labo_04_carvalho_bruno_gallay_david

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

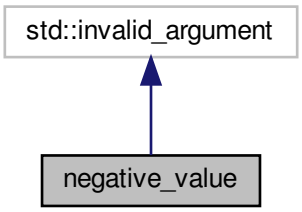
Chapter 4

Class Documentation

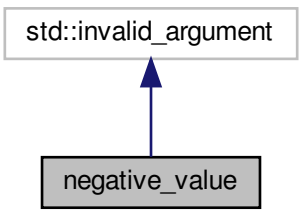
4.1 negative_value Class Reference

`#include <exceptions.h>`

Inheritance diagram for negative_value:



Collaboration diagram for negative_value:



Public Member Functions

- negative_value (const char *msg)
  Constructor.
- negative_value (std::string msg)
  Constructor.

4.1.1 Constructor & Destructor Documentation

4.1.1.1 negative_value() [1/2]

```
negative_value::negative_value (
            const char * msg )  [inline]
```

Constructor.

Parameters

msg	error message

4.1.1.2 negative_value() [2/2]

```
negative_value::negative_value (
            std::string msg )  [inline]
```

Constructor.

Parameters

msg	error message

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

- exceptions.h

4.2 Temps< T > Class Template Reference

```
#include <temps.h>
```

Public Member Functions

- Temps (T heure=0, T minute=0, T seconde=0)

  constructor of class Temps
- template<typename U >
  Temps (const Temps< U > &temps)

  constructor by copy
- Temps (const Temps &temps)

  constructor by copy
- Temps (T seconde)

  construct a full time HH:MM:SS just by converting seconds
- T getHeure () const

  getter for hours
- T getMinute () const

  getter for minutes
- T getSeconde () const

  getter for seconds
- T asHeure () const

  convert into hours representation
- T asMinute () const

  convert into minutes representation
- T asSeconde () const

  convert into seconds representation
- std::ostream & print (std::ostream &stream=std::cout) const

  print object representation to stream
- void setHeure (T heure)

  allow to modify value of hours
- void setMinute (T minute)

  allow to modify value of minutes
- void setSeconde (T seconde)

  allow to modify value of seconds
- template<typename U >
  Temps & operator= (const Temps< U > &temps)

  surcharge of operator =, allow us to copy an object Temps into an other
- Temps & operator= (const Temps &temps)

  surcharge of operator =, allow us to copy an object Temps into an other
- Temps & operator+= (const Temps &temps)

  surcharge of operator +=, allow us to addition object
- Temps & operator-= (const Temps &temps)

  surcharge of operator -=, allow us to substract object
- template<typename U >
  Temps & operator+= (const Temps< U > &temps)

  surcharge of operator +=, allow us to addition object
- template<typename U >
  Temps & operator-= (const Temps< U > &temps)

  surcharge of operator -=, allow us to substract object
- bool operator== (const Temps &temps) const

  surcharge of operator ==, allow us to compare two object
- bool operator!= (const Temps &temps) const

  surcharge of operator !=, allow us to compare two object
- operator float () const

cast object into float, it means it convert an HH:MM:SS into h.xxxxx

- operator double () const

  cast object into double, it means it convert an HH:MM:SS into h.xxxx

- operator long double () const

  cast object into long double, it means it convert HH:MM:SS into h.xxxx

- operator long long () const

  cast object into long long int, convert our full time HH:MM:SS into seconds

- operator std::string () const

  cast object into string, allow us to print it into HH:MM:SS format

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Temps() [1/4]

```
template<typename T>
Temps< T >::Temps (
            T heure = 0,
            T minute = 0,
            T seconde = 0 )
```

constructor of class Temps

Parameters

heure	
minute	
seconde	

Exceptions

negative_value	resulting time must be positive

4.2.1.2 Temps() [2/4]

```
template<typename T>
template<typename U >
Temps< T >::Temps (
            const Temps< U > & temps )
```

