ДОКУМЕНТАЦИЯ

Проект: Големи числа

ООП / Компютърни науки / I курс / 2014/2015

**Здравко Андонов**

# Class Documentation

## BigInteger Class Reference

### Public Member Functions

**BigInteger** (int=0, int base=10)

**BigInteger** (const **BigInteger** &)

**~BigInteger** ()

**BigInteger** & **operator=** (long int)

**BigInteger** & **operator=** (const **BigInteger** &)

**BigInteger** **getAbs** () const

bool **isPositive** () const

int **getSize** () const

**ulong** **getBigDigit** (int index) const

**BigInteger** **operator+** (const **BigInteger** &) const

**BigInteger** & **operator+=** (const **BigInteger** &)

**BigInteger** **operator-** () const

**BigInteger** **operator-** (const **BigInteger** &) const

**BigInteger** & **operator-=** (const **BigInteger** &)

**BigInteger** **operator\*** (const **BigInteger** &) const

**BigInteger** **operator\*** (char) const

**BigInteger** & **operator\*=** (const **BigInteger** &)

**BigInteger** **operator/** (const **BigInteger** &) const

**BigInteger** & **operator/=** (const **BigInteger** &)

**BigInteger** **operator%** (const **BigInteger** &) const

**BigInteger** & **operator%=** (const **BigInteger** &)

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

bool **operator!=** (const **BigInteger** &) const

bool **operator<** (const **BigInteger** &) const

bool **operator>** (const **BigInteger** &) const

bool **operator<=** (const **BigInteger** &) const

bool **operator>=** (const **BigInteger** &) const

int **operator[]** (int) const

**BigInteger** **operator~** ()

**BigInteger** **operator&** (**BigInteger** &) const

**BigInteger** & **operator&=** (**BigInteger** &)

**BigInteger** **operator|** (**BigInteger** &) const

**BigInteger** & **operator|=** (**BigInteger** &)

**BigInteger** **operator^** (**BigInteger** &) const

**BigInteger** & **operator^=** (**BigInteger** &)

**BigInteger** **operator<<** (int) const

**BigInteger** & **operator<<=** (int)

**BigInteger** **operator>>** (int) const

**BigInteger** & **operator>>=** (int)

### Friends

istream & **operator>>** (istream &, **BigInteger** &)

ostream & **operator<<** (ostream &, **BigInteger** &)

### Constructor & Destructor Documentation

#### BigInteger::BigInteger (int *number* = 0, int *base* = 10)

Създава BigInteger равно на number във съответната бройна система, винаги след това го превръща в десетична бройа система

#### BigInteger::BigInteger (const BigInteger & *other*)

Създава копие като запазва само стойността

#### BigInteger::~BigInteger ()

### Member Function Documentation

#### BigInteger BigInteger::getAbs () const

Връща абсолютната стойност на числото

#### int BigInteger::getSize () const

Връща броя на цифрите на числото

#### bool BigInteger::isPositive () const

Връща true, ако числото е положително, и false, ако числото е отрицателно

#### bool BigInteger::operator!= (const BigInteger & *other*) const

Използва ! и ==

#### BigInteger BigInteger::operator% (const BigInteger & *other*) const

Връща остатък при целочислено деление

Използва %=

#### BigInteger & BigInteger::operator%= (const BigInteger & *other*)

Връща остатък при целочислено деление и го присвоява на \*this

Използва /=, -= и \*

(\*this) -= ((\*this) / other) \* other;

За всички побитови операции използваме вектор от unsigned long long int - bigDigits, в който генерираме представянето на числата в бройна система с основа 2^32, където всеки елемент от масива е цифра в тази бройна система. Тъй като основата е степен на 2, то можем да прилагаме вградените побитови операции за цели числа върху всяка цифра от масива и след това пак да преобразуваме числото в десетична бройна система.

