3949

3950 String toString(char in) { return toStreamLit(static\_cast<signed>(in)); }

3951 String toString(char signed in) { return toStreamLit(static\_cast<signed>(in)); }

3952 String toString(char unsigned in) { return toStreamLit(static\_cast<unsigned>(in)); }

3953 String toString(short in) { return toStreamLit(in); }

3954 String toString(short unsigned in) { return toStreamLit(in); }

3955 String toString(signed in) { return toStreamLit(in); }

3956 String toString(unsigned in) { return toStreamLit(in); }

3957 String toString(long in) { return toStreamLit(in); }

3958 String toString(long unsigned in) { return toStreamLit(in); }

3959 String toString(long long in) { return toStreamLit(in); }

3960 String toString(long long unsigned in) { return toStreamLit(in); }

3961

3962 Approx::Approx(double value)

3963 : m\_epsilon(static\_cast<double>(std::numeric\_limits<float>::epsilon()) \* 100)

3964 , m\_scale(1.0)

3965 , m\_value(value) {}

3966

3967 Approx Approx::operator()(double value) const {

3968 Approx approx(value);

3969 approx.epsilon(m\_epsilon);

3970 approx.scale(m\_scale);

3971 return approx;

3972 }

3973

3974 Approx& Approx::epsilon(double newEpsilon) {

3975 m\_epsilon = newEpsilon;

3976 return \*this;

3977 }

3978 Approx& Approx::scale(double newScale) {

3979 m\_scale = newScale;

3980 return \*this;

3981 }

3982

3983 bool operator==(double lhs, const Approx& rhs) {

3984 // Thanks to Richard Harris for his help refining this formula

3985 return std::fabs(lhs - rhs.m\_value) <

3986 rhs.m\_epsilon \* (rhs.m\_scale + std::max<double>(std::fabs(lhs), std::fabs(rhs.m\_value)));

3987 }

3988 bool operator==(const Approx& lhs, double rhs) { return operator==(rhs, lhs); }

3989 bool operator!=(double lhs, const Approx& rhs) { return !operator==(lhs, rhs); }

3990 bool operator!=(const Approx& lhs, double rhs) { return !operator==(rhs, lhs); }

3991 bool operator<=(double lhs, const Approx& rhs) { return lhs < rhs.m\_value || lhs == rhs; }

3992 bool operator<=(const Approx& lhs, double rhs) { return lhs.m\_value < rhs || lhs == rhs; }

3993 bool operator>=(double lhs, const Approx& rhs) { return lhs > rhs.m\_value || lhs == rhs; }

3994 bool operator>=(const Approx& lhs, double rhs) { return lhs.m\_value > rhs || lhs == rhs; }

3995 bool operator<(double lhs, const Approx& rhs) { return lhs < rhs.m\_value && lhs != rhs; }

3996 bool operator<(const Approx& lhs, double rhs) { return lhs.m\_value < rhs && lhs != rhs; }

3997 bool operator>(double lhs, const Approx& rhs) { return lhs > rhs.m\_value && lhs != rhs; }

3998 bool operator>(const Approx& lhs, double rhs) { return lhs.m\_value > rhs && lhs != rhs; }

3999

4000 String toString(const Approx& in) {

4001 return "Approx( " + doctest::toString(in.m\_value) + " )";

4002 }

4003 const ContextOptions\* getContextOptions() { return DOCTEST\_BRANCH\_ON\_DISABLED(nullptr, g\_cs); }

4004

4005 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4738)

4006 template <typename F>

4007 IsNaN<F>::operator bool() const {

4008 return std::isnan(value) ^ flipped;

4009 }

4010 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

4011 template struct DOCTEST\_INTERFACE\_DEF IsNaN<float>;

4012 template struct DOCTEST\_INTERFACE\_DEF IsNaN<double>;

4013 template struct DOCTEST\_INTERFACE\_DEF IsNaN<long double>;

4014 template <typename F>

4015 String toString(IsNaN<F> in) { return String(in.flipped ? "! " : "") + "IsNaN( " + doctest::toString(in.value) + " )"; }

4016 String toString(IsNaN<float> in) { return toString<float>(in); }

4017 String toString(IsNaN<double> in) { return toString<double>(in); }

4018 String toString(IsNaN<double long> in) { return toString<double long>(in); }

4019

4020 } // namespace doctest

4021

4022 #ifdef DOCTEST\_CONFIG\_DISABLE

4023 namespace doctest {

4024 Context::Context(int, const char\* const\*) {}

4025 Context::~Context() = default;

4026 void Context::applyCommandLine(int, const char\* const\*) {}

4027 void Context::addFilter(const char\*, const char\*) {}

4028 void Context::clearFilters() {}

4029 void Context::setOption(const char\*, bool) {}

4030 void Context::setOption(const char\*, int) {}

4031 void Context::setOption(const char\*, const char\*) {}

4032 bool Context::shouldExit() { return false; }

4033 void Context::setAsDefaultForAssertsOutOfTestCases() {}

4034 void Context::setAssertHandler(detail::assert\_handler) {}

4035 void Context::setCout(std::ostream\*) {}

4036 int Context::run() { return 0; }

4037

4038 int IReporter::get\_num\_active\_contexts() { return 0; }

4039 const IContextScope\* const\* IReporter::get\_active\_contexts() { return nullptr; }

4040 int IReporter::get\_num\_stringified\_contexts() { return 0; }

4041 const String\* IReporter::get\_stringified\_contexts() { return nullptr; }

4042

4043 int registerReporter(const char\*, int, IReporter\*) { return 0; }

4044

4045 } // namespace doctest

4046 #else // DOCTEST\_CONFIG\_DISABLE

4047

4048 #if !defined(DOCTEST\_CONFIG\_COLORS\_NONE)

4049 #if !defined(DOCTEST\_CONFIG\_COLORS\_WINDOWS) && !defined(DOCTEST\_CONFIG\_COLORS\_ANSI)

4050 #ifdef DOCTEST\_PLATFORM\_WINDOWS

4051 #define DOCTEST\_CONFIG\_COLORS\_WINDOWS

4052 #else // linux

4053 #define DOCTEST\_CONFIG\_COLORS\_ANSI

4054 #endif // platform

4055 #endif // DOCTEST\_CONFIG\_COLORS\_WINDOWS && DOCTEST\_CONFIG\_COLORS\_ANSI

4056 #endif // DOCTEST\_CONFIG\_COLORS\_NONE

4057

4058 namespace doctest\_detail\_test\_suite\_ns {

4059 // holds the current test suite

4060 doctest::detail::TestSuite& getCurrentTestSuite() {

4061 static doctest::detail::TestSuite data{};

4062 return data;

4063 }

4064 } // namespace doctest\_detail\_test\_suite\_ns

4065

4066 namespace doctest {

4067 namespace {

4068 // the int (priority) is part of the key for automatic sorting - sadly one can register a

4069 // reporter with a duplicate name and a different priority but hopefully that won't happen often :|

4070 using reporterMap = std::map<std::pair<int, String>, reporterCreatorFunc>;

4071

4072 reporterMap& getReporters() {

4073 static reporterMap data;

4074 return data;

4075 }

4076 reporterMap& getListeners() {

4077 static reporterMap data;

4078 return data;

4079 }

4080 } // namespace

4081 namespace detail {

4082 #define DOCTEST\_ITERATE\_THROUGH\_REPORTERS(function, ...) \

4083 for(auto& curr\_rep : g\_cs->reporters\_currently\_used) \

4084 curr\_rep->function(\_\_VA\_ARGS\_\_)

4085

4086 bool checkIfShouldThrow(assertType::Enum at) {

4087 if(at & assertType::is\_require)

4088 return true;

4089

4090 if((at & assertType::is\_check)

4091 && getContextOptions()->abort\_after > 0 &&

4092 (g\_cs->numAssertsFailed + g\_cs->numAssertsFailedCurrentTest\_atomic) >=

4093 getContextOptions()->abort\_after)

4094 return true;

4095

4096 return false;

4097 }

4098

4099 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

4100 DOCTEST\_NORETURN void throwException() {

4101 g\_cs->shouldLogCurrentException = false;

4102 throw TestFailureException(); // NOLINT(hicpp-exception-baseclass)

4103 }

4104 #else // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

4105 void throwException() {}

4106 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

4107 } // namespace detail

4108

4109 namespace {

4110 using namespace detail;

4111 // matching of a string against a wildcard mask (case sensitivity configurable) taken from

4112 // https://www.codeproject.com/Articles/1088/Wildcard-string-compare-globbing

4113 int wildcmp(const char\* str, const char\* wild, bool caseSensitive) {

4114 const char\* cp = str;

4115 const char\* mp = wild;

4116

4117 while((\*str) && (\*wild != '\*')) {

4118 if((caseSensitive ? (\*wild != \*str) : (tolower(\*wild) != tolower(\*str))) &&

4119 (\*wild != '?')) {

4120 return 0;

4121 }

4122 wild++;

4123 str++;

4124 }

4125

4126 while(\*str) {

4127 if(\*wild == '\*') {

4128 if(!\*++wild) {

4129 return 1;

4130 }

4131 mp = wild;

4132 cp = str + 1;

4133 } else if((caseSensitive ? (\*wild == \*str) : (tolower(\*wild) == tolower(\*str))) ||

4134 (\*wild == '?')) {

4135 wild++;

4136 str++;

4137 } else {

4138 wild = mp;

4139 str = cp++;

4140 }

4141 }

4142

4143 while(\*wild == '\*') {

4144 wild++;

4145 }

4146 return !\*wild;

4147 }

4148

4149 // checks if the name matches any of the filters (and can be configured what to do when empty)

4150 bool matchesAny(const char\* name, const std::vector<String>& filters, bool matchEmpty,

4151 bool caseSensitive) {

4152 if (filters.empty() && matchEmpty)

4153 return true;

4154 for (auto& curr : filters)

4155 if (wildcmp(name, curr.c\_str(), caseSensitive))

4156 return true;

4157 return false;

4158 }

4159

4160 DOCTEST\_NO\_SANITIZE\_INTEGER

4161 unsigned long long hash(unsigned long long a, unsigned long long b) {

4162 return (a << 5) + b;

4163 }

4164

4165 // C string hash function (djb2) - taken from http://www.cse.yorku.ca/~oz/hash.html

4166 DOCTEST\_NO\_SANITIZE\_INTEGER

4167 unsigned long long hash(const char\* str) {

4168 unsigned long long hash = 5381;

4169 char c;

4170 while ((c = \*str++))

4171 hash = ((hash << 5) + hash) + c; // hash \* 33 + c

4172 return hash;

4173 }

4174

4175 unsigned long long hash(const SubcaseSignature& sig) {

4176 return hash(hash(hash(sig.m\_file), hash(sig.m\_name.c\_str())), sig.m\_line);

4177 }

4178

4179 unsigned long long hash(const std::vector<SubcaseSignature>& sigs, size\_t count) {

4180 unsigned long long running = 0;

4181 auto end = sigs.begin() + count;

4182 for (auto it = sigs.begin(); it != end; it++) {

4183 running = hash(running, hash(\*it));

4184 }

4185 return running;

4186 }

4187

4188 unsigned long long hash(const std::vector<SubcaseSignature>& sigs) {

4189 unsigned long long running = 0;

4190 for (const SubcaseSignature& sig : sigs) {

4191 running = hash(running, hash(sig));

4192 }

4193 return running;

4194 }

4195 } // namespace

4196 namespace detail {

4197 bool Subcase::checkFilters() {

4198 if (g\_cs->subcaseStack.size() < size\_t(g\_cs->subcase\_filter\_levels)) {

4199 if (!matchesAny(m\_signature.m\_name.c\_str(), g\_cs->filters[6], true, g\_cs->case\_sensitive))

4200 return true;

4201 if (matchesAny(m\_signature.m\_name.c\_str(), g\_cs->filters[7], false, g\_cs->case\_sensitive))

4202 return true;

4203 }

4204 return false;

4205 }

4206

4207 Subcase::Subcase(const String& name, const char\* file, int line)

4208 : m\_signature({name, file, line}) {

4209 if (!g\_cs->reachedLeaf) {

4210 if (g\_cs->nextSubcaseStack.size() <= g\_cs->subcaseStack.size()

4211 || g\_cs->nextSubcaseStack[g\_cs->subcaseStack.size()] == m\_signature) {

4212 // Going down.

4213 if (checkFilters()) { return; }

4214

4215 g\_cs->subcaseStack.push\_back(m\_signature);

4216 g\_cs->currentSubcaseDepth++;

4217 m\_entered = true;

4218 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(subcase\_start, m\_signature);

4219 }

4220 } else {

4221 if (g\_cs->subcaseStack[g\_cs->currentSubcaseDepth] == m\_signature) {

4222 // This subcase is reentered via control flow.

4223 g\_cs->currentSubcaseDepth++;

4224 m\_entered = true;

4225 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(subcase\_start, m\_signature);

4226 } else if (g\_cs->nextSubcaseStack.size() <= g\_cs->currentSubcaseDepth

4227 && g\_cs->fullyTraversedSubcases.find(hash(hash(g\_cs->subcaseStack, g\_cs->currentSubcaseDepth), hash(m\_signature)))

4228 == g\_cs->fullyTraversedSubcases.end()) {

4229 if (checkFilters()) { return; }

4230 // This subcase is part of the one to be executed next.

4231 g\_cs->nextSubcaseStack.clear();

4232 g\_cs->nextSubcaseStack.insert(g\_cs->nextSubcaseStack.end(),

4233 g\_cs->subcaseStack.begin(), g\_cs->subcaseStack.begin() + g\_cs->currentSubcaseDepth);

4234 g\_cs->nextSubcaseStack.push\_back(m\_signature);

4235 }

4236 }

4237 }

4238

4239 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4996) // std::uncaught\_exception is deprecated in C++17

4240 DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wdeprecated-declarations")

4241 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wdeprecated-declarations")

4242

4243 Subcase::~Subcase() {

4244 if (m\_entered) {

4245 g\_cs->currentSubcaseDepth--;

4246

4247 if (!g\_cs->reachedLeaf) {

4248 // Leaf.

4249 g\_cs->fullyTraversedSubcases.insert(hash(g\_cs->subcaseStack));

4250 g\_cs->nextSubcaseStack.clear();

4251 g\_cs->reachedLeaf = true;

4252 } else if (g\_cs->nextSubcaseStack.empty()) {

4253 // All children are finished.

4254 g\_cs->fullyTraversedSubcases.insert(hash(g\_cs->subcaseStack));

4255 }

4256

4257 #if defined(\_\_cpp\_lib\_uncaught\_exceptions) && \_\_cpp\_lib\_uncaught\_exceptions >= 201411L && (!defined(\_\_MAC\_OS\_X\_VERSION\_MIN\_REQUIRED) || \_\_MAC\_OS\_X\_VERSION\_MIN\_REQUIRED >= 101200)

4258 if(std::uncaught\_exceptions() > 0

4259 #else

4260 if(std::uncaught\_exception()

4261 #endif

4262 && g\_cs->shouldLogCurrentException) {

4263 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(

4264 test\_case\_exception, {"exception thrown in subcase - will translate later "

4265 "when the whole test case has been exited (cannot "

4266 "translate while there is an active exception)",

4267 false});

4268 g\_cs->shouldLogCurrentException = false;

4269 }

4270

4271 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(subcase\_end, DOCTEST\_EMPTY);

4272 }

4273 }

4274

4275 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

4276 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

4277 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

4278

4279 Subcase::operator bool() const { return m\_entered; }

4280

4281 Result::Result(bool passed, const String& decomposition)

4282 : m\_passed(passed)

4283 , m\_decomp(decomposition) {}

4284

4285 ExpressionDecomposer::ExpressionDecomposer(assertType::Enum at)

4286 : m\_at(at) {}

4287

4288 TestSuite& TestSuite::operator\*(const char\* in) {

4289 m\_test\_suite = in;

4290 return \*this;

4291 }

4292

4293 TestCase::TestCase(funcType test, const char\* file, unsigned line, const TestSuite& test\_suite,

4294 const String& type, int template\_id) {

4295 m\_file = file;

4296 m\_line = line;

4297 m\_name = nullptr; // will be later overridden in operator\*

4298 m\_test\_suite = test\_suite.m\_test\_suite;

4299 m\_description = test\_suite.m\_description;

4300 m\_skip = test\_suite.m\_skip;

4301 m\_no\_breaks = test\_suite.m\_no\_breaks;

4302 m\_no\_output = test\_suite.m\_no\_output;

4303 m\_may\_fail = test\_suite.m\_may\_fail;

4304 m\_should\_fail = test\_suite.m\_should\_fail;

4305 m\_expected\_failures = test\_suite.m\_expected\_failures;

4306 m\_timeout = test\_suite.m\_timeout;

4307

4308 m\_test = test;

4309 m\_type = type;

4310 m\_template\_id = template\_id;

4311 }

4312

4313 TestCase::TestCase(const TestCase& other)

4314 : TestCaseData() {

4315 \*this = other;

4316 }

4317

4318 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(26434) // hides a non-virtual function

4319 TestCase& TestCase::operator=(const TestCase& other) {

4320 TestCaseData::operator=(other);

4321 m\_test = other.m\_test;

4322 m\_type = other.m\_type;

4323 m\_template\_id = other.m\_template\_id;

4324 m\_full\_name = other.m\_full\_name;

4325

4326 if(m\_template\_id != -1)

4327 m\_name = m\_full\_name.c\_str();

4328 return \*this;

4329 }

4330 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

4331

4332 TestCase& TestCase::operator\*(const char\* in) {

4333 m\_name = in;

4334 // make a new name with an appended type for templated test case

4335 if(m\_template\_id != -1) {

4336 m\_full\_name = String(m\_name) + "<" + m\_type + ">";

4337 // redirect the name to point to the newly constructed full name

4338 m\_name = m\_full\_name.c\_str();

4339 }

4340 return \*this;

4341 }

4342

4343 bool TestCase::operator<(const TestCase& other) const {

4344 // this will be used only to differentiate between test cases - not relevant for sorting

4345 if(m\_line != other.m\_line)

4346 return m\_line < other.m\_line;

4347 const int name\_cmp = strcmp(m\_name, other.m\_name);

4348 if(name\_cmp != 0)

4349 return name\_cmp < 0;

4350 const int file\_cmp = m\_file.compare(other.m\_file);

4351 if(file\_cmp != 0)

4352 return file\_cmp < 0;

4353 return m\_template\_id < other.m\_template\_id;

4354 }

4355

4356 // all the registered tests

4357 std::set<TestCase>& getRegisteredTests() {

4358 static std::set<TestCase> data;

4359 return data;

4360 }

4361 } // namespace detail

4362 namespace {

4363 using namespace detail;

4364 // for sorting tests by file/line

4365 bool fileOrderComparator(const TestCase\* lhs, const TestCase\* rhs) {

4366 // this is needed because MSVC gives different case for drive letters

4367 // for \_\_FILE\_\_ when evaluated in a header and a source file

4368 const int res = lhs->m\_file.compare(rhs->m\_file, bool(DOCTEST\_MSVC));

4369 if(res != 0)

4370 return res < 0;

4371 if(lhs->m\_line != rhs->m\_line)

4372 return lhs->m\_line < rhs->m\_line;

4373 return lhs->m\_template\_id < rhs->m\_template\_id;

4374 }

4375

4376 // for sorting tests by suite/file/line

4377 bool suiteOrderComparator(const TestCase\* lhs, const TestCase\* rhs) {

4378 const int res = std::strcmp(lhs->m\_test\_suite, rhs->m\_test\_suite);

4379 if(res != 0)

4380 return res < 0;

4381 return fileOrderComparator(lhs, rhs);

4382 }

4383

4384 // for sorting tests by name/suite/file/line

4385 bool nameOrderComparator(const TestCase\* lhs, const TestCase\* rhs) {

4386 const int res = std::strcmp(lhs->m\_name, rhs->m\_name);

4387 if(res != 0)

4388 return res < 0;

4389 return suiteOrderComparator(lhs, rhs);

4390 }

4391

4392 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wdeprecated-declarations")

4393 void color\_to\_stream(std::ostream& s, Color::Enum code) {

4394 static\_cast<void>(s); // for DOCTEST\_CONFIG\_COLORS\_NONE or DOCTEST\_CONFIG\_COLORS\_WINDOWS

4395 static\_cast<void>(code); // for DOCTEST\_CONFIG\_COLORS\_NONE

4396 #ifdef DOCTEST\_CONFIG\_COLORS\_ANSI

4397 if(g\_no\_colors ||

4398 (isatty(STDOUT\_FILENO) == false && getContextOptions()->force\_colors == false))

4399 return;

4400

4401 auto col = "";

4402 // clang-format off

4403 switch(code) {

4404 case Color::Red: col = "[0;31m"; break;

4405 case Color::Green: col = "[0;32m"; break;

4406 case Color::Blue: col = "[0;34m"; break;

4407 case Color::Cyan: col = "[0;36m"; break;

4408 case Color::Yellow: col = "[0;33m"; break;

4409 case Color::Grey: col = "[1;30m"; break;

4410 case Color::LightGrey: col = "[0;37m"; break;

4411 case Color::BrightRed: col = "[1;31m"; break;

4412 case Color::BrightGreen: col = "[1;32m"; break;

4413 case Color::BrightWhite: col = "[1;37m"; break;

4414 case Color::Bright: // invalid

4415 case Color::None:

4416 case Color::White:

4417 default: col = "[0m";

4418 }

4419 // clang-format on

4420 s << "\033" << col;

