ETO

234567

Generated by Doxygen 1.9.8

Chapter 1

NewTec Dokumentation

Willkommen auf der Code Dokumentationsseite für die NewTec !!

1.1 Inhaltsverzeichnis

1.2 Übersicht

Willkommen zur Code-Dokumentationsseite der NewTec GmbH!

Unser Projekt, das Code-Dokumentationstool der NewTec GmbH, ist eine Anwendung, die auf Grundlage von Doxygen weiterentwickelt wurde, um Entwicklern und Teams bei der Erstellung und Verwaltung von hochwertiger Code-Dokumentation zu helfen. Mit diesem Tool können Sie mühelos Ihre Code-Basis dokumentieren, um die Wartbarkeit, Zusammenarbeit und Nachvollziehbarkeit Ihrer Softwareprojekte zu verbessern.

1.3 Funktionen

- Die Dokumentation ist in NewTec-Farben gestaltet
- Dark Mode verfügbar
- Sie können enthaltene Codefragmente kopieren.
- Sie können jede Klasse, Methode oder Funktion im Projekt mit der Suchleiste finden
- In der Suchleiste haben Sie die Möglichkeit zwischen diesen drei zu filtern
- Warnungen, Hinweise oder veraltete Teile sind farblich hervorgehoben

1.4 Konfiguration

Um die Dokumentation auf Ihr persönliches Projekt zu personalisieren können Sie einfach in Ihrer Kommandozeile den Befehl : ./config.sh ausführen. Daraufhin werden Sie aufgefordert den Projektnamen sowie die Projektnummer einzugben, bestätigen Sie beide Eingaben mit Enter und Ihr Projekt ist nun personalisiert.

1.5 Demo

1.5.1 Kopier und Bearbeitungs Beispiel

```
Dieser Text kann mittels der Button ganz rechts kopiert und bearbeitet werden!
```

1.5.2 Besondere Hinweise

1.5.2.1 1. Notes

Note

Diese Klasse wurde zu Demo Zwecken erstellt.
```
int var = 0;
```
Hiermit können Sie Notizen erstellen als Hinweise oder Erklärungen zu den dargestellten Funktionen.

Click to reveal source code
```
@note Diese Klasse wurde zu Demo Zwecken erstellt.
@code int var = 0; @endcode
Hiermit können sie Notizen erstellen als Hinweise oder Erklärungen zu den dargestellten Funktionen.
@note
```

1.5.2.2 2. Warnings

Warning

Hiermit können Sie auf potentielle Probleme und Fehler hinweisen!

Click to reveal source code
```
@warning Hiermit können Sie auf potentielle Probleme und Fehler hinweisen!
```

1.5.2.3 3. Deprecated

Deprecated Dieser Abschnitt weist auf veraltete Funktionen hin.

Click to reveal source code
```
@deprecated Dieser Abschnitt weist auf veraltete Funktionen hin.
```

1.5.2.4 4. Bug

Bug Die ist ein Beispiel Bug.

Click to reveal source code
```
@bug Dies ist ein Beispiel Bug.
```

1.5.2.5 5. Unveränderlich

Invariant

Dies ist eine unveränderliche Größe.

Click to reveal source code

```
@invariant Dies ist eine unveränderliche Größe.
```

1.5.2.6 6. Bedingung

Precondition

Dies ist eine Bedingung vor der Ausführung des Codes.

Click to reveal source code

```
@pre Dies ist eine Bedingung vor der Ausführung des Codes.
```

1.5.2.7 7. Nachbedingung

Postcondition

Dies eine Bedinung nach der Ausführung des Codes.

Click to reveal source code

```
@post Dies eine Bedinung nach der Ausführung des Codes.
```

1.5.2.8 8. Zu erledigen

Todo Weitere zu bearbeitende Themen.

Click to reveal source code

```
@todo Weiter zu bearbeitende Themen
```

1.5.2.9 9. Bemerkungen

Remarks

Viel Spaß mit dem Theme.

Click to reveal source code

```
@remark Viel Spaß mit dem Theme
```

Chapter 2

Deprecated List

Member Bruch::Bruch (int z=0, int n=1)

Dieser Abschnitt weist auf veraltete Funktionen hin.

page NewTec Dokumentation

Dieser Abschnitt weist auf veraltete Funktionen hin.

Chapter 3

Bug List

Member Bruch::ggt (int a, int b)

Dies ist ein Beispiel Bug

page NewTec Dokumentation

Die ist ein Beispiel Bug.

Chapter 4

Todo List

page NewTec Dokumentation

Weitere zu bearbeitende Themen.

Chapter 5

Topic Index

5.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

File Index

7.1 File List

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

Chapter 8

Topic Documentation

8.1 Bruch testen

Dieses Modul enthält die Dokumentation für das Bruch-Testsystem.

Classes

- class Bruch

  This is a class which represents a fracture.

8.1.1 Detailed Description

Dieses Modul enthält die Dokumentation für das Bruch-Testsystem.

