Задание.

Задания

Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

– метод инициализации init;
– ввод с клавиатуры read;
– вывод на экран display;
– преобразование в строку toString.

Все задания должны быть реализованы двумя способами:

- тип данных представляется структурой с необходимыми полями, а операции реализуются как внешние функции, которые получают объекты данного типа в качестве аргументов;
– как класс с закрытыми полями, где операции реализуются как методы класса.

15. Создать класс Fraction для работы с дробными числами. Число должно быть представлено двумя полями: целая часть – длинное целое со знаком, дробная часть – беззнаковое короткое целое. Реализовать арифметические операции сложения, вычитания, умножения и операции сравнения.

Код.

```
#include <iostream>

using namespace std;

class Fraction
{
   long int whole_part;
   unsigned short int fraction_part;

   void fractional_cast(Fraction &a);
   void fraction_reduction();

   public:
        void init(int wp, int fp);
        void display();
        void read();

        void add(Fraction &a);
        void sub(Fraction &a);
        void mul(Fraction &a);
        void equal(Fraction &a);
};

void Fraction::fractional_cast(Fraction &a)
{
```

```
    this->whole_part *= a.fraction_part;
    a.whole_part *= this->fraction_part;
    this->fraction_part *= a.fraction_part;
    a.fraction_part = this->fraction_part;
}

void Fraction::fraction_reduction()
{
    int x = abs(this->whole_part), y = this->fraction_part;
    while (x != 0 && y != 0)
    {
        (x > y) ? x%= y : y%=x;
    }
    this->whole_part /= (x + y);
    this->fraction_part /= (x + y);
}

void Fraction::init(int wp, int fp)
{
    if (fp > 0)
    {
        this->whole_part = wp;
        this->fraction_part = fp;
        return;
    }
    cout << "Incorrect data! Default values were setted!\n";
    this->whole_part = 1;
    this->fraction_part = 1;
}

void Fraction::display()
{
    cout << this->whole_part << "/" << this->fraction_part <<  "\n";
}

void Fraction::read()
{
    int wp, fp;
    cout << "Enter whole part: ";
    cin >> wp;
    cout << "Enter fraction part: ";
    cin >> fp;
    this->init(wp, fp);
```

```
}

void Fraction::add(Fraction &a)
{
   Fraction b;
   b.init(a.whole_part, a.fraction_part);
   this->fractional_cast(b);
   this->whole_part += b.whole_part;
   this->fraction_reduction();
}

void Fraction::sub(Fraction &a)
{
   Fraction b;
   b.init(a.whole_part, a.fraction_part);
   this->fractional_cast(b);
   this->whole_part -= b.whole_part;
   this->fraction_reduction();
}

void Fraction::mul(Fraction &a)
{
   this->whole_part *= a.whole_part;
   this->fraction_part *= a.fraction_part;
   this->fraction_reduction();
}

void Fraction::equal(Fraction &a)
{
   Fraction c, d;
   c.init(this->whole_part, this->fraction_part);
   d.init(a.whole_part, a.fraction_part);
   c.fractional_cast(d);
   if (c.whole_part > d.whole_part)
   {
       c.fraction_reduction(); d.fraction_reduction();
       cout << c.whole_part << "/" << c.fraction_part << " is bigger
that " << d.whole_part << "/" << d.fraction_part << "\n";
   }
   else if (c.whole_part < d.whole_part)
   {
       c.fraction_reduction(); d.fraction_reduction();
```

```
        cout << d.whole_part << "/" << d.fraction_part << " is bigger
that " << c.whole_part << "/" << c.fraction_part << "\n";
    }
    else
    {
        c.fraction_reduction(); d.fraction_reduction();
        cout << c.whole_part << "/" << c.fraction_part << " Equal " <<
d.whole_part << "/" << d.fraction_part << "\n";
    }
}

int main()
{
    Fraction tests[2];
    for (int i = 0; i < 2; i++)
    {
        tests[i].read();
    }
    tests[0].add(tests[1]);
    tests[0].display();
    tests[1].sub(tests[0]);
    tests[1].display();
    tests[0].mul(tests[1]);
    tests[0].display();
    tests[0].equal(tests[1]);
    tests[0].equal(tests[0]);
    return 0;
}
```

Результат.

```
Enter whole part: 1
Enter fraction part: 2
Enter whole part: 1
Enter fraction part: 4
3/4
-1/2
-3/8
-3/8 is bigger that -1/2
-3/8 Equal -3/8
```