constructor by copy

Parameters

temps	

4.2.1.3 Temps() [3/4]

```
template<typename T>
Temps< T >::Temps (
            const Temps< T > & temps )
```

constructor by copy

Parameters

temps	

4.2.1.4 Temps() [4/4]

```
template<typename T>
Temps< T >::Temps (
            T seconde )
```

construct a full time HH:MM:SS just by converting seconds

Parameters

seconde	

Exceptions

negative_value	resulting time must be positive

4.2.2 Member Function Documentation

4.2.2.1 asHeure()

```
template<typename T>
T Temps< T >::asHeure ( ) const
```

convert into hours representation

Returns

hours

4.2.2.2 asMinute()

```
template<typename T>
T Temps< T >::asMinute ( ) const
```

convert into minutes representation

Returns

minutes

4.2.2.3 asSeconde()

```
template<typename T>
T Temps< T >::asSeconde ( ) const
```

convert into seconds representation

Returns

seconds

4.2.2.4 getHeure()

```
template<typename T>
T Temps< T >::getHeure ( ) const
```

getter for hours

Returns

heure

4.2.2.5 getMinute()

```
template<typename T>
T Temps< T >::getMinute ( ) const
```

getter for minutes

Returns

minute

4.2.2.6 getSeconde()

```
template<typename T>
T Temps< T >::getSeconde ( ) const
```

getter for seconds

Returns

seconde

4.2.2.7 operator double()

```
template<typename T>
Temps< T >::operator double ( ) const
```

cast object into double, it means it convert an HH:MM:SS into h.xxxx

Returns

casted value to double

4.2.2.8 operator float()

```
template<typename T>
Temps< T >::operator float ( ) const
```

cast object into float, it means it convert an HH:MM:SS into h.xxxxx

Returns

casted value to float

4.2.2.9 operator long double()

```
template<typename T>
Temps< T >::operator long double ( ) const
```

cast object into long double, it means it convert HH:MM:SS into h.xxxx

Returns

casted value to long double

4.2.2.10 operator long long()

```
template<typename T>
Temps< T >::operator long long ( ) const
```

cast object into long long int, convert our full time HH:MM:SS into seconds

Returns

casted value to long long

4.2.2.11 operator std::string()

```
template<typename T>
Temps< T >::operator std::string ( ) const
```

cast object into string, allow us to print it into HH:MM:SS format

Returns

casted value to std::string

4.2.2.12 operator"!=()

```
template<typename T>
bool Temps< T >::operator!= (
            const Temps< T > & temps ) const
```

surcharge of operator !=, allow us to compare two object

Parameters

temps	

Returns

true if objects are different

4.2.2.13 operator+=() [1/2]

```
template<typename T>
Temps& Temps< T >::operator+= (
            const Temps< T > & temps )
```

surcharge of operator +=, allow us to addition object

Parameters

temps	

Returns

reference on itself

4.2.2.14 operator+=() [2/2]

```
template<typename T>
template<typename U >
Temps& Temps< T >::operator+= (
            const Temps< U > & temps )
```

surcharge of operator +=, allow us to addition object

Parameters

temps	

Returns

reference on itself

4.2.2.15 operator-=() [1/2]

```
template<typename T>
Temps& Temps< T >::operator-= (
            const Temps< T > & temps )
```

surcharge of operator -=, allow us to substract object

Parameters

temps	

Returns

reference on itself

4.2.2.16 operator-=() [2/2]

```
template<typename T>
template<typename U >
Temps& Temps< T >::operator-= (
            const Temps< U > & temps )
```

surcharge of operator -=, allow us to substract object

Parameters

temps	

Returns

reference on itself

4.2.2.17 operator=() [1/2]

```
template<typename T>
template<typename U >
Temps& Temps< T >::operator= (
            const Temps< U > & temps )
```

surcharge of operator =, allow us to copy an object Temps into an other

Parameters

temps	

Returns

reference on itself

4.2.2.18 operator=() [2/2]

```
template<typename T>
Temps& Temps< T >::operator= (
            const Temps< T > & temps )
```

surcharge of operator =, allow us to copy an object Temps into an other

Parameters

temps	

Returns

reference on itself

4.2.2.19 operator==()

```
template<typename T>
bool Temps< T >::operator== (
            const Temps< T > & temps ) const
```

surcharge of operator ==, allow us to compare two object

Parameters

temps	

Returns

true if objects are equal

4.2.2.20 print()

```
template<typename T>
std::ostream& Temps< T >::print (
            std::ostream & stream = std::cout ) const
```

