Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №10

Тема: «Сохранение данных в файле с использованием потоков»

Выполнил

Студент группы РИС-22-2б

Максимов А.А.

Проверил доц. Кафедры ИТАС

Полякова О.А.

Пермь 2023

# Постановка задачи

# Создать пользовательский класс с минимальной функциональностью.

# Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке (файле).

# Написать функцию для чтения и просмотра объектов из потока.

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

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

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

# Для вызова функций в основной программе предусмотреть меню.

ВАРИАНТ 15:

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: вычитание пар чисел добавление константы к паре (увеличивается первое число, если константа целая, второе, если константа вещественная).

Задание:

Удалить все записи меньшие заданного значения.

Увеличить все записи с заданным значением на число L.

Добавить K записей после элемента с заданным номером.

**Контрольные вопросы**

*1. Что такое поток?*

Поток – это средство передачи данных между программой и устройством ввода-вывода.

*2. Какие типы потоков существуют?*

Существует 3 типа потоков: входящий (std::cin), исходящий (std::cout), и ошибок (std::cerr).

*3. Какую библиотеку надо подключить при использовании стандартных потоков?*

Для использования стандартных потоков надо подключить библиотеку <iostream>.

*4. Какую библиотеку надо подключить при использовании файловых потоков?*

Для использования файловых потоков надо подключить библиотеку <fstream>.

*5. Какую библиотеку надо подключить при использовании строковых потоков?*

Для использования строковых потоков надо подключить библиотеку <sstream>.

*6. Какая операция используется при выводе в форматированный поток?*

Для вывода в форматированный поток используется оператор <<, который перегружен для каждого типа данных в библиотеке <iostream>.

*7. Какая операция используется при вводе из форматированных потоков?*

Для ввода из форматированных потоков используется оператор >>, который перегружен для каждого типа данных в библиотеке <iostream>.

*8. Какие методы используются при выводе в форматированный поток?*

Для вывода в форматированный поток используются методы форматирования, такие как std::setw() и std::setprecision()

*9. Какие методы используется при вводе из форматированного потока?*

Для ввода из форматированного потока используются методы извлечения, такие как std::getline() и std::ignore().

*10. Какие режимы для открытия файловых потоков существуют?*

В языке С++ для открытия файлов используются следующие режимы доступа:

- std::ios::in – файл открывается для чтения;

- std::ios::out – файл открывается для записи;

- std::ios::app – запись добавляется в конец файла;

- std::ios::binary – файл открывается в двоичном режиме;

- std::ios::ate – указатель файла устанавливается в конец при открытии файла.

*11. Какой режим используется для добавления записей в файл?*

Для добавления записей в файл используется режим ios::app.

*12. Какой режим (комбинация режимов) используется в конструкторе ifstream file(“f.txt”)?*

Комбинация режимов ios::in и ios::binary используется в конструкторе ifstream file("f.txt").

*13. Какой режим (комбинация режимов) используется в конструкторе fstream file(“f.txt”)?*

Комбинация режимов ios::in | ios::out и ios::binary используется в конструкторе fstream file("f.txt").

*14. Какой режим (комбинация режимов) используется в конструкторе ofstream file(“f.txt”)?*

Режим ios::out используется в конструкторе ofstream file("f.txt").

*15. Каким образом открывается поток в режиме ios::out|ios::app?*

Для открытия потока в режиме ios::out|ios::app, нужно использовать следующую команду: ofstream file("filename.txt", ios::out|ios::app);

*16. Каким образом открывается поток в режиме ios::out |ios::trunc?*

Для открытия потока в режиме ios::out|ios::trunc, нужно использовать следующую команду: ofstream file("filename.txt", ios::out|ios::trunc);

*17. Каким образом открывается поток в режиме ios::out |ios::in|ios::trunk?*

Для открытия потока в режиме ios::out|ios::in|ios::trunk, нужно использовать следующую команду: fstream file("filename.txt", ios::out|ios::in|ios::trunc);

*18. Каким образом можно открыть файл для чтения?*

Чтобы открыть файл для чтения, нужно использовать объект ifstream и вызвать его конструктор с параметром, который представляет имя файла: ifstream file("filename.txt");

*19. Каким образом можно открыть файл для записи?*

Чтобы открыть файл для записи, нужно использовать объект ofstream и вызвать его конструктор с параметром, который представляет имя файла: ofstream file("filename.txt");

*20. Привести примеры открытия файловых потоков в различных режимах.*

Примеры открытия файловых потоков в различных режимах:

• ifstream file("filename.txt"); // открыть файл для чтения

• ofstream file("filename.txt"); // открыть файл для записи

• ofstream file("filename.txt", ios::app); // открыть файл для добавления записей в конец

• fstream file("filename.txt", ios::in | ios::out | ios::binary); // открыть файл для чтения и записи в двоичном режиме

• fstream file("filename.txt", ios::out | ios::trunc); // открыть файл для записи, если файл существует, то удалить все его содержимое