4421 #endif // DOCTEST\_CONFIG\_COLORS\_ANSI

4422

4423 #ifdef DOCTEST\_CONFIG\_COLORS\_WINDOWS

4424 if(g\_no\_colors ||

4425 (\_isatty(\_fileno(stdout)) == false && getContextOptions()->force\_colors == false))

4426 return;

4427

4428 static struct ConsoleHelper {

4429 HANDLE stdoutHandle;

4430 WORD origFgAttrs;

4431 WORD origBgAttrs;

4432

4433 ConsoleHelper() {

4434 stdoutHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

4435 CONSOLE\_SCREEN\_BUFFER\_INFO csbiInfo;

4436 GetConsoleScreenBufferInfo(stdoutHandle, &csbiInfo);

4437 origFgAttrs = csbiInfo.wAttributes & ~(BACKGROUND\_GREEN | BACKGROUND\_RED |

4438 BACKGROUND\_BLUE | BACKGROUND\_INTENSITY);

4439 origBgAttrs = csbiInfo.wAttributes & ~(FOREGROUND\_GREEN | FOREGROUND\_RED |

4440 FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

4441 }

4442 } ch;

4443

4444 #define DOCTEST\_SET\_ATTR(x) SetConsoleTextAttribute(ch.stdoutHandle, x | ch.origBgAttrs)

4445

4446 // clang-format off

4447 switch (code) {

4448 case Color::White: DOCTEST\_SET\_ATTR(FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_BLUE); break;

4449 case Color::Red: DOCTEST\_SET\_ATTR(FOREGROUND\_RED); break;

4450 case Color::Green: DOCTEST\_SET\_ATTR(FOREGROUND\_GREEN); break;

4451 case Color::Blue: DOCTEST\_SET\_ATTR(FOREGROUND\_BLUE); break;

4452 case Color::Cyan: DOCTEST\_SET\_ATTR(FOREGROUND\_BLUE | FOREGROUND\_GREEN); break;

4453 case Color::Yellow: DOCTEST\_SET\_ATTR(FOREGROUND\_RED | FOREGROUND\_GREEN); break;

4454 case Color::Grey: DOCTEST\_SET\_ATTR(0); break;

4455 case Color::LightGrey: DOCTEST\_SET\_ATTR(FOREGROUND\_INTENSITY); break;

4456 case Color::BrightRed: DOCTEST\_SET\_ATTR(FOREGROUND\_INTENSITY | FOREGROUND\_RED); break;

4457 case Color::BrightGreen: DOCTEST\_SET\_ATTR(FOREGROUND\_INTENSITY | FOREGROUND\_GREEN); break;

4458 case Color::BrightWhite: DOCTEST\_SET\_ATTR(FOREGROUND\_INTENSITY | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_BLUE); break;

4459 case Color::None:

4460 case Color::Bright: // invalid

4461 default: DOCTEST\_SET\_ATTR(ch.origFgAttrs);

4462 }

4463 // clang-format on

4464 #endif // DOCTEST\_CONFIG\_COLORS\_WINDOWS

4465 }

4466 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

4467

4468 std::vector<const IExceptionTranslator\*>& getExceptionTranslators() {

4469 static std::vector<const IExceptionTranslator\*> data;

4470 return data;

4471 }

4472

4473 String translateActiveException() {

4474 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

4475 String res;

4476 auto& translators = getExceptionTranslators();

4477 for(auto& curr : translators)

4478 if(curr->translate(res))

4479 return res;

4480 // clang-format off

4481 DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wcatch-value")

4482 try {

4483 throw;

4484 } catch(std::exception& ex) {

4485 return ex.what();

4486 } catch(std::string& msg) {

4487 return msg.c\_str();

4488 } catch(const char\* msg) {

4489 return msg;

4490 } catch(...) {

4491 return "unknown exception";

4492 }

4493 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

4494 // clang-format on

4495 #else // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

4496 return "";

4497 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

4498 }

4499 } // namespace

4500

4501 namespace detail {

4502 // used by the macros for registering tests

4503 int regTest(const TestCase& tc) {

4504 getRegisteredTests().insert(tc);

4505 return 0;

4506 }

4507

4508 // sets the current test suite

4509 int setTestSuite(const TestSuite& ts) {

4510 doctest\_detail\_test\_suite\_ns::getCurrentTestSuite() = ts;

4511 return 0;

4512 }

4513

4514 #ifdef DOCTEST\_IS\_DEBUGGER\_ACTIVE

4515 bool isDebuggerActive() { return DOCTEST\_IS\_DEBUGGER\_ACTIVE(); }

4516 #else // DOCTEST\_IS\_DEBUGGER\_ACTIVE

4517 #ifdef DOCTEST\_PLATFORM\_LINUX

4518 class ErrnoGuard {

4519 public:

4520 ErrnoGuard() : m\_oldErrno(errno) {}

4521 ~ErrnoGuard() { errno = m\_oldErrno; }

4522 private:

4523 int m\_oldErrno;

4524 };

4525 // See the comments in Catch2 for the reasoning behind this implementation:

4526 // https://github.com/catchorg/Catch2/blob/v2.13.1/include/internal/catch\_debugger.cpp#L79-L102

4527 bool isDebuggerActive() {

4528 ErrnoGuard guard;

4529 std::ifstream in("/proc/self/status");

4530 for(std::string line; std::getline(in, line);) {

4531 static const int PREFIX\_LEN = 11;

4532 if(line.compare(0, PREFIX\_LEN, "TracerPid:\t") == 0) {

4533 return line.length() > PREFIX\_LEN && line[PREFIX\_LEN] != '0';

4534 }

4535 }

4536 return false;

4537 }

4538 #elif defined(DOCTEST\_PLATFORM\_MAC)

4539 // The following function is taken directly from the following technical note:

4540 // https://developer.apple.com/library/archive/qa/qa1361/\_index.html

4541 // Returns true if the current process is being debugged (either

4542 // running under the debugger or has a debugger attached post facto).

4543 bool isDebuggerActive() {

4544 int mib[4];

4545 kinfo\_proc info;

4546 size\_t size;

4547 // Initialize the flags so that, if sysctl fails for some bizarre

4548 // reason, we get a predictable result.

4549 info.kp\_proc.p\_flag = 0;

4550 // Initialize mib, which tells sysctl the info we want, in this case

4551 // we're looking for information about a specific process ID.

4552 mib[0] = CTL\_KERN;

4553 mib[1] = KERN\_PROC;

4554 mib[2] = KERN\_PROC\_PID;

4555 mib[3] = getpid();

4556 // Call sysctl.

4557 size = sizeof(info);

4558 if(sysctl(mib, DOCTEST\_COUNTOF(mib), &info, &size, 0, 0) != 0) {

4559 std::cerr << "\nCall to sysctl failed - unable to determine if debugger is active \*\*\n";

4560 return false;

4561 }

4562 // We're being debugged if the P\_TRACED flag is set.

4563 return ((info.kp\_proc.p\_flag & P\_TRACED) != 0);

4564 }

4565 #elif DOCTEST\_MSVC || defined(\_\_MINGW32\_\_) || defined(\_\_MINGW64\_\_)

4566 bool isDebuggerActive() { return ::IsDebuggerPresent() != 0; }

4567 #else

4568 bool isDebuggerActive() { return false; }

4569 #endif // Platform

4570 #endif // DOCTEST\_IS\_DEBUGGER\_ACTIVE

4571

4572 void registerExceptionTranslatorImpl(const IExceptionTranslator\* et) {

4573 if(std::find(getExceptionTranslators().begin(), getExceptionTranslators().end(), et) ==

4574 getExceptionTranslators().end())

4575 getExceptionTranslators().push\_back(et);

4576 }

4577

4578 DOCTEST\_THREAD\_LOCAL std::vector<IContextScope\*> g\_infoContexts; // for logging with INFO()

4579

4580 ContextScopeBase::ContextScopeBase() {

4581 g\_infoContexts.push\_back(this);

4582 }

4583

4584 ContextScopeBase::ContextScopeBase(ContextScopeBase&& other) noexcept {

4585 if (other.need\_to\_destroy) {

4586 other.destroy();

4587 }

4588 other.need\_to\_destroy = false;

4589 g\_infoContexts.push\_back(this);

4590 }

4591

4592 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4996) // std::uncaught\_exception is deprecated in C++17

4593 DOCTEST\_GCC\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wdeprecated-declarations")

4594 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_WITH\_PUSH("-Wdeprecated-declarations")

4595

4596 // destroy cannot be inlined into the destructor because that would mean calling stringify after

4597 // ContextScope has been destroyed (base class destructors run after derived class destructors).

4598 // Instead, ContextScope calls this method directly from its destructor.

4599 void ContextScopeBase::destroy() {

4600 #if defined(\_\_cpp\_lib\_uncaught\_exceptions) && \_\_cpp\_lib\_uncaught\_exceptions >= 201411L && (!defined(\_\_MAC\_OS\_X\_VERSION\_MIN\_REQUIRED) || \_\_MAC\_OS\_X\_VERSION\_MIN\_REQUIRED >= 101200)

4601 if(std::uncaught\_exceptions() > 0) {

4602 #else

4603 if(std::uncaught\_exception()) {

4604 #endif

4605 std::ostringstream s;

4606 this->stringify(&s);

4607 g\_cs->stringifiedContexts.push\_back(s.str().c\_str());

4608 }

4609 g\_infoContexts.pop\_back();

4610 }

4611

4612 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

4613 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

4614 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

4615 } // namespace detail

4616 namespace {

4617 using namespace detail;

4618

4619 #if !defined(DOCTEST\_CONFIG\_POSIX\_SIGNALS) && !defined(DOCTEST\_CONFIG\_WINDOWS\_SEH)

4620 struct FatalConditionHandler

4621 {

4622 static void reset() {}

4623 static void allocateAltStackMem() {}

4624 static void freeAltStackMem() {}

4625 };

4626 #else // DOCTEST\_CONFIG\_POSIX\_SIGNALS || DOCTEST\_CONFIG\_WINDOWS\_SEH

4627

4628 void reportFatal(const std::string&);

4629

4630 #ifdef DOCTEST\_PLATFORM\_WINDOWS

4631

4632 struct SignalDefs

4633 {

4634 DWORD id;

4635 const char\* name;

4636 };

4637 // There is no 1-1 mapping between signals and windows exceptions.

4638 // Windows can easily distinguish between SO and SigSegV,

4639 // but SigInt, SigTerm, etc are handled differently.

4640 SignalDefs signalDefs[] = {

4641 {static\_cast<DWORD>(EXCEPTION\_ILLEGAL\_INSTRUCTION),

4642 "SIGILL - Illegal instruction signal"},

4643 {static\_cast<DWORD>(EXCEPTION\_STACK\_OVERFLOW), "SIGSEGV - Stack overflow"},

4644 {static\_cast<DWORD>(EXCEPTION\_ACCESS\_VIOLATION),

4645 "SIGSEGV - Segmentation violation signal"},

4646 {static\_cast<DWORD>(EXCEPTION\_INT\_DIVIDE\_BY\_ZERO), "Divide by zero error"},

4647 };

4648

4649 struct FatalConditionHandler

4650 {

4651 static LONG CALLBACK handleException(PEXCEPTION\_POINTERS ExceptionInfo) {

4652 // Multiple threads may enter this filter/handler at once. We want the error message to be printed on the

4653 // console just once no matter how many threads have crashed.

4654 DOCTEST\_DECLARE\_STATIC\_MUTEX(mutex)

4655 static bool execute = true;

4656 {

4657 DOCTEST\_LOCK\_MUTEX(mutex)

4658 if(execute) {

4659 bool reported = false;

4660 for(size\_t i = 0; i < DOCTEST\_COUNTOF(signalDefs); ++i) {

4661 if(ExceptionInfo->ExceptionRecord->ExceptionCode == signalDefs[i].id) {

4662 reportFatal(signalDefs[i].name);

4663 reported = true;

4664 break;

4665 }

4666 }

4667 if(reported == false)

4668 reportFatal("Unhandled SEH exception caught");

4669 if(isDebuggerActive() && !g\_cs->no\_breaks)

4670 DOCTEST\_BREAK\_INTO\_DEBUGGER();

4671 }

4672 execute = false;

4673 }

4674 std::exit(EXIT\_FAILURE);

4675 }

4676

4677 static void allocateAltStackMem() {}

4678 static void freeAltStackMem() {}

4679

4680 FatalConditionHandler() {

4681 isSet = true;

4682 // 32k seems enough for doctest to handle stack overflow,

4683 // but the value was found experimentally, so there is no strong guarantee

4684 guaranteeSize = 32 \* 1024;

4685 // Register an unhandled exception filter

4686 previousTop = SetUnhandledExceptionFilter(handleException);

4687 // Pass in guarantee size to be filled

4688 SetThreadStackGuarantee(&guaranteeSize);

4689

4690 // On Windows uncaught exceptions from another thread, exceptions from

4691 // destructors, or calls to std::terminate are not a SEH exception

4692

4693 // The terminal handler gets called when:

4694 // - std::terminate is called FROM THE TEST RUNNER THREAD

4695 // - an exception is thrown from a destructor FROM THE TEST RUNNER THREAD

4696 original\_terminate\_handler = std::get\_terminate();

4697 std::set\_terminate([]() DOCTEST\_NOEXCEPT {

4698 reportFatal("Terminate handler called");

4699 if(isDebuggerActive() && !g\_cs->no\_breaks)

4700 DOCTEST\_BREAK\_INTO\_DEBUGGER();

4701 std::exit(EXIT\_FAILURE); // explicitly exit - otherwise the SIGABRT handler may be called as well

4702 });

4703

4704 // SIGABRT is raised when:

4705 // - std::terminate is called FROM A DIFFERENT THREAD

4706 // - an exception is thrown from a destructor FROM A DIFFERENT THREAD

4707 // - an uncaught exception is thrown FROM A DIFFERENT THREAD

4708 prev\_sigabrt\_handler = std::signal(SIGABRT, [](int signal) DOCTEST\_NOEXCEPT {

4709 if(signal == SIGABRT) {

4710 reportFatal("SIGABRT - Abort (abnormal termination) signal");

4711 if(isDebuggerActive() && !g\_cs->no\_breaks)

4712 DOCTEST\_BREAK\_INTO\_DEBUGGER();

4713 std::exit(EXIT\_FAILURE);

4714 }

4715 });

4716

4717 // The following settings are taken from google test, and more

4718 // specifically from UnitTest::Run() inside of gtest.cc

4719

4720 // the user does not want to see pop-up dialogs about crashes

4721 prev\_error\_mode\_1 = SetErrorMode(SEM\_FAILCRITICALERRORS | SEM\_NOALIGNMENTFAULTEXCEPT |

4722 SEM\_NOGPFAULTERRORBOX | SEM\_NOOPENFILEERRORBOX);

4723 // This forces the abort message to go to stderr in all circumstances.

4724 prev\_error\_mode\_2 = \_set\_error\_mode(\_OUT\_TO\_STDERR);

4725 // In the debug version, Visual Studio pops up a separate dialog

4726 // offering a choice to debug the aborted program - we want to disable that.

4727 prev\_abort\_behavior = \_set\_abort\_behavior(0x0, \_WRITE\_ABORT\_MSG | \_CALL\_REPORTFAULT);

4728 // In debug mode, the Windows CRT can crash with an assertion over invalid

4729 // input (e.g. passing an invalid file descriptor). The default handling

4730 // for these assertions is to pop up a dialog and wait for user input.

4731 // Instead ask the CRT to dump such assertions to stderr non-interactively.

4732 prev\_report\_mode = \_CrtSetReportMode(\_CRT\_ASSERT, \_CRTDBG\_MODE\_FILE | \_CRTDBG\_MODE\_DEBUG);

4733 prev\_report\_file = \_CrtSetReportFile(\_CRT\_ASSERT, \_CRTDBG\_FILE\_STDERR);

4734 }

4735

4736 static void reset() {

4737 if(isSet) {

4738 // Unregister handler and restore the old guarantee

4739 SetUnhandledExceptionFilter(previousTop);

4740 SetThreadStackGuarantee(&guaranteeSize);

4741 std::set\_terminate(original\_terminate\_handler);

4742 std::signal(SIGABRT, prev\_sigabrt\_handler);

4743 SetErrorMode(prev\_error\_mode\_1);

4744 \_set\_error\_mode(prev\_error\_mode\_2);

4745 \_set\_abort\_behavior(prev\_abort\_behavior, \_WRITE\_ABORT\_MSG | \_CALL\_REPORTFAULT);

4746 static\_cast<void>(\_CrtSetReportMode(\_CRT\_ASSERT, prev\_report\_mode));

4747 static\_cast<void>(\_CrtSetReportFile(\_CRT\_ASSERT, prev\_report\_file));

4748 isSet = false;

4749 }

4750 }

4751

4752 ~FatalConditionHandler() { reset(); }

4753

4754 private:

4755 static UINT prev\_error\_mode\_1;

4756 static int prev\_error\_mode\_2;

4757 static unsigned int prev\_abort\_behavior;

4758 static int prev\_report\_mode;

4759 static \_HFILE prev\_report\_file;

4760 static void (DOCTEST\_CDECL \*prev\_sigabrt\_handler)(int);

4761 static std::terminate\_handler original\_terminate\_handler;

4762 static bool isSet;

4763 static ULONG guaranteeSize;

4764 static LPTOP\_LEVEL\_EXCEPTION\_FILTER previousTop;

4765 };

4766

4767 UINT FatalConditionHandler::prev\_error\_mode\_1;

4768 int FatalConditionHandler::prev\_error\_mode\_2;

4769 unsigned int FatalConditionHandler::prev\_abort\_behavior;

4770 int FatalConditionHandler::prev\_report\_mode;

4771 \_HFILE FatalConditionHandler::prev\_report\_file;

4772 void (DOCTEST\_CDECL \*FatalConditionHandler::prev\_sigabrt\_handler)(int);

4773 std::terminate\_handler FatalConditionHandler::original\_terminate\_handler;

4774 bool FatalConditionHandler::isSet = false;

4775 ULONG FatalConditionHandler::guaranteeSize = 0;

4776 LPTOP\_LEVEL\_EXCEPTION\_FILTER FatalConditionHandler::previousTop = nullptr;

4777

4778 #else // DOCTEST\_PLATFORM\_WINDOWS

4779

4780 struct SignalDefs

4781 {

4782 int id;

4783 const char\* name;

4784 };

4785 SignalDefs signalDefs[] = {{SIGINT, "SIGINT - Terminal interrupt signal"},

4786 {SIGILL, "SIGILL - Illegal instruction signal"},

4787 {SIGFPE, "SIGFPE - Floating point error signal"},

4788 {SIGSEGV, "SIGSEGV - Segmentation violation signal"},

4789 {SIGTERM, "SIGTERM - Termination request signal"},

4790 {SIGABRT, "SIGABRT - Abort (abnormal termination) signal"}};

4791

4792 struct FatalConditionHandler

4793 {

4794 static bool isSet;

4795 static struct sigaction oldSigActions[DOCTEST\_COUNTOF(signalDefs)];

4796 static stack\_t oldSigStack;

4797 static size\_t altStackSize;

4798 static char\* altStackMem;

4799

4800 static void handleSignal(int sig) {

4801 const char\* name = "<unknown signal>";

4802 for(std::size\_t i = 0; i < DOCTEST\_COUNTOF(signalDefs); ++i) {

4803 SignalDefs& def = signalDefs[i];

4804 if(sig == def.id) {

4805 name = def.name;

4806 break;

4807 }

4808 }

4809 reset();

4810 reportFatal(name);

4811 raise(sig);

4812 }

4813

4814 static void allocateAltStackMem() {

4815 altStackMem = new char[altStackSize];

4816 }

4817

4818 static void freeAltStackMem() {

4819 delete[] altStackMem;

4820 }

4821

4822 FatalConditionHandler() {

4823 isSet = true;

4824 stack\_t sigStack;

4825 sigStack.ss\_sp = altStackMem;

4826 sigStack.ss\_size = altStackSize;

4827 sigStack.ss\_flags = 0;

4828 sigaltstack(&sigStack, &oldSigStack);

4829 struct sigaction sa = {};

4830 sa.sa\_handler = handleSignal;

4831 sa.sa\_flags = SA\_ONSTACK;

4832 for(std::size\_t i = 0; i < DOCTEST\_COUNTOF(signalDefs); ++i) {

4833 sigaction(signalDefs[i].id, &sa, &oldSigActions[i]);

4834 }

4835 }

4836

4837 ~FatalConditionHandler() { reset(); }

4838 static void reset() {

4839 if(isSet) {

4840 // Set signals back to previous values -- hopefully nobody overwrote them in the meantime

4841 for(std::size\_t i = 0; i < DOCTEST\_COUNTOF(signalDefs); ++i) {

4842 sigaction(signalDefs[i].id, &oldSigActions[i], nullptr);

4843 }

4844 // Return the old stack

4845 sigaltstack(&oldSigStack, nullptr);

4846 isSet = false;

4847 }

4848 }

4849 };

4850

4851 bool FatalConditionHandler::isSet = false;

4852 struct sigaction FatalConditionHandler::oldSigActions[DOCTEST\_COUNTOF(signalDefs)] = {};

4853 stack\_t FatalConditionHandler::oldSigStack = {};

4854 size\_t FatalConditionHandler::altStackSize = 4 \* SIGSTKSZ;

4855 char\* FatalConditionHandler::altStackMem = nullptr;

