Класът BigNumber позволява работа с големи цели числа в 10-ична и 16-ична бройна система, както и с десетични дроби. За целта всеки обект от този тип съдържа масиви, които съхраняват цифрите на представянето му в съответната система, като цифрите преди и след десетичната запетая са в два отделни масива. За улеснение на някои операции като събиране, умножение, преминаване от една система в друга цифрите се съхраняват в обратен ред (например 123 се представя с низа „321").

Число от класа може да се **конструира**:

- По подадено цяло число (long), като в такъв случай като опционални параметри може да се подаде padding(отместване, или умножение по съответната степен на 10; например BigNumber(123,8) дава 123*10^8) и бройна система base(10 по подразбиране)

  BigNumber(long=0 , int=0, int=10)

- По подадено дробно число, като задължително като втори аргумент се посочва прецизност – например ако подадем (124.6532,4) получвамае 124,6532, но ако подадем с втори аргумент 3, ще имаме 124.653

  BigNumber(double,int)

Това конструиране става по конвенционалния алгоритъм – в обратен ред(т.е. последователно при нашето представяне) се запълва масива от цифри с остатъка от делене на числото на базата, след което продължаваме върху частното от деленето до получаване на 0. При дробно число вземаме цялата част, прилагаме върху нея алгоритъма, а дробната умножаваме по съответната точност и прилагаме върху нея алгоритъма, като обаче накрая обръщаме масива при записването му(за дробната част по удобно е представянето в правата посока); това се извършва от същия помощен метод, но двата случая се разграничават с булев параметър reversed

Класът предоставя голяма четворка и извеждане на стандартния изход, като то се определя от текущата бройна система, в която се разглежда числото (присъства като state variable в обекта). **Входът от клавиатурата** се извършва с помощта на структура-буфер, която въвежда до срещане на разделител, различен от цифра или десетична точка. След това масивът се разделя по точката, ако я има, и цифрите се записват в подходящия ред в подходящия масив в зависимост от бройната система.

Обектите от класа поддържат всички стандартни аритметични действия, вкл инфиксен минус, събиране и изваждане, умножение, деление и деление с остатък. **Събиране и изваждане** се реализират със стандартния алгоритъм – събиране на съответните цифри с преноси. Това се извършва и върху цялата, и върху дробната част. **Умножението** се осъществява рекурсивно – масивите от цифрите на всяко число се разделят по средата и се свежда до по-малки случаи, като се ползва равенството ab*cd = (a*base^k +b)(c*base^m+d) = (ac)*base^(k+m) + (ad)*base^k + (bc)*base^m + bd. Използва се, че ac,ad,bc,bd са умножение на числа с по-малък брой цифри, а

умножението със степен на основата се извършва просто като добавяне на padding. Рекурсията завършва в базовия случай, когато едно от числата е едноцифрено – тогава просто умножаваме по всяка цифра с преноси. Сходен алгоритъм се реализира и при умножението на дроби, като първо премества десетичната запетая така че числата да са цели, извършваме умножението по този алгоритъм, после връщаме десетичната запетая на мястото ѝ.

Делението се реализира по алгоритъм long division. За целта изравняваме броя на цифрите на числото отдясно с делимото и натрупваме във временна променлива кратни на делителя, докато надхвърлим даденото. Тогава се връщаме стъпка назад и записваме броя пъти, по които сме умножили, като съответна цифра к. Вадим к пъти делимото от нашето число и получаваме остатъка, за който продължаваме да прилагаме алгоритъма: изравняваме броя на цифрите, делим и така нататък. Продължаваме, докато не получим число, по-малко от самия делител. Тогава това е остатъкът, а полученото дотук число от масива от натрупани цифри е частното. При дробни числа отново умножаваме по степен на базата, за да сведем до съответната задача за цели числа, но вземаме предвид и точността – брой цифри след десетичната запетая, зададена като статична променлива precision на класа.

Възможен е също **преход от една бройна система в друга**, като от 10-ична към 16-ична използваме стандартния алгоритъм: делим числото на 16, запазваме остатъка, продължаваме с частното. За обратния преход използваме алгоритъма, базиран на метода на Хорнер: започваме отзад напред и на всяка стъпка умножаваме полученото дотук по 16 и прибавяме следващата цифра. Накрая получаваме желаното представяне в десетична бройна система. Можем както да конвертираме дадено число, като сменим и текущата му база, така и да получим негово копие в съответната бройна система.

```
void convertToDecimal();
void convertToBase();
BigNumber getDecimal() const;
BigNumber getHex() const;
```

Като бонус са реализирани и **побитовите операции**. Те използват 16-ичното представяне на числото, за да получат от него двоично, прилагат побитово операциите върху него чрез помощни функции и конвертират полученото двоично число до 16-ично, а при необходимост и до десетично с функциите за преминаване в бройни системи. За целта се използва бързия преход между 2-ична и 16-ична система – всяка 16-ична цифра се замества с поредицата от четирите двоични, които отговарят на стойността ѝ, и обратното.

Класът предоставя **операции за сравнение**, които действат по очаквания начин – първо гледат знака на числото, после броя цифри, а при равен брой цифри сравняват и лексикографски целите, а при нужда и дробните части. Освен това има и някои други помощни методи – достъпване на цифра по позиция (директно поради избраната имплементация), проверка за знак, за равенство с нула, дали числото е цяло, с колко цифри се записва в подадена като параметър бройна система. Функцията abs връща абсолютната стойност на числото.

```
char& operator[](int);
char at(int) const;

bool isPositive() const { return sign==1; }
bool isNegative() const { return sign==-1; }
bool isZero() const { return sign == 0; }
bool isInteger() const { return numberOfFracDigits==0; }
BigNumber abs(BigNumber const&) const;
```