#### BigInteger BigInteger::operator& (BigInteger & *other*) const

Побитово и

Използва &=

#### BigInteger & BigInteger::operator&= (BigInteger & *other*)

Побитово и

Извършваме го последователно за всяка от цифрите (елементите на bigDigits), използвайки вграденото в c++ за цели числа

#### BigInteger BigInteger::operator\* (const BigInteger & *other*) const

Умножение с друго голямо число

Използваме \*=

#### BigInteger BigInteger::operator\* (char *c*) const

Умножение с цифра

char c е символът, представящ цифрата

Използваме стандартният алгоритъм, учен в училище

#### BigInteger & BigInteger::operator\*= (const BigInteger & *other*)

Умножение с друго голямо число

Използваме умножение със цифра и стандартния алгоритъм, учен в училище

#### BigInteger BigInteger::operator+ (const BigInteger & *other*) const

Събиране

Използваме +=

#### BigInteger & BigInteger::operator+= (const BigInteger & *other*)

Събиране

Използваме стандартния алгоритъм, учен в училище и в зависимост от знаците на числата може да използваме -=

#### BigInteger BigInteger::operator- () const

Унарен оператор за промяна на знака

#### BigInteger BigInteger::operator- (const BigInteger & *other*) const

Изваждане

Използваме -=

#### BigInteger & BigInteger::operator-= (const BigInteger & *other*)

Използваме стандартния алгоритъм, учен в училище и в зависимост от знаците на числата може да използваме +=

#### BigInteger BigInteger::operator/ (const BigInteger & *other*) const

Деление

Използваме /=

#### BigInteger & BigInteger::operator/= (const BigInteger & *other*)

Деление

Използваме стандартния алгоритъм, учен в училище

#### bool BigInteger::operator< (const BigInteger & *other*) const

Първо проверява знаците

Второ проверява дължината на числата

Проверява цифра по цифра

#### BigInteger BigInteger::operator<< (int *n*) const

Използва <<=

#### BigInteger & BigInteger::operator<<= (int *n*)

Използва маски и bigDigits по подобен начин на останалите побитови операции

#### bool BigInteger::operator<= (const BigInteger & *other*) const

Използва ! и >

#### BigInteger & BigInteger::operator= (long int *number*)

Присвоява стойността на number

#### BigInteger & BigInteger::operator= (const BigInteger & *other*)

Присвоява стойността на other

#### bool BigInteger::operator== (const BigInteger & *other*) const

Връща true, ако двете числа са равни, и false, ако не са

Сравнява цифра по цифра

#### bool BigInteger::operator> (const BigInteger & *other*) const

Първо проверява знаците

Второ проверява дължината на числата

Проверява цифра по цифра

#### bool BigInteger::operator>= (const BigInteger & *other*) const

Използва ! и <

#### BigInteger BigInteger::operator>> (int *n*) const

Използва >>=

#### BigInteger & BigInteger::operator>>= (int *n*)

Работи аналогично на останалите побитови операции, използвайки маски и bigDigits

#### int BigInteger::operator[] (int *index*) const

Връща поредната цифра на позиция index от числото

#### BigInteger BigInteger::operator^ (BigInteger & *other*) const

XOR

Използва ^=

#### BigInteger & BigInteger::operator^= (BigInteger & *other*)

XOR

Използва аналогичен подход както при останалите побитови операции, използващи bigDigits

#### BigInteger BigInteger::operator| (BigInteger & *other*) const

Използва |=

#### BigInteger & BigInteger::operator|= (BigInteger & *other*)

Аналогичен на останалите побитови оператори

Използва bigDigits

#### BigInteger BigInteger::operator~ ()

Използва bigDigits аналогично на останалите побитови оператори

### Friends And Related Function Documentation

#### ostream& operator<< (ostream & , BigInteger & )[friend]

Извежда в десетична или шестнайсетична бройна система в зависимост от настройките на потока

#### istream& operator>> (istream & , BigInteger & )[friend]

Въвежда число в десетична или шестнайсетична бройна система