4856

4857 #endif // DOCTEST\_PLATFORM\_WINDOWS

4858 #endif // DOCTEST\_CONFIG\_POSIX\_SIGNALS || DOCTEST\_CONFIG\_WINDOWS\_SEH

4859

4860 } // namespace

4861

4862 namespace {

4863 using namespace detail;

4864

4865 #ifdef DOCTEST\_PLATFORM\_WINDOWS

4866 #define DOCTEST\_OUTPUT\_DEBUG\_STRING(text) ::OutputDebugStringA(text)

4867 #else

4868 // TODO: integration with XCode and other IDEs

4869 #define DOCTEST\_OUTPUT\_DEBUG\_STRING(text)

4870 #endif // Platform

4871

4872 void addAssert(assertType::Enum at) {

4873 if((at & assertType::is\_warn) == 0)

4874 g\_cs->numAssertsCurrentTest\_atomic++;

4875 }

4876

4877 void addFailedAssert(assertType::Enum at) {

4878 if((at & assertType::is\_warn) == 0)

4879 g\_cs->numAssertsFailedCurrentTest\_atomic++;

4880 }

4881

4882 #if defined(DOCTEST\_CONFIG\_POSIX\_SIGNALS) || defined(DOCTEST\_CONFIG\_WINDOWS\_SEH)

4883 void reportFatal(const std::string& message) {

4884 g\_cs->failure\_flags |= TestCaseFailureReason::Crash;

4885

4886 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_case\_exception, {message.c\_str(), true});

4887

4888 while (g\_cs->subcaseStack.size()) {

4889 g\_cs->subcaseStack.pop\_back();

4890 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(subcase\_end, DOCTEST\_EMPTY);

4891 }

4892

4893 g\_cs->finalizeTestCaseData();

4894

4895 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_case\_end, \*g\_cs);

4896

4897 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_run\_end, \*g\_cs);

4898 }

4899 #endif // DOCTEST\_CONFIG\_POSIX\_SIGNALS || DOCTEST\_CONFIG\_WINDOWS\_SEH

4900 } // namespace

4901

4902 AssertData::AssertData(assertType::Enum at, const char\* file, int line, const char\* expr,

4903 const char\* exception\_type, const StringContains& exception\_string)

4904 : m\_test\_case(g\_cs->currentTest), m\_at(at), m\_file(file), m\_line(line), m\_expr(expr),

4905 m\_failed(true), m\_threw(false), m\_threw\_as(false), m\_exception\_type(exception\_type),

4906 m\_exception\_string(exception\_string) {

4907 #if DOCTEST\_MSVC

4908 if (m\_expr[0] == ' ') // this happens when variadic macros are disabled under MSVC

4909 ++m\_expr;

4910 #endif // MSVC

4911 }

4912

4913 namespace detail {

4914 ResultBuilder::ResultBuilder(assertType::Enum at, const char\* file, int line, const char\* expr,

4915 const char\* exception\_type, const String& exception\_string)

4916 : AssertData(at, file, line, expr, exception\_type, exception\_string) { }

4917

4918 ResultBuilder::ResultBuilder(assertType::Enum at, const char\* file, int line, const char\* expr,

4919 const char\* exception\_type, const Contains& exception\_string)

4920 : AssertData(at, file, line, expr, exception\_type, exception\_string) { }

4921

4922 void ResultBuilder::setResult(const Result& res) {

4923 m\_decomp = res.m\_decomp;

4924 m\_failed = !res.m\_passed;

4925 }

4926

4927 void ResultBuilder::translateException() {

4928 m\_threw = true;

4929 m\_exception = translateActiveException();

4930 }

4931

4932 bool ResultBuilder::log() {

4933 if(m\_at & assertType::is\_throws) {

4934 m\_failed = !m\_threw;

4935 } else if((m\_at & assertType::is\_throws\_as) && (m\_at & assertType::is\_throws\_with)) {

4936 m\_failed = !m\_threw\_as || !m\_exception\_string.check(m\_exception);

4937 } else if(m\_at & assertType::is\_throws\_as) {

4938 m\_failed = !m\_threw\_as;

4939 } else if(m\_at & assertType::is\_throws\_with) {

4940 m\_failed = !m\_exception\_string.check(m\_exception);

4941 } else if(m\_at & assertType::is\_nothrow) {

4942 m\_failed = m\_threw;

4943 }

4944

4945 if(m\_exception.size())

4946 m\_exception = "\"" + m\_exception + "\"";

4947

4948 if(is\_running\_in\_test) {

4949 addAssert(m\_at);

4950 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(log\_assert, \*this);

4951

4952 if(m\_failed)

4953 addFailedAssert(m\_at);

4954 } else if(m\_failed) {

4955 failed\_out\_of\_a\_testing\_context(\*this);

4956 }

4957

4958 return m\_failed && isDebuggerActive() && !getContextOptions()->no\_breaks &&

4959 (g\_cs->currentTest == nullptr || !g\_cs->currentTest->m\_no\_breaks); // break into debugger

4960 }

4961

4962 void ResultBuilder::react() const {

4963 if(m\_failed && checkIfShouldThrow(m\_at))

4964 throwException();

4965 }

4966

4967 void failed\_out\_of\_a\_testing\_context(const AssertData& ad) {

4968 if(g\_cs->ah)

4969 g\_cs->ah(ad);

4970 else

4971 std::abort();

4972 }

4973

4974 bool decomp\_assert(assertType::Enum at, const char\* file, int line, const char\* expr,

4975 const Result& result) {

4976 bool failed = !result.m\_passed;

4977

4978 // ###################################################################################

4979 // IF THE DEBUGGER BREAKS HERE - GO 1 LEVEL UP IN THE CALLSTACK FOR THE FAILING ASSERT

4980 // THIS IS THE EFFECT OF HAVING 'DOCTEST\_CONFIG\_SUPER\_FAST\_ASSERTS' DEFINED

4981 // ###################################################################################

4982 DOCTEST\_ASSERT\_OUT\_OF\_TESTS(result.m\_decomp);

4983 DOCTEST\_ASSERT\_IN\_TESTS(result.m\_decomp);

4984 return !failed;

4985 }

4986

4987 MessageBuilder::MessageBuilder(const char\* file, int line, assertType::Enum severity) {

4988 m\_stream = tlssPush();

4989 m\_file = file;

4990 m\_line = line;

4991 m\_severity = severity;

4992 }

4993

4994 MessageBuilder::~MessageBuilder() {

4995 if (!logged)

4996 tlssPop();

4997 }

4998

4999 DOCTEST\_DEFINE\_INTERFACE(IExceptionTranslator)

5000

5001 bool MessageBuilder::log() {

5002 if (!logged) {

5003 m\_string = tlssPop();

5004 logged = true;

5005 }

5006

5007 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(log\_message, \*this);

5008

5009 const bool isWarn = m\_severity & assertType::is\_warn;

5010

5011 // warn is just a message in this context so we don't treat it as an assert

5012 if(!isWarn) {

5013 addAssert(m\_severity);

5014 addFailedAssert(m\_severity);

5015 }

5016

5017 return isDebuggerActive() && !getContextOptions()->no\_breaks && !isWarn &&

5018 (g\_cs->currentTest == nullptr || !g\_cs->currentTest->m\_no\_breaks); // break into debugger

5019 }

5020

5021 void MessageBuilder::react() {

5022 if(m\_severity & assertType::is\_require)

5023 throwException();

5024 }

5025 } // namespace detail

5026 namespace {

5027 using namespace detail;

5028

5029 // clang-format off

5030

5031 // =================================================================================================

5032 // The following code has been taken verbatim from Catch2/include/internal/catch\_xmlwriter.h/cpp

5033 // This is done so cherry-picking bug fixes is trivial - even the style/formatting is untouched.

5034 // =================================================================================================

5035

5036 class XmlEncode {

5037 public:

5038 enum ForWhat { ForTextNodes, ForAttributes };

5039

5040 XmlEncode( std::string const& str, ForWhat forWhat = ForTextNodes );

5041

5042 void encodeTo( std::ostream& os ) const;

5043

5044 friend std::ostream& operator << ( std::ostream& os, XmlEncode const& xmlEncode );

5045

5046 private:

5047 std::string m\_str;

5048 ForWhat m\_forWhat;

5049 };

5050

5051 class XmlWriter {

5052 public:

5053

5054 class ScopedElement {

5055 public:

5056 ScopedElement( XmlWriter\* writer );

5057

5058 ScopedElement( ScopedElement&& other ) DOCTEST\_NOEXCEPT;

5059 ScopedElement& operator=( ScopedElement&& other ) DOCTEST\_NOEXCEPT;

5060

5061 ~ScopedElement();

5062

5063 ScopedElement& writeText( std::string const& text, bool indent = true );

5064

5065 template<typename T>

5066 ScopedElement& writeAttribute( std::string const& name, T const& attribute ) {

5067 m\_writer->writeAttribute( name, attribute );

5068 return \*this;

5069 }

5070

5071 private:

5072 mutable XmlWriter\* m\_writer = nullptr;

5073 };

5074

5075 #ifndef DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

5076 XmlWriter( std::ostream& os = std::cout );

5077 #else // DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

5078 XmlWriter( std::ostream& os );

5079 #endif // DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

5080 ~XmlWriter();

5081

5082 XmlWriter( XmlWriter const& ) = delete;

5083 XmlWriter& operator=( XmlWriter const& ) = delete;

5084

5085 XmlWriter& startElement( std::string const& name );

5086

5087 ScopedElement scopedElement( std::string const& name );

5088

5089 XmlWriter& endElement();

5090

5091 XmlWriter& writeAttribute( std::string const& name, std::string const& attribute );

5092

5093 XmlWriter& writeAttribute( std::string const& name, const char\* attribute );

5094

5095 XmlWriter& writeAttribute( std::string const& name, bool attribute );

5096

5097 template<typename T>

5098 XmlWriter& writeAttribute( std::string const& name, T const& attribute ) {

5099 std::stringstream rss;

5100 rss << attribute;

5101 return writeAttribute( name, rss.str() );

5102 }

5103

5104 XmlWriter& writeText( std::string const& text, bool indent = true );

5105

5106 //XmlWriter& writeComment( std::string const& text );

5107

5108 //void writeStylesheetRef( std::string const& url );

5109

5110 //XmlWriter& writeBlankLine();

5111

5112 void ensureTagClosed();

5113

5114 void writeDeclaration();

5115

5116 private:

5117

5118 void newlineIfNecessary();

5119

5120 bool m\_tagIsOpen = false;

5121 bool m\_needsNewline = false;

5122 std::vector<std::string> m\_tags;

5123 std::string m\_indent;

5124 std::ostream& m\_os;

5125 };

5126

5127 // =================================================================================================

5128 // The following code has been taken verbatim from Catch2/include/internal/catch\_xmlwriter.h/cpp

5129 // This is done so cherry-picking bug fixes is trivial - even the style/formatting is untouched.

5130 // =================================================================================================

5131

5132 using uchar = unsigned char;

5133

5134 namespace {

5135

5136 size\_t trailingBytes(unsigned char c) {

5137 if ((c & 0xE0) == 0xC0) {

5138 return 2;

5139 }

5140 if ((c & 0xF0) == 0xE0) {

5141 return 3;

5142 }

5143 if ((c & 0xF8) == 0xF0) {

5144 return 4;

5145 }

5146 DOCTEST\_INTERNAL\_ERROR("Invalid multibyte utf-8 start byte encountered");

5147 }

5148

5149 uint32\_t headerValue(unsigned char c) {

5150 if ((c & 0xE0) == 0xC0) {

5151 return c & 0x1F;

5152 }

5153 if ((c & 0xF0) == 0xE0) {

5154 return c & 0x0F;

5155 }

5156 if ((c & 0xF8) == 0xF0) {

5157 return c & 0x07;

5158 }

5159 DOCTEST\_INTERNAL\_ERROR("Invalid multibyte utf-8 start byte encountered");

5160 }

5161

5162 void hexEscapeChar(std::ostream& os, unsigned char c) {

5163 std::ios\_base::fmtflags f(os.flags());

5164 os << "\\x"

5165 << std::uppercase << std::hex << std::setfill('0') << std::setw(2)

5166 << static\_cast<int>(c);

5167 os.flags(f);

5168 }

5169

5170 } // anonymous namespace

5171

5172 XmlEncode::XmlEncode( std::string const& str, ForWhat forWhat )

5173 : m\_str( str ),

5174 m\_forWhat( forWhat )

5175 {}

5176

5177 void XmlEncode::encodeTo( std::ostream& os ) const {

5178 // Apostrophe escaping not necessary if we always use " to write attributes

5179 // (see: https://www.w3.org/TR/xml/#syntax)

5180

5181 for( std::size\_t idx = 0; idx < m\_str.size(); ++ idx ) {

5182 uchar c = m\_str[idx];

5183 switch (c) {

5184 case '<': os << "&lt;"; break;

5185 case '&': os << "&amp;"; break;

5186

5187 case '>':

5188 // See: https://www.w3.org/TR/xml/#syntax

5189 if (idx > 2 && m\_str[idx - 1] == ']' && m\_str[idx - 2] == ']')

5190 os << "&gt;";

5191 else

5192 os << c;

5193 break;

5194

5195 case '\"':

5196 if (m\_forWhat == ForAttributes)

5197 os << "&quot;";

5198 else

5199 os << c;

5200 break;

5201

5202 default:

5203 // Check for control characters and invalid utf-8

5204

5205 // Escape control characters in standard ascii

5206 // see https://stackoverflow.com/questions/404107/why-are-control-characters-illegal-in-xml-1-0

5207 if (c < 0x09 || (c > 0x0D && c < 0x20) || c == 0x7F) {

5208 hexEscapeChar(os, c);

5209 break;

5210 }

5211

5212 // Plain ASCII: Write it to stream

5213 if (c < 0x7F) {

5214 os << c;

5215 break;

5216 }

5217

5218 // UTF-8 territory

5219 // Check if the encoding is valid and if it is not, hex escape bytes.

5220 // Important: We do not check the exact decoded values for validity, only the encoding format

5221 // First check that this bytes is a valid lead byte:

5222 // This means that it is not encoded as 1111 1XXX

5223 // Or as 10XX XXXX

5224 if (c < 0xC0 ||

5225 c >= 0xF8) {

5226 hexEscapeChar(os, c);

5227 break;

5228 }

5229

5230 auto encBytes = trailingBytes(c);

5231 // Are there enough bytes left to avoid accessing out-of-bounds memory?

5232 if (idx + encBytes - 1 >= m\_str.size()) {

5233 hexEscapeChar(os, c);

5234 break;

5235 }

5236 // The header is valid, check data

5237 // The next encBytes bytes must together be a valid utf-8

5238 // This means: bitpattern 10XX XXXX and the extracted value is sane (ish)

5239 bool valid = true;

5240 uint32\_t value = headerValue(c);

5241 for (std::size\_t n = 1; n < encBytes; ++n) {

5242 uchar nc = m\_str[idx + n];

5243 valid &= ((nc & 0xC0) == 0x80);

5244 value = (value << 6) | (nc & 0x3F);

5245 }

5246

5247 if (

5248 // Wrong bit pattern of following bytes

5249 (!valid) ||

5250 // Overlong encodings

5251 (value < 0x80) ||

5252 ( value < 0x800 && encBytes > 2) || // removed "0x80 <= value &&" because redundant

5253 (0x800 < value && value < 0x10000 && encBytes > 3) ||

5254 // Encoded value out of range

5255 (value >= 0x110000)

5256 ) {

5257 hexEscapeChar(os, c);

5258 break;

5259 }

5260

5261 // If we got here, this is in fact a valid(ish) utf-8 sequence

5262 for (std::size\_t n = 0; n < encBytes; ++n) {

5263 os << m\_str[idx + n];

5264 }

5265 idx += encBytes - 1;

5266 break;

5267 }

5268 }

5269 }

5270

5271 std::ostream& operator << ( std::ostream& os, XmlEncode const& xmlEncode ) {

5272 xmlEncode.encodeTo( os );

5273 return os;

5274 }

5275

5276 XmlWriter::ScopedElement::ScopedElement( XmlWriter\* writer )

5277 : m\_writer( writer )

5278 {}

5279

5280 XmlWriter::ScopedElement::ScopedElement( ScopedElement&& other ) DOCTEST\_NOEXCEPT

5281 : m\_writer( other.m\_writer ){

5282 other.m\_writer = nullptr;

5283 }

5284 XmlWriter::ScopedElement& XmlWriter::ScopedElement::operator=( ScopedElement&& other ) DOCTEST\_NOEXCEPT {

5285 if ( m\_writer ) {

5286 m\_writer->endElement();

5287 }

5288 m\_writer = other.m\_writer;

5289 other.m\_writer = nullptr;

5290 return \*this;

5291 }

5292

5293

5294 XmlWriter::ScopedElement::~ScopedElement() {

5295 if( m\_writer )

5296 m\_writer->endElement();

5297 }

5298

5299 XmlWriter::ScopedElement& XmlWriter::ScopedElement::writeText( std::string const& text, bool indent ) {

5300 m\_writer->writeText( text, indent );

5301 return \*this;

5302 }

5303

5304 XmlWriter::XmlWriter( std::ostream& os ) : m\_os( os )

5305 {

5306 // writeDeclaration(); // called explicitly by the reporters that use the writer class - see issue #627

5307 }

5308

5309 XmlWriter::~XmlWriter() {

5310 while( !m\_tags.empty() )

5311 endElement();

5312 }

5313

5314 XmlWriter& XmlWriter::startElement( std::string const& name ) {

5315 ensureTagClosed();

5316 newlineIfNecessary();

5317 m\_os << m\_indent << '<' << name;

5318 m\_tags.push\_back( name );

5319 m\_indent += " ";

5320 m\_tagIsOpen = true;

5321 return \*this;

5322 }

5323

5324 XmlWriter::ScopedElement XmlWriter::scopedElement( std::string const& name ) {

5325 ScopedElement scoped( this );

5326 startElement( name );

5327 return scoped;

5328 }

5329

5330 XmlWriter& XmlWriter::endElement() {

5331 newlineIfNecessary();

5332 m\_indent = m\_indent.substr( 0, m\_indent.size()-2 );

5333 if( m\_tagIsOpen ) {

5334 m\_os << "/>";

5335 m\_tagIsOpen = false;

5336 }

5337 else {

5338 m\_os << m\_indent << "</" << m\_tags.back() << ">";

5339 }

5340 m\_os << std::endl;

5341 m\_tags.pop\_back();

5342 return \*this;

5343 }

5344

5345 XmlWriter& XmlWriter::writeAttribute( std::string const& name, std::string const& attribute ) {

5346 if( !name.empty() && !attribute.empty() )

5347 m\_os << ' ' << name << "=\"" << XmlEncode( attribute, XmlEncode::ForAttributes ) << '"';

5348 return \*this;

5349 }

5350

5351 XmlWriter& XmlWriter::writeAttribute( std::string const& name, const char\* attribute ) {

5352 if( !name.empty() && attribute && attribute[0] != '\0' )

5353 m\_os << ' ' << name << "=\"" << XmlEncode( attribute, XmlEncode::ForAttributes ) << '"';

5354 return \*this;

5355 }

5356

5357 XmlWriter& XmlWriter::writeAttribute( std::string const& name, bool attribute ) {

5358 m\_os << ' ' << name << "=\"" << ( attribute ? "true" : "false" ) << '"';

5359 return \*this;

5360 }

5361

5362 XmlWriter& XmlWriter::writeText( std::string const& text, bool indent ) {

5363 if( !text.empty() ){

5364 bool tagWasOpen = m\_tagIsOpen;

5365 ensureTagClosed();

5366 if( tagWasOpen && indent )

5367 m\_os << m\_indent;

5368 m\_os << XmlEncode( text );

5369 m\_needsNewline = true;

5370 }

5371 return \*this;

5372 }

5373

5374 //XmlWriter& XmlWriter::writeComment( std::string const& text ) {

5375 // ensureTagClosed();

5376 // m\_os << m\_indent << "<!--" << text << "-->";

5377 // m\_needsNewline = true;

5378 // return \*this;

5379 //}

5380

5381 //void XmlWriter::writeStylesheetRef( std::string const& url ) {

5382 // m\_os << "<?xml-stylesheet type=\"text/xsl\" href=\"" << url << "\"?>\n";

5383 //}

5384

5385 //XmlWriter& XmlWriter::writeBlankLine() {

5386 // ensureTagClosed();

5387 // m\_os << '\n';

5388 // return \*this;

5389 //}

5390

5391 void XmlWriter::ensureTagClosed() {

5392 if( m\_tagIsOpen ) {

5393 m\_os << ">" << std::endl;

5394 m\_tagIsOpen = false;

5395 }

5396 }

5397

5398 void XmlWriter::writeDeclaration() {

5399 m\_os << "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";

5400 }

5401

5402 void XmlWriter::newlineIfNecessary() {

5403 if( m\_needsNewline ) {

5404 m\_os << std::endl;

5405 m\_needsNewline = false;

5406 }

5407 }

5408

5409 // =================================================================================================

5410 // End of copy-pasted code from Catch

5411 // =================================================================================================

5412

5413 // clang-format on

5414

5415 struct XmlReporter : public IReporter

5416 {

5417 XmlWriter xml;

5418 DOCTEST\_DECLARE\_MUTEX(mutex)

5419

5420 // caching pointers/references to objects of these types - safe to do

5421 const ContextOptions& opt;

5422 const TestCaseData\* tc = nullptr;

