Date und Time für C++

Christian Reinbothe
Sudermanplatz 8 - 10

50670 Köln
Germany

mailto:Christian.Reinbothe@T-Online.DE
http://WWW.Reinbothe.DE

1. Überblick über die Klassen

1. Konstruktoren für die Klasse Date:

   ```
   Date::Date()
   Date::Date(int year, int month, int day)
   Date::Date(const char *S)
   ```

   Der 1. Konstruktor liefert das aktuelle Datum.
   Der 2. Konstruktor ist klar.
   Der 3. Konstruktor liefert das durch "2021-04-23" oder "23.04.2021" beabsichtigte Datum.

   Alle Konstruktoren testen auf Validität.
   Im Falle eines Fehlers werfen Sie eine Exception aus.

   Bitte:

   Bevor Sie einen Konstruktor aufrufen, testen Sie das beabsichtigte Datum mit einer der Methoden IsDate…(…). Sie sind mit "static" deklariert, so dass sie ohne Instanz (also mit Date::IsDate("2021-04-23")) aufgerufen werden können.

2. Konstruktoren für die Klasse Time:

   ```
   Time::Time()
   Time::Time(int hour, int minute, int second)
   Time::Time(const char *S)
   ```

   Der 1. Konstruktor liefert die aktuelle Zeit.
   Der 2. Konstruktor ist klar.
   Der 3. Konstruktor liefert die durch "17:45:00" beabsichtigte Zeit.

   Alle Konstruktoren testen auf Validität.
   Im Falle eines Fehlers werfen Sie eine Exception aus.

   Bitte:

   Bevor Sie einen Konstruktor aufrufen, testen Sie das beabsichtigte Datum mit einer der Methoden IsTime…(…). Sie sind mit "static" deklariert, so dass sie ohne Instanz (also mit Time::IsTime("17:45:00")) aufgerufen werden können.

3. Die Klasse DateTime kombiniert die Klassen Date und Time.

   Die korrekte Form für eine Instanz der Klasse DateTime ist "2021-04-23, 17:45:00" oder "23.04.2021, 17:45:00".

2. Operatoren auf Klasse Date

1. `Date& Date::operator ++=()`
   `Date Date::operator ++(int i)`

   Diese Operatoren addieren einen Tag zu der Instanz. Sie verändern die Instanz.

2. `Date& Date::operator --=()`
   `Date Date::operator --(int i)`

   Diese Operatoren subtrahieren einen Tag von der Instanz. Sie verändern die Instanz.

4. `int Date::operator –(const Date& D)`

   Dieser Operator berechnet die Anzahl von Tagen zwischen zwei Instanzen. Er verändert die Instanz nicht.

4. `Date Date::operator +(int ndays)`

   Dieser Operator addiert ndays Tage zu der Instant. Er verändert die Instanz nicht.

   (ndays < 0) ist möglich.

5. `Date Date::operator -(int ndays)`

   Dieser Operator subtrahier ndays Tage von der Instanz. Er verändert die Instanz nicht.

   (ndays < 0) ist möglich.

6. `Bool Date::operator ==(const Date& D)`
   `Bool Date::operator !=(const Date& D)`
   `Bool Date::operator <(const Date& D)`
   `Bool Date::operator >(const Date& D)`
   `Bool Date::operator <=(const Date& D)`
   `Bool Date::operator >=(const Date& D)`

   Diese Operatoren vergleichen zwei Instanzen der class Date. Sie verändern die Instanz nicht.

3. Operatoren auf Klasse Time

```
Bool Time::operator ==(const Time& T)
Bool Time::operator !=(const Time& T)
Bool Time::operator <(const Time& T)
Bool Time::operator >(const Time& T)
Bool Time::operator <=(const Time& T)
Bool Time::operator >=(const Time& T)
```

Diese Operatoren vergleichen zwei Instanzen der class Time. Sie verändern die Instanz nicht.

4. Operatoren auf Klasse DateTime

```
Bool DateTime::operator ==(const DateTime& DT)
Bool DateTime::operator !=(const DateTime& DT)
Bool DateTime::operator <(const DateTime& DT)
Bool DateTime::operator >(const DateTime& DT)
Bool DateTime::operator <=(const DateTime& DT)
Bool DateTime::operator >=(const DateTime& DT)
```

Diese Operatoren vergleichen zwei Instanzen der class DateTime. Sie verändern die Instanz nicht.

5. Methoden der Klasse Date

1. `bool Date::IsDate(int year, int month, int day)`
   `bool Date::IsDate(const char * S)`

   Diese Methoden sind klar.

2. `bool Date::IsDateUS(int year, int month, int day)`
   `bool Date::IsDateUS(const char * S)`

   Diese Methoden testen auf die US-Form von Date's (eine valide Form ist z.B. "2021-04-23".

3. `bool Date::IsDateDE(int year, int month, int day)`
   `bool Date::IsDateDE(const char * S)`

   Diese Methoden testen auf die DE-Form von Date's (eine valide Form ist z.B. "23.04.2021").

4. `std::string Date::to_string_US()`

   Diese Methode liefert das Datum der Instanz in US-Form.

5. `std::string Date::to_string_DE()`

   Diese Methode liefert das Datum der Instanz in DE-Form.

6. `Date Date::EasternJulianic(int year)`

   Diese Methode berechnet das Datum des Ostersonntags des Jahres year im Julianschen Kalender.
   Siehe "www.wikipedia.org"!

   Sie ist mit "static" deklariert, so dass sie ohne Instanz aufgerufen mit Date::EasternJulianic(int year)) aufgerufen werden können.

   Sie verändert die Instanz nicht.

7. `Date Date::EasternGregorianic(int year)`

   Diese Methode berechnet das Datum des Ostersonntags des Jahres year im Gregorianschen Kalender.
   Siehe "www.wikipedia.org"!

   Sie ist mit "static" deklariert, so dass sie ohne Instanz aufgerufen mit Date::EasternGregorianic(int year)) aufgerufen werden können.

   Sie verändert die Instanz nicht.

6. Methoden der Klasse Time

1. bool Time::IsTime(int hour, int minute, int second)
   bool Time::IsTime(const char * S)

   Diese Methoden sind klar.

2. int Time::to_seconds()

   Diese Methode berechnet die Instanz in Sekunden.

3. double Time::to_minutes()

   Diese Methode berechnet die Instanz in Minuten.

4. Double Time::to:hours()

   Diese Methode berechnet die Instanz in Stunden.

5. std::string Time::to_string()

   Diese Methode liefert die Zeit der Instanz.

7. Klasse DateTime

Die Klasse DateTime hat zwei Members:

```
Date D
Time T
```

Man kann alle Operatoren und Methoden der Klassen Date und Time anwenden.

Die Konstruktoren der Klasse DateTime sind ähnlich aufgebaut wie die Konstruktoren der Klassen Date und Time.

Die Methoden DateTime::IsDateTime(…) der Klassen DateTime sind ähnlich gebaut wie die Methoden Date::IsDate…(…) der Klasse Date und Time::IsTime…(…) der Klasse Time.

Die Methoden DateTime::to_string_US() und DateTime::to_string_DE() sind klar.