print object representation to stream

Parameters

stream	

Returns

reference on the stream parameter

4.2.2.21 setHeure()

```
template<typename T>
void Temps< T >::setHeure (
            T heure )
```

allow to modify value of hours

Parameters

heure	

Exceptions

negative_value	hours must be positives

4.2.2.22 setMinute()

```
template<typename T>
void Temps< T >::setMinute (
            T minute )
```

allow to modify value of minutes

Parameters

minute	

Exceptions

std::invalid_argument	minutes must be in [0, 60[

4.2.2.23 setSeconde()

```
template<typename T>
void Temps< T >::setSeconde (
            T seconde )
```

allow to modify value of seconds

Parameters

seconde	

Exceptions

std::invalid_argument	seconds must be in [0, 60[

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

- temps.h

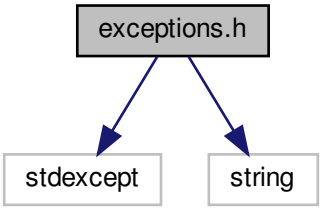
Chapter 5

File Documentation

5.1 exceptions.h File Reference

```
#include <stdexcept>
#include <string>
```

Include dependency graph for exceptions.h:



Classes

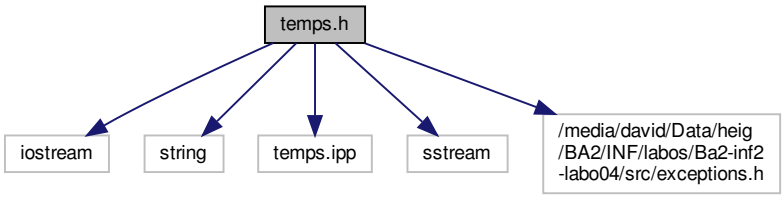
- class negative_value

5.2 temps.h File Reference

```
#include <iostream>
#include <string>
```

```
#include "temps.ipp"
```

Include dependency graph for temps.h:



Classes

- class Temps< T >

Functions

- template<typename T1 , typename T2 >
  Temps< T1 > operator+ (Temps< T1 > t1, const Temps< T2 > &t2)

  return the difference without check for overflow Object is granted to be positiv
- template<typename T1 , typename T2 >
  Temps< T1 > operator- (Temps< T1 > t1, const Temps< T2 > &t2)

  return the difference without check for underflow checks could be done using operator< Object is granted to be positiv
- template<typename T1 , typename T2 >
  Temps< T1 > operator< (Temps< T1 > t1, const Temps< T2 > &t2)
- template<typename T1 , typename T2 >
  Temps< T1 > operator> (Temps< T1 > t1, const Temps< T2 > &t2)
- template<typename T >
  std::ostream & operator<< (std::ostream &stream, Temps< T > temps)

  print temps object representation to stream

5.2.1 Function Documentation

5.2.1.1 operator+()

```
template<typename T1 , typename T2 >
Temps<T1> operator+ (
            Temps< T1 > t1,
            const Temps< T2 > & t2 )
```

return the difference without check for overflow Object is granted to be positiv

Parameters

t1	instance of Temps
t2	instance of Temps

Returns

return the difference between t1 and t2.

5.2.1.2 operator-()

```
template<typename T1 , typename T2 >
Temps<T1> operator- (
            Temps< T1 > t1,
            const Temps< T2 > & t2 )
```

return the difference without check for underflow checks could be done using operator< Object is granted to be positiv

Parameters

t1	instance of Temps
t2	instance of Temps

Returns

return the difference between t1 and t2.

5.2.1.3 operator<()

```
template<typename T1 , typename T2 >
Temps<T1> operator< (
            Temps< T1 > t1,
            const Temps< T2 > & t2 )
```

Parameters

t1	instance of Temps
t2	instance of Temps

Returns

true if t1 is smaller than t2

5.2.1.4 operator<<()

```
template<typename T >
std::ostream& operator<< (
             std::ostream & stream,
             Temps< T > temps )
```

print temps object representation to stream

Parameters

stream	

Returns

reference on the stream parameter

5.2.1.5 operator>()

```
template<typename T1 , typename T2 >
Temps<T1> operator> (
             Temps< T1 > t1,
             const Temps< T2 > & t2 )
```

Parameters

t1	instance of Temps
t2	instance of Temps

Returns

true if t1 is bigger than t2

Index