5423

5424 XmlReporter(const ContextOptions& co)

5425 : xml(\*co.cout)

5426 , opt(co) {}

5427

5428 void log\_contexts() {

5429 int num\_contexts = get\_num\_active\_contexts();

5430 if(num\_contexts) {

5431 auto contexts = get\_active\_contexts();

5432 std::stringstream ss;

5433 for(int i = 0; i < num\_contexts; ++i) {

5434 contexts[i]->stringify(&ss);

5435 xml.scopedElement("Info").writeText(ss.str());

5436 ss.str("");

5437 }

5438 }

5439 }

5440

5441 unsigned line(unsigned l) const { return opt.no\_line\_numbers ? 0 : l; }

5442

5443 void test\_case\_start\_impl(const TestCaseData& in) {

5444 bool open\_ts\_tag = false;

5445 if(tc != nullptr) { // we have already opened a test suite

5446 if(std::strcmp(tc->m\_test\_suite, in.m\_test\_suite) != 0) {

5447 xml.endElement();

5448 open\_ts\_tag = true;

5449 }

5450 }

5451 else {

5452 open\_ts\_tag = true; // first test case ==> first test suite

5453 }

5454

5455 if(open\_ts\_tag) {

5456 xml.startElement("TestSuite");

5457 xml.writeAttribute("name", in.m\_test\_suite);

5458 }

5459

5460 tc = &in;

5461 xml.startElement("TestCase")

5462 .writeAttribute("name", in.m\_name)

5463 .writeAttribute("filename", skipPathFromFilename(in.m\_file.c\_str()))

5464 .writeAttribute("line", line(in.m\_line))

5465 .writeAttribute("description", in.m\_description);

5466

5467 if(Approx(in.m\_timeout) != 0)

5468 xml.writeAttribute("timeout", in.m\_timeout);

5469 if(in.m\_may\_fail)

5470 xml.writeAttribute("may\_fail", true);

5471 if(in.m\_should\_fail)

5472 xml.writeAttribute("should\_fail", true);

5473 }

5474

5475 // =========================================================================================

5476 // WHAT FOLLOWS ARE OVERRIDES OF THE VIRTUAL METHODS OF THE REPORTER INTERFACE

5477 // =========================================================================================

5478

5479 void report\_query(const QueryData& in) override {

5480 test\_run\_start();

5481 if(opt.list\_reporters) {

5482 for(auto& curr : getListeners())

5483 xml.scopedElement("Listener")

5484 .writeAttribute("priority", curr.first.first)

5485 .writeAttribute("name", curr.first.second);

5486 for(auto& curr : getReporters())

5487 xml.scopedElement("Reporter")

5488 .writeAttribute("priority", curr.first.first)

5489 .writeAttribute("name", curr.first.second);

5490 } else if(opt.count || opt.list\_test\_cases) {

5491 for(unsigned i = 0; i < in.num\_data; ++i) {

5492 xml.scopedElement("TestCase").writeAttribute("name", in.data[i]->m\_name)

5493 .writeAttribute("testsuite", in.data[i]->m\_test\_suite)

5494 .writeAttribute("filename", skipPathFromFilename(in.data[i]->m\_file.c\_str()))

5495 .writeAttribute("line", line(in.data[i]->m\_line))

5496 .writeAttribute("skipped", in.data[i]->m\_skip);

5497 }

5498 xml.scopedElement("OverallResultsTestCases")

5499 .writeAttribute("unskipped", in.run\_stats->numTestCasesPassingFilters);

5500 } else if(opt.list\_test\_suites) {

5501 for(unsigned i = 0; i < in.num\_data; ++i)

5502 xml.scopedElement("TestSuite").writeAttribute("name", in.data[i]->m\_test\_suite);

5503 xml.scopedElement("OverallResultsTestCases")

5504 .writeAttribute("unskipped", in.run\_stats->numTestCasesPassingFilters);

5505 xml.scopedElement("OverallResultsTestSuites")

5506 .writeAttribute("unskipped", in.run\_stats->numTestSuitesPassingFilters);

5507 }

5508 xml.endElement();

5509 }

5510

5511 void test\_run\_start() override {

5512 xml.writeDeclaration();

5513

5514 // remove .exe extension - mainly to have the same output on UNIX and Windows

5515 std::string binary\_name = skipPathFromFilename(opt.binary\_name.c\_str());

5516 #ifdef DOCTEST\_PLATFORM\_WINDOWS

5517 if(binary\_name.rfind(".exe") != std::string::npos)

5518 binary\_name = binary\_name.substr(0, binary\_name.length() - 4);

5519 #endif // DOCTEST\_PLATFORM\_WINDOWS

5520

5521 xml.startElement("doctest").writeAttribute("binary", binary\_name);

5522 if(opt.no\_version == false)

5523 xml.writeAttribute("version", DOCTEST\_VERSION\_STR);

5524

5525 // only the consequential ones (TODO: filters)

5526 xml.scopedElement("Options")

5527 .writeAttribute("order\_by", opt.order\_by.c\_str())

5528 .writeAttribute("rand\_seed", opt.rand\_seed)

5529 .writeAttribute("first", opt.first)

5530 .writeAttribute("last", opt.last)

5531 .writeAttribute("abort\_after", opt.abort\_after)

5532 .writeAttribute("subcase\_filter\_levels", opt.subcase\_filter\_levels)

5533 .writeAttribute("case\_sensitive", opt.case\_sensitive)

5534 .writeAttribute("no\_throw", opt.no\_throw)

5535 .writeAttribute("no\_skip", opt.no\_skip);

5536 }

5537

5538 void test\_run\_end(const TestRunStats& p) override {

5539 if(tc) // the TestSuite tag - only if there has been at least 1 test case

5540 xml.endElement();

5541

5542 xml.scopedElement("OverallResultsAsserts")

5543 .writeAttribute("successes", p.numAsserts - p.numAssertsFailed)

5544 .writeAttribute("failures", p.numAssertsFailed);

5545

5546 xml.startElement("OverallResultsTestCases")

5547 .writeAttribute("successes",

5548 p.numTestCasesPassingFilters - p.numTestCasesFailed)

5549 .writeAttribute("failures", p.numTestCasesFailed);

5550 if(opt.no\_skipped\_summary == false)

5551 xml.writeAttribute("skipped", p.numTestCases - p.numTestCasesPassingFilters);

5552 xml.endElement();

5553

5554 xml.endElement();

5555 }

5556

5557 void test\_case\_start(const TestCaseData& in) override {

5558 test\_case\_start\_impl(in);

5559 xml.ensureTagClosed();

5560 }

5561

5562 void test\_case\_reenter(const TestCaseData&) override {}

5563

5564 void test\_case\_end(const CurrentTestCaseStats& st) override {

5565 xml.startElement("OverallResultsAsserts")

5566 .writeAttribute("successes",

5567 st.numAssertsCurrentTest - st.numAssertsFailedCurrentTest)

5568 .writeAttribute("failures", st.numAssertsFailedCurrentTest)

5569 .writeAttribute("test\_case\_success", st.testCaseSuccess);

5570 if(opt.duration)

5571 xml.writeAttribute("duration", st.seconds);

5572 if(tc->m\_expected\_failures)

5573 xml.writeAttribute("expected\_failures", tc->m\_expected\_failures);

5574 xml.endElement();

5575

5576 xml.endElement();

5577 }

5578

5579 void test\_case\_exception(const TestCaseException& e) override {

5580 DOCTEST\_LOCK\_MUTEX(mutex)

5581

5582 xml.scopedElement("Exception")

5583 .writeAttribute("crash", e.is\_crash)

5584 .writeText(e.error\_string.c\_str());

5585 }

5586

5587 void subcase\_start(const SubcaseSignature& in) override {

5588 xml.startElement("SubCase")

5589 .writeAttribute("name", in.m\_name)

5590 .writeAttribute("filename", skipPathFromFilename(in.m\_file))

5591 .writeAttribute("line", line(in.m\_line));

5592 xml.ensureTagClosed();

5593 }

5594

5595 void subcase\_end() override { xml.endElement(); }

5596

5597 void log\_assert(const AssertData& rb) override {

5598 if(!rb.m\_failed && !opt.success)

5599 return;

5600

5601 DOCTEST\_LOCK\_MUTEX(mutex)

5602

5603 xml.startElement("Expression")

5604 .writeAttribute("success", !rb.m\_failed)

5605 .writeAttribute("type", assertString(rb.m\_at))

5606 .writeAttribute("filename", skipPathFromFilename(rb.m\_file))

5607 .writeAttribute("line", line(rb.m\_line));

5608

5609 xml.scopedElement("Original").writeText(rb.m\_expr);

5610

5611 if(rb.m\_threw)

5612 xml.scopedElement("Exception").writeText(rb.m\_exception.c\_str());

5613

5614 if(rb.m\_at & assertType::is\_throws\_as)

5615 xml.scopedElement("ExpectedException").writeText(rb.m\_exception\_type);

5616 if(rb.m\_at & assertType::is\_throws\_with)

5617 xml.scopedElement("ExpectedExceptionString").writeText(rb.m\_exception\_string.c\_str());

5618 if((rb.m\_at & assertType::is\_normal) && !rb.m\_threw)

5619 xml.scopedElement("Expanded").writeText(rb.m\_decomp.c\_str());

5620

5621 log\_contexts();

5622

5623 xml.endElement();

5624 }

5625

5626 void log\_message(const MessageData& mb) override {

5627 DOCTEST\_LOCK\_MUTEX(mutex)

5628

5629 xml.startElement("Message")

5630 .writeAttribute("type", failureString(mb.m\_severity))

5631 .writeAttribute("filename", skipPathFromFilename(mb.m\_file))

5632 .writeAttribute("line", line(mb.m\_line));

5633

5634 xml.scopedElement("Text").writeText(mb.m\_string.c\_str());

5635

5636 log\_contexts();

5637

5638 xml.endElement();

5639 }

5640

5641 void test\_case\_skipped(const TestCaseData& in) override {

5642 if(opt.no\_skipped\_summary == false) {

5643 test\_case\_start\_impl(in);

5644 xml.writeAttribute("skipped", "true");

5645 xml.endElement();

5646 }

5647 }

5648 };

5649

5650 DOCTEST\_REGISTER\_REPORTER("xml", 0, XmlReporter);

5651

5652 void fulltext\_log\_assert\_to\_stream(std::ostream& s, const AssertData& rb) {

5653 if((rb.m\_at & (assertType::is\_throws\_as | assertType::is\_throws\_with)) ==

5654 0)

5655 s << Color::Cyan << assertString(rb.m\_at) << "( " << rb.m\_expr << " ) "

5656 << Color::None;

5657

5658 if(rb.m\_at & assertType::is\_throws) {

5659 s << (rb.m\_threw ? "threw as expected!" : "did NOT throw at all!") << "\n";

5660 } else if((rb.m\_at & assertType::is\_throws\_as) &&

5661 (rb.m\_at & assertType::is\_throws\_with)) {

5662 s << Color::Cyan << assertString(rb.m\_at) << "( " << rb.m\_expr << ", \""

5663 << rb.m\_exception\_string.c\_str()

5664 << "\", " << rb.m\_exception\_type << " ) " << Color::None;

5665 if(rb.m\_threw) {

5666 if(!rb.m\_failed) {

5667 s << "threw as expected!\n";

5668 } else {

5669 s << "threw a DIFFERENT exception! (contents: " << rb.m\_exception << ")\n";

5670 }

5671 } else {

5672 s << "did NOT throw at all!\n";

5673 }

5674 } else if(rb.m\_at &

5675 assertType::is\_throws\_as) {

5676 s << Color::Cyan << assertString(rb.m\_at) << "( " << rb.m\_expr << ", "

5677 << rb.m\_exception\_type << " ) " << Color::None

5678 << (rb.m\_threw ? (rb.m\_threw\_as ? "threw as expected!" :

5679 "threw a DIFFERENT exception: ") :

5680 "did NOT throw at all!")

5681 << Color::Cyan << rb.m\_exception << "\n";

5682 } else if(rb.m\_at &

5683 assertType::is\_throws\_with) {

5684 s << Color::Cyan << assertString(rb.m\_at) << "( " << rb.m\_expr << ", \""

5685 << rb.m\_exception\_string.c\_str()

5686 << "\" ) " << Color::None

5687 << (rb.m\_threw ? (!rb.m\_failed ? "threw as expected!" :

5688 "threw a DIFFERENT exception: ") :

5689 "did NOT throw at all!")

5690 << Color::Cyan << rb.m\_exception << "\n";

5691 } else if(rb.m\_at & assertType::is\_nothrow) {

5692 s << (rb.m\_threw ? "THREW exception: " : "didn't throw!") << Color::Cyan

5693 << rb.m\_exception << "\n";

5694 } else {

5695 s << (rb.m\_threw ? "THREW exception: " :

5696 (!rb.m\_failed ? "is correct!\n" : "is NOT correct!\n"));

5697 if(rb.m\_threw)

5698 s << rb.m\_exception << "\n";

5699 else

5700 s << " values: " << assertString(rb.m\_at) << "( " << rb.m\_decomp << " )\n";

5701 }

5702 }

5703

5704 // TODO:

5705 // - log\_message()

5706 // - respond to queries

5707 // - honor remaining options

5708 // - more attributes in tags

5709 struct JUnitReporter : public IReporter

5710 {

5711 XmlWriter xml;

5712 DOCTEST\_DECLARE\_MUTEX(mutex)

5713 Timer timer;

5714 std::vector<String> deepestSubcaseStackNames;

5715

5716 struct JUnitTestCaseData

5717 {

5718 static std::string getCurrentTimestamp() {

5719 // Beware, this is not reentrant because of backward compatibility issues

5720 // Also, UTC only, again because of backward compatibility (%z is C++11)

5721 time\_t rawtime;

5722 std::time(&rawtime);

5723 auto const timeStampSize = sizeof("2017-01-16T17:06:45Z");

5724

5725 std::tm timeInfo;

5726 #ifdef DOCTEST\_PLATFORM\_WINDOWS

5727 gmtime\_s(&timeInfo, &rawtime);

5728 #else // DOCTEST\_PLATFORM\_WINDOWS

5729 gmtime\_r(&rawtime, &timeInfo);

5730 #endif // DOCTEST\_PLATFORM\_WINDOWS

5731

5732 char timeStamp[timeStampSize];

5733 const char\* const fmt = "%Y-%m-%dT%H:%M:%SZ";

5734

5735 std::strftime(timeStamp, timeStampSize, fmt, &timeInfo);

5736 return std::string(timeStamp);

5737 }

5738

5739 struct JUnitTestMessage

5740 {

5741 JUnitTestMessage(const std::string& \_message, const std::string& \_type, const std::string& \_details)

5742 : message(\_message), type(\_type), details(\_details) {}

5743

5744 JUnitTestMessage(const std::string& \_message, const std::string& \_details)

5745 : message(\_message), type(), details(\_details) {}

5746

5747 std::string message, type, details;

5748 };

5749

5750 struct JUnitTestCase

5751 {

5752 JUnitTestCase(const std::string& \_classname, const std::string& \_name)

5753 : classname(\_classname), name(\_name), time(0), failures() {}

5754

5755 std::string classname, name;

5756 double time;

5757 std::vector<JUnitTestMessage> failures, errors;

5758 };

5759

5760 void add(const std::string& classname, const std::string& name) {

5761 testcases.emplace\_back(classname, name);

5762 }

5763

5764 void appendSubcaseNamesToLastTestcase(std::vector<String> nameStack) {

5765 for(auto& curr: nameStack)

5766 if(curr.size())

5767 testcases.back().name += std::string("/") + curr.c\_str();

5768 }

5769

5770 void addTime(double time) {

5771 if(time < 1e-4)

5772 time = 0;

5773 testcases.back().time = time;

5774 totalSeconds += time;

5775 }

5776

5777 void addFailure(const std::string& message, const std::string& type, const std::string& details) {

5778 testcases.back().failures.emplace\_back(message, type, details);

5779 ++totalFailures;

5780 }

5781

5782 void addError(const std::string& message, const std::string& details) {

5783 testcases.back().errors.emplace\_back(message, details);

5784 ++totalErrors;

5785 }

5786

5787 std::vector<JUnitTestCase> testcases;

5788 double totalSeconds = 0;

5789 int totalErrors = 0, totalFailures = 0;

5790 };

5791

5792 JUnitTestCaseData testCaseData;

5793

5794 // caching pointers/references to objects of these types - safe to do

5795 const ContextOptions& opt;

5796 const TestCaseData\* tc = nullptr;

5797

5798 JUnitReporter(const ContextOptions& co)

5799 : xml(\*co.cout)

5800 , opt(co) {}

5801

5802 unsigned line(unsigned l) const { return opt.no\_line\_numbers ? 0 : l; }

5803

5804 // =========================================================================================

5805 // WHAT FOLLOWS ARE OVERRIDES OF THE VIRTUAL METHODS OF THE REPORTER INTERFACE

5806 // =========================================================================================

5807

5808 void report\_query(const QueryData&) override {

5809 xml.writeDeclaration();

5810 }

5811

5812 void test\_run\_start() override {

5813 xml.writeDeclaration();

5814 }

5815

5816 void test\_run\_end(const TestRunStats& p) override {

5817 // remove .exe extension - mainly to have the same output on UNIX and Windows

5818 std::string binary\_name = skipPathFromFilename(opt.binary\_name.c\_str());

5819 #ifdef DOCTEST\_PLATFORM\_WINDOWS

5820 if(binary\_name.rfind(".exe") != std::string::npos)

5821 binary\_name = binary\_name.substr(0, binary\_name.length() - 4);

5822 #endif // DOCTEST\_PLATFORM\_WINDOWS

5823 xml.startElement("testsuites");

5824 xml.startElement("testsuite").writeAttribute("name", binary\_name)

5825 .writeAttribute("errors", testCaseData.totalErrors)

5826 .writeAttribute("failures", testCaseData.totalFailures)

5827 .writeAttribute("tests", p.numAsserts);

5828 if(opt.no\_time\_in\_output == false) {

5829 xml.writeAttribute("time", testCaseData.totalSeconds);

5830 xml.writeAttribute("timestamp", JUnitTestCaseData::getCurrentTimestamp());

5831 }

5832 if(opt.no\_version == false)

5833 xml.writeAttribute("doctest\_version", DOCTEST\_VERSION\_STR);

5834

5835 for(const auto& testCase : testCaseData.testcases) {

5836 xml.startElement("testcase")

5837 .writeAttribute("classname", testCase.classname)

5838 .writeAttribute("name", testCase.name);

5839 if(opt.no\_time\_in\_output == false)

5840 xml.writeAttribute("time", testCase.time);

5841 // This is not ideal, but it should be enough to mimic gtest's junit output.