Dieses Modul bietet eine Dokumentation für die Bruchklasse und die dazugehörigen Tests.

- Bruchklasse
  - Konstruktor
  - Wert abrufen
  - Multiplikation
- Testklasse
  - Tests ausführen

8.1.2 Bruchklasse

Die Bruchklasse repräsentiert eine einfache Bruchzahl.

8.1.2.1 Konstruktor

Der Konstruktor erzeugt eine Bruchinstanz mit einem gegebenen Zähler und Nenner.

```
Bruch(int numerator, int denominator);
```

8.1.2.2 Wert abrufen

Die Methode `getValue` gibt den Wert des Bruchs als Fließkommazahl zurück.

```
double getValue() const;
```

8.1.2.3 Multiplikation

Die Methode `multiply` ermöglicht die Multiplikation dieses Bruchs mit einem anderen Bruch.

```
Bruch multiply(const Bruch& other) const;
```

8.1.3 Testklasse

Die Testklasse enthält Tests für die Bruchklasse.

8.1.3.1 Tests ausführen

Die Methode `runTests` führt Tests für die Bruchklasse aus und gibt das Ergebnis aus.

```
static void runTests();
```

Chapter 9

Class Documentation

9.1 Bruch Class Reference

9.1.1 Detailed Description

This is a class which represents a fracture.

Class holds internally a counter and a denominator both are integer.

`#include <Bruch.h>`

Public Member Functions

- Bruch (int z=0, int n=1)

  Constructor for class Constructor gets counter and denominator evalutes greatest common divisor and checks if the denominator is greater then zero.
- int **nenner** () const
- int **zaehler** () const
- Bruch & operator∗= (const Bruch &b)

  Copy Constructor.
- int ggt (int a, int b)

  Inline Function to determine the greatest common divisor. The Function gets the counter and denominator and determines the greatest common divisor.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 Bruch()

```
Bruch::Bruch (
            int z = 0,
            int n = 1 )  [inline]
```

Constructor for class Constructor gets counter and denominator evalutes greatest common divisor and checks if the denominator is greater then zero.

Parameters

in	*z*	counter if no value is given it is 0 per default
in,out	*n*	denominator if no value is given it is 1 per default

Note

Diese Klasse wurde zu Demo Zwecken erstellt.

```
int var = 0;
```

Notizen dienen als Hinweise oder Erklärungen zu den dargestellten Funktionen.

```
    Bruch(int z = 0, int n = 1) {
        int var = 0;

        if (cond == false ||
            cond2 == true)
        {

        }
    }
   Bruch(int z = 0, int n = 1) {
    int var = 0;
    if (cond == false ||
        cond2 == true)
    {
    }
}
```

Warning

Warnings dienen dazu Entwickler auf potentielle Probleme und Fehler hinzuweisen!

Deprecated Dieser Abschnitt weist auf veraltete Funktionen hin.

9.1.3 Member Function Documentation

9.1.3.1 ggt()

```
int Bruch::ggt (
            int a,
            int b )  [inline]
```

Inline Function to determine the greatest common divisor. The Function gets the counter and denominator and determines the greatest common divisor.

Parameters

a	= counter
b	= denominator

Returns

Greatest common divisor

Bug Dies ist ein Beispiel Bug

```
int var = 0;
```

9.1.3.2 operator∗=()

```
Bruch & Bruch::operator*= (
            const Bruch & b )  [inline]
```

Copy Constructor.

Returns

Object

```
Bruch& operator ==(const Bruch& b){
        float b = 12;
        return true;
}
```

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

- source/Bruch.h

9.2 Test Class Reference

9.2.1 Detailed Description

This is a test class to show what doxygen does.

It contains four attributes which are one int, one float, one char and one int array

```
#include <test_class.h>
```

Public Member Functions

- Test (int a, float b, char c, int d)
  This is the constructor of the class.
- ∼**Test** ()
  Destructor for Test class Deletes test4 array.
- int **test1** () const
  returns the value of test1
- float **test2** () const
  returns the value of test2
- char **test3** () const
  returns the value of test3

9.2.2 Constructor & Destructor Documentation

9.2.2.1 Test()

```
Test::Test (
            int a,
            float b,
            char c,
            int d )  [inline]
```

This is the constructor of the class.

It initializes all member attributes and creates an int array

Parameters

a	is for test1
b	is for test2
c	is for test3
d	is the size of the int array

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

- source/test_class.h
- source/test_class.cpp

9.3 Time Class Reference

9.3.1 Detailed Description

This is a time class which tracks time in hours, minutes and seconds.

