МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

# **Лабораторная работа № 5**

# **Шаблоны классов. Работа с исключительными**

# **ситуациями языка С++**

Руководитель,

старший преподаватель Петрова О.Б.

подпись, дата

Исполнитель,

группа ИКПИ-33 К о н ь к о в М . Д .

подпись, дата

**Условия работы:**

****1.** Постановка задачи: В рамках лабораторной работы необходимо выполнить следующие задания:**

* Преобразовать в шаблон класс, разработанный в первой лабораторной работе (в данном случае — класс Rational), чтобы он поддерживал работу с различными числовыми типами (например, int, float и double).
* Разработать шаблонный класс для контейнера. Выбрана **задача №2**: разработать шаблон класса для статического одномерного массива. При этом требуется предусмотреть обработку исключительных ситуаций.

**Описание функциональности:**

* **Класс** Rational**:** Представляет рациональные числа и позволяет выполнять операции сложения, вычитания, умножения, деления и сравнения. Реализованы проверки деления на ноль через исключения.
* **Класс** StaticArray**:** Контейнер для хранения элементов фиксированного размера. Предусмотрена защита от выхода за пределы массива через обработку исключений.

### 2. Описание классов с их составляющими:

#### 2.1. Класс Rational

**Назначение:** Реализует работу с рациональными числами (дробями) с поддержкой шаблонных типов.

**Состав:**

* **Поля:**
  + T numerator — числитель.
  + T denominator — знаменатель.
* **Методы:**
  + void reduce() — приватный метод для сокращения дроби.
  + Rational() — конструктор по умолчанию, создаёт дробь 0/1.
  + Rational(T num, T den) — конструктор с параметрами. Проверяет знаменатель на равенство нулю, выбрасывая исключение std::invalid\_argument.
  + Арифметические операторы (+, -, \*, /) — реализуют сложение, вычитание, умножение и деление дробей.
  + Операторы сравнения (==, !=, <, >, <=, >=) — поддерживают сравнение дробей.
* **Исключительные ситуации:**
  + Проверка знаменателя на ноль при создании дроби или делении (выбрасывается std::invalid\_argument).

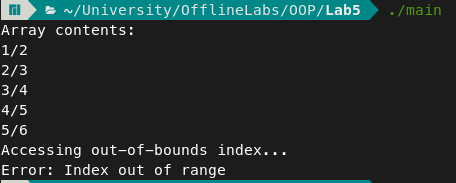
#### 2.2. Класс StaticArray

**Назначение:** Реализует статический одномерный массив фиксированного размера, способный работать с любым типом данных.

**Состав:**

* **Поля:**
  + T data[N] — массив фиксированного размера.
  + size\_t N — размер массива (определяется шаблоном).
* **Методы:**
  + StaticArray() — конструктор по умолчанию, заполняет массив значениями по умолчанию для типа T.
  + T& operator[](size\_t index) — оператор доступа к элементу. Выбрасывает исключение std::out\_of\_range, если индекс выходит за пределы массива.
  + size\_t size() const — возвращает размер массива.
  + void display() const — выводит содержимое массива.
* **Исключительные ситуации:**
  + Проверка индекса на выход за пределы массива. При некорректном доступе выбрасывается std::out\_of\_range.

**Результат работы и проверки:**



**Листинг кода:**

**main.cpp:**

#include "Rational.h"

#include "StaticArray.h"

#include <iostream>

int main() {

try {

// Создаём массив рациональных чисел

StaticArray<Rational<int>, 5> array;

// Заполняем массив

for (size\_t i = 0; i < array.size(); i++) {

array[i] = Rational<int>(i + 1, i + 2); // Пример: 1/2, 2/3, 3/4, ...

