ДОКУМЕНТАЦИЯ

Проект: Дати

ООП / Компютърни науки / I курс / 2014/2015

**Здравко Андонов**

# Class Documentation

## Date Class Reference

### Public Types

* enum **workdaySearchType** { **none**, **closest**, **forward**, **backward** }

### Public Member Functions

* **Date** (int day, int month, int year)

Ако датата е коректна, day, month, year приемат съответните стойности, а ако не – приемат стойности 0, 0, 0

* **Date** (char q, int dayCode, int month, int year)

Създава дата по описание с ден от седмицата. Например:

втората събота от месец октомври

последният понеделник от месец ноември

q = ‘1’, ‘2’, ‘3’, … или ‘L’ за последния такъв ден

* **~Date** ()

Унищожава целия обект

* int **getDay** () const

връща деня от датата

* int **getMonth** () const

връща месеца от датата

* int **getYear** () const

връща годината от датата

* int **getWeekday** () const

връща кода на деня от седмицата за датата:

0 – неделя, 1 – понеделник, … 6 - събота

* bool **isWorkday** () const

връща true, ако деня е работен и false, ако не е

Проверява дали деня не е събота или неделя, дали не е ден за отработване, дали не е допълнително добавен почивен ден, дали не е официален почивен ден и дали не е разпети петък.

ПОЯСНЕНИЕ:

* Празниците и дните за отработване се запазват в vector<Date>, който при добавяне на нова дата се сортира със стандартния алгоритъм sort от <algorithm>
* Търсенето във вектора е двоично – сложност O(logn)
* **Date** **getClosestWorkday** ()

Намира най-близкия работен ден, като използва getClosestWorkday(forward) и getClosestWorkday(backward)

* **Date** **getClosestWorkday** (**workdaySearchType**)

Намира най-близкия работен ден в зависимост от посоката – преминава през всеки следващ ден и проверява дали е работен

* void **fixDateWorkday** ()

Наглася текущата дата към най-близкия работен ден в зависимост от посоката (**workdaySearch**)

* bool **getWorkdaysOnlyState** ()

Връща дали се работи само със работни дни или не

True – ако се работи само със работни дни

False – ако не

* void **setDay** (int day)

Променя деня, ако датата остава коректна

* void **setMonth** (int month)

Променя месеца, ако датата остава коректна

* void **setYear** (int year)

Променя годината, ако датата остава коректна

* void **setPrintFormatCode** (int code)

Задава формата за извеждане на дата

* void **toggleWorkdaysOnly** ()

Започва работа само с работни дни или спира работата само с работни дни

* void **setWorkdaySearch** (**workdaySearchType**)

Задава посоката на работа при нагласяне към най-близкия работен ден

По подразбиране не настроено на **none**, тоест да не се наглася

Възможностите са none, closest, forward, backward

* bool **operator==** (const **Date** &) const

Връща дали две дати са равни

* **Date** **operator+** (int days)

Добавя дни – използва +=

* **Date** & **operator+=** (int days)

Добавя дни, като първо добавя годините, после месеците и накрая дните по-малки от цял месец и ако трябва наглася до най-близкия работен ден

Ако е включен режим на работа само с работни дни добавя всеки ден и проверява дали е работен и ако е намалява дните, които трябва да се добавят

* **Date** & **operator++** ()

Добавя един ден и проверява граничните случаи за преминаване в нов месец или година и ги поправя

* **Date** **operator++** (int)

Добавя един ден, използвайки **operator++()**

* **Date** **operator-** (int days)

Аналогично на +

* **Date** & **operator-=** (int days)

Аналогично на +=

* **Date** & **operator--** ()

Аналогично на ++

* **Date** **operator--** (int)

Аналогично на ++

* **Date** **operator\*** (int months)

Добавя месеци, използвайки \*=

* **Date** & **operator\*=** (int months)

Добавя месеци и ако новата дата не е в месеца, деня става последния ден от месеца

* **Date** **operator/** (int months)

Изважда месеци, използвайки /=

* **Date** & **operator/=** (int months)

Изважда месеци аналогично на \*=

Операторите за години използват добавянето на месеци, като за всяка година добавят по 12 месеца

* **Date** **operator&** (int years)
* **Date** & **operator&=** (int years)
* **Date** **operator|** (int years)
* **Date** & **operator|=** (int years)

Следващите оператори за седмици използват операторите за дни – за всяка седмица добавят по 7 дни:

* **Date** **operator^** (int weeks)
* **Date** & **operator^=** (int weeks)
* **Date** **operator%** (int weeks)
* **Date** & **operator%=** (int weeks)
* int **operator-** (**Date** &date2)

Смята дните от Христа до едната дата, дните от Христа до другата дата и връша разликата – винаги положително число

* int **operator/** (**Date** &date2)

Връща само пълни месеци

* int **operator|** (**Date** &date2)

Връща само пълни години, използва /

* int **operator%** (**Date** &date2)

Връща само пълни седмици, използва -

### Static Public Member Functions

* static void **addHoliday** (**Date** &holiday, **Date** &workOff)

добавяне на почивен ден и ден за отработване

* static **Date** **calcEaster** (int year)

пресмята датата на Великден по нов стил в православната църква, по алгоритъма на Гаус

### Friends

* ostream & **operator<<** (ostream &out, const **Date** &date)

Формата се задава чрез **setPrintFormatCode**()

* istream & **operator>>** (istream &in, **Date** &date)

При коректно зададена дата тя се записва в date и настройките за формата на извеждане се връщат към тези по подразбиране

### Member Enumeration Documentation

#### enum Date::workdaySearchType

**Enumerator**

***none*** - не се наглася до работен ден

***closest*** - наглася се до най-близкия работен ден без значение от посоката

***forward*** - наглася се до най-близкия работен ден в бъдещето

***backward*** - наглася се до най-близкия работен ден в миналото

## Utility Class Reference

### Static Public Member Functions

* static int **charToInt** (char c)

Превръща символ (напр. ‚1‘) в число (1)