There are three member variables hours, minutes and seconds, They are all integer. You can add seconds, minutes and hours to to an existing time. It is also possible only to check the time.

```
#include <time.h>
```

Public Member Functions

- Time (int h=0, int m=0, int s=0)
  Constructor for Time Class.
- ~Time ()
- int hour () const
  Member method.
- int minute () const
  Member method.
- int second () const
  Member method.
- int just_seconds ()
  Methode to give the time in seconds.
- int just_minutes ()
  Methode to give the time in minutes.
- void **add_hours** (int n)
  add hours to existing time
- void **add_minutes** (int n)
  add minutes to existing time
- void **add_seconds** (int n)
  add seconds to existing time
- void **check_time** (int h, int m, int s)

9.3.2 Constructor & Destructor Documentation

9.3.2.1 Time()

```
Time::Time (
              int h = 0,
              int m = 0,
              int s = 0 )  [inline]
```

Constructor for Time Class.

It initializes the three member variables.

Parameters

h	= hour
m	= minutes
s	= secounds

```
Time(int var = 0){
afh

fas


}
```

9.3.2.2 ~Time()

```
Time::~Time ( )  [inline]
  Time(int var = 0){
afh

fas


}
```

9.3.3 Member Function Documentation

9.3.3.1 hour()

```
int Time::hour ( ) const  [inline]
```

Member method.

Returns

Attribute for hours

9.3.3.2 just_minutes()

```
int Time::just_minutes ( )
```

Methode to give the time in minutes.

Returns

Saved time in minutes

9.3.3.3 just_seconds()

```
int Time::just_seconds ( )
```

Methode to give the time in seconds.

Returns

Saved time in seconds

9.3.3.4 minute()

```
int Time::minute ( ) const  [inline]
```

Member method.

Returns

Attribute for minutes

9.3.3.5 second()

```
int Time::second ( ) const  [inline]
```

Member method.

Returns

Attribute for seconds

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

- source/time.h
- source/time.cpp

Chapter 10

File Documentation

10.1 Bruch.h

```
00001
00009 #ifndef Bruch_H_
00010 #define Bruch_H_
00011 class Bruch {
00012 public:
00055     Bruch(int z = 0, int n = 1) :
00056             _nenner { n / ggt(z, n) }, _zaehler { z / ggt(z, n) } {
00057         if (_nenner == 0) {
00058             _nenner = 1;
00059         }
00060         if (_nenner < 0) {
00061             _nenner *= -1;
00062             _zaehler *= -1;
00063         } else
00064             ;
00065
00066     }
00067     int nenner() const {
00068         return _nenner;
00069     }
00070     int zaehler() const {
00071         return _zaehler;
00072     }
00083     Bruch& operator*= (const Bruch& b){
00084         this->_zaehler *= b._zaehler;
00085         this->_nenner *= b._nenner;
00086         int a = ggt(_zaehler, _nenner);
00087         this->_zaehler = this->_zaehler/a;
00088         this->_nenner = this->_nenner/a;
00089         return *this;
00090     }
00105 inline int ggt(int a, int b) {
00106     while (a != 0) {
00107         a %= b;
00108     }
00109     return b;
00110 }
00111 private:
00112     int _nenner;
00113     int _zaehler;
00114 };
00115 inline Bruch operator*(const Bruch &b, const Bruch &c) {
00116     Bruch ret(b.zaehler() * c.zaehler(), b.nenner() * c.nenner());
00117     return ret;
00118
00119 }
00120 inline int operator==(const Bruch &b, const Bruch &c) {
00121     float b1 = b.zaehler() / b.nenner();
00122     float b2 = c.zaehler()/c.nenner();
00123     if (b1 == b2) {
00124         return true;
00125     } else
00126         return false;
00127
00128 }
00129 inline int operator!=(const Bruch &b,const Bruch& c){
00130         float b1 = b.zaehler()/b.nenner();
00131         float b2 = c.zaehler()/c.nenner();
```

```
00132         if (b1 != b2){
00133             return true;
00134         }else
00135             return false;
00136 }
00137
00138
00139 #endif /* Bruch_H_ */
```

10.2 test_class.h

```
00001
00010 class Test{
00014     int test1;
00018     float test2;
00022     char test3;
00026     int* test4;
00027 public:
00037     Test(int a, float b, char c,int d): test1(a) , test2(b), test3(c){
00038         test4 = new int [d];
00039         for(int i = 0; i < d; ++i){
00040             test4[i] = 0;
00041         }
00042     }
00047     ~Test(){};
00051     int test1()const {return test1;}
00055     float test2()const{return test2;}
00059     char test3()const{return test3;}
00060 };
```

10.3 time.h

```
00001
00008 #ifndef TIME_H_
00009 #define TIME_H_
00010 class Time{
00011 public:
00032     Time(int h=0,int m=0,int s=0){
00033         _std=h;
00034         _min=m;
00035         _sec=s;
00036     }
00050     ~Time(){
00051         int var;
00052     }
00058     int hour() const {return _std;}
00064     int minute ()const {return _min;}
00070     int second ()const {return _sec;}
00076     int just_seconds();
00082     int just_minutes();
00086     void add_hours (int n);
00090     void add_minutes (int n);
00094     void add_seconds (int n);
00095     void check_time (int h,int m,int s);
00096
00097 private:
00098     int _std;
00099     int _min;
00100     int _sec;
00101 };
00102 #endif /* TIME_H_ */
```