5842 xml.writeAttribute("status", "run");

5843

5844 for(const auto& failure : testCase.failures) {

5845 xml.scopedElement("failure")

5846 .writeAttribute("message", failure.message)

5847 .writeAttribute("type", failure.type)

5848 .writeText(failure.details, false);

5849 }

5850

5851 for(const auto& error : testCase.errors) {

5852 xml.scopedElement("error")

5853 .writeAttribute("message", error.message)

5854 .writeText(error.details);

5855 }

5856

5857 xml.endElement();

5858 }

5859 xml.endElement();

5860 xml.endElement();

5861 }

5862

5863 void test\_case\_start(const TestCaseData& in) override {

5864 testCaseData.add(skipPathFromFilename(in.m\_file.c\_str()), in.m\_name);

5865 timer.start();

5866 }

5867

5868 void test\_case\_reenter(const TestCaseData& in) override {

5869 testCaseData.addTime(timer.getElapsedSeconds());

5870 testCaseData.appendSubcaseNamesToLastTestcase(deepestSubcaseStackNames);

5871 deepestSubcaseStackNames.clear();

5872

5873 timer.start();

5874 testCaseData.add(skipPathFromFilename(in.m\_file.c\_str()), in.m\_name);

5875 }

5876

5877 void test\_case\_end(const CurrentTestCaseStats&) override {

5878 testCaseData.addTime(timer.getElapsedSeconds());

5879 testCaseData.appendSubcaseNamesToLastTestcase(deepestSubcaseStackNames);

5880 deepestSubcaseStackNames.clear();

5881 }

5882

5883 void test\_case\_exception(const TestCaseException& e) override {

5884 DOCTEST\_LOCK\_MUTEX(mutex)

5885 testCaseData.addError("exception", e.error\_string.c\_str());

5886 }

5887

5888 void subcase\_start(const SubcaseSignature& in) override {

5889 deepestSubcaseStackNames.push\_back(in.m\_name);

5890 }

5891

5892 void subcase\_end() override {}

5893

5894 void log\_assert(const AssertData& rb) override {

5895 if(!rb.m\_failed) // report only failures & ignore the `success` option

5896 return;

5897

5898 DOCTEST\_LOCK\_MUTEX(mutex)

5899

5900 std::ostringstream os;

5901 os << skipPathFromFilename(rb.m\_file) << (opt.gnu\_file\_line ? ":" : "(")

5902 << line(rb.m\_line) << (opt.gnu\_file\_line ? ":" : "):") << std::endl;

5903

5904 fulltext\_log\_assert\_to\_stream(os, rb);

5905 log\_contexts(os);

5906 testCaseData.addFailure(rb.m\_decomp.c\_str(), assertString(rb.m\_at), os.str());

5907 }

5908

5909 void log\_message(const MessageData& mb) override {

5910 if(mb.m\_severity & assertType::is\_warn) // report only failures

5911 return;

5912

5913 DOCTEST\_LOCK\_MUTEX(mutex)

5914

5915 std::ostringstream os;

5916 os << skipPathFromFilename(mb.m\_file) << (opt.gnu\_file\_line ? ":" : "(")

5917 << line(mb.m\_line) << (opt.gnu\_file\_line ? ":" : "):") << std::endl;

5918

5919 os << mb.m\_string.c\_str() << "\n";

5920 log\_contexts(os);

5921

5922 testCaseData.addFailure(mb.m\_string.c\_str(),

5923 mb.m\_severity & assertType::is\_check ? "FAIL\_CHECK" : "FAIL", os.str());

5924 }

5925

5926 void test\_case\_skipped(const TestCaseData&) override {}

5927

5928 void log\_contexts(std::ostringstream& s) {

5929 int num\_contexts = get\_num\_active\_contexts();

5930 if(num\_contexts) {

5931 auto contexts = get\_active\_contexts();

5932

5933 s << " logged: ";

5934 for(int i = 0; i < num\_contexts; ++i) {

5935 s << (i == 0 ? "" : " ");

5936 contexts[i]->stringify(&s);

5937 s << std::endl;

5938 }

5939 }

5940 }

5941 };

5942

5943 DOCTEST\_REGISTER\_REPORTER("junit", 0, JUnitReporter);

5944

5945 struct Whitespace

5946 {

5947 int nrSpaces;

5948 explicit Whitespace(int nr)

5949 : nrSpaces(nr) {}

5950 };

5951

5952 std::ostream& operator<<(std::ostream& out, const Whitespace& ws) {

5953 if(ws.nrSpaces != 0)

5954 out << std::setw(ws.nrSpaces) << ' ';

5955 return out;

5956 }

5957

5958 struct ConsoleReporter : public IReporter

5959 {

5960 std::ostream& s;

5961 bool hasLoggedCurrentTestStart;

5962 std::vector<SubcaseSignature> subcasesStack;

5963 size\_t currentSubcaseLevel;

5964 DOCTEST\_DECLARE\_MUTEX(mutex)

5965

5966 // caching pointers/references to objects of these types - safe to do

5967 const ContextOptions& opt;

5968 const TestCaseData\* tc;

5969

5970 ConsoleReporter(const ContextOptions& co)

5971 : s(\*co.cout)

5972 , opt(co) {}

5973

5974 ConsoleReporter(const ContextOptions& co, std::ostream& ostr)

5975 : s(ostr)

5976 , opt(co) {}

5977

5978 // =========================================================================================

5979 // WHAT FOLLOWS ARE HELPERS USED BY THE OVERRIDES OF THE VIRTUAL METHODS OF THE INTERFACE

5980 // =========================================================================================

5981

5982 void separator\_to\_stream() {

5983 s << Color::Yellow

5984 << "==============================================================================="

5985 "\n";

5986 }

5987

5988 const char\* getSuccessOrFailString(bool success, assertType::Enum at,

5989 const char\* success\_str) {

5990 if(success)

5991 return success\_str;

5992 return failureString(at);

5993 }

5994

5995 Color::Enum getSuccessOrFailColor(bool success, assertType::Enum at) {

5996 return success ? Color::BrightGreen :

5997 (at & assertType::is\_warn) ? Color::Yellow : Color::Red;

5998 }

5999

6000 void successOrFailColoredStringToStream(bool success, assertType::Enum at,

6001 const char\* success\_str = "SUCCESS") {

6002 s << getSuccessOrFailColor(success, at)

6003 << getSuccessOrFailString(success, at, success\_str) << ": ";

6004 }

6005

6006 void log\_contexts() {

6007 int num\_contexts = get\_num\_active\_contexts();

6008 if(num\_contexts) {

6009 auto contexts = get\_active\_contexts();

6010

6011 s << Color::None << " logged: ";

6012 for(int i = 0; i < num\_contexts; ++i) {

6013 s << (i == 0 ? "" : " ");

6014 contexts[i]->stringify(&s);

6015 s << "\n";

6016 }

6017 }

6018

6019 s << "\n";

6020 }

6021

6022 // this was requested to be made virtual so users could override it

6023 virtual void file\_line\_to\_stream(const char\* file, int line,

6024 const char\* tail = "") {

6025 s << Color::LightGrey << skipPathFromFilename(file) << (opt.gnu\_file\_line ? ":" : "(")

6026 << (opt.no\_line\_numbers ? 0 : line) // 0 or the real num depending on the option

6027 << (opt.gnu\_file\_line ? ":" : "):") << tail;

6028 }

6029

6030 void logTestStart() {

6031 if(hasLoggedCurrentTestStart)

6032 return;

6033

6034 separator\_to\_stream();

6035 file\_line\_to\_stream(tc->m\_file.c\_str(), tc->m\_line, "\n");

6036 if(tc->m\_description)

6037 s << Color::Yellow << "DESCRIPTION: " << Color::None << tc->m\_description << "\n";

6038 if(tc->m\_test\_suite && tc->m\_test\_suite[0] != '\0')

6039 s << Color::Yellow << "TEST SUITE: " << Color::None << tc->m\_test\_suite << "\n";

6040 if(strncmp(tc->m\_name, " Scenario:", 11) != 0)

6041 s << Color::Yellow << "TEST CASE: ";

6042 s << Color::None << tc->m\_name << "\n";

6043

6044 for(size\_t i = 0; i < currentSubcaseLevel; ++i) {

6045 if(subcasesStack[i].m\_name[0] != '\0')

6046 s << " " << subcasesStack[i].m\_name << "\n";

6047 }

6048

6049 if(currentSubcaseLevel != subcasesStack.size()) {

6050 s << Color::Yellow << "\nDEEPEST SUBCASE STACK REACHED (DIFFERENT FROM THE CURRENT ONE):\n" << Color::None;

6051 for(size\_t i = 0; i < subcasesStack.size(); ++i) {

6052 if(subcasesStack[i].m\_name[0] != '\0')

6053 s << " " << subcasesStack[i].m\_name << "\n";

6054 }

6055 }

6056

6057 s << "\n";

6058

6059 hasLoggedCurrentTestStart = true;

6060 }

6061

6062 void printVersion() {

6063 if(opt.no\_version == false)

6064 s << Color::Cyan << "[doctest] " << Color::None << "doctest version is \""

6065 << DOCTEST\_VERSION\_STR << "\"\n";

6066 }

6067

6068 void printIntro() {

6069 if(opt.no\_intro == false) {

6070 printVersion();

6071 s << Color::Cyan << "[doctest] " << Color::None

6072 << "run with \"--" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "help\" for options\n";

6073 }

6074 }

6075

6076 void printHelp() {

6077 int sizePrefixDisplay = static\_cast<int>(strlen(DOCTEST\_OPTIONS\_PREFIX\_DISPLAY));

6078 printVersion();

6079 // clang-format off

6080 s << Color::Cyan << "[doctest]\n" << Color::None;

6081 s << Color::Cyan << "[doctest] " << Color::None;

6082 s << "boolean values: \"1/on/yes/true\" or \"0/off/no/false\"\n";

6083 s << Color::Cyan << "[doctest] " << Color::None;

6084 s << "filter values: \"str1,str2,str3\" (comma separated strings)\n";

6085 s << Color::Cyan << "[doctest]\n" << Color::None;

6086 s << Color::Cyan << "[doctest] " << Color::None;

6087 s << "filters use wildcards for matching strings\n";

6088 s << Color::Cyan << "[doctest] " << Color::None;

6089 s << "something passes a filter if any of the strings in a filter matches\n";

6090 #ifndef DOCTEST\_CONFIG\_NO\_UNPREFIXED\_OPTIONS

6091 s << Color::Cyan << "[doctest]\n" << Color::None;

6092 s << Color::Cyan << "[doctest] " << Color::None;

6093 s << "ALL FLAGS, OPTIONS AND FILTERS ALSO AVAILABLE WITH A \"" DOCTEST\_CONFIG\_OPTIONS\_PREFIX "\" PREFIX!!!\n";

6094 #endif

6095 s << Color::Cyan << "[doctest]\n" << Color::None;

6096 s << Color::Cyan << "[doctest] " << Color::None;

6097 s << "Query flags - the program quits after them. Available:\n\n";

6098 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "?, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "help, -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "h "

6099 << Whitespace(sizePrefixDisplay\*0) << "prints this message\n";

6100 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "v, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "version "

6101 << Whitespace(sizePrefixDisplay\*1) << "prints the version\n";

6102 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "c, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "count "

6103 << Whitespace(sizePrefixDisplay\*1) << "prints the number of matching tests\n";

6104 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "ltc, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "list-test-cases "

6105 << Whitespace(sizePrefixDisplay\*1) << "lists all matching tests by name\n";

6106 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "lts, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "list-test-suites "

6107 << Whitespace(sizePrefixDisplay\*1) << "lists all matching test suites\n";

6108 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "lr, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "list-reporters "

6109 << Whitespace(sizePrefixDisplay\*1) << "lists all registered reporters\n\n";

6110 // ================================================================================== << 79

6111 s << Color::Cyan << "[doctest] " << Color::None;

6112 s << "The available <int>/<string> options/filters are:\n\n";

6113 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "tc, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "test-case=<filters> "

6114 << Whitespace(sizePrefixDisplay\*1) << "filters tests by their name\n";

6115 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "tce, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "test-case-exclude=<filters> "

6116 << Whitespace(sizePrefixDisplay\*1) << "filters OUT tests by their name\n";

6117 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "sf, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "source-file=<filters> "

6118 << Whitespace(sizePrefixDisplay\*1) << "filters tests by their file\n";

6119 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "sfe, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "source-file-exclude=<filters> "

6120 << Whitespace(sizePrefixDisplay\*1) << "filters OUT tests by their file\n";

6121 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "ts, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "test-suite=<filters> "

6122 << Whitespace(sizePrefixDisplay\*1) << "filters tests by their test suite\n";

6123 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "tse, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "test-suite-exclude=<filters> "

6124 << Whitespace(sizePrefixDisplay\*1) << "filters OUT tests by their test suite\n";

6125 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "sc, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "subcase=<filters> "

6126 << Whitespace(sizePrefixDisplay\*1) << "filters subcases by their name\n";

6127 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "sce, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "subcase-exclude=<filters> "

6128 << Whitespace(sizePrefixDisplay\*1) << "filters OUT subcases by their name\n";

6129 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "r, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "reporters=<filters> "

6130 << Whitespace(sizePrefixDisplay\*1) << "reporters to use (console is default)\n";

6131 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "o, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "out=<string> "

6132 << Whitespace(sizePrefixDisplay\*1) << "output filename\n";

6133 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "ob, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "order-by=<string> "

6134 << Whitespace(sizePrefixDisplay\*1) << "how the tests should be ordered\n";

6135 s << Whitespace(sizePrefixDisplay\*3) << " <string> - [file/suite/name/rand/none]\n";

6136 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "rs, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "rand-seed=<int> "

6137 << Whitespace(sizePrefixDisplay\*1) << "seed for random ordering\n";

6138 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "f, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "first=<int> "

6139 << Whitespace(sizePrefixDisplay\*1) << "the first test passing the filters to\n";

6140 s << Whitespace(sizePrefixDisplay\*3) << " execute - for range-based execution\n";

6141 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "l, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "last=<int> "

6142 << Whitespace(sizePrefixDisplay\*1) << "the last test passing the filters to\n";

6143 s << Whitespace(sizePrefixDisplay\*3) << " execute - for range-based execution\n";

6144 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "aa, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "abort-after=<int> "

6145 << Whitespace(sizePrefixDisplay\*1) << "stop after <int> failed assertions\n";

6146 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "scfl,--" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "subcase-filter-levels=<int> "

6147 << Whitespace(sizePrefixDisplay\*1) << "apply filters for the first <int> levels\n";

6148 s << Color::Cyan << "\n[doctest] " << Color::None;

6149 s << "Bool options - can be used like flags and true is assumed. Available:\n\n";

6150 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "s, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "success=<bool> "

6151 << Whitespace(sizePrefixDisplay\*1) << "include successful assertions in output\n";

6152 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "cs, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "case-sensitive=<bool> "

6153 << Whitespace(sizePrefixDisplay\*1) << "filters being treated as case sensitive\n";

6154 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "e, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "exit=<bool> "

6155 << Whitespace(sizePrefixDisplay\*1) << "exits after the tests finish\n";

6156 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "d, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "duration=<bool> "

6157 << Whitespace(sizePrefixDisplay\*1) << "prints the time duration of each test\n";

6158 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "m, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "minimal=<bool> "

6159 << Whitespace(sizePrefixDisplay\*1) << "minimal console output (only failures)\n";

6160 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "q, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "quiet=<bool> "

6161 << Whitespace(sizePrefixDisplay\*1) << "no console output\n";

6162 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "nt, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-throw=<bool> "

6163 << Whitespace(sizePrefixDisplay\*1) << "skips exceptions-related assert checks\n";

6164 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "ne, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-exitcode=<bool> "

6165 << Whitespace(sizePrefixDisplay\*1) << "returns (or exits) always with success\n";

6166 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "nr, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-run=<bool> "

6167 << Whitespace(sizePrefixDisplay\*1) << "skips all runtime doctest operations\n";

6168 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "ni, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-intro=<bool> "

6169 << Whitespace(sizePrefixDisplay\*1) << "omit the framework intro in the output\n";

6170 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "nv, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-version=<bool> "

6171 << Whitespace(sizePrefixDisplay\*1) << "omit the framework version in the output\n";

6172 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "nc, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-colors=<bool> "

6173 << Whitespace(sizePrefixDisplay\*1) << "disables colors in output\n";

6174 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "fc, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "force-colors=<bool> "

6175 << Whitespace(sizePrefixDisplay\*1) << "use colors even when not in a tty\n";

6176 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "nb, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-breaks=<bool> "

6177 << Whitespace(sizePrefixDisplay\*1) << "disables breakpoints in debuggers\n";

6178 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "ns, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-skip=<bool> "

6179 << Whitespace(sizePrefixDisplay\*1) << "don't skip test cases marked as skip\n";

6180 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "gfl, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "gnu-file-line=<bool> "

6181 << Whitespace(sizePrefixDisplay\*1) << ":n: vs (n): for line numbers in output\n";

6182 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "npf, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-path-filenames=<bool> "

6183 << Whitespace(sizePrefixDisplay\*1) << "only filenames and no paths in output\n";

6184 s << " -" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "nln, --" DOCTEST\_OPTIONS\_PREFIX\_DISPLAY "no-line-numbers=<bool> "

6185 << Whitespace(sizePrefixDisplay\*1) << "0 instead of real line numbers in output\n";

6186 // ================================================================================== << 79

6187 // clang-format on

6188

6189 s << Color::Cyan << "\n[doctest] " << Color::None;

6190 s << "for more information visit the project documentation\n\n";

6191 }

6192

6193 void printRegisteredReporters() {

6194 printVersion();

6195 auto printReporters = [this] (const reporterMap& reporters, const char\* type) {

6196 if(reporters.size()) {

6197 s << Color::Cyan << "[doctest] " << Color::None << "listing all registered " << type << "\n";

6198 for(auto& curr : reporters)

6199 s << "priority: " << std::setw(5) << curr.first.first

6200 << " name: " << curr.first.second << "\n";

6201 }

6202 };

6203 printReporters(getListeners(), "listeners");

6204 printReporters(getReporters(), "reporters");

6205 }

6206

6207 // =========================================================================================

6208 // WHAT FOLLOWS ARE OVERRIDES OF THE VIRTUAL METHODS OF THE REPORTER INTERFACE

6209 // =========================================================================================

6210

6211 void report\_query(const QueryData& in) override {

6212 if(opt.version) {

6213 printVersion();

6214 } else if(opt.help) {

6215 printHelp();

6216 } else if(opt.list\_reporters) {

6217 printRegisteredReporters();

6218 } else if(opt.count || opt.list\_test\_cases) {

6219 if(opt.list\_test\_cases) {

6220 s << Color::Cyan << "[doctest] " << Color::None

6221 << "listing all test case names\n";

6222 separator\_to\_stream();

6223 }

6224

6225 for(unsigned i = 0; i < in.num\_data; ++i)

6226 s << Color::None << in.data[i]->m\_name << "\n";

6227

6228 separator\_to\_stream();

6229

6230 s << Color::Cyan << "[doctest] " << Color::None

6231 << "unskipped test cases passing the current filters: "

6232 << g\_cs->numTestCasesPassingFilters << "\n";

6233

6234 } else if(opt.list\_test\_suites) {

6235 s << Color::Cyan << "[doctest] " << Color::None << "listing all test suites\n";

6236 separator\_to\_stream();

6237

6238 for(unsigned i = 0; i < in.num\_data; ++i)

6239 s << Color::None << in.data[i]->m\_test\_suite << "\n";

6240

6241 separator\_to\_stream();

6242

6243 s << Color::Cyan << "[doctest] " << Color::None

6244 << "unskipped test cases passing the current filters: "

6245 << g\_cs->numTestCasesPassingFilters << "\n";

6246 s << Color::Cyan << "[doctest] " << Color::None

6247 << "test suites with unskipped test cases passing the current filters: "

6248 << g\_cs->numTestSuitesPassingFilters << "\n";

6249 }

6250 }

6251

6252 void test\_run\_start() override {

6253 if(!opt.minimal)

6254 printIntro();

6255 }

6256

6257 void test\_run\_end(const TestRunStats& p) override {

6258 if(opt.minimal && p.numTestCasesFailed == 0)

6259 return;

6260

6261 separator\_to\_stream();

6262 s << std::dec;

6263

6264 auto totwidth = int(std::ceil(log10(static\_cast<double>(std::max(p.numTestCasesPassingFilters, static\_cast<unsigned>(p.numAsserts))) + 1)));

6265 auto passwidth = int(std::ceil(log10(static\_cast<double>(std::max(p.numTestCasesPassingFilters - p.numTestCasesFailed, static\_cast<unsigned>(p.numAsserts - p.numAssertsFailed))) + 1)));

6266 auto failwidth = int(std::ceil(log10(static\_cast<double>(std::max(p.numTestCasesFailed, static\_cast<unsigned>(p.numAssertsFailed))) + 1)));

6267 const bool anythingFailed = p.numTestCasesFailed > 0 || p.numAssertsFailed > 0;

6268 s << Color::Cyan << "[doctest] " << Color::None << "test cases: " << std::setw(totwidth)

6269 << p.numTestCasesPassingFilters << " | "

6270 << ((p.numTestCasesPassingFilters == 0 || anythingFailed) ? Color::None :

6271 Color::Green)

6272 << std::setw(passwidth) << p.numTestCasesPassingFilters - p.numTestCasesFailed << " passed"

6273 << Color::None << " | " << (p.numTestCasesFailed > 0 ? Color::Red : Color::None)

6274 << std::setw(failwidth) << p.numTestCasesFailed << " failed" << Color::None << " |";

6275 if(opt.no\_skipped\_summary == false) {

6276 const int numSkipped = p.numTestCases - p.numTestCasesPassingFilters;

6277 s << " " << (numSkipped == 0 ? Color::None : Color::Yellow) << numSkipped

6278 << " skipped" << Color::None;

6279 }

6280 s << "\n";

6281 s << Color::Cyan << "[doctest] " << Color::None << "assertions: " << std::setw(totwidth)

6282 << p.numAsserts << " | "

6283 << ((p.numAsserts == 0 || anythingFailed) ? Color::None : Color::Green)

6284 << std::setw(passwidth) << (p.numAsserts - p.numAssertsFailed) << " passed" << Color::None

6285 << " | " << (p.numAssertsFailed > 0 ? Color::Red : Color::None) << std::setw(failwidth)

6286 << p.numAssertsFailed << " failed" << Color::None << " |\n";

6287 s << Color::Cyan << "[doctest] " << Color::None

6288 << "Status: " << (p.numTestCasesFailed > 0 ? Color::Red : Color::Green)

6289 << ((p.numTestCasesFailed > 0) ? "FAILURE!" : "SUCCESS!") << Color::None << std::endl;

6290 }

6291

6292 void test\_case\_start(const TestCaseData& in) override {

6293 hasLoggedCurrentTestStart = false;

6294 tc = &in;

6295 subcasesStack.clear();

6296 currentSubcaseLevel = 0;

6297 }

6298

6299 void test\_case\_reenter(const TestCaseData&) override {

6300 subcasesStack.clear();

6301 }

6302

6303 void test\_case\_end(const CurrentTestCaseStats& st) override {

6304 if(tc->m\_no\_output)

6305 return;

6306

6307 // log the preamble of the test case only if there is something

6308 // else to print - something other than that an assert has failed

6309 if(opt.duration ||

6310 (st.failure\_flags && st.failure\_flags != static\_cast<int>(TestCaseFailureReason::AssertFailure)))

6311 logTestStart();

6312

6313 if(opt.duration)

6314 s << Color::None << std::setprecision(6) << std::fixed << st.seconds

6315 << " s: " << tc->m\_name << "\n";

6316

6317 if(st.failure\_flags & TestCaseFailureReason::Timeout)

6318 s << Color::Red << "Test case exceeded time limit of " << std::setprecision(6)

6319 << std::fixed << tc->m\_timeout << "!\n";

6320

6321 if(st.failure\_flags & TestCaseFailureReason::ShouldHaveFailedButDidnt) {

6322 s << Color::Red << "Should have failed but didn't! Marking it as failed!\n";

6323 } else if(st.failure\_flags & TestCaseFailureReason::ShouldHaveFailedAndDid) {

