МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения

вычислительной техники и автоматизированных

систем

Лабораторная работа №8

по дисциплине: ООП

тема: «**Создание шаблонов классов в C++**»

Выполнил: студент группы
ПВ-233
Мороз Роман Алексеевич

Проверили:
Морозов Данила Александрович

Белгород 2025

Лабораторная работа №8

Создание шаблонов классов в C++

Цель работы: Получение теоретических знаний о шаблонах классов в C++. Получение практических навыков по созданию классов-шаблонов C++.

Задания к лабораторной работе

1. Изучить теоретические сведения о шаблонах классов в C++.
2. Разработать программу в соответствии с заданным вариантом задания.
3. Оформить отчет.

Задание 1

Реализовать шаблон класса в соответствии с указанным вариантом. Предусмотреть необходимые методы для работы со структурой данных, указанной в варианте. Предусмотреть исключительные ситуации, которые могут возникнуть в процессе работы.

1. очередь

```
template <typename T>
class Queue {
private:
    struct Node {
        T data;
        Node* next;
        Node(const T& data) : data(data), next(nullptr) {}
    };
    Node* frontPtr;
    Node* rearPtr;
    size_t count;

public:
    Queue() : frontPtr(nullptr), rearPtr(nullptr), count(0) {}

    ~Queue() {
        while (!isEmpty()) {
            dequeue();
```

```
    }
}

void enqueue(const T& value) {
    Node* newNode = new Node(value);
    if (isEmpty()) {
        frontPtr = rearPtr = newNode;
    } else {
        rearPtr->next = newNode;
        rearPtr = newNode;
    }

    count++;
}

T dequeue() {
    if (isEmpty()) {
        throw std::runtime_error("Queue is empty");
    }

    Node* temp = frontPtr;
    T data = temp->data;

    frontPtr = frontPtr->next;
    if (frontPtr == nullptr) {
        rearPtr = nullptr;
    }

    delete temp;
    count--;

    return data;
}

inline bool isEmpty() const {
    return frontPtr == nullptr;
}

inline size_t size() const {
    return count;
}

class Iterator {
```

```
    private:
        Node* current;
    public:
        Iterator(Node* node) : current(node) {}

        Iterator& operator++() {
            if (current) {
                current = current->next;
            }
            return *this;
        }

        bool operator!=(const Iterator& other) const {
            return current != other.current;
        }

        T& operator*() const {
            return current->data;
        }
    };

    Iterator begin() {
        return Iterator(frontPtr);
    }

    Iterator end() {
        return Iterator(nullptr);
    }
};
```

Задание 2.

На основе разработанного шаблона решить прикладную задачу в соответствии с выбранным вариантом?

3. Дан файл структур с тремя полями: фамилия, возраст, оценка. Реализовать структуру для упорядоченного поиска элемента из файла по возрасту, оценки, фамилии. В качестве метода сравнения двух элементов обязательно использовать template функцию.

```cpp
#include "queue.hpp"

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>


struct Student {
    std::string surname;
    int age;
    int score;

    Student() : age(0), score(0) {}
    Student(const std::string& s, int a, int sc) : surname(s), age(a),
score(sc) {}
};

Queue<Student> readStudentsFromFile(const std::string& filename) {
    Queue<Student> queue;
    std::ifstream file(filename);

    if (!file.is_open()) {
        throw std::runtime_error("Cannot open file: " + filename);
    }

    std::string line;

    while (std::getline(file, line)) {
        std::istringstream iss(line);
        std::string surname;
        int age, score;
        if (iss >> surname >> age >> score) {
            queue.enqueue(Student(surname, age, score));
        }
    }

    return queue;
}

template<typename T, typename V>
bool compareField(const T& obj, V value, V T::* member) {
    return obj.*member == value;
```

```
}

template<typename T, typename V>
std::vector<T> searchInQueue(Queue<T>& queue, V value, V T::* member) {
    std::vector<T> results;
    for (auto it = queue.begin(); it != queue.end(); ++it) {
        if (compareField(*it, value, member)) {
            results.push_back(*it);
        }
    }

    return results;
}


int main() {
    try {
        Queue<Student> students = readStudentsFromFile("students.txt");

        int targetAge = 20;
        auto byAge = searchInQueue(students, targetAge, &Student::age);
        std::cout << "Students with age " << targetAge << ":\n";
        for (const auto& s : byAge) {
            std::cout << s.surname << " " << s.age << " " << s.score <<
std::endl;
        }

        int targetScore = 5;
        auto byScore = searchInQueue(students, targetScore,
&Student::score);
        std::cout << "Students with score " << targetScore << ":\n";
        for (const auto& s : byScore) {
            std::cout << s.surname << " " << s.age << " " << s.score <<
std::endl;
        }

        std::string targetSurname = "Ivanov";
        auto bySurname = searchInQueue(students, targetSurname,
&Student::surname);
        std::cout << "Students with surname " << targetSurname << ":\n";
        for (const auto& s : bySurname) {
            std::cout << s.surname << " " << s.age << " " << s.score <<
std::endl;
```

```
        }

    } catch (const std::exception& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}
```

```
❯ ./a.out
Students with age 20:
Ivan 20 0
Students with score 5:
Ivanov 22 5
Students with surname Ivanov:
Ivanov 22 5
```

Вывод: Получили теоретические знания о шаблонах классов в C++. Получили практические навыки по созданию классов-шаблонов C++.