Date

v1.0

Généré par Doxygen 1.8.13

Table des matières

1 Une classe pour représenter une date

Un bon point d'entrée est celui de la documentation de l'espace de nom nvs.

2 Index des espaces de nommage

2.1 Liste des espaces de nommage

Liste de tous les espaces de nommage documentés avec une brève description :

3 Index des classes

3.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

4 Index des fichiers

4.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

5 Documentation des espaces de nommage

5.1 Référence de l'espace de nommage nvs

Espace de nom de Nicolas Vansteenkiste.

Classes

— class Date
Classe représentant une date.

Fonctions

- **std : :ostream** & operator<< (**std : :ostream** &out, const Date &in)
  Opérateur d'injection d'une Date dans un flux en sortie.
- bool operator< (const Date &lhs, const Date &rhs)
  Opérateur de comparaison de deux dates.
- bool operator== (const Date &lhs, const Date &rhs)
  Opérateur de test d'égalité de deux dates.
- **std : :string** to_string (const Date &in)
  Fonction de conversion d'une nvs : :Date en ***std : :string****.*
- Date today ()
  Fonction retournant la date actuelle (au moment de l'exécution) sous la forme d'une Date.
- bool leapYear (int year)
  Fonction pour déterminer si une année est bissextile.
- template<typename T >
  bool between (const T &value, T min, T max)
  Évaluation si une valeur est comprise entre deux autres.
- template<typename T >
  std : :string error_message (const T &value, T min, T max, const **std : :string** &msg="error : ")
  Production d'un message d'erreur.
- template<typename T >
  T validate (const T &value, const T &min, const T &max, const **std : :string** &msg="error : ")
  Validation d'une valeur entre deux bornes.

5.1.1 Description détaillée

Espace de nom de Nicolas Vansteenkiste.

5.1.2 Documentation des fonctions

5.1.2.1 between()

```
template<typename T >
bool nvs::between (
            const T & value,
            T min,
            T max )
```

Évaluation si une valeur est comprise entre deux autres.

Les opérateurs < et == doivent être disponibles pour le type `T`.

Paramètres

value	valeur à tester.
min	borne minimale.
max	borne maximale.

Renvoie

`true` si `min <= value <= max`, `false` sinon.

5.1.2.2 error_message()

```
template<typename T >
std::string nvs::error_message (
            const T & value,
            T min,
            T max,
            const  std::string & msg = "error : " )
```

Production d'un message d'erreur.

La fonction `to_string()` doit être disponible pour le type `T`.

Paramètres

value	valeur.
min	borne minimale.
max	borne maximale.
msg	en-tête du message.

Renvoie

message d'erreur.

Voir également

validate()

5.1.2.3 leapYear()

```
bool nvs::leapYear (
            int year )  [inline]
```

Fonction pour déterminer si une année est bissextile.

Paramètres

year	l'année à tester.

Renvoie

`true` si `year` est bissextile, `false` sinon.

Voir également

Date : :leapYear().

5.1.2.4 operator<()

```
bool nvs::operator< (
            const Date & lhs,
            const Date & rhs )  [inline]
```

Opérateur de comparaison de deux dates.

La valeur `true` est retournée si `lhs` est *strictement antérieure* à `rhs`. Lorsque `lhs` est *égale ou postérieure* à `rhs`, `false` est retourné.

Paramètres

lhs	une Date.
rhs	une autre Date.

Renvoie

`true` si lhs est strictement antérieure à rhs, `false` sinon.

5.1.2.5 operator<<()

```
std::ostream& nvs::operator<< (
             std::ostream & out,
            const Date & in )
```

Opérateur d'injection d'une Date dans un flux en sortie.

La Date est injectée sous la forme `jj/mm/aaaa`, par exemple `26/04/2016`.

Il s'agit de la même forme que celle retournée par nvs : :Date : :to_string().

Paramètres

out	le flux dans lequel la Date est injectée.
in	la Date injectée dans le flux en sortie.

Renvoie

le flux dans lequel la Date a été injectée.

5.1.2.6 operator==()

```
bool nvs::operator== (
            const Date & lhs,
            const Date & rhs )  [inline]
```

Opérateur de test d'égalité de deux dates.

Paramètres

lhs	une Date.
rhs	une autre Date.

Renvoie

`true` si les deux dates sont identiques, `false` sinon.

5.1.2.7 to_string()

```
std::string nvs::to_string (
            const Date & in )  [inline]
```

Fonction de conversion d'une nvs : :Date en **std : :string**.

La **`std::string`** retournée est `"jj/mm/aaaa"`, par exemple `"26/04/2016"`.

Il s'agit de la même forme que celle lors de l'injection d'une Date dans un flux en sortie (voir operator<<(std↩ : :ostream &, const Date &)).

