Задание.

Задания

В каждом упражнении требуется реализовать в том или ином виде определение нового класса. Во всех заданиях необходимо реализовать конструктор инициализации (один или несколько), конструктор копирования и конструктор без аргументов.

Для демонстрации работы с объектами нового типа во всех заданиях требуется написать главную функцию. В программе обязательно должны быть продемонстрированы различные способы создания объектов и массивов объектов. Программа должна демонстрировать использование всех функций и методов.

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>
#include <cmath>
#include <string>

using namespace std;

class Fraction
{
   long int whole_part; // более конкретно - числитель.
   unsigned short int fractional_part; // более конкретно -
знаменатель.

   void fractional_cast(Fraction &a); // приведение дробей к общему
знаменателю.
   void fractional_reduction(); // сокращение дроби при помощи
алгоритма Евклида.

   public:
       // перегрузка конструкторов требуется в задании.
       Fraction(); // конструктор без параметров.
       Fraction(int wp, int fp); // "стандартный" конструктор.
       Fraction(const Fraction &a); // конструктор копирования.
       // перегрузка для строковых параметров.
       Fraction(string wp, string fp);
       Fraction(string wp, int fp);
       Fraction(int wp, string fp);

       void read();
       void display();
       void add(Fraction &a);
```

```
        void sub(Fraction &a);
        void mul(Fraction &a);
        void equal(Fraction &a);

};

Fraction::Fraction()
{
    this->whole_part = 1;
    this->fractional_part = 1;
}

Fraction::Fraction(int wp, int fp)
{
    if (fp > 0){ this->whole_part = wp; this->fractional_part = fp;
return; }
    this->whole_part = 1;
    this->fractional_part = 1;
    cout << "Object was not created correctly(were setted default
values)! You've entered a wrong data!\n";
}

Fraction::Fraction(const Fraction &a)
{
    this->whole_part = a.whole_part;
    this->fractional_part = a.fractional_part;
}

Fraction::Fraction(string wp, string fp)
{
    int WP = stoi(wp), FP = stoi(fp);
    if (FP > 0){ this->whole_part = WP; this->fractional_part = FP;
return; }
    this->whole_part = 1;
    this->fractional_part = 1;
    cout << "Object was not created correctly(were setted default
values)! You've entered a wrong data!\n";
}

Fraction::Fraction(string wp, int fp)
{
    int WP = stoi(wp);
```

```
    if (fp > 0){ this->whole_part = WP; this->fractional_part = fp;
return; }
    this->whole_part = 1;
    this->fractional_part = 1;
    cout << "Object was not created correctly(were setted default
values)! You've entered a wrong data!\n";
}

Fraction::Fraction(int wp, string fp)
{
    int FP = stoi(fp);
    if (FP > 0){ this->whole_part = wp; this->fractional_part = FP;
return; }
    this->whole_part = 1;
    this->fractional_part = 1;
    cout << "Object was not created correctly(were setted default
values)! You've entered a wrong data!\n";
}

void Fraction::fractional_cast(Fraction &a)
{
    this->whole_part *= a.fractional_part;
    a.whole_part *= this->fractional_part;
    this->fractional_part *= a.fractional_part;
    a.fractional_part = this->fractional_part;
}

void Fraction::fractional_reduction()
{
    int x = abs(this->whole_part), y = this->fractional_part;
    while (x != 0 && y != 0)
    {
        (x > y) ? x %= y : y %= x;
    }
    // (x + y) - наибольший общий делитель.
    this->whole_part /= (x + y);
    this->fractional_part /= (x + y);
}

void Fraction::read()
{
    int wp, fp;
    cout << "Enter a whole part: ";
```

```
    cin >> wp;
    cout << "Enter a fractional part: ";
    cin >> fp;
    if (fp > 0){ this->whole_part = wp; this->fractional_part = fp;
return; }
    cout << "Incorrect values!\n";
}

void Fraction::display()
{
    if (this->fractional_part != 0){ cout << this->whole_part << "/" <<
this->fractional_part << "\n"; }
    else cout << "None\n";
}

void Fraction::add(Fraction &a)
{
    this->fractional_cast(a);
    this->whole_part += a.whole_part;
    this->fractional_reduction();
}

void Fraction::sub(Fraction &a)
{
    this->fractional_cast(a);
    this->whole_part -= a.whole_part;
    this->fractional_reduction();
}

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

void Fraction::equal(Fraction &a)
{
    int x = this->whole_part * a.fractional_part, y = a.whole_part *
this->fractional_part;
    if (x > y) cout << "Greater\n";
    else if (x < y) cout << "Less\n";
    else cout << "Equal\n";
```

```
}

int main()
{
   Fraction fractions[2], num1("1", "3"), num2(4, 16),
num3(fractions[0]);
   cout << "Massive of Fraction objects:\n";
   for (int i = 0; i < 2; i++){ fractions[i].read(); cout <<
"fraction[" << i << "] + num1 = " ; fractions[i].add(num1);
fractions[i].display(); }
   cout << "num2 - num3\n";
   num2.sub(num3);
   num2.display();
   cout << "fractions[1] ? num3\n";
   fractions[1].equal(num3);
   cout << "num3 * num1\n";
   num3.mul(num1);
   num3.display();
   return 0;
}
```

Результат.

```
Massive of Fraction objects:
Enter a whole part: 1
Enter a fractional part: 2
fraction[0] + num1 = 5/6
Enter a whole part: 2
Enter a fractional part: 3
fraction[1] + num1 = 1/1
num2 - num3
-3/4
fractions[1] ? num3
Equal
num3 * num1
1/3
```