

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа №2
по курсу «ООП»

Тема:
Перегрузка операторов

Студент:	Вахрамьян К.О.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	3
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

Trational.h:

```
#ifndef _CLASS_H_
#define _CLASS_H_
#include <iostream>
#include <cstdlib>
#include <math.h>
#include <cstdint>
#include <cstdio>

class TRational {
private:
    int a;
    int b;
public:
    TRational() {
        a = 0;
        b = 0;
    }
    TRational(int a, int b) : a(a), b(b) {}
    friend TRational operator + (const TRational& , const TRational& );
    friend TRational operator - (const TRational& , const TRational& );
    friend TRational operator / (const TRational& , const TRational& );
    friend TRational operator * (const TRational& , const TRational& );
    friend bool operator > (const TRational& , const TRational& );
    friend bool operator == (const TRational& , const TRational& );
    friend bool operator < (const TRational& , const TRational& );
    friend std::ostream& operator << (std::ostream& out, const TRational& Rational);
    friend std::istream& operator >> (std::istream &in, TRational& Rational);

    TRational& operator *= (unsigned long long num);
    TRational Add(const TRational &d1, const TRational &d2) const;
    TRational Div(const TRational &d1, const TRational &d2) const;
    TRational Sub(const TRational &d1, const TRational &d2) const;
    TRational Mul(const TRational &d1, const TRational &d2) const;
    int Compare(const TRational &d1) const;
    void Reduce();
    void Print() const;
};

TRational operator ""_xn (unsigned long long first);
```

```
#endif
```

TRational.cpp:

```
#include "TRational2.h"
```

```
TRational operator + (const TRational& d1, const TRational& d2) {  
    return TRational(d1.a * d2.b + d1.b * d2.a, d1.b * d2.b);  
}
```

```
TRational operator - (const TRational& d1, const TRational& d2) {  
    return TRational(d1.a * d2.b - d1.b * d2.a, d1.b * d2.b);  
}
```

```
TRational operator / (const TRational& d1, const TRational& d2) {  
    return TRational(d1.a * d2.b, d1.b * d2.a);  
}
```

```
TRational operator * (const TRational& d1, const TRational& d2) {  
    return TRational(d1.a * d2.a, d1.b * d2.b);  
}
```

```
TRational& TRational::operator*= (unsigned long long num) {  
    a = a * num;  
    b = b * 1;  
    return *this;  
}
```

```
TRational operator"" _xn(unsigned long long first) {  
    TRational P(first, 2);  
  
    return P;  
}
```

```
std::ostream& operator << (std::ostream& out, const TRational& Rational) {  
    out << Rational.a << "/" << Rational.b;  
}  
std::istream& operator >> (std::istream &in, TRational& Rational) {  
    char tmp;
```

```

    in >> Rational.a >> tmp >> Rational.b;

}

bool operator > (const TRational& d1, const TRational& d2) {
    if (d1.a * d2.b > d1.b * d2.a) {

        return true;
    }
    return false;
}

bool operator == (const TRational& d1, const TRational& d2) {
    if (d1.a * d2.b == d1.b * d2.a) {
        return true;
    }
    return false;
}

bool operator < (const TRational& d1, const TRational& d2) {
    if (d1.a * d2.b < d1.b * d2.a) {
        return true;
    }
    return false;
}

}

TRational TRational::Add(const TRational &d1, const TRational &d2) const{
    TRational tmp;
    tmp = d1 + d2;
    tmp.Reduce();
    return tmp;

}

TRational TRational::Div(const TRational &d1, const TRational &d2) const{
    TRational tmp;
    tmp = d1 / d2;
    tmp.Reduce();
    return tmp;
}

TRational TRational::Sub(const TRational &d1, const TRational &d2) const{
    TRational tmp;
    tmp = d1 - d2;
    tmp.Reduce();
    return tmp;
}

```

```

TRational TRational::Mul(const TRational &d1, const TRational &d2) const{
    TRational tmp;
    tmp = d1 * d2;
    tmp.Reduce();
    return tmp;
}

```

```

int TRational::Compare(const TRational &d1) const{
    if (this->b == 0) {
        return 2;
    } else if (d1.b == 0) {
        return 1;
    }
    if (*this > d1) {
        return 1;
    } else if (*this == d1) {
        return 0;
    } else {
        return 2;
    }
}

```

```

void TRational::Reduce(){
    int x = abs(this->a);
    int y = abs(this->b);
    if (x == 0 || y == 0) {
        return;
    }
    while (x != 0 && y != 0) {
        if (x > y) {
            x = x % y;
        } else {
            y = y % x;
        }
    }
    this->a = this->a / (x + y);
    this->b = this->b / (x + y);
}

```

```

void TRational::Print() const{
    TRational tmp = *this;
    if (tmp.b == 0) {
        std::cout << " -nan\n";
        return;
    }
}

```

```

    } else if (tmp.a == 0) {
        std::cout << " 0\n";
        return;
    } else if (tmp.a < 0 and tmp.b < 0) {
        tmp.a = (-1) * tmp.a;
        tmp.b = (-1) * tmp.b;
        std::cout << tmp << "\n";
        return;
    }
    std::cout << tmp << "\n";
}

```

main2.cpp:

```

#include "TRational2.h"
#include <sstream>
#include <iostream>
#include <cmath>

```

```

int main()
{
    TRational d1, d2, c;
    std::cin >> d1 >> d2;
    std::cout << "Add = ";
    (d1.Add(d1, d2)).Print();
    std::cout << "Sub = ";
    (d1.Sub(d1, d2)).Print();
    std::cout << "Mul = ";
    (d1.Mul(d1, d2)).Print();
    std::cout << "Div = ";
    (d1.Div(d1, d2)).Print();
    if (d1.Compare(d2) == 1) {
        std::cout << d1 << " > " << d2 << "\n";
    } else if (d1.Compare(d2) == 0) {
        std::cout << d1 << " = " << d2 << "\n";
    } else if (d1.Compare(d2) == 2) {
        std::cout << d1 << " < " << d2 << "\n";
    }
    c = 3_xn;

    std::cout << c;
}

```

}

2. Ссылка на репозиторий на GitHub.

https://github.com/vebcreatex7/oop_exercise_02

3. Набор testcases.

Test_01.txt

1/2 3/4

Add = 5/4

Sub = -1/4

Div = 3/8

Mul = 2/3

1/2 < 3/4

Test_02.txt

1/1 1/1

Add = 2/1

Sub = 0

Div = 1/1

Mul = 1/1

1/1 = 1/1

Test_03.txt

1/0 2/3

Add = -nan

Sub = -nan

Div = -nan

Mul = -nan

1/0 < 2/3

Test_04.txt

0/2 3/4

Add = 3/4

Sub = -3/4

Div = 0

Mul = 0

0/2 < 3/4

Test_05.txt

$-1/3 - 1/4$

Add = $-7/12$

Sub = $-1/12$

Div = $1/12$

Mul = $4/3$

$-1/3 < -1/4$

4. Объяснение результатов работы программы.

Вторая лабораторная работа в сущности выполняет ту же работу, что и первая. Однако арифметические операции реализованы при помощи перегрузки операторов.

5. Вывод.

Выполняя данную лабораторную я получил опыт работы с простыми классами, с системой сборки Cmake, с системой контроля версий git, а также изучил основы работы с классами в C++. Создал класс, научился перегружать операторы.