**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ’ЯЗКУ**

**Звіт**

**з дисципліни Проектний Практикум**

**Лабораторна робота №4**

**на тему: «Об’єктно орієнтоване програмування»»**

Виконав: студент групи ІПЗ-3.04

Бухта М.М

       \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив: Багачук Д.Г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Одеса  2023**

**Завдання №1**

**Опис завдання:**

1. Напишіть клас Fraction, який має два цілочисельних члени: чисельнік і займенник. Реалізуйте функцію print(), яка виводитиме дріб.
2. Додайте перевантаження оператора множення ( \* ) для виконання операції множення об’єкту класу Fraction на цілочисельне значення і для множення двох об’єктів класу Fraction. Використовуйте спосіб перевантаження оператора через дружню функцію.

**Код програми:**

**main.cpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #4;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\* Task 1;                          \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#include "Fraction.hpp"

#include <utility>

#include <iostream>

using namespace lab\_4;

void do\_fragment\_a(void) {

    std::cout << "Start task 1(a):" << std::endl;

    Fraction f1{1, 4};

    f1.print();

    Fraction f2{1, 2};

    f2.print();

}

void do\_fragment\_b(void) {

    std::cout << "Start task 1(b):" << std::endl;

    Fraction f1{3, 4};

    f1.print();

    Fraction f2{2, 7};

    f2.print();

    Fraction f3{f1 \* f2};

    f3.print();

    Fraction f4{f1 \* 3};

    f4.print();

    Fraction f5{3 \* f2};

    f5.print();

    Fraction f6{Fraction{1, 2} \* Fraction{2, 3} \* Fraction{3, 4}};

    f6.print();

}

int main(int argc, char \*\*argv) {

    do\_fragment\_a();

    do\_fragment\_b();

    return 0;

}

**Fraction.hpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #4;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\* Task 1;                          \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#ifndef BUKHTAMYKYTA\_LAB\_4\_TASK\_1\_FRACTION\_HPP

#define BUKHTAMYKYTA\_LAB\_4\_TASK\_1\_FRACTION\_HPP

#include <inttypes.h>

namespace lab\_4 {

class Fraction {

public:

    explicit Fraction(int32\_t numerator = 0, int32\_t pronount = 0);

    Fraction operator\* (const Fraction &other) const noexcept;

    Fraction operator\* (int32\_t numerator) const noexcept;

    friend Fraction operator\* (int32\_t numerator, const Fraction &fraction) noexcept;

    void print(void) const;

private:

    int32\_t m\_numerator;

    int32\_t m\_pronount;

};

} *// !lab\_4;*

#endif *// !BUKHTAMYKYTA\_LAB\_4\_TASK\_1\_FRACTION\_HPP;*

**Fraction.cpp**

#include "Fraction.hpp"

#include <iostream>

namespace lab\_4 {

Fraction::Fraction(int32\_t numerator, int32\_t pronount)

    : m\_numerator{numerator}, m\_pronount(pronount)

{

}

Fraction Fraction::operator\* (const Fraction &other) const noexcept{

    return Fraction{

        this->m\_numerator \* other.m\_numerator,

        this->m\_pronount \* other.m\_pronount

    };

}

Fraction Fraction::operator\* (int32\_t numerator) const noexcept {

    return Fraction{

        this->m\_numerator \* numerator,

        this->m\_pronount

    };

}

Fraction operator\* (int32\_t numerator, const Fraction &fraction) noexcept{

    return fraction \* numerator;

}

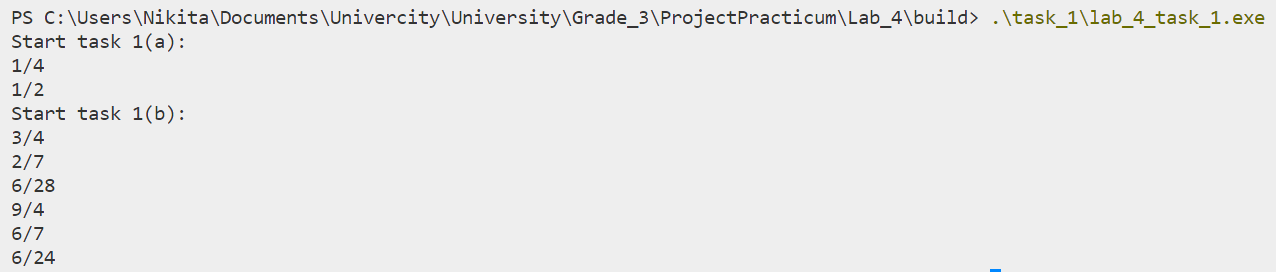
void Fraction::print(void) const {

    std::cout << m\_numerator << '/' << m\_pronount << std::endl;

}

} *// lab\_4;*

**Результат виконання програми:**



**Завдання №2**

**Опис завдання:**

Використовуючи клас Fraction , представлений у лабораторній роботі, додайте перевантаження операторів << і >> .