L'appel :

```
to_string(in);
```

est équivalent à celui-ci :

```
in.to_string();
```

Paramètres

in	la Date à convertir.

Renvoie

une représentation de la date courante sous la forme d'une **std : :string**.

Voir également

Date : :to_string().

5.1.2.8 today()

```
Date nvs::today ( )
```

Fonction retournant la date actuelle (au moment de l'exécution) sous la forme d'une Date.

Renvoie

la date au moment de l'exécution comme une Date.

Voir également

http://en.cppreference.com/w/cpp/chrono/time_point
http://en.cppreference.com/w/cpp/chrono/c/time
http://en.cppreference.com/w/cpp/chrono/c/localtime

5.1.2.9 validate()

```
template<typename T >
T nvs::validate (
            const T & value,
            const T & min,
            const T & max,
            const  std::string & msg = "error : " )
```

Validation d'une valeur entre deux bornes.

Paramètres

value	valeur à valider.
min	borne minimale.
max	borne maximale.
msg	en-tête du message en cas de problème.

Renvoie

`value` si `min <= value <= max`.

Exceptions

std : :invalid_argument	si value $\notin$ [`min`, `max`]

Voir également

between()

6 Documentation des classes

6.1 Référence de la classe nvs : :Date

Classe représentant une date.

```
#include <date.h>
```

Fonctions membres publiques

- Date (int year, unsigned month, unsigned day)
  Constructeur.
- int year () const
  Accesseur en lecture de l'année.
- unsigned month () const
  Accesseur en lecture du mois.
- unsigned day () const
  Accesseur en lecture du jour.
- bool leapYear () const
  Méthode pour déterminer si l'année courante est bissextile.
- unsigned dayNumberInMonth () const
  Méthode pour connaître le nombre de jours du mois courant.
- **std : :string** to_string () const
  Méthode pour obtenir la date courante sous la forme d'une ***std : :string****.*

Attributs publics statiques

- static constexpr int MINIMUM_YEAR { **std : :numeric_limits**<int> : :min() }
  Valeur minimale acceptée pour l'année.
- static constexpr int MAXIMUM_YEAR { **std : :numeric_limits**<int> : :max() }
  Valeur maximale acceptée pour l'année.
- static constexpr unsigned MINIMUM_MONTH { 1 }
  Valeur minimale acceptée pour le mois.
- static constexpr unsigned MAXIMUM_MONTH { 12 }
  Valeur maximale acceptée pour le mois.
- static constexpr unsigned MINIMUM_DAY { 1 }
  Valeur minimale acceptée pour le jour.
- static constexpr **std : :array**< unsigned, MAXIMUM_MONTH+1 > MAXIMUM_DAY {{ 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }}
  Nombre de jours par mois, hors années bissextiles.

Attributs privés

- const int year_
  L'année.
- const unsigned month_
  Le mois.
- const unsigned day_
  Le jour.

6.1.1 Description détaillée

Classe représentant une date.

On ne se soucie pas ici du type de calendrier et de sa pertinence historique. Il n'y a pas d'années minimale ou maximale autre que celles impliquées par la taille du type sous-jacent.

Les attributs year_, month_ et day_ sont constants. Les instances de Date sont donc `immuables`.

6.1.2 Documentation des constructeurs et destructeur

6.1.2.1 Date()

```
nvs::Date::Date (
            int year,
            unsigned month,
            unsigned day )
```

Constructeur.

Ce constructeur valide ses arguments à l'aide du modèle de fonction validate(). Dès lors, une exception de type **`std::invalid_argument`** peut être levée lors de son utilisation.

On considère qu'il y a une année 0.

Paramètres

year	l'année.
month	le mois.
day	le jour.

Exceptions

std : :invalid_argument	si : — year ∉ [Date : :MINIMUM_YEAR, Date : :MAXIMUM_YEAR] — month ∉ [Date : :MINIMUM_MONTH, Date : :MAXIMUM_MONTH] — day ∉ [Date : :MINIMUM_DAY, Date : :MAXIMUM_DAY[month]] tenant compte des années bissextiles.

Voir également

year_, month_, day_.

6.1.3 Documentation des fonctions membres

6.1.3.1 day()

```
unsigned nvs::Date::day ( ) const  [inline]
```

Accesseur en lecture du jour.

Renvoie

le jour.

6.1.3.2 dayNumberInMonth()

```
unsigned nvs::Date::dayNumberInMonth ( ) const  [inline]
```

Méthode pour connaître le nombre de jours du mois courant.

Renvoie

le nombre de jours du mois courant.

6.1.3.3 leapYear()

```
bool nvs::Date::leapYear ( ) const  [inline]
```

Méthode pour déterminer si l'année courante est bissextile.