}

// Выводим содержимое массива

std::cout << "Array contents:" << std::endl;

for (size\_t i = 0; i < array.size(); i++) {

array[i].display();

}

// Пробуем обратиться к недопустимому индексу (генерация исключения)

std::cout << "Accessing out-of-bounds index..." << std::endl;

array[10].display();

} catch (const std::exception& e) {

std::cerr << "Error: " << e.what() << std::endl;

}

return 0;

}

**Rational.h:**

#ifndef RATIONAL\_H

#define RATIONAL\_H

#include <iostream>

#include <numeric>

#include <stdexcept>

template <typename T>

class Rational {

private:

T numerator; // Числитель

T denominator; // Знаменатель

// Приватная функция для сокращения дроби

void reduce() {

T gcd = std::gcd(numerator, denominator);

numerator /= gcd;

denominator /= gcd;

// Если знаменатель отрицательный, переносим знак в числитель

if (denominator < 0) {

numerator = -numerator;

denominator = -denominator;

}

}

public:

// Конструктор по умолчанию (создаёт 0/1)

Rational() : numerator(0), denominator(1) {}

// Конструктор с параметрами

Rational(T num, T den) : numerator(num), denominator(den) {

if (den == 0) {

throw std::invalid\_argument("Denominator cannot be zero.");

}

reduce();

}

// Метод для отображения дроби

void display() const {

if (denominator == 1) {

std::cout << numerator << std::endl;

} else {

std::cout << numerator << "/" << denominator << std::endl;

}

}

// Арифметические операторы

Rational operator+(const Rational& other) const {

T num = numerator \* other.denominator + other.numerator \* denominator;

T den = denominator \* other.denominator;

return Rational(num, den);

}

Rational operator-(const Rational& other) const {

T num = numerator \* other.denominator - other.numerator \* denominator;

T den = denominator \* other.denominator;

return Rational(num, den);

}

Rational operator\*(const Rational& other) const {

T num = numerator \* other.numerator;

T den = denominator \* other.denominator;

return Rational(num, den);

}

Rational operator/(const Rational& other) const {

if (other.numerator == 0) {

throw std::invalid\_argument("Cannot divide by zero.");

}

T num = numerator \* other.denominator;

T den = denominator \* other.numerator;

return Rational(num, den);

}

};

#endif // RATIONAL\_H

**StaticArray.h:**

#ifndef STATICARRAY\_H

#define STATICARRAY\_H

#include <iostream>

#include <stdexcept>

template <typename T, size\_t N>

class StaticArray {

private:

T data[N]; // Массив фиксированного размера

public:

// Конструктор по умолчанию

StaticArray() {

for (size\_t i = 0; i < N; i++) {

data[i] = T(); // Инициализация значением по умолчанию

}

}

// Доступ к элементу (оператор [])

T& operator[](size\_t index) {

if (index >= N) {

throw std::out\_of\_range("Index out of range");

}

return data[index];

}

// Доступ к элементу (оператор [] для константного объекта)

const T& operator[](size\_t index) const {

if (index >= N) {

throw std::out\_of\_range("Index out of range");

}

return data[index];

}

// Метод для получения размера массива

size\_t size() const {

return N;

}

// Метод для вывода содержимого массива

void display() const {

for (size\_t i = 0; i < N; i++) {

std::cout << data[i] << " ";

}

std::cout << std::endl;

}

};

#endif // STATICARRAY\_H

**Компиляция и запуск:**

**makefile:**

# Компилятор и его флаги

CXX = g++

CXXFLAGS = -std=c++17 -Wall -Wextra -g

# Название выходного файла

TARGET = main

# Список исходных файлов

SRCS = main.cpp StaticArray.h Rational.h

# Правило сборки

$(TARGET): main.cpp

$(CXX) $(CXXFLAGS) -o $(TARGET) main.cpp

# Очистка скомпилированных файлов

clean:

rm -f $(TARGET)

**Вывод:**

Обе задачи лабораторной работы выполнены. Шаблонные классы демонстрируют гибкость и позволяют работать с различными типами данных. Исключения реализованы как часть ключевой функциональности, обеспечивая безопасное использование контейнера и рациональных чисел.