The class has various methods for performing operations on dates and overloads several operators for comparison and arithmetic operations.

The leapyearcheck function is a helper function that returns true if the input year is a leap year, and false otherwise.

The monthmask array is a constant array that holds the number of days in each month. It is indexed from 0 to 11, with 0 corresponding to January and 11 corresponding to December.

The date class has three private member variables \_y, \_m, and \_d, which represent the year, month, and day of the date, respectively. The class has several constructors, including a default constructor, a constructor that takes in a year, month, and day, a constructor that takes in a tm struct from the ctime library, and a copy constructor.

The date class overloads various operators, including the increment and decrement operators, the addition and subtraction operators, and comparison operators such as ==, !=, <, >, <=, and >= with 2 variations(tm struct, date). It also has conversion operators to convert a date object to an int(calculating the day number of the Julian calendar from the Gregorian date) or a string, a default constructor that reads a date from the users pc, constructor with parameters - a constructor that initializes a date with the given year, month, and day values, a constructor that initializes a date with a tm struct representing a date, date(const date& A) - a copy constructor that initializes a date with another date object.

Class methods

date& todate(tm\* date) - initializes a date with a tm struct representing a date.

void correctdate() – corrects wrong date.

void readdate() - reads a date from the users pc.

char\* dayofweek() const - return day of week.

get/set methods.

Use of this library

1. Calendar and scheduling applications: library can be used in applications that need to schedule or display dates and times, such as appointment booking systems, task managers, project management tools, and event calendars.
2. Financial and accounting software: library can be useful for calculating interest rates, amortization schedules, payment due dates, and other date-related calculations used in financial and accounting applications.
3. IoT and sensor systems: library can be used to handle and manage timestamps generated by sensors or other devices. This is especially useful when working with data that needs to be recorded and processed over time, such as weather sensors, traffic sensors, and energy usage meters.
4. Data analysis and reporting: library can be used to perform time-series analysis, trend analysis, and other types of data analysis that involve working with dates and times.
5. Gaming applications: library can be useful in games that require time-based events, such as turn-based games, real-time strategy games, and simulations.
6. Aerospace and defense applications: library can be useful in aerospace and defense applications that need to manage mission-critical events, such as launch dates, flight schedules, and mission timelines.
7. Educational software: library can be used in educational software that deals with history, timelines, and other date-related topics.

Overall, this C++ library for working with dates can be useful in any project that involves date and time calculations or management, also can be rewritten on Java, Python, C#, etc. really fast.