Разработать ООП вычисления дополнения обыкновенной дроби до ближайшего целого числа, значение которого будет не меньше ее величины по модулю. Символьная запись дроби должна передаваться программе в качестве аргумента командной строки, где числитель и знаменатель разделены знаком ‘/’. Результирующее дополнение должно отображаться строкой стандартного вывода в том же символьном формате. Программная реализация вычислений должна быть основана на разработке класса обыкновенной дроби с приватными полями целочисленных данных для ее числителя и знаменателя, а также публичным компонентным методом перегрузки оператора ‘~’. Конструкторы этого класса должны обеспечивать преобразование символьной записи дроби в числовой формат и инициализацию значений ее числителя и знаменателя. Кроме того, нужно предусмотреть компонентный метод приведения дроби к несократимому виду, использующий алгоритм Евклида для вычисления наибольшего общего делителя (НОД) ее числителя и знаменателя, а также компонентный оператор преобразования числовой записи дроби в формат символьной строки для стандартного вывода результатов вычислений.

```
1:  #include <stdio.h>
2:  #include <string.h>
3:  #include <stdlib.h>
4:  #include <iostream>
5:
6:  using std::cout;
7:  using std::endl;
8:  int euclide(int, int);
9:
10: class Fraction
11: {
12:   private:
13:     int nom;
14:     int den;
15:   public:
16:     Fraction(char*);
17:     Fraction(int n=0, int m=1) : nom(n), den(m) {};
18:     Fraction operator~();
19:     operator char*();
20:     void reduce();
21: };
22:
23: Fraction::Fraction(char* s)
24: {
25:   char* p = strchr(s, '/');
26:   den = 1;
27:   if(p != NULL)
28:   {
29:     *p++ = '\0';
30:      den = atoi(p);
31:   }
32:  nom = atoi(s);
33: }
34:
35: Fraction Fraction::operator~()
```

```
36:  {
37:    int k = 0;
38:    int sign = (nom < 0) ? -1 : 1;
39:    int n = sign * nom;
40:    while(k < n)
41:       k += den;
42:    n = (k - n);
43:   return Fraction(n * sign, den);
44:  }
45:
46:  Fraction::operator char* ()
47:  {
48:    static char s[32];
49:    sprintf(s, "%d/%d", nom, den);
50:   return s;
51:  }
52:
53:  void Fraction::reduce()
54:  {
55:    int gcd;
56:    gcd = euclide(abs(nom), den);
57:    nom /= gcd;
58:    den /= gcd;
59:    return;
60:  }
61:
62:  int euclide(int n, int m)
63:  {
64:    int r = 1;
65:    while(n != 0)
66:    {
67:       r = m % n;
68:       m = n;
69:       n = r;
70:    }
71:    return(m);
72:  }
73:
74:  int main(int argc, char* argv[])
75:  {
76:    if(argc < 2)
77:       return(puts("Usage: complement nomerator/denomerator"));
78:    Fraction x(argv[1]);
79:    Fraction y;
80:    x.reduce();
81:    y = ~x;
82:    cout << (char*) y << endl;
83:   return(0);
84:  }
```

Обыкновенные дроби – это записи вида $\frac{m}{n}$ (или *m/n*), где *m* и *n* – любые натуральные числа.

Например, *5/10*, $\frac{11}{8002}$, $\frac{35}{35}$, *21/1*, *9/4*, $\frac{555}{37}$ - обыкновенные дроби, а записи $\frac{\sqrt{3}}{2}$; $\frac{12}{36}$; $\frac{3}{\frac{5}{4}}$; $\frac{1,2}{7,6}$; $\frac{-7}{9}$ не подходят под определение обыкновенных дробей.

Обыкновенные дроби подразделяются на сократимые и несократимые. По названиям можно догадаться, что сократимые дроби можно сократить, а несократимые – нельзя.

Сократить дробь значит разделить ее числитель и знаменатель на их положительный и отличный от единицы общий делитель. В результате сокращения дроби получается новая обыкновенная дробь с меньшим числителем и знаменателем, причем полученная дробь равна исходной.

Обычно конечной целью сокращения дроби является получение несократимой дроби, которая равна исходной сократимой дроби. Эта цель может быть достигнута, если провести сокращение исходной сократимой дроби на наибольший общий делитель (НОД) ее числителя и знаменателя. В результате такого сокращения всегда получается несократимая дробь

$$\frac{a : НОД(a, b)}{b : НОД(a, b)}$$

Несократимость дроби гарантирует тот факт, что $a : НОД(a, b)$ и $b : НОД(a, b)$ - взаимно простые числа.

Наибольший общий делитель (НОД) – это число, которое делит без остатка два числа и делится само без остатка на любой другой делитель данных двух чисел. Проще говоря, это самое большое число, на которое можно без остатка разделить два числа, для которых ищется НОД.

Алгоритм Евклида позволяет с легкостью вычислить наибольший общий делитель для двух положительных чисел. Суть алгоритма

заключается в том, чтобы последовательно проводить деление с остатком, в ходе которого получается ряд равенств вида:

$$
\begin{aligned}
&a = b \cdot q_1 + r_1,\ 0 < r_1 < b \\
&b = r_1 \cdot q_2 + r_2,\ 0 < r_2 < r_1 \\
&r_1 = r_2 \cdot q_3 + r_3,\ 0 < r_3 < r_2 \\
&r_2 = r_3 \cdot q_4 + r_4,\ 0 < r_4 < r_3 \\
&\vdots \\
&r_{k-2} = r_{k-1} \cdot q_k + r_k,\ 0 < r_k < r_{k-1} \\
&r_{k-1} = r_k \cdot q_{k+1}
\end{aligned}
$$

Деление заканчивается при $r_{k+1} = 0$

При этом НОД определяется как

$$r_k = НОД(a,\ b)$$

Пример нахождения НОД чисел 645 и 381 алгоритмом Евклида:

Решение:

645 : 381 = 1 (ост.294)
381 : 294 = 1 (ост.87)
294 : 87 =3 (ост. 33)
33 : 3 = 11

Следовательно, НОД(645,381)=3.