6324 s << Color::Yellow << "Failed as expected so marking it as not failed\n";

6325 } else if(st.failure\_flags & TestCaseFailureReason::CouldHaveFailedAndDid) {

6326 s << Color::Yellow << "Allowed to fail so marking it as not failed\n";

6327 } else if(st.failure\_flags & TestCaseFailureReason::DidntFailExactlyNumTimes) {

6328 s << Color::Red << "Didn't fail exactly " << tc->m\_expected\_failures

6329 << " times so marking it as failed!\n";

6330 } else if(st.failure\_flags & TestCaseFailureReason::FailedExactlyNumTimes) {

6331 s << Color::Yellow << "Failed exactly " << tc->m\_expected\_failures

6332 << " times as expected so marking it as not failed!\n";

6333 }

6334 if(st.failure\_flags & TestCaseFailureReason::TooManyFailedAsserts) {

6335 s << Color::Red << "Aborting - too many failed asserts!\n";

6336 }

6337 s << Color::None; // lgtm [cpp/useless-expression]

6338 }

6339

6340 void test\_case\_exception(const TestCaseException& e) override {

6341 DOCTEST\_LOCK\_MUTEX(mutex)

6342 if(tc->m\_no\_output)

6343 return;

6344

6345 logTestStart();

6346

6347 file\_line\_to\_stream(tc->m\_file.c\_str(), tc->m\_line, " ");

6348 successOrFailColoredStringToStream(false, e.is\_crash ? assertType::is\_require :

6349 assertType::is\_check);

6350 s << Color::Red << (e.is\_crash ? "test case CRASHED: " : "test case THREW exception: ")

6351 << Color::Cyan << e.error\_string << "\n";

6352

6353 int num\_stringified\_contexts = get\_num\_stringified\_contexts();

6354 if(num\_stringified\_contexts) {

6355 auto stringified\_contexts = get\_stringified\_contexts();

6356 s << Color::None << " logged: ";

6357 for(int i = num\_stringified\_contexts; i > 0; --i) {

6358 s << (i == num\_stringified\_contexts ? "" : " ")

6359 << stringified\_contexts[i - 1] << "\n";

6360 }

6361 }

6362 s << "\n" << Color::None;

6363 }

6364

6365 void subcase\_start(const SubcaseSignature& subc) override {

6366 subcasesStack.push\_back(subc);

6367 ++currentSubcaseLevel;

6368 hasLoggedCurrentTestStart = false;

6369 }

6370

6371 void subcase\_end() override {

6372 --currentSubcaseLevel;

6373 hasLoggedCurrentTestStart = false;

6374 }

6375

6376 void log\_assert(const AssertData& rb) override {

6377 if((!rb.m\_failed && !opt.success) || tc->m\_no\_output)

6378 return;

6379

6380 DOCTEST\_LOCK\_MUTEX(mutex)

6381

6382 logTestStart();

6383

6384 file\_line\_to\_stream(rb.m\_file, rb.m\_line, " ");

6385 successOrFailColoredStringToStream(!rb.m\_failed, rb.m\_at);

6386

6387 fulltext\_log\_assert\_to\_stream(s, rb);

6388

6389 log\_contexts();

6390 }

6391

6392 void log\_message(const MessageData& mb) override {

6393 if(tc->m\_no\_output)

6394 return;

6395

6396 DOCTEST\_LOCK\_MUTEX(mutex)

6397

6398 logTestStart();

6399

6400 file\_line\_to\_stream(mb.m\_file, mb.m\_line, " ");

6401 s << getSuccessOrFailColor(false, mb.m\_severity)

6402 << getSuccessOrFailString(mb.m\_severity & assertType::is\_warn, mb.m\_severity,

6403 "MESSAGE") << ": ";

6404 s << Color::None << mb.m\_string << "\n";

6405 log\_contexts();

6406 }

6407

6408 void test\_case\_skipped(const TestCaseData&) override {}

6409 };

6410

6411 DOCTEST\_REGISTER\_REPORTER("console", 0, ConsoleReporter);

6412

6413 #ifdef DOCTEST\_PLATFORM\_WINDOWS

6414 struct DebugOutputWindowReporter : public ConsoleReporter

6415 {

6416 DOCTEST\_THREAD\_LOCAL static std::ostringstream oss;

6417

6418 DebugOutputWindowReporter(const ContextOptions& co)

6419 : ConsoleReporter(co, oss) {}

6420

6421 #define DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(func, type, arg) \

6422 void func(type arg) override { \

6423 bool with\_col = g\_no\_colors; \

6424 g\_no\_colors = false; \

6425 ConsoleReporter::func(arg); \

6426 if(oss.tellp() != std::streampos{}) { \

6427 DOCTEST\_OUTPUT\_DEBUG\_STRING(oss.str().c\_str()); \

6428 oss.str(""); \

6429 } \

6430 g\_no\_colors = with\_col; \

6431 }

6432

6433 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(test\_run\_start, DOCTEST\_EMPTY, DOCTEST\_EMPTY)

6434 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(test\_run\_end, const TestRunStats&, in)

6435 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(test\_case\_start, const TestCaseData&, in)

6436 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(test\_case\_reenter, const TestCaseData&, in)

6437 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(test\_case\_end, const CurrentTestCaseStats&, in)

6438 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(test\_case\_exception, const TestCaseException&, in)

6439 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(subcase\_start, const SubcaseSignature&, in)

6440 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(subcase\_end, DOCTEST\_EMPTY, DOCTEST\_EMPTY)

6441 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(log\_assert, const AssertData&, in)

6442 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(log\_message, const MessageData&, in)

6443 DOCTEST\_DEBUG\_OUTPUT\_REPORTER\_OVERRIDE(test\_case\_skipped, const TestCaseData&, in)

6444 };

6445

6446 DOCTEST\_THREAD\_LOCAL std::ostringstream DebugOutputWindowReporter::oss;

6447 #endif // DOCTEST\_PLATFORM\_WINDOWS

6448

6449 // the implementation of parseOption()

6450 bool parseOptionImpl(int argc, const char\* const\* argv, const char\* pattern, String\* value) {

6451 // going from the end to the beginning and stopping on the first occurrence from the end

6452 for(int i = argc; i > 0; --i) {

6453 auto index = i - 1;

6454 auto temp = std::strstr(argv[index], pattern);

6455 if(temp && (value || strlen(temp) == strlen(pattern))) {

6456 // eliminate matches in which the chars before the option are not '-'

6457 bool noBadCharsFound = true;

6458 auto curr = argv[index];

6459 while(curr != temp) {

6460 if(\*curr++ != '-') {

6461 noBadCharsFound = false;

6462 break;

6463 }

6464 }

6465 if(noBadCharsFound && argv[index][0] == '-') {

6466 if(value) {

6467 // parsing the value of an option

6468 temp += strlen(pattern);

6469 const unsigned len = strlen(temp);

6470 if(len) {

6471 \*value = temp;

6472 return true;

6473 }

6474 } else {

6475 // just a flag - no value

6476 return true;

6477 }

6478 }

6479 }

6480 }

6481 return false;

6482 }

6483

6484 // parses an option and returns the string after the '=' character

6485 bool parseOption(int argc, const char\* const\* argv, const char\* pattern, String\* value = nullptr,

6486 const String& defaultVal = String()) {

6487 if(value)

6488 \*value = defaultVal;

6489 #ifndef DOCTEST\_CONFIG\_NO\_UNPREFIXED\_OPTIONS

6490 // offset (normally 3 for "dt-") to skip prefix

6491 if(parseOptionImpl(argc, argv, pattern + strlen(DOCTEST\_CONFIG\_OPTIONS\_PREFIX), value))

6492 return true;

6493 #endif // DOCTEST\_CONFIG\_NO\_UNPREFIXED\_OPTIONS

6494 return parseOptionImpl(argc, argv, pattern, value);

6495 }

6496

6497 // locates a flag on the command line

6498 bool parseFlag(int argc, const char\* const\* argv, const char\* pattern) {

6499 return parseOption(argc, argv, pattern);

6500 }

6501

6502 // parses a comma separated list of words after a pattern in one of the arguments in argv

6503 bool parseCommaSepArgs(int argc, const char\* const\* argv, const char\* pattern,

6504 std::vector<String>& res) {

6505 String filtersString;

6506 if(parseOption(argc, argv, pattern, &filtersString)) {

6507 // tokenize with "," as a separator, unless escaped with backslash

6508 std::ostringstream s;

6509 auto flush = [&s, &res]() {

6510 auto string = s.str();

6511 if(string.size() > 0) {

6512 res.push\_back(string.c\_str());

6513 }

6514 s.str("");

6515 };

6516

6517 bool seenBackslash = false;

6518 const char\* current = filtersString.c\_str();

6519 const char\* end = current + strlen(current);

6520 while(current != end) {

6521 char character = \*current++;

6522 if(seenBackslash) {

6523 seenBackslash = false;

6524 if(character == ',' || character == '\\') {

6525 s.put(character);

6526 continue;

6527 }

6528 s.put('\\');

6529 }

6530 if(character == '\\') {

6531 seenBackslash = true;

6532 } else if(character == ',') {

6533 flush();

6534 } else {

6535 s.put(character);

6536 }

6537 }

6538

6539 if(seenBackslash) {

6540 s.put('\\');

6541 }

6542 flush();

6543 return true;

6544 }

6545 return false;

6546 }

6547

6548 enum optionType

6549 {

6550 option\_bool,

6551 option\_int

6552 };

6553

6554 // parses an int/bool option from the command line

6555 bool parseIntOption(int argc, const char\* const\* argv, const char\* pattern, optionType type,

6556 int& res) {

6557 String parsedValue;

6558 if(!parseOption(argc, argv, pattern, &parsedValue))

6559 return false;

6560

6561 if(type) {

6562 // integer

6563 // TODO: change this to use std::stoi or something else! currently it uses undefined behavior - assumes '0' on failed parse...

6564 int theInt = std::atoi(parsedValue.c\_str());

6565 if (theInt != 0) {

6566 res = theInt;

6567 return true;

6568 }

6569 } else {

6570 // boolean

6571 const char positive[][5] = { "1", "true", "on", "yes" }; // 5 - strlen("true") + 1

6572 const char negative[][6] = { "0", "false", "off", "no" }; // 6 - strlen("false") + 1

6573

6574 // if the value matches any of the positive/negative possibilities

6575 for (unsigned i = 0; i < 4; i++) {

6576 if (parsedValue.compare(positive[i], true) == 0) {

6577 res = 1;

6578 return true;

6579 }

6580 if (parsedValue.compare(negative[i], true) == 0) {

6581 res = 0;

6582 return true;

6583 }

6584 }

6585 }

6586 return false;

6587 }

6588 } // namespace

6589

6590 Context::Context(int argc, const char\* const\* argv)

6591 : p(new detail::ContextState) {

6592 parseArgs(argc, argv, true);

6593 if(argc)

6594 p->binary\_name = argv[0];

6595 }

6596

6597 Context::~Context() {

6598 if(g\_cs == p)

6599 g\_cs = nullptr;

6600 delete p;

6601 }

6602

6603 void Context::applyCommandLine(int argc, const char\* const\* argv) {

6604 parseArgs(argc, argv);

6605 if(argc)

6606 p->binary\_name = argv[0];

6607 }

6608

6609 // parses args

6610 void Context::parseArgs(int argc, const char\* const\* argv, bool withDefaults) {

6611 using namespace detail;

6612

6613 // clang-format off

6614 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "source-file=", p->filters[0]);

6615 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "sf=", p->filters[0]);

6616 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "source-file-exclude=",p->filters[1]);

6617 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "sfe=", p->filters[1]);

6618 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "test-suite=", p->filters[2]);

6619 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "ts=", p->filters[2]);

6620 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "test-suite-exclude=", p->filters[3]);

6621 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "tse=", p->filters[3]);

6622 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "test-case=", p->filters[4]);

6623 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "tc=", p->filters[4]);

6624 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "test-case-exclude=", p->filters[5]);

6625 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "tce=", p->filters[5]);

6626 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "subcase=", p->filters[6]);

6627 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "sc=", p->filters[6]);

6628 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "subcase-exclude=", p->filters[7]);

6629 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "sce=", p->filters[7]);

6630 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "reporters=", p->filters[8]);

6631 parseCommaSepArgs(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "r=", p->filters[8]);

6632 // clang-format on

6633

6634 int intRes = 0;

6635 String strRes;

6636

6637 #define DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG(name, sname, var, default) \

6638 if(parseIntOption(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX name "=", option\_bool, intRes) || \

6639 parseIntOption(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX sname "=", option\_bool, intRes)) \

6640 p->var = static\_cast<bool>(intRes); \

6641 else if(parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX name) || \

6642 parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX sname)) \

6643 p->var = true; \

6644 else if(withDefaults) \

6645 p->var = default

6646

6647 #define DOCTEST\_PARSE\_INT\_OPTION(name, sname, var, default) \

6648 if(parseIntOption(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX name "=", option\_int, intRes) || \

6649 parseIntOption(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX sname "=", option\_int, intRes)) \

6650 p->var = intRes; \

6651 else if(withDefaults) \

6652 p->var = default

6653

6654 #define DOCTEST\_PARSE\_STR\_OPTION(name, sname, var, default) \

6655 if(parseOption(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX name "=", &strRes, default) || \

6656 parseOption(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX sname "=", &strRes, default) || \

6657 withDefaults) \

6658 p->var = strRes

6659

6660 // clang-format off

6661 DOCTEST\_PARSE\_STR\_OPTION("out", "o", out, "");

6662 DOCTEST\_PARSE\_STR\_OPTION("order-by", "ob", order\_by, "file");

6663 DOCTEST\_PARSE\_INT\_OPTION("rand-seed", "rs", rand\_seed, 0);

6664

6665 DOCTEST\_PARSE\_INT\_OPTION("first", "f", first, 0);

6666 DOCTEST\_PARSE\_INT\_OPTION("last", "l", last, UINT\_MAX);

6667

6668 DOCTEST\_PARSE\_INT\_OPTION("abort-after", "aa", abort\_after, 0);

6669 DOCTEST\_PARSE\_INT\_OPTION("subcase-filter-levels", "scfl", subcase\_filter\_levels, INT\_MAX);

6670

6671 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("success", "s", success, false);

6672 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("case-sensitive", "cs", case\_sensitive, false);

6673 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("exit", "e", exit, false);

6674 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("duration", "d", duration, false);

6675 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("minimal", "m", minimal, false);

6676 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("quiet", "q", quiet, false);

6677 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-throw", "nt", no\_throw, false);

6678 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-exitcode", "ne", no\_exitcode, false);

6679 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-run", "nr", no\_run, false);

6680 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-intro", "ni", no\_intro, false);

6681 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-version", "nv", no\_version, false);

6682 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-colors", "nc", no\_colors, false);

6683 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("force-colors", "fc", force\_colors, false);

6684 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-breaks", "nb", no\_breaks, false);

6685 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-skip", "ns", no\_skip, false);

6686 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("gnu-file-line", "gfl", gnu\_file\_line, !bool(DOCTEST\_MSVC));

6687 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-path-filenames", "npf", no\_path\_in\_filenames, false);

6688 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-line-numbers", "nln", no\_line\_numbers, false);

6689 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-debug-output", "ndo", no\_debug\_output, false);

6690 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-skipped-summary", "nss", no\_skipped\_summary, false);

6691 DOCTEST\_PARSE\_AS\_BOOL\_OR\_FLAG("no-time-in-output", "ntio", no\_time\_in\_output, false);

6692 // clang-format on

6693

6694 if(withDefaults) {

6695 p->help = false;

6696 p->version = false;

6697 p->count = false;

6698 p->list\_test\_cases = false;

6699 p->list\_test\_suites = false;

6700 p->list\_reporters = false;

6701 }

6702 if(parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "help") ||

6703 parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "h") ||

6704 parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "?")) {

6705 p->help = true;

6706 p->exit = true;

6707 }

6708 if(parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "version") ||

6709 parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "v")) {

6710 p->version = true;

6711 p->exit = true;

6712 }

6713 if(parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "count") ||

6714 parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "c")) {

6715 p->count = true;

6716 p->exit = true;

6717 }

6718 if(parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "list-test-cases") ||

6719 parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "ltc")) {

6720 p->list\_test\_cases = true;

6721 p->exit = true;

6722 }

6723 if(parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "list-test-suites") ||

6724 parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "lts")) {

6725 p->list\_test\_suites = true;

6726 p->exit = true;

6727 }

6728 if(parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "list-reporters") ||

6729 parseFlag(argc, argv, DOCTEST\_CONFIG\_OPTIONS\_PREFIX "lr")) {

6730 p->list\_reporters = true;

6731 p->exit = true;

6732 }

6733 }

6734

6735 // allows the user to add procedurally to the filters from the command line

6736 void Context::addFilter(const char\* filter, const char\* value) { setOption(filter, value); }

6737

6738 // allows the user to clear all filters from the command line

6739 void Context::clearFilters() {

6740 for(auto& curr : p->filters)

6741 curr.clear();

6742 }

6743

6744 // allows the user to override procedurally the bool options from the command line

6745 void Context::setOption(const char\* option, bool value) {

6746 setOption(option, value ? "true" : "false");

6747 }

6748

6749 // allows the user to override procedurally the int options from the command line

6750 void Context::setOption(const char\* option, int value) {

6751 setOption(option, toString(value).c\_str());

6752 }

6753

6754 // allows the user to override procedurally the string options from the command line

6755 void Context::setOption(const char\* option, const char\* value) {

6756 auto argv = String("-") + option + "=" + value;

6757 auto lvalue = argv.c\_str();

6758 parseArgs(1, &lvalue);

6759 }

6760

6761 // users should query this in their main() and exit the program if true

6762 bool Context::shouldExit() { return p->exit; }

6763

6764 void Context::setAsDefaultForAssertsOutOfTestCases() { g\_cs = p; }

6765

6766 void Context::setAssertHandler(detail::assert\_handler ah) { p->ah = ah; }

6767

6768 void Context::setCout(std::ostream\* out) { p->cout = out; }

6769

6770 static class DiscardOStream : public std::ostream

6771 {

6772 private:

6773 class : public std::streambuf

6774 {

6775 private:

6776 // allowing some buffering decreases the amount of calls to overflow

6777 char buf[1024];

6778

6779 protected:

6780 std::streamsize xsputn(const char\_type\*, std::streamsize count) override { return count; }

6781

6782 int\_type overflow(int\_type ch) override {

6783 setp(std::begin(buf), std::end(buf));

6784 return traits\_type::not\_eof(ch);

6785 }

6786 } discardBuf;

6787

6788 public:

6789 DiscardOStream()

6790 : std::ostream(&discardBuf) {}

6791 } discardOut;

6792

6793 // the main function that does all the filtering and test running

6794 int Context::run() {

6795 using namespace detail;

6796

6797 // save the old context state in case such was setup - for using asserts out of a testing context

6798 auto old\_cs = g\_cs;

6799 // this is the current contest

6800 g\_cs = p;

6801 is\_running\_in\_test = true;

6802

6803 g\_no\_colors = p->no\_colors;

6804 p->resetRunData();

6805

6806 std::fstream fstr;

6807 if(p->cout == nullptr) {

6808 if(p->quiet) {

6809 p->cout = &discardOut;

6810 } else if(p->out.size()) {

6811 // to a file if specified

6812 fstr.open(p->out.c\_str(), std::fstream::out);

6813 p->cout = &fstr;

6814 } else {

6815 #ifndef DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

6816 // stdout by default

6817 p->cout = &std::cout;

6818 #else // DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

6819 return EXIT\_FAILURE;

6820 #endif // DOCTEST\_CONFIG\_NO\_INCLUDE\_IOSTREAM

6821 }

6822 }

6823

6824 FatalConditionHandler::allocateAltStackMem();

6825

6826 auto cleanup\_and\_return = [&]() {

6827 FatalConditionHandler::freeAltStackMem();

6828

6829 if(fstr.is\_open())

6830 fstr.close();

6831

6832 // restore context

6833 g\_cs = old\_cs;

6834 is\_running\_in\_test = false;

6835

6836 // we have to free the reporters which were allocated when the run started

6837 for(auto& curr : p->reporters\_currently\_used)

6838 delete curr;

6839 p->reporters\_currently\_used.clear();

6840

6841 if(p->numTestCasesFailed && !p->no\_exitcode)

6842 return EXIT\_FAILURE;

6843 return EXIT\_SUCCESS;

6844 };

6845

6846 // setup default reporter if none is given through the command line

6847 if(p->filters[8].empty())

6848 p->filters[8].push\_back("console");

6849

6850 // check to see if any of the registered reporters has been selected

6851 for(auto& curr : getReporters()) {

6852 if(matchesAny(curr.first.second.c\_str(), p->filters[8], false, p->case\_sensitive))

6853 p->reporters\_currently\_used.push\_back(curr.second(\*g\_cs));

6854 }

6855

6856 // TODO: check if there is nothing in reporters\_currently\_used

6857

6858 // prepend all listeners

6859 for(auto& curr : getListeners())

6860 p->reporters\_currently\_used.insert(p->reporters\_currently\_used.begin(), curr.second(\*g\_cs));

6861

6862 #ifdef DOCTEST\_PLATFORM\_WINDOWS

6863 if(isDebuggerActive() && p->no\_debug\_output == false)

6864 p->reporters\_currently\_used.push\_back(new DebugOutputWindowReporter(\*g\_cs));

