

Отчёт по лабораторной работе № 01 по курсу 2

студента группы М80-208Б-18, № по списку 3

Адреса www, e-mail, jabber, skype

anisimov.valera2000@yandex.ru

Работа выполнена: "16" сентября 2019г.

1. Тема:

Простые
классы

2. Цель работы: Изучение системы сборки на языке C++, изучение систем контроля версии. Изучение основ работы с классами в C++.

3. Задание (вариант № 3):

Рациональная (несократимая) дробь представляется парой целых чисел (a, b) , где a — числитель, b — знаменатель. Создать класс **Rational** для работы с рациональными дробями. Обязательно должны быть реализованы операции:

- сложения **add**, $(a, b) + (c, d) = (ad + bc, bd)$;
- вычитания **sub**, $(a, b) - (c, d) = (ad - bc, bd)$;
- умножения **mul**, $(a, b) \times (c, d) = (ac, bd)$;
- деления **div**, $(a, b) / (c, d) = (ad, bc)$;
- операции сравнения.

Должна быть реализована функция сокращения дроби `reduce()`, которая обязательно вызывается при выполнении арифметических операций.

4. Адрес репозитория на GitHub https://github.com/wAlienUFOx/oop_exercise_01

5. Код программы на C++

main.cpp

```
#include <iostream>
#include "fraction.h"
```

```
int main() {
    fractions f;
    fractions f1;
    std::cout << "Введите первую дробь\n";
    for(int i = 0; i < 2; i++)
        f.set(i);
    std::cout << "Введите вторую дробь\n";
    for(int i = 0; i < 2; i++)
        f1.set(i);
    std::cout << "Первая дробь\n";
    for(int i = 0; i < 2; i++)
    {
        std::cout << f.get(i);
        if(i < 1)
            std::cout << ' ';
    }
    std::cout << "\n";
    std::cout << "Вторая дробь\n";
    for(int i = 0; i < 2; i++)
    {
        std::cout << f1.get(i);
        if(i < 1)
```

```

        std::cout << '/';
    }
    std::cout << "\n";
    fractions sum = f._add(f1);
    sum._reduce(sum);
    std::cout << "Сумма\n";
    for(int i = 0; i < 2; i++)
    {
        std::cout << sum.get(i);
        if(i < 1)
            std::cout << '/';
    }
    std::cout << "\n";
    fractions raz = f._sub(f1);
    raz._reduce(raz);
    std::cout << "Разность\n";
    for(int i = 0; i < 2; i++)
    {
        std::cout << raz.get(i);
        if(i < 1)
            std::cout << '/';
    }
    std::cout << "\n";
    fractions pro = f._mult(f1);
    pro._reduce(pro);
    std::cout << "Произведение\n";
    for(int i = 0; i < 2; i++)
    {
        std::cout << pro.get(i);
        if(i < 1)
            std::cout << '/';
    }
    std::cout << "\n";
    fractions del = f._div(f1);
    del._reduce(del);
    std::cout << "Частное\n";
    for(int i = 0; i < 2; i++)
    {
        std::cout << del.get(i);
        if(i < 1)
            std::cout << '/';
    }
    std::cout << "\n";
    f._sravn(f1);
}

```

fraction.h

```

#ifndef D_FRACTIONS_H
#define D_FRACTIONS_H

#include <iostream>

struct fractions{
    fractions();
    fractions(int a, int b);

    int get(int i);
    void set(int i);

```

```

fractions _add(const fractions& dr) const;
fractions _sub(const fractions& dr) const;
fractions _mult(const fractions& dr) const;
fractions _div(const fractions& dr) const;
fractions _reduce(fractions& res) const;
void _srafn(const fractions& dr) const;

```

```

public:
    int arr[2];
};

```

```

#endif

```

```

fraction.cpp

```

```

#include "fraction.h"

```

```

fractions::fractions(): arr{0, 0} {}
fractions::fractions(int a, int b): arr{a, b} {}

```

```

int fractions::get(int i){
    return arr[i];
}
void fractions::set(int i){
    std::cin >> arr[i];
}

```

```

fractions fractions::_add(const fractions& dr) const{
    fractions result{};
    result.arr[0] = (arr[0] * dr.arr[1]) + (arr[1] * dr.arr[0]);
    result.arr[1] = arr[1] * dr.arr[1];
    if(arr[1] == 0){
        result.arr[0] = dr.arr[0];
        result.arr[1] = dr.arr[1];
    }
    if(dr.arr[1] == 0){
        result.arr[0] = arr[0];
        result.arr[1] = arr[1];
    }
    return result;
}
fractions fractions::_sub(const fractions& dr) const{
    fractions result{};
    result.arr[0] = (arr[0] * dr.arr[1]) - (arr[1] * dr.arr[0]);
    result.arr[1] = arr[1] * dr.arr[1];
    if(arr[1] == 0){
        result.arr[0] = -dr.arr[0];
        result.arr[1] = dr.arr[1];
    }
    if(dr.arr[1] == 0){
        result.arr[0] = arr[0];
        result.arr[1] = arr[1];
    }
    return result;
}
fractions fractions::_mult(const fractions& dr) const{
    fractions result{};
    for(int i = 0; i < 2; i++)
    {
        result.arr[i] = arr[i] * dr.arr[i];
    }
    return result;
}

