Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра информационных технологий и автоматизированных систем

Лабораторная работа №10

" Сохранение данных в файле с использованием потоков "

Вариант: 12

Выполнил студент ИВТ-24-2б:
Шишкин Максим Григорьевич

_____
(дата, подпись)

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

_____
(дата, подпись)

Пермь 2025

Содержание

1 Постановка задачи

1. Создать пользовательский класс с минимальной функциональностью.
2. Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке (файле).

3. Написать функцию для чтения и просмотра объектов из потока.

4. Написать функцию для удаления объектов из потока в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

5. Написать функцию для добавления объектов в поток в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

6. Написать функцию для изменения объектов в потоке в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

7. Для вызова функций в основной программе предусмотреть меню.

Вариант: Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать:

- операции сравнения (<, >).
- операция ++, которая работает следующим образом: если форма операции префиксная, то увеличивается первое число, если форма операции постфиксная, то увеличивается второе число.

Задание:

- Удалить все записи большие заданного значения.
- Увеличить все записи с заданным значением на число L.
- Добавить K записей после записи с номером N.

2 Код на C++

main.cpp | pair.h | pair.cpp | functions.h | functions.cpp

10лабапеределка

```cpp
#include <iostream>
#include <vector>
#include "pair.h"
#include "functions.h"

using namespace std;

int main() {
    setlocale(LC_ALL, "Russian");

    vector<Pair> pairs;
    string filename = "pairs.txt";
    bool running = true;

    while (running) {
        cout << "\nМеню:\n"
             << "1. Добавить пары и сохранить в файл\n"
             << "2. Просмотреть пары из файла\n"
             << "3. Удалить записи больше заданного\n"
             << "4. Увеличить записи равные заданному\n"
             << "5. Добавить К записей после N-й\n"
             << "0. Выход\n"
             << "Выбор: ";
        int choice;
        cin >> choice;

        switch (choice) {
        case 1: {
            int n;
            cout << "Сколько пар добавить: ";
            cin >> n;
            pairs.clear();
            for (int i = 0; i < n; ++i) {
                Pair p;
                cin >> p;
                pairs.push_back(p);
            }
            savePairsToFile(filename, pairs);
            break;
        }
        case 2: {
            pairs = loadPairsFromFile(filename);
            printPairs(pairs);
            break;
        }
        case 3: {
            Pair val;
            cout << "Введите значение для сравнения: ";
            cin >> val;
            pairs = loadPairsFromFile(filename);
            deleteGreater(pairs, val);
            savePairsToFile(filename, pairs);
            break;
        }
        case 4: {
            Pair val;
            int L;
            cout << "Введите значение для сравнения: ";
            cin >> val;
            cout << "Введите число L: ";
            cin >> L;
            pairs = loadPairsFromFile(filename);
            increaseEqual(pairs, val, L);
            savePairsToFile(filename, pairs);
            break;
        }
        case 5: {
            int N, K;
            cout << "Введите номер элемента после которого вставить: ";
            cin >> N;
            cout << "Сколько пар вставить: ";
            cin >> K;
            pairs = loadPairsFromFile(filename);
            insertAfter(pairs, N, K);
            savePairsToFile(filename, pairs);
            break;
        }
        case 0:
            running = false;
            break;
        default:
            cout << "Неверный выбор!\n";
        }
    }

    return 0;
}
```

main.cpp | pair.h | pair.cpp | functions.h | functions.cpp

10лабапеределка — (Глобальная область)

```cpp
#pragma once
#include <iostream>

class Pair {
private:
    int first;
    double second;

public:
    Pair();
    Pair(int f, double s);

    int getFirst() const;
    double getSecond() const;

    void setFirst(int f);
    void setSecond(double s);
    void increase(int val);

    bool operator>(const Pair& other) const;
    bool operator<(const Pair& other) const;
    bool operator==(const Pair& other) const;

    friend std::istream& operator>>(std::istream& in, Pair& p);
    friend std::ostream& operator<<(std::ostream& out, const Pair& p);
};
```

main.cpp | pair.h | pair.cpp | functions.h | functions.cpp

10лабапеределка — (Глобальная обла

```cpp
#include "pair.h"

Pair::Pair() : first(0), second(0.0) {}
Pair::Pair(int f, double s) : first(f), second(s) {}

int Pair::getFirst() const { return first; }
double Pair::getSecond() const { return second; }

void Pair::setFirst(int f) { first = f; }
void Pair::setSecond(double s) { second = s; }

void Pair::increase(int val) {
    first += val;
    second += val;
}

bool Pair::operator>(const Pair& other) const {
    if (first > other.first) return true;
    if (first == other.first) return second > other.second;
    return false;
}

bool Pair::operator<(const Pair& other) const {
    if (first < other.first) return true;
    if (first == other.first) return second < other.second;
    return false;
}

bool Pair::operator==(const Pair& other) const {
    return first == other.first && second == other.second;
}

std::istream& operator>>(std::istream& in, Pair& p) {
    std::cout << "Введите пару (int double): ";
    in >> p.first >> p.second;
    return in;
}

std::ostream& operator<<(std::ostream& out, const Pair& p) {
    out << "(" << p.first << ", " << p.second << ")";
    return out;
}
```