Renvoie

`true` si l'année courante est bissextile, `false` sinon.

Voir également

leapYear(int).

6.1.3.4 month()

```
unsigned nvs::Date::month ( ) const  [inline]
```

Accesseur en lecture du mois.

Renvoie

le mois.

6.1.3.5 to_string()

```
std::string nvs::Date::to_string ( ) const
```

Méthode pour obtenir la date courante sous la forme d'une **std : :string**.

La **`std::string`** retournée est `"jj/mm/aaaa"`, par exemple `"26/04/2016"`.

Il s'agit de la même forme que celle lors de l'injection d'une Date dans un flux en sortie (voir operator<<(std↩ : :ostream &, const Date &)).

Renvoie

une représentation de la date courante sous la forme d'une **std : :string**.

Voir également

to_string(const Date &).

6.1.3.6 year()

```
int nvs::Date::year ( ) const  [inline]
```

Accesseur en lecture de l'année.

Renvoie

l'année.

6.1.4 Documentation des données membres

6.1.4.1 day_

```
const unsigned nvs::Date::day_  [private]
```

Le jour.

Cet attribut est constant.

Le premier jour du mois a comme valeur MINIMUM_DAY, le suivant MINIMUM_DAY + 1, etc., jusqu'au dernier du mois, dépendant du mois et de l'année et fourni par dayNumberInMonth().

6.1.4.2 MAXIMUM_DAY

```
constexpr  std::array< unsigned, MAXIMUM_MONTH + 1 > nvs::Date::MAXIMUM_DAY {{ 0, 31, 28, 31,
30, 31, 30, 31, 31, 30, 31, 30, 31 }}  [static]
```

Nombre de jours par mois, *hors* années bissextiles.

Il s'agit des valeurs maximales acceptées pour le jour en fonction du mois, *sans* tenir compte des années bissextiles. Pour connaître le nombre de jours d'un mois en tenant compte des années bissextiles, il faut utiliser la méthode dayNumberInMonth().

La valeur initiale nulle est présente pour permettre d'utiliser directement la valeur du mois comme index de la **std::array**.

6.1.4.3 month_

```
const unsigned nvs::Date::month_  [private]
```

Le mois.

Cet attribut est constant.

Janvier a comme valeur MINIMUM_MONTH, février MINIMUM_MONTH + 1, etc., jusquà décembre de valeur MAXIMUM_MONTH.

6.1.4.4 year_

```
const int nvs::Date::year_  [private]
```

L'année.

Cet attribut est constant.

Elle est comprise entre MINIMUM_YEAR et MAXIMUM_YEAR, ces valeurs comprises.

On considère qu'il y a une année 0.

La documentation de cette classe a été générée à partir du fichier suivant :
- date.h

7 Documentation des fichiers

7.1 Référence du fichier date.h

Définition de la classe nvs : :Date et de fonctions utilisant ce type.

```
#include <limits>
#include <array>
#include <string>
#include <ostream>
```

Classes

- class nvs : :Date
  Classe représentant une date.

Espaces de nommage

- nvs
  Espace de nom de Nicolas Vansteenkiste.

Fonctions

- **std : :ostream** & nvs : :operator<< (**std : :ostream** &out, const Date &in)
  Opérateur d'injection d'une Date dans un flux en sortie.
- bool nvs : :operator< (const Date &lhs, const Date &rhs)
  Opérateur de comparaison de deux dates.
- bool nvs : :operator== (const Date &lhs, const Date &rhs)
  Opérateur de test d'égalité de deux dates.
- **std : :string** nvs : :to_string (const Date &in)
  Fonction de conversion d'une nvs : :Date en ***std : :string****.*
- Date nvs : :today ()
  Fonction retournant la date actuelle (au moment de l'exécution) sous la forme d'une Date.
- bool nvs : :leapYear (int year)
  Fonction pour déterminer si une année est bissextile.

7.1.1 Description détaillée

Définition de la classe nvs : :Date et de fonctions utilisant ce type.

7.2 Référence du fichier validation.hpp

Fonctions d'aide à la validation.

```
#include <functional>
#include <utility>
#include <string>
#include <stdexcept>
```

Espaces de nommage

- nvs
  Espace de nom de Nicolas Vansteenkiste.

Fonctions

- template<typename T >
  bool nvs : :between (const T &value, T min, T max)
  Évaluation si une valeur est comprise entre deux autres.
- template<typename T >
  std : :string nvs : :error_message (const T &value, T min, T max, const **std : :string** &msg="error : ")
  Production d'un message d'erreur.
- template<typename T >
  T nvs : :validate (const T &value, const T &min, const T &max, const **std : :string** &msg="error : ")
  Validation d'une valeur entre deux bornes.

7.2.1 Description détaillée

Fonctions d'aide à la validation.

Index