1. **Тема**: Простые классы
2. **Цель работы**: Изучение системы сборки на языке C++, изучение систем контроля версии. Изучение основ работы с классами в С++.
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*)  (*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.

1. **Адрес репозитория на GitHub** <https://github.com/wAlienUFOx/oop_exercise_01>
2. **Код программы на С++**

main.cpp

#include <iostream>

#include "fraction.h"

int main() {

fractions f;

fractions f1;

std::cout << "Введите первую дробь\n";

f.\_read(std::cin);

std::cout << "Введите вторую дробь\n";

f1.\_read(std::cin);

std::cout << "Первая дробь\n";

f.\_write(std::cout);

std::cout << "Вторая дробь\n";

f1.\_write(std::cout);

fractions sum = f.\_add(f1);

std::cout << "Сумма\n";

sum.\_write(std::cout);

fractions raz = f.\_sub(f1);

std::cout << "Разность\n";

raz.\_write(std::cout);

fractions pro = f.\_mult(f1);

std::cout << "Произведение\n";

pro.\_write(std::cout);

fractions del = f.\_div(f1);

std::cout << "Частное\n";

del.\_write(std::cout);

int r = f.\_sravn(f1);

if (r == -1)

std::cout << "Первая дробь меньше\n";

else if (r == 1)

std::cout << "Первая дробь больше\n";

else

std::cout << "Дроби равны\n";

}

fraction.h

#ifndef D\_FRACTIONS\_H

#define D\_FRACTIONS\_H

#include <iostream>

struct fractions{

fractions();

fractions(int a, int b);

void \_read(std::istream& is);

void \_write(std::ostream& os) const;

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();

int \_sravn(const fractions& dr) const;

public:

int arr[2];

};

#endif

fraction.cpp

#include "fraction.h"

#include <algorithm>

fractions::fractions(): arr{0, 0} {}

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

void fractions::\_read(std::istream& is){

for(int i = 0; i < 2; i++)

is >> arr[i];

}

void fractions::\_write(std::ostream& os) const{

for(int i = 0; i < 2; i++)

{

os << arr[i];

if(i < 1)

os << '/';

}

os << '\n';

}

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];

result.\_reduce();

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];

result.\_reduce();

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];

}

result.\_reduce();

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];

result.\_reduce();

return result;

}

fractions fractions::\_reduce() {

int g = std::\_\_gcd(arr[0], arr[1]);

arr[0] /= g;

arr[1] /= g;

return \*this;

}

int fractions::\_sravn(const fractions& dr) const{

int result;

if(arr[0] \* dr.arr[1] < arr[1]\*dr.arr[0]){

result = -1;

}

else if(arr[0] \* dr.arr[1] > arr[1]\*dr.arr[0]){

result = 1;

}

else

result = 0;

return result;

}

CMakeLists.txt

cmake\_minimum\_required (VERSION 3.5)

project(lab1)

add\_executable(oop\_exercise\_01

main.cpp

fraction.cpp)

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -Wall -Wextra")

set\_target\_properties(oop\_exercise\_01 PROPERTIES CXX\_STANDART 14 CXX\_STANDART\_REQUIRED ON)

1. **Набор testcases**

test\_01.txt Ожидаемое действие Ожидаемый результат

0 1 add(0/0, 1/3) 1/3

1 3

sub(0/0, 1/3) -1/3

mult(0/0, 1/3) 0/0

div(0/0, 1/3) 0/0

sravn(0/0, 1/3) Первая дробь меньше

test\_02.txt Ожидаемое действие Ожидаемый результат

2 3 add(2/3, 5/15) 1/1

5 15

sub(2/3, 5/15) 1/3

mult(2/3, 5/15) 2/9

div(2/3, 5/15) 2/1

sravn(2/3, 5/15) Первая дробь больше

test\_03.txt Ожидаемое действие Ожидаемый результат

16 10 add(16/10, 16/10) 16/5

16 10

sub(16/10, 16/10) 0/0

mult(16/10, 16/10) 64/25

div(16/10, 16/10) 1/1

sravn(16/10, 16/10) Дроби равны

1. **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

1. **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 метода 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.