main.cpp pair.h pair.cpp functions.h functions.cpp

10лабапеределка (Глобальная область)

```cpp
#pragma once
#include <vector>
#include <string>
#include "pair.h"

void savePairsToFile(const std::string& filename, const std::vector<Pair>& vec);
std::vector<Pair> loadPairsFromFile(const std::string& filename);
void printPairs(const std::vector<Pair>& vec);
void deleteGreater(std::vector<Pair>& vec, const Pair& value);
void increaseEqual(std::vector<Pair>& vec, const Pair& target, int L);
void insertAfter(std::vector<Pair>& vec, int N, int K);
```

main.cpp pair.h pair.cpp functions.h functions.cpp

10лабапеределка (Глобальная область)

```cpp
#include "functions.h"
#include <fstream>
#include <iostream>
#include <algorithm>

void savePairsToFile(const std::string& filename, const std::vector<Pair>& vec) {
    std::ofstream out(filename);
    for (const auto& p : vec)
        out << p.getFirst() << " " << p.getSecond() << "\n";
}

std::vector<Pair> loadPairsFromFile(const std::string& filename) {
    std::ifstream in(filename);
    std::vector<Pair> vec;
    int f;
    double s;
    while (in >> f >> s)
        vec.emplace_back(f, s);
    return vec;
}

void printPairs(const std::vector<Pair>& vec) {
    for (size_t i = 0; i < vec.size(); ++i)
        std::cout << i + 1 << ") " << vec[i] << "\n";
}

void deleteGreater(std::vector<Pair>& vec, const Pair& value) {
    vec.erase(std::remove_if(vec.begin(), vec.end(),
        [&](const Pair& p) { return p > value; }), vec.end());
}

void increaseEqual(std::vector<Pair>& vec, const Pair& target, int L) {
    for (auto& p : vec)
        if (p == target) p.increase(L);
}

void insertAfter(std::vector<Pair>& vec, int N, int K) {
    if (N < 1 || N > vec.size()) {
        std::cout << "Неверный номер элемента.\n";
        return;
    }

    std::vector<Pair> newPairs;
    for (int i = 0; i < K; ++i) {
        Pair p;
        std::cin >> p;
        newPairs.push_back(p);
    }
    vec.insert(vec.begin() + N, newPairs.begin(), newPairs.end());
}
```

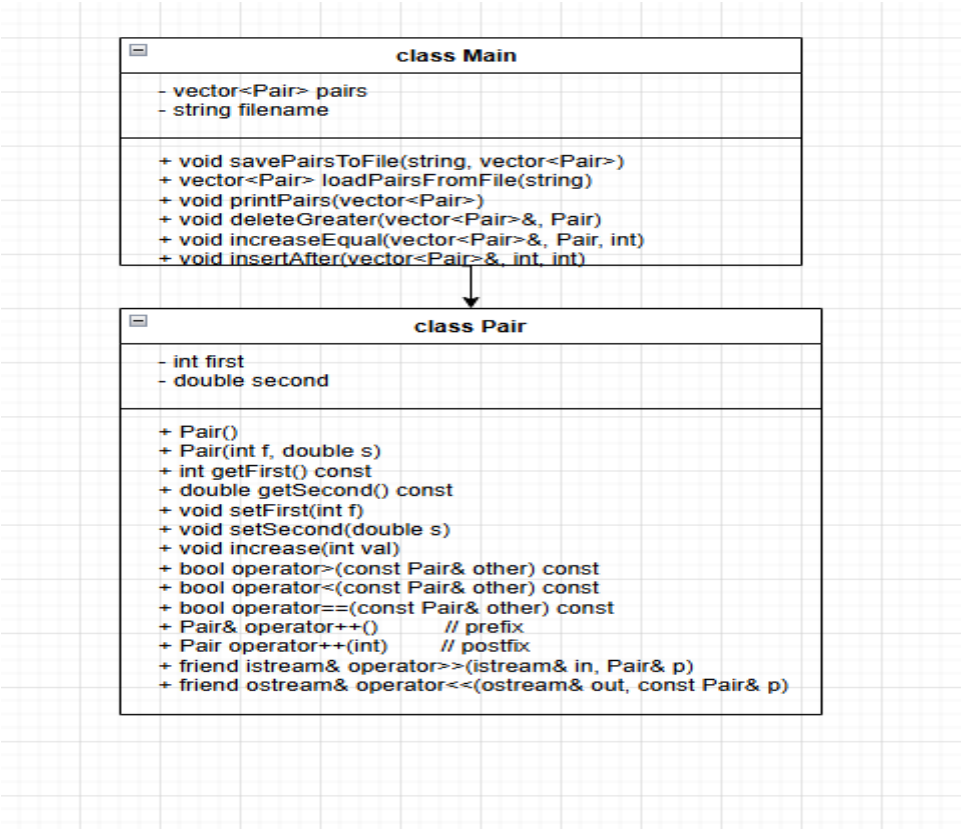
3 Результаты работы

```
Меню:
1. Добавить пары и сохранить в файл
2. Просмотреть пары из файла
3. Удалить записи больше заданного
4. Увеличить записи равные заданному
5. Добавить K записей после N-й
0. Выход
Выбор: 1
Сколько пар добавить: 1
Введите пару (int double): 2 2.2

Меню:
1. Добавить пары и сохранить в файл
2. Просмотреть пары из файла
3. Удалить записи больше заданного
4. Увеличить записи равные заданному
5. Добавить K записей после N-й
0. Выход
Выбор: 2
1) (2, 2.2)

Меню:
1. Добавить пары и сохранить в файл
2. Просмотреть пары из файла
3. Удалить записи больше заданного
4. Увеличить записи равные заданному
5. Добавить K записей после N-й
0. Выход
Выбор: 0
```

