Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Информатика»

Тема: Сохранение данных в файле с использованием потоков

Семестр 2

Выполнил работу

Студент группы ИВТ-22-2б

Ищенко Дарья Олеговна

Проверил

Доцент кафедры ИТАС

Полякова О. A.

Г. Пермь-2023

**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.

**Описание класса**

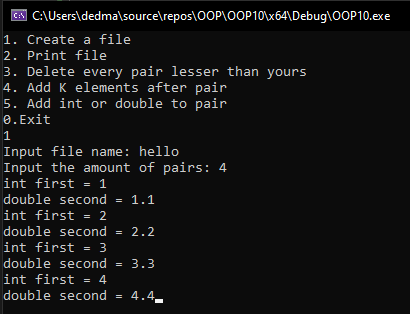
Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать:

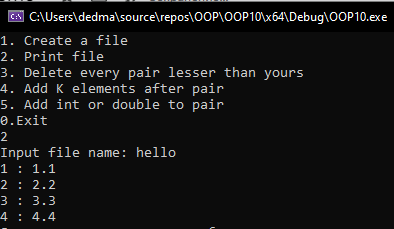
* вычитание пар чисел
* добавление константы к паре (увеличивается первое число, если константа целая, второе, если константа вещественная).

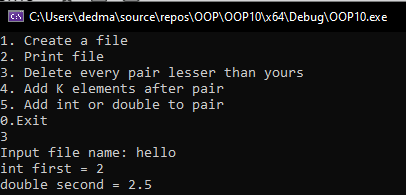
Задание:

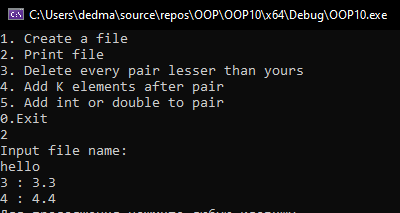
* Удалить все записи меньшие заданного значения.
* Увеличить все записи с заданным значением на число L.
* Добавить K записей после элемента с заданным номером.

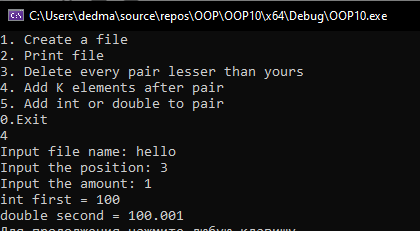
**Скриншот работы программы**

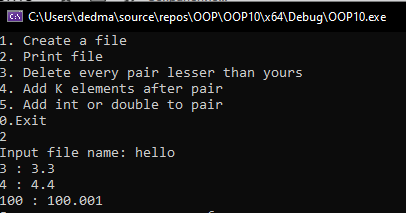
****

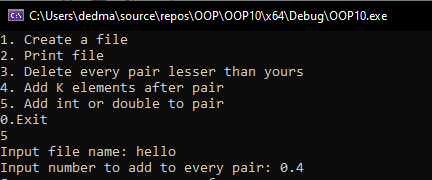
****

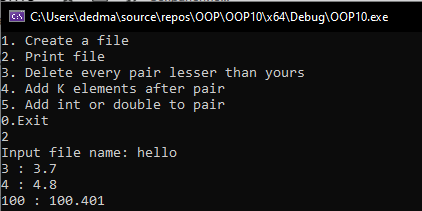
****

****

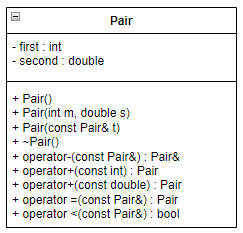
****

****

****

****

**UML-диаграмма**

****

**Контрольные вопросы**

1. Что такое поток?

Поток - определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен (оперативная память, файл на диске, клавиатура или принтер). Обмен с потоком для увеличения скорости передачи данных производится, как правило, через специальную область оперативной памяти — буфер. Буфер накапливает байты, и фактическая передача данных выполняется после заполнения буфера. При вводе это дает возможность исправить ошибки, если данные из буфера еще не отправлены в программу.

2. Какие типы потоков существуют?

- Стандартные: только однонаправленные, либо входные, либо выходные. Привязаны к стандартным устройствам: клавиатуре и экрану

- Строковые: могут быть и однонаправленными и двунаправленными

- Файловые: могут быть и однонаправленными и двунаправленными. Переменная файлового потока связывается со стандартным файлом на диске.

3. Какую библиотеку надо подключить при использовании стандартных потоков?

#include <iostream>

4. Какую библиотеку надо подключить при использовании файловых потоков?

#include <fstream>

5. Какую библиотеку надо подключить при использовании строковых потоков?

#include <sstream>

6. Какая операция используется при выводе в форматированный поток?

Для форматируемых потоков вывод, как правило, осуществляется перегруженной операцией сдвига влево operator<<.

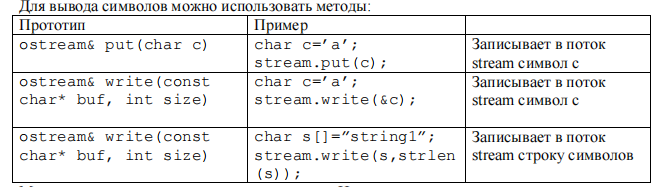
stream << 404.404 <<’\n’;

7. Какая операция используется при вводе из форматированных потоков?

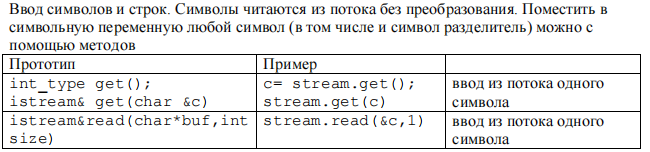
Для форматируемых потоков ввод перегруженной операцией сдвига вправо operator>>.

