# Date and Time for C++

**Christian Reinbothe**
**Sudermanplatz 8 - 10**

**50670 Köln**
**Germany**

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

# 1. Provided Classes

There are 3 classes with several interessant contructors:

1. Constructors for class Date:

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

   The 1st constructor constructs the actual date now.
   The 2nd constructor is clear.
   The 3rd constructor constructs from i.e. "2021-04-23" or "23.04.2021".

   All contructors tests for validity of the date.
   They throw an exception on error.

   **Please:**

   Before you contruct, test the intendet dates with the methods ns IsDate…(…). They have a "static" (so you can call this methods without an instance (i.e. Date::IsDate("2021-04-23"))).

2. Constructors for class Time:

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

   The 1st constructor constructs the actual time now.
   The 2nd constructor is clear.
   The 3rd constructor constructs from i.e. "17:45:00".

   All contructors tests for validity of the date.
   They throw an exception on error.

   **Please:**

   Before you contruct, test the intendet times with the methods IsTime…(…). They have a "static" (so you can call this methods without an instance (i.e. Time::IsTime("17:45:00"))).

3. The class DateTime combines the classes Date and Time.

   The correct forms of DateTime are "2021-04-23, 17:45:00" or "23.04.2021, 17:45:00".

# 2. operators on class Date

We have several operators for the class Date:

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

   They add a day to the instance of the Date. They modify the instance.

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

   They subtract a day from the instance of the Date. They modify the instance

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

   It computes the number of days between two Date-instances. It does not modify the instance.

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

   It adds ndays to the instance of the Date and returns the result. It does not modify the instance.

   (ndays < 0) is possible.

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

   It subtracts ndays from the instance of the Date and returns the result. It does not modify the instance.

   (ndays < 0) is possible.

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)`

   They compare two instances of Date. They do not modify the instance.

# 3. operators on class Time

We have several operators for the class 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)
```

They compare two instances of Time. They do not modify the instances.

# 4. operators on class DateTime

We have several operators for the class Time:

```
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)
```

They compare two instances of DateTime. They do not modify the instances.

# 5. Methods in class Date

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

   These methods are clear.

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

   These methods tests for the US-form for Dates (i.e. "2021-04-23").

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

   These methods tests for the DE-form for Dates (i.e. "23.04.2021").

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

   It returns the date of the instance in US-form.

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

   It returns the date of the instance in DE-form.

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

   It generates a date for Eastern (sunday) in the Julianic calender. See "www.wikipedia.org"!

   It has a "static" (so you can call this method without an instance with `Date::EasternJulianic(int year)`).

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

   It generates a date for Eastern (sunday) in the Gregorianic calender. See "www.wikipedia.org"!

   It has a "static" (so you can call this method without an instance with `Date::EasternGregorianic(int year)`).

# 6. Methods in class Time

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

   These methods are clear.

2. int Time::to_seconds()

   It evaluates the instance in seconds.

3. double Time::to_minutes()

   It evaluates the instance in minutes.

4. Double Time::to:hours()

   It evaluates the instance in hours.

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

   It returns the time of the instance.

# 7. class DateTime

The class DateTime has two members:

```
Date D
Time T
```

So you can apply all operators and methods of class Date and class Time.

The constructors of class DateTime are similar to the constructors of class Date and class Time.

The methods of type DateTime::IsDateTime(…) are similar to the methods Date::IsDate…(…) of class Date and Time::IsTime…(…) of class Time.

Finally the methods DateTime::to_string_US() and DateTime::to_string_DE() are clear.