4 UML-диаграмма классов



5 Ответы на контрольные вопросы

1. Поток — это абстракция, представляющая собой источник или получатель данных. В C++ потоки используются для ввода и вывода информации.
2. Существуют следующие типы потоков:
- Стандартные потоки (ввод/вывод с консоли)
- Файловые потоки
- Строковые потоки
3. Для использования стандартных потоков подключается библиотека: #include
4. Для работы с файловыми потоками подключается библиотека: #include
5. Для работы со строковыми потоками подключается библиотека: #include
6. Для вывода в поток используется операция: << (оператор вывода)
7. Для ввода из потока используется операция:
- (оператор ввода)
8. Методы вывода в форматированный поток:
- std::cout << переменная;
- std::setw, std::setprecision, std::fixed (из <iomanip>)
9. Методы ввода из форматированного потока:
- std::cin >> переменная;
- std::getline, std::stoi, std::stod (если читается строка и преобразуется вручную)
10. Режимы открытия файловых потоков:
- ios::in — для чтения
- ios::out — для записи
- ios::app — добавление в конец
- ios::trunc — удаление содержимого
- ios::binary — бинарный режим
- ios::ate — установить курсор в конец

11. Для добавления записей используется режим: ios::app
12. Конструктор ifstream file("f.txt"); использует режим: ios::in
13. Конструктор fstream file("f.txt"); использует режим по умолчанию: ios::in | ios::out
14. Конструктор ofstream file("f.txt"); использует режим: ios::out | ios::trunc
15. Поток открывается так: ofstream file("f.txt", ios::out | ios::app);
16. Поток открывается так: ofstream file("f.txt", ios::out | ios::trunc);
17. Поток открывается так: fstream file("f.txt", ios::out | ios::in | ios::trunc);
18. Для чтения: ifstream file("f.txt"); или fstream file("f.txt", ios::in);
19. Для записи: ofstream file("f.txt"); или fstream file("f.txt", ios::out);

20

- ifstream in("file.txt"); // только чтение
- ofstream out("file.txt", ios::app); // добавление
- fstream inout("file.txt", ios::in | ios::out); // чтение и запись

21

- Pair p;
- ifstream in("file.txt");
- in >> p;

22

- Pair p(1, 2.5);
- ofstream out("file.txt");
- out << p;

23 Алгоритм удаления записей:

- Открыть файл для чтения.
- Считать все записи в контейнер.
- Удалить все записи, удовлетворяющие условию.
- Открыть файл в режиме `ios::trunc` и записать обновлённые данные.

24 Алгоритм добавления записей:

- Считать данные из файла в контейнер.
- Добавить новые записи в нужное место.
- Перезаписать файл с новыми данными.

25 Алгоритм изменения записей:

- Прочитать файл в контейнер.
- Найти и изменить нужные записи.
- Сохранить контейнер обратно в файл.

6 Ссылка на github

ссылка на github - https://github.com/MAKSPOWERO/mas1