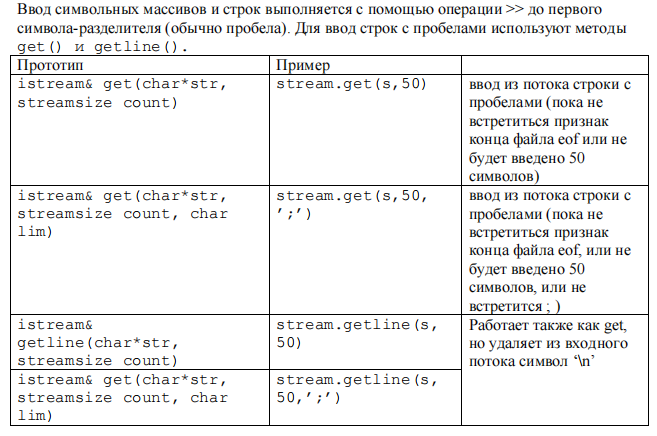
stream >> num;

8. Какие методы используются при выводе в форматированный поток?

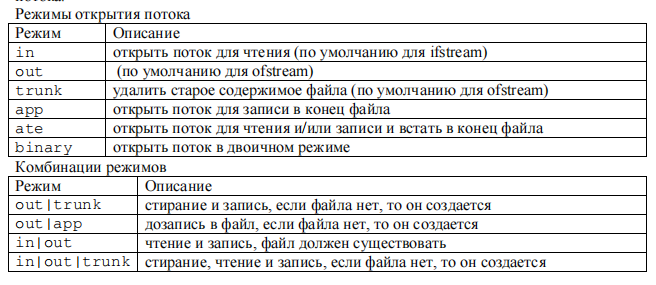


9. Какие методы используется при вводе из форматированного потока?





10. Какие режимы для открытия файловых потоков существуют?



11. Какой режим используется для добавления записей в файл?

Для добавления записи в конец файла используется режим app.

Также, используется режим ate для чтения и/или записи в конец файла.

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

Используется режим in - открыть поток для чтения (по умолчанию для ifstream).

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

Используется комбинация режимов in | out - чтение и запись, файл должен существовать.

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

Используется режим out - открыть поток для записи в файл (по умолчанию для ofstream).

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

Поток открывается для дополнительной записи в файл, если файла нет, то он создаётся.

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

Поток открывается для стирания и записи, если файла нет, то он создаётся.

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

Поток открывается для стирания, чтения и записи, если файла нет, то он создаётся

18. Каким образом можно открыть файл для чтения?

Файл может быть открыт либо явно методом open(), либо неявно – конструктором при создании потока.

Для открытия файла для чтения можно использовать различные режимы:

in - открыть поток для чтения (по умолчанию для ifstream);

ate - открыть поток для чтения и/или записи и встать в конец файла.

19. Каким образом можно открыть файл для записи?

Файл может быть открыт либо явно методом open(), либо неявно – конструктором при создании потока.

Для открытия файла для записи можно использовать различные режимы:

out - открыть поток для записи в файл (по умолчанию для ofstream);

trunk - удалить старое содержимое файла (по умолчанию для оfstream);

app - открыть поток для записи в конец файла

ate - открыть поток для чтения и/или записи и встать в конец файла

20. Привести примеры открытия файловых потоков в различных режимах.

*Пример*

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream file("MyFile.txt");

if (!file.is\_open()) cout << "Error" << endl;

else {

file << "Hello, world!" << endl;

file.close(); }

return 0;

}

*Пример*

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream file("MyFile.bin", ios::binary);

if (!file.is\_open()) cout << "Error" << endl;

else {

char buffer[256]; // чтение данных из файла в буфер

file.read(buffer, sizeof(buffer)); // обработка данных из буфера

file.close(); }

return 0;

}

21. Привести примеры чтения объектов из потока.

*Пример*

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream file("MyFile.txt");

if (!file.is\_open()) cout << “Error" << endl;

else {

string line; //чтение строк

while (getline(file, line)) cout << line << endl;

file.close(); }

return 0;

}

22. Привести примеры записи объектов в поток.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Person {

public:

Person(string name, int age) : m\_name(name), m\_age(age) {}

string getName() const { return m\_name; }

int getAge() const { return m\_age; }

private:

string m\_name;

int m\_age;

};

int main() {

ofstream file("MyFile.bin", ios::binary);

if (!file.is\_open()) cout << "Error" << endl;

else {

Person A("John", 30);

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

file.close(); }

return 0;

}

23. Сформулировать алгоритм удаления записей из файла.

1. Открыть файл для чтения и записи.

2. Проверить, что файл успешно открыт

3. Создать временный файл.

4. Прочитать записи из исходного файла и проверять каждую запись на соответствие критерию удаления.

5. Записать нужные записи во временный файл.

6. Закрыть исходный файл и временный файл.

7. Удалить исходный файл с помощью функции.

8. Переименовать временный файл в имя исходного файла.

24. Сформулировать алгоритм добавления записей в файл.

1. Открыть файл для записи с помощью функции

2. Проверить, что файл успешно открыт с помощью функции

3. Создать объект записи, который будет содержать данные для записи в файл.

4. Заполнить объект записи данными.

5. Записать объект записи в файл с помощью оператора или функции.

6. Проверить, что запись прошла успешно с помощью функции.

7. Повторять шаги 3-6 для каждой новой записи.

8. Закрыть файл с помощью функции

25. Сформулировать алгоритм изменения записей в файле.

1. Открыть файл для чтения и записи с помощью функции std::fstream.

2. Проверить, что файл успешно открыт.

3. Считать данные из файла в структуру или массив данных.

4. Изменить необходимые данные в структуре или массиве.

5. Перезаписать измененные данные в файл.

6. Закрыть файл.