6865 #endif // DOCTEST\_PLATFORM\_WINDOWS

6866

6867 // handle version, help and no\_run

6868 if(p->no\_run || p->version || p->help || p->list\_reporters) {

6869 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(report\_query, QueryData());

6870

6871 return cleanup\_and\_return();

6872 }

6873

6874 std::vector<const TestCase\*> testArray;

6875 for(auto& curr : getRegisteredTests())

6876 testArray.push\_back(&curr);

6877 p->numTestCases = testArray.size();

6878

6879 // sort the collected records

6880 if(!testArray.empty()) {

6881 if(p->order\_by.compare("file", true) == 0) {

6882 std::sort(testArray.begin(), testArray.end(), fileOrderComparator);

6883 } else if(p->order\_by.compare("suite", true) == 0) {

6884 std::sort(testArray.begin(), testArray.end(), suiteOrderComparator);

6885 } else if(p->order\_by.compare("name", true) == 0) {

6886 std::sort(testArray.begin(), testArray.end(), nameOrderComparator);

6887 } else if(p->order\_by.compare("rand", true) == 0) {

6888 std::srand(p->rand\_seed);

6889

6890 // random\_shuffle implementation

6891 const auto first = &testArray[0];

6892 for(size\_t i = testArray.size() - 1; i > 0; --i) {

6893 int idxToSwap = std::rand() % (i + 1);

6894

6895 const auto temp = first[i];

6896

6897 first[i] = first[idxToSwap];

6898 first[idxToSwap] = temp;

6899 }

6900 } else if(p->order\_by.compare("none", true) == 0) {

6901 // means no sorting - beneficial for death tests which call into the executable

6902 // with a specific test case in mind - we don't want to slow down the startup times

6903 }

6904 }

6905

6906 std::set<String> testSuitesPassingFilt;

6907

6908 bool query\_mode = p->count || p->list\_test\_cases || p->list\_test\_suites;

6909 std::vector<const TestCaseData\*> queryResults;

6910

6911 if(!query\_mode)

6912 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_run\_start, DOCTEST\_EMPTY);

6913

6914 // invoke the registered functions if they match the filter criteria (or just count them)

6915 for(auto& curr : testArray) {

6916 const auto& tc = \*curr;

6917

6918 bool skip\_me = false;

6919 if(tc.m\_skip && !p->no\_skip)

6920 skip\_me = true;

6921

6922 if(!matchesAny(tc.m\_file.c\_str(), p->filters[0], true, p->case\_sensitive))

6923 skip\_me = true;

6924 if(matchesAny(tc.m\_file.c\_str(), p->filters[1], false, p->case\_sensitive))

6925 skip\_me = true;

6926 if(!matchesAny(tc.m\_test\_suite, p->filters[2], true, p->case\_sensitive))

6927 skip\_me = true;

6928 if(matchesAny(tc.m\_test\_suite, p->filters[3], false, p->case\_sensitive))

6929 skip\_me = true;

6930 if(!matchesAny(tc.m\_name, p->filters[4], true, p->case\_sensitive))

6931 skip\_me = true;

6932 if(matchesAny(tc.m\_name, p->filters[5], false, p->case\_sensitive))

6933 skip\_me = true;

6934

6935 if(!skip\_me)

6936 p->numTestCasesPassingFilters++;

6937

6938 // skip the test if it is not in the execution range

6939 if((p->last < p->numTestCasesPassingFilters && p->first <= p->last) ||

6940 (p->first > p->numTestCasesPassingFilters))

6941 skip\_me = true;

6942

6943 if(skip\_me) {

6944 if(!query\_mode)

6945 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_case\_skipped, tc);

6946 continue;

6947 }

6948

6949 // do not execute the test if we are to only count the number of filter passing tests

6950 if(p->count)

6951 continue;

6952

6953 // print the name of the test and don't execute it

6954 if(p->list\_test\_cases) {

6955 queryResults.push\_back(&tc);

6956 continue;

6957 }

6958

6959 // print the name of the test suite if not done already and don't execute it

6960 if(p->list\_test\_suites) {

6961 if((testSuitesPassingFilt.count(tc.m\_test\_suite) == 0) && tc.m\_test\_suite[0] != '\0') {

6962 queryResults.push\_back(&tc);

6963 testSuitesPassingFilt.insert(tc.m\_test\_suite);

6964 p->numTestSuitesPassingFilters++;

6965 }

6966 continue;

6967 }

6968

6969 // execute the test if it passes all the filtering

6970 {

6971 p->currentTest = &tc;

6972

6973 p->failure\_flags = TestCaseFailureReason::None;

6974 p->seconds = 0;

6975

6976 // reset atomic counters

6977 p->numAssertsFailedCurrentTest\_atomic = 0;

6978 p->numAssertsCurrentTest\_atomic = 0;

6979

6980 p->fullyTraversedSubcases.clear();

6981

6982 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_case\_start, tc);

6983

6984 p->timer.start();

6985

6986 bool run\_test = true;

6987

6988 do {

6989 // reset some of the fields for subcases (except for the set of fully passed ones)

6990 p->reachedLeaf = false;

6991 // May not be empty if previous subcase exited via exception.

6992 p->subcaseStack.clear();

6993 p->currentSubcaseDepth = 0;

6994

6995 p->shouldLogCurrentException = true;

6996

6997 // reset stuff for logging with INFO()

6998 p->stringifiedContexts.clear();

6999

7000 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

7001 try {

7002 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

7003 // MSVC 2015 diagnoses fatalConditionHandler as unused (because reset() is a static method)

7004 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4101) // unreferenced local variable

7005 FatalConditionHandler fatalConditionHandler; // Handle signals

7006 // execute the test

7007 tc.m\_test();

7008 fatalConditionHandler.reset();

7009 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

7010 #ifndef DOCTEST\_CONFIG\_NO\_EXCEPTIONS

7011 } catch(const TestFailureException&) {

7012 p->failure\_flags |= TestCaseFailureReason::AssertFailure;

7013 } catch(...) {

7014 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_case\_exception,

7015 {translateActiveException(), false});

7016 p->failure\_flags |= TestCaseFailureReason::Exception;

7017 }

7018 #endif // DOCTEST\_CONFIG\_NO\_EXCEPTIONS

7019

7020 // exit this loop if enough assertions have failed - even if there are more subcases

7021 if(p->abort\_after > 0 &&

7022 p->numAssertsFailed + p->numAssertsFailedCurrentTest\_atomic >= p->abort\_after) {

7023 run\_test = false;

7024 p->failure\_flags |= TestCaseFailureReason::TooManyFailedAsserts;

7025 }

7026

7027 if(!p->nextSubcaseStack.empty() && run\_test)

7028 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_case\_reenter, tc);

7029 if(p->nextSubcaseStack.empty())

7030 run\_test = false;

7031 } while(run\_test);

7032

7033 p->finalizeTestCaseData();

7034

7035 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_case\_end, \*g\_cs);

7036

7037 p->currentTest = nullptr;

7038

7039 // stop executing tests if enough assertions have failed

7040 if(p->abort\_after > 0 && p->numAssertsFailed >= p->abort\_after)

7041 break;

7042 }

7043 }

7044

7045 if(!query\_mode) {

7046 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(test\_run\_end, \*g\_cs);

7047 } else {

7048 QueryData qdata;

7049 qdata.run\_stats = g\_cs;

7050 qdata.data = queryResults.data();

7051 qdata.num\_data = unsigned(queryResults.size());

7052 DOCTEST\_ITERATE\_THROUGH\_REPORTERS(report\_query, qdata);

7053 }

7054

7055 return cleanup\_and\_return();

7056 }

7057

7058 DOCTEST\_DEFINE\_INTERFACE(IReporter)

7059

7060 int IReporter::get\_num\_active\_contexts() { return detail::g\_infoContexts.size(); }

7061 const IContextScope\* const\* IReporter::get\_active\_contexts() {

7062 return get\_num\_active\_contexts() ? &detail::g\_infoContexts[0] : nullptr;

7063 }

7064

7065 int IReporter::get\_num\_stringified\_contexts() { return detail::g\_cs->stringifiedContexts.size(); }

7066 const String\* IReporter::get\_stringified\_contexts() {

7067 return get\_num\_stringified\_contexts() ? &detail::g\_cs->stringifiedContexts[0] : nullptr;

7068 }

7069

7070 namespace detail {

7071 void registerReporterImpl(const char\* name, int priority, reporterCreatorFunc c, bool isReporter) {

7072 if(isReporter)

7073 getReporters().insert(reporterMap::value\_type(reporterMap::key\_type(priority, name), c));

7074 else

7075 getListeners().insert(reporterMap::value\_type(reporterMap::key\_type(priority, name), c));

7076 }

7077 } // namespace detail

7078

7079 } // namespace doctest

7080

7081 #endif // DOCTEST\_CONFIG\_DISABLE

7082

7083 #ifdef DOCTEST\_CONFIG\_IMPLEMENT\_WITH\_MAIN

7084 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_WITH\_PUSH(4007) // 'function' : must be 'attribute' - see issue #182

7085 int main(int argc, char\*\* argv) { return doctest::Context(argc, argv).run(); }

7086 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

7087 #endif // DOCTEST\_CONFIG\_IMPLEMENT\_WITH\_MAIN

7088

7089 DOCTEST\_CLANG\_SUPPRESS\_WARNING\_POP

7090 DOCTEST\_MSVC\_SUPPRESS\_WARNING\_POP

7091 DOCTEST\_GCC\_SUPPRESS\_WARNING\_POP

7092

7093 DOCTEST\_SUPPRESS\_COMMON\_WARNINGS\_POP

7094

7095 #endif // DOCTEST\_LIBRARY\_IMPLEMENTATION

7096 #endif // DOCTEST\_CONFIG\_IMPLEMENT

7097

7098 #ifdef DOCTEST\_UNDEF\_WIN32\_LEAN\_AND\_MEAN

7099 #undef WIN32\_LEAN\_AND\_MEAN

7100 #undef DOCTEST\_UNDEF\_WIN32\_LEAN\_AND\_MEAN

7101 #endif // DOCTEST\_UNDEF\_WIN32\_LEAN\_AND\_MEAN

7102

7103 #ifdef DOCTEST\_UNDEF\_NOMINMAX

7104 #undef NOMINMAX

7105 #undef DOCTEST\_UNDEF\_NOMINMAX

7106 #endif // DOCTEST\_UNDEF\_NOMINMAX

## Farmer.cpp

1 #include <iostream>

2

3 #include "Farmer.hpp"

4 #include "Item.hpp"

5 #include "Inventory.hpp"

6 #include "ItemsStorage.hpp"

7 #include "Player.hpp"

8 #include "TerminalPalette.hpp"

9

10 void Farmer::getWheat(int quantidade, Player \*p, ItemsStorage \*storage)

11 {

12 try

13 {

14 if (p->getInventory()->hasItem(0, quantidade))

15 {

16 for (int i = 0; i < quantidade; i++)

17 {

18 p->getInventory()->remove(0, 1);

19 Item item = storage->findByName("Trigo");

20 p->getInventory()->insert(item.getId(), item);

21 p->addXp(15.0);

22 }

23 }

24 else

25 {

26 throw std::runtime\_error("Não foi encontrada semente");

27 }

28 }

29 catch (std::runtime\_error const &e)

30 {

31 std::cout << color::redi << "Nao foi possivel plantar o trigo... verifique se possui todos os itens necessarios" << color::off << std::endl;

32 return;

33 }

34 std::cout << color::cyan << "Trigos adicionados ao seu inventario!" << color::off << std::endl;

35 }

36

37 void Farmer::getCarrot(int quantidade, Player \*p, ItemsStorage \*storage)

38 {

39 try

40 {

41 if (p->getInventory()->hasItem(0, quantidade))

42 {

43 for (int i = 0; i < quantidade; i++)

44 {

45 Item cenoura = storage->findByName("Cenoura");

46 p->getInventory()->insert(cenoura.getId(), cenoura);

47 p->getInventory()->remove(0, 1);

48 p->addXp(15.0);

49 }

50 }

51 else

52 {

53 throw std::runtime\_error("Não foi encontrada semente");

54 }

55 }

56 catch (std::runtime\_error const &e)

57 {

58 std::cout << color::redi << "Nao foi possivel plantar a cenoura... verifique se possui todos os itens necessarios" << color::off << std::endl;

59 return;

60 }

61 std::cout << color::cyan << "Cenouras adicionadas ao seu inventario!" << color::off << std::endl;

62 }

## Farmer.hpp File Reference

Contém a definição da classe **Farmer**.

#include "Item.hpp"

#include "Inventory.hpp"

### Classes

class **Farmer***Representa um agricultor no jogo.*

### Detailed Description

Contém a definição da classe **Farmer**.

## Farmer.hpp

Go to the documentation of this file.

1 #pragma once

2 #include "Item.hpp"

3 #include "Inventory.hpp"

4

14 class Farmer

15 {

16 public:

23 void getWheat(int quantidade, Player \*player, ItemsStorage \*storage);

24

31 void getCarrot(int quantidade, Player \*player, ItemsStorage \*storage);

32 };

## Fisherman.cpp

1 #include <iostream>

2 #include <ctime>

3 #include <cstdlib>

4 #include <algorithm>

5 #include <vector>

6

7 #include "Player.hpp"

8 #include "ItemsStorage.hpp"

9 #include "Fisherman.hpp"

10 #include "Item.hpp"

11 #include "Inventory.hpp"

12 #include "TerminalPalette.hpp"

13

14 void Fisherman::fish(Player \*p, ItemsStorage \*storage)

15 {

16 Item peixe = storage->findRandom(1);

17 p->getInventory()->insert(peixe.getId(), peixe);

18 p->addXp(peixe.getUnitaryPrice() \* 2.0);

19 std::cout << color::cyan << "Voce pescou um " << peixe.getName() << "!" << color::off << std::endl;

20 }

21

22 void Fisherman::cleanFish(Player \*p)

23 {

24 Inventory \*playerInventory = p->getInventory();

25 bool limpou = false;

26 for (auto &entry : playerInventory->list())

27 {

28 Item &item = entry.second.first;

29

30 if ((item.getId() >= 5 && item.getId() <= 11) && !item.getStatus())

31 {

32 double cleanedPrice = item.getUnitaryPrice() \* 1.1;

33 playerInventory->updateItemPrice(item.getId(), cleanedPrice);

34 playerInventory->updateItemStatus(item.getId());

35 limpou = true;

36 }

37 }

38 if (limpou)

39 {

40 p->addXp(15.0);

41 std::cout << color::cyan << "Agora seus peixes valem mais!" << color::off << std::endl;

42 }

43 else

44 {

45 std::cout << color::red << "Ops, parece que todos os seus peixes ja estao limpos..." << color::off << std::endl;

46 }

47 }

## Fisherman.hpp File Reference

Contém a definição da classe **Fisherman**.

#include "Item.hpp"

#include "Inventory.hpp"

#include "ItemsStorage.hpp"

### Classes

class **Fisherman***Representa um pescador no jogo.*

### Detailed Description

Contém a definição da classe **Fisherman**.

## Fisherman.hpp

Go to the documentation of this file.

1 #pragma once

2 #include "Item.hpp"

3 #include "Inventory.hpp"

4 #include "ItemsStorage.hpp"

5

15 class Fisherman

16 {

17 public:

23 void fish(Player \*player, ItemsStorage \*storage);

24

29 void cleanFish(Player \*player);

30 };

## Inventory.cpp

1 #include "Inventory.hpp"

2 #include "Item.hpp"

3 #include "TerminalPalette.hpp"

4 #include <iostream>

5 #include <set>

6

7 Inventory::Inventory(unsigned capacity)

8 {

9 \_inventorySize = capacity;

10 }

11

12 bool Inventory::hasItem(int id, unsigned quantidade)

13 {

14 auto find = \_inventory.find(id);

15 return (find != \_inventory.end() && find->second.second >= quantidade);

16 }

17

18 bool Inventory::insert(int id, Item item)

19 {

20 // std::cout <<color::grayf << "Item atual sendo inserido : " << color::off << std::endl;

21

22 auto find = \_inventory.find(id);

23 if (\_inventory.find(id) != \_inventory.end())

24 find->second.second += 1;

25 else

26 {

27 \_inventory.insert({id, std::make\_pair(item, 1)});

28 find = \_inventory.find(id);

29 }

30

31 // std::cout <<color::grayf << "Qtd do item atual:" << find->second.second << color::off << std::endl;

32

33 return true;

34 }

35

36 bool Inventory::remove(int id, unsigned quantidade)

37 {

38 auto find = \_inventory.find(id);

39 if (find != \_inventory.end())

40 {

41 if (find->second.second >= quantidade)

42 {

43 find->second.second -= quantidade;

44 }

45 else

46 {

47 return false;

48 }

49 }

50 return true;

51 }

52

53 void Inventory::printInventory()

54 {

55 std::cout << color::yellows << "--------------- LISTAGEM DO INVENTARIO ---------------" << color::off << std::endl;

56 for (auto stack : \_inventory)

57 {

58 std::cout << "Item: " << stack.second.first.getName() << " | Qtd:" << stack.second.second << " | Preco total:" << stack.second.first.getUnitaryPrice() \* static\_cast<double>(stack.second.second) << std::endl;

59 }

60 std::cout << std::endl;

61 }

62

63 std::map<int, std::pair<Item, unsigned>> Inventory::list()

64 {

65 return \_inventory;

66 }

67

68 void Inventory::updateItemPrice(int id, double novoPreco)

69 {

70 auto it = \_inventory.find(id);

71

72 if (it != \_inventory.end())

73 {

74 it->second.first.changePrice(novoPreco);

75 }

76 }

77

78 void Inventory::updateItemStatus(int id)

79 {

80 auto it = \_inventory.find(id);

81

82 if (it != \_inventory.end())

83 {

84 it->second.first.changeStatus();

85 }

86 }

## Inventory.hpp File Reference

Contém a definição da classe **Inventory**.

#include <map>

#include "Item.hpp"

### Classes

class **Inventory***Representa o inventário de um jogador.*

### Detailed Description

Contém a definição da classe **Inventory**.

## Inventory.hpp

Go to the documentation of this file.

1 #pragma once

2 #include <map>

3 #include "Item.hpp"

4

14 class Inventory

15 {

16 public:

21 Inventory(unsigned capacity);

22

29 bool hasItem(int id, unsigned quantidade);

30

37 bool insert(int id, Item item);

38

45 bool remove(int id, unsigned quantidade);

46

50 void printInventory();

51

56 std::map<int, std::pair<Item, unsigned>> list();

57

63 void updateItemPrice(int id, double novoPreco);

64

69 void updateItemStatus(int id);

70

71 private:

72 unsigned \_inventorySize;

73 std::map<int, std::pair<Item, unsigned>> \_inventory;

74 };

## Item.cpp

1 #include <string>

2

3 #include "Item.hpp"

4

5 Item::Item(int id, std::string name, double price, bool status)

6 {

7 \_id = id;

8 \_name = name;

9 \_price = price;

10 \_status = status;

11 }

12

13 std::string Item::getName()

14 {

15 return \_name;

16 }

17

18 int Item::getId()

19 {

20 return \_id;

21 }

22

23 double Item::getUnitaryPrice()

24 {

25 return \_price;

26 }

27

28 bool Item::getStatus()

29 {

30 return \_status;

31 }

32

33 void Item::changePrice(double novo)

34 {

35 \_price = novo;

36 }

37

38 void Item::changeStatus()

39 {

40 \_status = !\_status;

41 }

## Item.hpp File Reference

Contém a definição da classe **Item**.

#include <map>

#include <string>

### Classes

class **Item***Representa um item no sistema.*

### Detailed Description

Contém a definição da classe **Item**.

## Item.hpp

Go to the documentation of this file.

1 #pragma once

2 #include <map>

3 #include <string>

4

14 class Item {

15 public:

23 Item(int id, std::string name, double price, bool status);

24

29 std::string getName();

30

35 double getUnitaryPrice();

36

41 int getId();

42

47 bool getStatus();

48

53 void changePrice(double novo);

54

58 void changeStatus();

59

60 private:

61 int \_id;

62 std::string \_name;

63 double \_price;

64 bool \_status;

65 };

## ItemsStorage.hpp File Reference

Contém a definição da classe **ItemsStorage**, uma subclasse de **StorageManager**.

#include "StorageManager.hpp"

#include "ItemsStorage.hpp"

#include "Item.hpp"

### Classes

class **ItemsStorage***Gerencia o armazenamento de itens.*

### Detailed Description

Contém a definição da classe **ItemsStorage**, uma subclasse de **StorageManager**.

## ItemsStorage.hpp

Go to the documentation of this file.