• fstream file("filename.txt", ios::out | ios::in | ios::trunc); // открыть файл для чтения и записи, если файл существует, то удалить все его содержимое

*21. Привести примеры чтения объектов из потока.*

- Чтение строки из потока: std::string str; std::getline(std::cin, str);

- Чтение числа типа int из потока: int n; std::cin >> n;

- Чтение числа типа double из потока: double x; std::cin >> x;

- Чтение объекта типа Person из потока: Person p; std::cin >> p;

*22. Привести примеры записи объектов в поток.*

- Запись строки в поток: std::string str = "Hello, world!"; std::cout << str;

- Запись числа типа int в поток: int n = 42; std::cout << n;

- Запись числа типа double в поток: double x = 3.14; std::cout << x;

- Запись объекта типа Person в поток: Person p("John", "Doe"); std::cout << p;

*23. Сформулировать алгоритм удаления записей из файла.*

1. Открыть файл в режиме чтения и записи.

2. Создать временный файл для сохранения оставшихся записей.

3. Прочитать каждую запись из исходного файла.

4. Если запись должна быть удалена, пропустить ее.

5. Иначе записать эту запись во временный файл.

6. Закрыть оба файла.

7. Удалить исходный файл.

8. Переименовать временный файл в исходный файл.

*24. Сформулировать алгоритм добавления записей в файл.*

1. Открыть файл в режиме добавления.

2. Записать новую запись в конец файла.

3. Закрыть файл.

*25. Сформулировать алгоритм изменения записей в файле.*

1. Открыть файл в режиме чтения и записи.

2. Создать временный файл для сохранения измененных записей.

3. Прочитать каждую запись из исходного файла.

4. Если запись должна быть изменена, изменить ее.

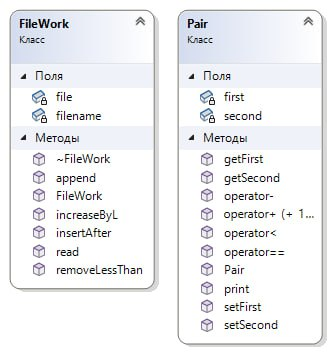
5. Иначе записать эту запись во временный файл.

6. Закрыть оба файла.

7. Удалить исходный файл.

8. Переименовать временный файл в исходный файл.

UML – диаграмма



Код программы

**file\_work.h**

#ifndef FILE\_WORK\_H

#define FILE\_WORK\_H

#include <fstream>

#include "pair.h"

class FileWork {

public:

FileWork(const char\* filename);

~FileWork();

void append(const Pair& pair);

void read();

void removeLessThan(const Pair& pair);

void increaseByL(const Pair& pair, const Pair& L);

void insertAfter(const Pair& pair, const Pair\* const values, const int& size);

private:

const char\* filename;

std::fstream file;

};

#endif // FILE\_WORK\_H

**file\_work.cpp**

#include "file\_work.h"

FileWork::FileWork(const char\* filename)

: filename(filename) {

file.open(filename, std::ios::in | std::ios::out | std::ios::binary);

}

FileWork::~FileWork() {

if (file.is\_open()) {

file.close();

}

}

void FileWork::append(const Pair& pair) {

file.seekp(0, std::ios::end);

file.write(reinterpret\_cast<const char\*>(&pair), sizeof(pair));

}

void FileWork::read() {

file.seekg(0, std::ios::beg);

Pair pair;

while (file.read(reinterpret\_cast<char\*>(&pair), sizeof(pair))) {

pair.print();

}

}

void FileWork::removeLessThan(const Pair& pair) {

std::fstream temp(filename, std::ios::in | std::ios::out | std::ios::binary | std::ios::trunc);

file.seekg(0, std::ios::beg);

Pair current\_pair;

while (file.read(reinterpret\_cast<char\*>(&current\_pair), sizeof(current\_pair))) {

if (!(current\_pair < pair)) {

temp.write(reinterpret\_cast<const char\*>(&current\_pair), sizeof(current\_pair));

}

}

file.close();

temp.close();

remove(filename);

rename("temp.bin", filename);

file.open(filename, std::ios::in | std::ios::out | std::ios::binary);

}

void FileWork::increaseByL(const Pair& pair, const Pair& L) {

std::fstream temp(filename, std::ios::in | std::ios::out | std::ios::binary | std::ios::trunc);

file.seekg(0, std::ios::beg);

Pair current\_pair;

while (file.read(reinterpret\_cast<char\*>(&current\_pair), sizeof(current\_pair))) {

if (current\_pair == pair) {

current\_pair.setFirst(current\_pair.getFirst() + L.getFirst());

current\_pair.setSecond(current\_pair.getSecond() + L.getSecond());

}

temp.write(reinterpret\_cast<const char\*>(&current\_pair), sizeof(current\_pair));