**Код програми:**

**main.cpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #4;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\* Task 2;                          \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#include "Fraction.hpp"

#include <iostream>

using namespace lab\_4;

int main(int argc, char \*\*argv) {

    Fraction f1;

    std::cout << "Input fraction 1: ";

    std::cin >> f1;

    Fraction f2;

    std::cout << "Input fraction 2: ";

    std::cin >> f2;

    std::cout << f1 << " \* " << f2 << " is " << (f1 \* f2) << std::endl;

    return 0;

}

**Fraction.hpp**

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\* Laboratory work #4;              \**

*\* Student Bukhta Mykyta;           \**

*\* Grade: 3;                        \**

*\* Group Software Engineering 3.04; \**

*\* Task 2;                          \**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*/*

#ifndef BUKHTAMYKYTA\_LAB\_4\_TASK\_1\_FRACTION\_HPP

#define BUKHTAMYKYTA\_LAB\_4\_TASK\_1\_FRACTION\_HPP

#include <istream>

#include <ostream>

namespace lab\_4 {

class Fraction {

private:

    int m\_numerator;

    int m\_denominator;

public:

    Fraction(int numerator = 0, int decominator = 1);

    friend Fraction operator\* (const Fraction &f1, const Fraction &f2);

    friend Fraction operator\* (const Fraction &f1, int value);

    friend Fraction operator\* (int value, const Fraction &f1);

    friend std::istream& operator>> (std::istream & in, Fraction &f1);

    friend std::ostream& operator<< (std::ostream & out, const Fraction &f1);

    static int nod(int a, int b);

    void reduce();

    void print();

};

} *// !lab\_4;*

#endif *// !BUKHTAMYKYTA\_LAB\_4\_TASK\_1\_FRACTION\_HPP;*

**Fraction.cpp**

#include "Fraction.hpp"

#include <iostream>

#include <sstream>

#include <array>

namespace lab\_4 {

Fraction::Fraction(int numerator, int decominator):

    m\_numerator(numerator), m\_denominator(decominator)

{

*// Ми помістили метод reduce() в конструктор, щоб переконатися, що всі дробі, які у нас є, будут зменшені.*

*// всі дробі, які будут перезаписані, повинні бути повторно зменшенні*

    reduce();

}

Fraction operator\* (const Fraction &f1, const Fraction &f2)

{

    return Fraction(f1.m\_numerator \* f2.m\_numerator, f1.m\_denominator \* f2.m\_denominator);

}

Fraction operator\* (const Fraction &f1, int value)

{

    return Fraction(f1.m\_numerator \* value, f1.m\_denominator);

}

Fraction operator\* (int value, const Fraction &f1)

{

    return Fraction(f1.m\_numerator \* value, f1.m\_denominator);

}

std::istream& operator>> (std::istream & in, Fraction &f1) {

    std::string segment;

    std::string input;

    in >> input;

    std::stringstream ss(input);

    std::array<int\*, 2> metadata = {&f1.m\_numerator, &f1.m\_denominator};

    int counter{0};

    while (std::getline(ss, segment, '/') && counter != metadata.size()) {

        \*metadata[counter++] = std::stoi(segment);

    }

    return in;

}

std::ostream& operator<< (std::ostream & out, const Fraction &f1) {

    out << f1.m\_numerator << '/' << f1.m\_denominator;

    return out;

}

int Fraction::nod(int a, int b)

{

    return b == 0 ? a : nod(b, a % b);

}

void Fraction::reduce()

{

    int nod = Fraction::nod(m\_numerator, m\_denominator);

    m\_numerator /= nod;

    m\_denominator /= nod;

}

void Fraction::print()

{

    std::cout << m\_numerator << "/" << m\_denominator << "\n";

}

} *// lab\_4;*

**Результат виконання програми:**