1 #pragma once

2 #include "StorageManager.hpp"

3 #include "ItemsStorage.hpp"

4 #include "Item.hpp"

14 class ItemsStorage : public StorageManager<Item>

15 {

16 public:

21 ItemsStorage(char const \*file\_path);

22

26 ~ItemsStorage();

27

32 void write(const Item &item) override;

33

38 void read(int totalRegistros) override;

39

46 Item findRandom(const int tipo);

47

48 Item findByName(const std::string name);

49 };

## Livestocker.cpp

1 #include <iostream>

2

3 #include "Player.hpp"

4 #include "Livestocker.hpp"

5 #include "Item.hpp"

6 #include "Inventory.hpp"

7 #include "ItemsStorage.hpp"

8 #include "TerminalPalette.hpp"

9

10 void Livestocker::getBacon(int quantidade, Player \*p, ItemsStorage \*storage)

11 {

12 try

13 {

14 if (p->getInventory()->hasItem(3, quantidade))

15 {

16 for (int i = 0; i < quantidade; i++)

17 {

18 Item bacon = storage->findByName("Bacon");

19 p->getInventory()->insert(bacon.getId(), bacon);

20 p->getInventory()->remove(3, 1);

21 }

22 }

23 else

24 {

25 throw std::runtime\_error("Não foi encontrada cenoura");

26 }

27 }

28 catch (std::runtime\_error const &e)

29 {

30 std::cout << color::redi << "Nao foi possivel produzir bacon... verifique se possui todos os itens necessarios" << color::off << std::endl;

31 return;

32 }

33 p->addXp(15.0);

34 std::cout << color::cyan << "Bacons adicionados ao seu inventario!" << color::off << std::endl;

35 }

36

37 void Livestocker::getEgg(int quantidade, Player \*p, ItemsStorage \*storage)

38 {

39 try

40 {

41 if (p->getInventory()->hasItem(3, quantidade))

42 {

43 for (int i = 0; i < quantidade; i++)

44 {

45 Item ovo = storage->findByName("Ovo");

46 p->getInventory()->insert(ovo.getId(), ovo);

47 p->getInventory()->remove(2, quantidade);

48 }

49 }

50 else

51 {

52 throw std::runtime\_error("Não foi encontrado trigo");

53 }

54 }

55 catch (std::runtime\_error const &e)

56 {

57 std::cout << color::redi << "Nao foi possivel produzir ovo... verifique se possui todos os itens necessarios" << color::off << std::endl;

58 return;

59 }

60 p->addXp(15.0);

61 std::cout << color::cyan << "Ovos adicionados ao seu inventario!" << color::off << std::endl;

62 }

## Livestocker.hpp File Reference

Contém a definição da classe **Livestocker**.

#include "Item.hpp"

#include "Inventory.hpp"

### Classes

class **Livestocker***Representa um criador de animais no jogo.*

### Detailed Description

Contém a definição da classe **Livestocker**.

## Livestocker.hpp

Go to the documentation of this file.

1 #pragma once

2 #include "Item.hpp"

3 #include "Inventory.hpp"

4

14 class Livestocker

15 {

16 public:

23 void getBacon(int quantidade, Player \*player, ItemsStorage \*storage);

24

31 void getEgg(int quantidade, Player \*player, ItemsStorage \*storage);

32 };

## LojaDeItens.cpp

1 #include "LojaDeItens.hpp"

2 #include <iostream>

3

4 LojaDeItens::LojaDeItens(Player\* player, Inventory\* inventory, std::vector<Item>\* items)

5 : \_player(player), \_inventory(inventory), \_items(items) {}

6

7 void LojaDeItens::exibirItens()

8 {

9 std::cout << "Itens da Loja:\n";

10 for (const auto& item : \*\_items) {

11 std::cout << item.\_id << " - " << item.\_name << " - $" << item.\_price << "\n";

12 }

13 std::cout << "\n";

14

15 std::cout << "Escolha uma opção:" << std::endl;

16 std::cout << "1. Comprar Item" << std::endl;

17 std::cout << "2. Vender Item" << std::endl;

18

19 int opcao;

20 std::cin >> opcao;

21

22 switch (opcao)

23 {

24 case 1:

25 comprarItem();

26 break;

27 case 2:

28 venderItem();

29 break;

30 default:

31 std::cout << "Opção inválida!" << std::endl;

32 }

33 }

34

35 void LojaDeItens::comprarItem()

36 {

37 \_inventory->list(); // Chama a função list do inventário para exibir os itens

38 // Adicione lógica para compra de item aqui

39

40 //Jogador deve escolher um item

41 int escolha;

42 std::cout << "Escolha um item para comprar: ";

43 std::cin >> escolha;

44

45 if (escolha >= 0 && escolha < \_items.size()) {

46 Item itemComprado = (\*\_items)[escolha];

47

48 // Verificar se o jogador tem dinheiro suficiente

49 if (\_player->\_money >= itemComprado.\_price) {

50 // Adicionar item ao inventário do jogador

51 \_player->inventory.insert(itemComprado);

52 // Remover dinheiro do jogador

53 \_player->\_money -= itemComprado.\_price;

54

55 std::cout << "Você comprou " << itemComprado.\_name << " por $" << itemComprado.\_price << "\n\n";

56 } else {

57 std::cout << "Dinheiro insuficiente!\n\n";

58 }

59 } else {

60 std::cout << "Escolha inválida!\n\n";

61 return comprarItem();

62 }

63 }

64 }

65

66 void LojaDeItens::venderItem()

67 {

68 \_inventory->list(); // Chama a função list do inventário para exibir os itens

69 // Adicione lógica para venda de item aqui

70 int escolha;

71 std::cout << "Escolha um item para vender: ";

72 std::cin >> escolha;

73

74 if (escolha >= 0 && escolha < \_player->inventory.size()) {

75 Item itemVendido = \_player->inventory[escolha];

76

77 //Adicionar os itens à loja

78 \_items->push\_back(itemVendido);

79

80 //Adicionar dinheiro ao jogador

81 \_player->\_money += itemVendido.\_price;

82

83 //Removendo item do inventário do jogador

84 \_player->inventory.remove(\_player->inventory.begin() + escolha);

85

86 std::cout << "Você vendeu seu item" << itemVendido.\_name << " por $" << itemVendido.\_price << " \n\n ";

87 } else {

88 std::cout << "Escolha inválida!\n\n";

89 return venderItem();

90 }

91 }

## LojaDeItens.hpp

1 #pragma once

2 #include <map>

3 #include <string>

4 #include "Item.hpp"

5 #include "Player.hpp"

6 #include "Inventory.hpp"

7

9 class LojaDeItens

10 {

11 public:

12

16 LojaDeItens(Player\* player, Inventory\* inventory, Item\* item);

17

19 void exibirItens();

21 void LojaDeItens::comprarItem();

23 void LojaDeItens::venderItem();

24

25 private:

26 Player\* \_player;

27 Inventory\* \_inventory;

28 Item\* \_items;

29 };

## Menu.cpp

1 #include "Menu.hpp"

2 #include "ItemsStorage.hpp"

3 #include "Sistema.hpp"

4 #include "Player.hpp"

5 #include "Farmer.hpp"

6 #include "Fisherman.hpp"

7 #include "Miner.hpp"

8 #include "Livestocker.hpp"

9 #include "TerminalPalette.hpp"

10 #include <iostream>

11

12 Menu::Menu(){};

13

14 void Menu::listarAtividades(Player \*p, ItemsStorage \*storage, Sistema \*sistema)

15 {

16

17 int option;

18 int quantidade;

19 Farmer f;

20 Livestocker l;

21 Fisherman i;

22 Miner m;

23 do

24 {

25 std::cout << color::greens << "Atividades disponiveis:" << color::off << std::endl;

26 std::cout << color::green;

27 sistema->printText(" 1- Colher trigo");

28 sistema->printText(" 2- Colher cenouras");

29 sistema->printText(" 3- Alimentar as galinhas");

30 sistema->printText(" 4- Cuidar dos porcos");

31 sistema->printText(" 5- Pescar");

32 sistema->printText(" 6- Limpar os peixes");

33 sistema->printText(" 7- Minerar");

34 sistema->printText(" 8- Refinar os minerios");

35 sistema->printText(" 9- Loja");

36 sistema->printText(" 10- Ver inventario");

37 sistema->printText(" 11- Ver perfil");

38 sistema->printText(" 0- Sair");

39 std::cout << color::off;

40

41 do

42 {

43 std::cin >> option;

44 if (std::cin.fail())

45 {

46 std::cin.clear();

47 std::cout << color::redf << "Digite uma tarefa válida" << color::off << std::endl;

48 }

49 else if (option < 0 || option > 11)

50 {

51 std::cout << color::grayb << "Uhmm, ainda nao sabemos fazer essa tarefa..." << color::off << std::endl;

52 }

53 } while (std::cin.fail() || option < 0 || option > 11);

54

55 switch (option)

56 {

57 case 1:

58 std::cout << color::grayb << "Quantos trigos deseja plantar?" << color::off << std::endl;

59 std::cin >> quantidade;

60 f.getWheat(quantidade, p, storage);

61 break;

62 case 2:

63 std::cout << color::grayb << "Quantas cenouras deseja plantar?" << color::off << std::endl;

64 std::cin >> quantidade;

65 f.getCarrot(quantidade, p, storage);

66 break;

67 case 3:

68 std::cout << color::grayb << "Quantas galinhas deseja alimentar?" << color::off << std::endl;

69 std::cin >> quantidade;

70 l.getEgg(quantidade, p, storage);

71 break;

72 case 4:

73 std::cout << color::grayb << "Quantos porcos deseja cuidar?" << color::off << std::endl;

74 std::cin >> quantidade;

75 l.getBacon(quantidade, p, storage);

76 break;

77 case 5:

78 i.fish(p, storage);

79 break;

80 case 6:

81 i.cleanFish(p);

82 break;

83 case 7:

84 m.ore(p, storage);

85 break;

86 case 8:

87 m.refine(p);

88 break;

89 case 9:

90 // exibir a loja

91 break;

92 case 10:

93 p->printInventory();

94 break;

95 case 11:

96 p->displayProfile();

97 break;

98 default:

99 break;

100 }

101 std::cout << std::endl;

102 } while (option != 0);

103 }

## Menu.hpp

1 #pragma once

2

3 #include <string>

4 class Player; // Forward declaration

5 class ItemsStorage; // Forward declaration

6 class Sistema; // Forward declaration

7

9 class Menu

10 {

11 public:

13 Menu();

14

17 void listarAtividades(Player \*p, ItemsStorage \*storage, Sistema \*sistema);

18

19 // Métodos comentados para remoção temporária

20 // virtual ~StorageManager();

21 // void write(const T &data);

22 // void read(const T &data, int totalRegistros);

23 // virtual void findOne(char const \*data);

24

25 private:

26 // std::ifstream \_file;

27 };

28

29 // Cabeçalho comentado para remoção temporária

30 // #include "StorageManager.tpp"

## Miner.cpp

1 #include <iostream>

2 #include <ctime>

3 #include <cstdlib>

4 #include <algorithm>

5 #include <vector>

6

7 #include "Player.hpp"

8 #include "Miner.hpp"

9 #include "Item.hpp"

10 #include "Inventory.hpp"

11 #include "ItemsStorage.hpp"

12 #include "TerminalPalette.hpp"

13

14 void Miner::ore(Player \*p, ItemsStorage \*storage)

15 {

16 Item minerio = storage->findRandom(2);

17 p->getInventory()->insert(minerio.getId(), minerio);

18 p->addXp(minerio.getUnitaryPrice() \* 2.0);

19 std::cout << color::cyan << "Voce mineirou 1 " << minerio.getName() << "!" << color::off << std::endl;

20 }

21

22 void Miner::refine(Player \*p)

23 {

24 Inventory \*playerInventory = p->getInventory();

25 bool refinou = false;

26 for (auto &entry : playerInventory->list())

27 {

28 Item &item = entry.second.first;

29

30 if ((item.getId() >= 11 || item.getId() <= 16) && !item.getStatus())

31 {

32 double refinedPrice = item.getUnitaryPrice() \* 1.1;

33 playerInventory->updateItemPrice(item.getId(), refinedPrice);

34 playerInventory->updateItemStatus(item.getId());

35 refinou = true;

36 }

37 }

38 if (refinou)

39 {

40

41 p->addXp(15.0);

42 std::cout << color::cyan << "Agora seus minerios valem mais!" << color::off << std::endl;

43 }

44 else

45 std::cout << color::red << "Ops, parece que seus minerios ja estao refinados..." << color::off << std::endl;

46 }

## MIner.hpp

1 #pragma once

2 #include "Item.hpp"

3 #include "Inventory.hpp"

4

5 class Miner

6 {

7 public:

12 void ore(Player \*p, ItemsStorage \*storage);

13

17 void refine(Player \*p);

18 };

## Player.cpp

1 #include "Player.hpp"

2 #include "Inventory.hpp"

3 #include "Item.hpp"

4 #include "TerminalPalette.hpp"

5 #include <iostream>

6

7 Player::Player() : \_name(""), \_level(0.0), \_xp(0.0), \_inventario(0){};

8 Player::Player(const std::string name) : \_name(name), \_level(0.0), \_xp(0.0), \_inventario(10){};

9

10 void Player::functeste()

11 {

12 Item ITEMTESTE(0, "Item teste 1", 10.3, true);

13 \_inventario.insert(ITEMTESTE.getId(), ITEMTESTE);

14 \_inventario.list();

15 }

16 void Player::printInventory()

17 {

18 \_inventario.printInventory();

19 }

20

21 void Player::addXp(float newXp)

22 {

23 \_xp += newXp;

24 if(\_xp >= 150.0){

25 \_level++;

26 \_xp = 0;

27 std::cout<<color::bluen << "Nivel atualizado: " << \_level <<color::off<< std::endl;

28 std::cout<<color::bluen << "Pontos de experiencia atualizados: " << \_xp <<color::off<< std::endl;

29 }

30 }

31

32 float Player::getLevel()

33 {

34 return \_level;

35 }

36

37 Inventory\* Player::getInventory()

38 {

39 return &\_inventario;

40 }

41

42 void Player::displayProfile()

43 {

44 std::cout << color::blues << "--------------- MEU PERFIL ---------------" << color::off<< std::endl;

45 std::cout << color::bluen << "Nome de jogador: " << \_name << color::off<< std::endl;

46 std::cout << color::bluen << "Nivel: " << \_level << color::off<< std::endl;

47 std::cout << color::bluen << "Pontos de experiencia: " << \_xp << color::off<< std::endl;

48 std::cout << std::endl;

49 }

## Player.hpp

1 #pragma once

2

3 #include <string>

4 #include "Inventory.hpp"

5

7 class Player {

8 public:

10 Player();

11

14 Player(const std::string name);

15

17 void functeste();

18

20 void printInventory();

21

24 void addXp(float newXp);

25

27 void displayProfile();

28

31 float getLevel();

32

35 Inventory\* getInventory();

36

37 // Métodos comentados para remoção temporária

38 // virtual ~StorageManager();

39 // void write(const T &data);

40 // void read(const T &data, int totalRegistros);

41 // virtual void findOne(char const \*data);

42

43 private:

44 std::string \_name;

45 float \_level;

46 float \_xp;

47 Inventory \_inventario;

48 };

49

50 // Cabeçalho comentado para remoção temporária

51 // #include "StorageManager.tpp"

## Sistema.cpp

1 #include "Sistema.hpp"

2 #include "Player.hpp"

3 #include "Menu.hpp"

4 #include "ItemsStorage.hpp"

5 #include "TerminalPalette.hpp"

6 #include <iostream>

7 #include <iostream>

8 #include <cstdlib>

9 #include <ctime>

10 #include <windows.h>

11

12 Sistema::Sistema() : \_player(){};

13 void Sistema::criarPersonagem()

14 {

15 std::cout << color::purpleb << "Bem vindo(a) a sua nova fazenda!" << color::off << std::endl;

16 std::cout << "Informe um nome para o seu personagem:";

17 std::string nome;

18 std::cin >> nome;

19 // std::cout << "Escolha uma classe pro seu personagem:" << std::endl;

20 // std::cout << " 1- Fazendeiro" << std::endl;

21 // std::cout << " 2- Pescador" << std::endl;

22 // std::cout << " 3- Pecuarista" << std::endl;

23 // std::cout << " 4- Mineiro" << std::endl;

24 // int classe;

25 // do

26 // {

27 // std::cin >> classe;

28 // switch (classe)

29 // {

30 // case 1:

31 // std::cout << "Fazendeiro!" << std::endl;

32 // break;

33 // case 2:

34 // std::cout << "Pescador" << std::endl;

35 // /\* code \*/

36 // break;

37 // case 3:

38 // std::cout << "Pecuarista" << std::endl;

39 // /\* code \*/

40 // break;

41 // case 4:

42 // std::cout << "Mineiro" << std::endl;

43 // /\* code \*/

44 // break;

45

46 // default:

47 // std::cout << "Valor inválido, tente novamente." << std::endl;

48 // break;

49 // }

50 // } while (classe != 1 && classe != 2 && classe != 3 && classe != 4);

51 \_player = Player(nome);

52 ItemsStorage storage("./filetest.txt");

53 Menu menu = Menu();

54

55 menu.listarAtividades(&\_player, &storage, this);

56

57 \_player.printInventory();

58 };

59 void Sistema::printText(const std::string texto)

60 {

61 for (char x : texto)

62 {

63 unsigned seed = time(0);

64 std::cout << x;

65 srand(seed);

66 // std::cout << rand() % 300 << std::endl;

67 Sleep(rand() % 50);

68 }

69 std::cout << std::endl;

70 }

## Sistema.hpp

1 #pragma once

2 #include "Player.hpp"

3 #include <iostream>

4 #include <fstream>

5 #include <string>

6 #include <sstream>

7

9 class Sistema

10 {

11 public:

13 Sistema();

14

16 void criarPersonagem();

17

20 void printText(const std::string texto);

21

22 private:

23 Player \_player;

24 };

## StorageManager.hpp File Reference

Contém a definição da classe template **StorageManager**.

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include "StorageManager.tpp"

### Classes

class **StorageManager< T >***Classe modelo para gerenciamento de armazenamento.*

### Detailed Description

Contém a definição da classe template **StorageManager**.

## StorageManager.hpp

Go to the documentation of this file.

1 #include <iostream>

2 #include <fstream>

3 #include <string>

4 #include <sstream>

5

15 template <typename T>

16 class StorageManager

17 {

18 public:

23 StorageManager(char const \*file\_path);

24

25 // Métodos comentados para remoção temporária

26 // virtual ~StorageManager();

27

32 virtual void write(const T &data);

33

39 virtual void read(int totalRegistros);

40

45 // virtual T findRandom(const std::string classe);

46 // Métodos comentados para remoção temporária

47 // virtual void findOne(char const \*data);

48

49 protected:

50 std::ifstream \_file;

51 };

52

53 #include "StorageManager.tpp"

## TerminalPalette.hpp

1 #pragma once

2 #ifndef COLORS\_H

3 #define COLORS\_H

4 #include <iostream>

5

7 namespace color

8 {

9 const std::string gray = "\e[30;10m", // normal

10 grayn = "\e[30;1m", // negrito

11 grayf = "\e[30;2m", // fraco

12 grayi = "\e[30;3m", // itálico

13 grays = "\e[30;4m", // sublinhado

14 grayp = "\e[30;5m", // piscando

15 grayb = "\e[30;7m", // background

16 grayc = "\e[30;9m", // cancelado

17

18 red = "\e[31;10m", // normal

19 redn = "\e[31;1m", // negrito

20 redf = "\e[31;2m", // fraco

21 redi = "\e[31;3m", // itálico

22 reds = "\e[31;4m", // sublinhado

23 redp = "\e[31;5m", // piscando

24 redb = "\e[31;7m", // background

25 redc = "\e[31;9m", // cancelado

26

27 green = "\e[32;10m", // normal

28 greenn = "\e[32;1m", // negrito

29 greenf = "\e[32;2m", // fraco

30 greeni = "\e[32;3m", // itálico

31 greens = "\e[32;4m", // sublinhado

32 greenp = "\e[32;5m", // piscando

33 greenb = "\e[32;7m", // background

34 greenc = "\e[32;9m", // cancelado

35

36 yellow = "\e[33;10m", // normal

37 yellown = "\e[33;1m", // negrito

38 yellowf = "\e[33;2m", // fraco

39 yellowi = "\e[33;3m", // itálico

40 yellows = "\e[33;4m", // sublinhado

41 yellowp = "\e[33;5m", // piscando

42 yellowb = "\e[33;7m", // background

43 yellowc = "\e[33;9m", // cancelado

44

45 blue = "\e[34;10m", // normal

46 bluen = "\e[34;1m", // negrito

47 bluef = "\e[34;2m", // fraco

48 bluei = "\e[34;3m", // itálico

49 blues = "\e[34;4m", // sublinhado

50 bluep = "\e[34;5m", // piscando

51 blueb = "\e[34;7m", // background

52 bluec = "\e[34;9m", // cancelado

53

54 purple = "\e[35;10m", // normal

55 purplen = "\e[35;1m", // negrito

56 purplef = "\e[35;2m", // fraco

57 purplei = "\e[35;3m", // itálico

58 purples = "\e[35;4m", // sublinhado

59 purplep = "\e[35;5m", // piscando

60 purpleb = "\e[35;7m", // background

61 purplec = "\e[35;9m", // cancelado

62

63 cyan = "\e[36;10m", // normal

64 cyann = "\e[36;1m", // negrito

65 cyanf = "\e[36;2m", // fraco

66 cyani = "\e[36;3m", // itálico

67 cyans = "\e[36;4m", // sublinhado

68 cyanp = "\e[36;5m", // piscando

69 cyanb = "\e[36;7m", // background

70 cyanc = "\e[36;9m", // cancelado

71

72 white = "\e[38;10m", // normal

73 whiten = "\e[38;1m", // negrito

74 whitef = "\e[38;2m", // fraco

75 whitei = "\e[38;3m", // itálico

76 whites = "\e[38;4m", // sublinhado

77 whitep = "\e[38;5m", // piscando

78 whiteb = "\e[38;7m", // background

79 whitec = "\e[38;9m", // cancelado

80

81 off = "\e[m"; // desativa a cor personalizada

82 }

83

84 #endif

# Index