```

```

}
fractions fractions::_div(const fractions& dr) const{
    fractions result{};
    result.arr[0] = arr[0] * dr.arr[1];
    result.arr[1] = arr[1] * dr.arr[0];
    return result;
}
fractions fractions::_reduce(fractions& res) const{
    fractions result{};
    if(res.arr[1] == 0)
        res.arr[0] = 0;
    if(res.arr[0] == 0)
        res.arr[1] = 0;
    if(res.arr[0] >= res.arr[1])
    {
        if(res.arr[1] > 0){
            for(int i = res.arr[1]; i > 1; i--)
            {
                if(res.arr[0] % i == 0 && res.arr[1] % i == 0)
                {
                    res.arr[0] = res.arr[0] / i;
                    res.arr[1] = res.arr[1] / i;
                }
            }
        }
    }
    else{
        for(int i = res.arr[1]; i < -1; i++)
        {
            if(res.arr[0] % i == 0 && res.arr[1] % i == 0)
            {
                res.arr[0] = res.arr[0] / (-i);
                res.arr[1] = res.arr[1] / (-i);
            }
        }
    }
}

if (res.arr[0] < res.arr[1])
{
    if(res.arr[0] > 0)
    {
        for(int i = res.arr[0]; i > 1; i--)
        {
            if(res.arr[0] % i == 0 && res.arr[1] % i == 0)
            {
                res.arr[0] = res.arr[0] / i;
                res.arr[1] = res.arr[1] / i;
            }
        }
    }
    else{
        for(int i = res.arr[0]; i < -1; i++)
        {
            if(res.arr[0] % i == 0 && res.arr[1] % i == 0)
            {
                res.arr[0] = res.arr[0] / (-i);
                res.arr[1] = res.arr[1] / (-i);
            }
        }
    }
}
return result;

```

```

}
void fractions::_sravn(const fractions& dr) const{
    if(arr[1] * dr.arr[1] != 0){
        if ((arr[0] * dr.arr[1]) < (dr.arr[0] * arr[1])){
            std::cout << "Первая дробь меньше\n";
        }
        else if ((arr[0] * dr.arr[1]) > (dr.arr[0] * arr[1])){
            std::cout << "Первая дробь больше\n";
        }
        else{
            std::cout << "Дробь равны\n";
        }
    }
    else{
        if(arr[1] == dr.arr[1]){
            std::cout << "Дробь равны\n";
        }
        if(arr[1] == 0 && dr.arr[1] != 0){
            if(dr.arr[0] > 0)
                std::cout << "Первая дробь меньше\n";
            else
                std::cout << "Первая дробь больше\n";
        }
        if(arr[1] != 0 && dr.arr[1] == 0){
            if(arr[0] < 0)
                std::cout << "Первая дробь меньше\n";
            else
                std::cout << "Первая дробь больше\n";
        }
    }
}
}
}

```

CMakeLists.txt

```
project(lab1)
```

```
add_executable(oop_exercise_01
    main.cpp
    fraction.cpp)
```

```
set(CMAKE_CXX_FLAGS
    "${CMAKE_CXX_FLAGS} -Wall -Wextra")
```

6. Набор testcases

test_01.txt	Ожидаемое действие	Ожидаемый результат
0 0 1 3	add(0/0, 1/3)	1/3
	sub(0/0, 1/3)	-1/3
	mult(0/0, 1/3)	0/0

	<code>div(0/0, 1/3)</code>	0/0
	<code>sravn(0/0, 1/3)</code>	Первая дробь меньше
test_02.txt	Ожидаемое действие	Ожидаемый результат
2 3 5 15	<code>add(2/3, 5/15)</code>	1/1
	<code>sub(2/3, 5/15)</code>	1/3
	<code>mult(2/3, 5/15)</code>	2/9
	<code>div(2/3, 5/15)</code>	2/1
	<code>sravn(2/3, 5/15)</code>	Первая дробь больше
test_03.txt	Ожидаемое действие	Ожидаемый результат
16 10 16 10	<code>add(16/10, 16/10)</code>	16/5
	<code>sub(16/10, 16/10)</code>	0/0
	<code>mult(16/10, 16/10)</code>	64/25
	<code>div(16/10, 16/10)</code>	1/1
	<code>sravn(16/10, 16/10)</code>	Дробь равны

7. Результаты выполнения тестов

```
walien@PC-name:~/2kurs/CPP/lab1/tmp$ ./oop_exercise_01 < ~/2kurs/CPP/lab1/test_01.txt
```

Введите первую дробь

Введите вторую дробь

Первая дробь

0/0

Вторая дробь

1/3

Сумма

1/3

Разность

-1/3

Произведение

0/0

Частное

0/0

Первая дробь меньше

```
walien@PC-name:~/2kurs/CPP/lab1/tmp$ ./oop_exercise_01 < ~/2kurs/CPP/lab1/test_02.txt
```

Введите первую дробь

Введите вторую дробь

Первая дробь

2/3

Вторая дробь

5/15

Сумма

1/1

Разность

1/3

Произведение

2/9

Частное

2/1

Первая дробь больше

```
walien@PC-name:~/2kurs/CPP/lab1/tmp$ ./oop_exercise_01 < ~/2kurs/CPP/lab1/test_03.txt
```

Введите первую дробь

Введите вторую дробь

Первая дробь

16/10

Вторая дробь

16/10

Сумма

16/5

Разность

0/0

Произведение

64/25

Частное

1/1

Дроби равны

8. Объяснение результатов работы программы - вывод

В fractions.h были заданы, а в fractions.cpp описаны, методы и свойства этого класса, применяемые в main.cpp.

Классы, описывают методы и свойства объектов, позволяют работать с этими объектами, не вдаваясь в подробности их реализации, что является примером абстракции данных. Такой подход незаменим при работе в групповых проектах.