}

file.close();

temp.close();

remove(filename);

rename("temp.bin", filename);

file.open(filename, std::ios::in | std::ios::out | std::ios::binary);

}

void FileWork::insertAfter(const Pair& pair, const Pair\* const values, const int& size) {

std::fstream temp(filename, std::ios::in | std::ios::out | std::ios::binary | std::ios::trunc);

file.seekg(0, std::ios::beg);

Pair current\_pair;

while (file.read(reinterpret\_cast<char\*>(&current\_pair), sizeof(current\_pair))) {

temp.write(reinterpret\_cast<const char\*>(&current\_pair), sizeof(current\_pair));

if (current\_pair == pair) {

for (int i = 0; i < size; i++) {

temp.write(reinterpret\_cast<const char\*>(&values[i]), sizeof(values[i]));

}

}

}

file.close();

temp.close();

remove(filename);

rename("temp.bin", filename);

file.open(filename, std::ios::in | std::ios::out | std::ios::binary);

}

**pair.h**

#ifndef PAIR\_H

#define PAIR\_H

#include <iostream>

class Pair {

public:

Pair(int first = 0, double second = 0.0);

void print() const;

Pair operator-(const Pair& other) const;

Pair operator+(const int& num) const;

Pair operator+(const double& num) const;

bool operator<(const Pair& other) const;

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

int getFirst() const;

double getSecond() const;

void setFirst(int first);

void setSecond(double second);

private:

int first;

double second;

};

#endif // PAIR\_H

**pair.cpp**

#include "pair.h"

Pair::Pair(int first, double second)

: first(first), second(second) {}

void Pair::print() const {

std::cout << first << ":" << second << std::endl;

}

Pair Pair::operator-(const Pair& other) const {

return Pair(first - other.first, second - other.second);

}

Pair Pair::operator+(const int& num) const {

return Pair(first + num, second);

}

Pair Pair::operator+(const double& num) const {

return Pair(first, second + num);

}

bool Pair::operator<(const Pair& other) const {

if (first == other.first) {

return second < other.second;

}

return first < other.first;

}

bool Pair::operator==(const Pair& other) const {

return first == other.first && second == other.second;

}

int Pair::getFirst() const {

return first;

}

double Pair::getSecond() const {

return second;

}

void Pair::setFirst(int first) {

this->first = first;

}

void Pair::setSecond(double second) {

this->second = second;

}

**main.cpp**

#include <iostream>

#include "pair.h"

#include "file\_work.h"

void print\_menu() {

std::cout << "Choose an option:" << std::endl;

std::cout << "1. Append a pair to the file" << std::endl;

std::cout << "2. Read the file" << std::endl;

std::cout << "3. Remove pairs less than given" << std::endl;

std::cout << "4. Increase a pair by a given value" << std::endl;

std::cout << "5. Insert pairs after a given one" << std::endl;

std::cout << "0. Exit" << std::endl;

}

int main() {

const char\* filename = "pairs.bin";

FileWork file(filename);

int option;

do {

print\_menu();

std::cin >> option;

switch (option) {

case 1: {

int first;

double second;

std::cout << "Enter the first and second elements of the pair:" << std::endl;

std::cin >> first >> second;

Pair pair(first, second);

file.append(pair);

std::cout << "Pair appended to the file" << std::endl;

break;

}

case 2: {

file.read();

break;

}

case 3: {

int first;

double second;

std::cout << "Enter the first and second elements of the pair:" << std::endl;

std::cin >> first >> second;

Pair pair(first, second);

file.removeLessThan(pair);

std::cout << "Pairs less than " << first << ":" << second << " removed from the file" << std::endl;

break;

}

case 4: {

int first;

double second;

std::cout << "Enter the first and second elements of the pair to be increased:" << std::endl;

std::cin >> first >> second;

Pair pair(first, second);

std::cout << "Enter the value to increase the pair by:" << std::endl;

std::cin >> first >> second;

Pair L(first, second);

file.increaseByL(pair, L);

std::cout << "Pair " << first << ":" << second << " increased by " << L.getFirst() << ":" << L.getSecond() << std::endl;

break;

}

case 5: {

int first;

double second;

std::cout << "Enter the first and second elements of the pair after which new pairs will be inserted:" << std::endl;

std::cin >> first >> second;

Pair pair(first, second);

int size;

std::cout << "Enter the number of pairs to insert:" << std::endl;

std::cin >> size;

Pair\* values = new Pair[size];

for (int i = 0; i < size; i++) {

std::cout << "Enter the first and second elements of pair " << i + 1 << ":" << std::endl;

std::cin >> first >> second;

values[i] = Pair(first, second);

}

file.insertAfter(pair, values, size);

std::cout << size << " pairs inserted after " << first << ":" << second << std::endl;

delete[] values;

break;

}

case 0: {

std::cout << "Exiting program" << std::endl;

break;

}

default: {

std::cout << "Invalid option selected" << std::endl;

break;

}

}

} while (option != 0);

return 0;

}