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: «Основы алгоритмизации и программирования»

Тема: «Сохранение данных в файле с использованием потоков»

Выполнил работу

Студент группы РИС-23-3Б

Епин Т. Е.

Проверил

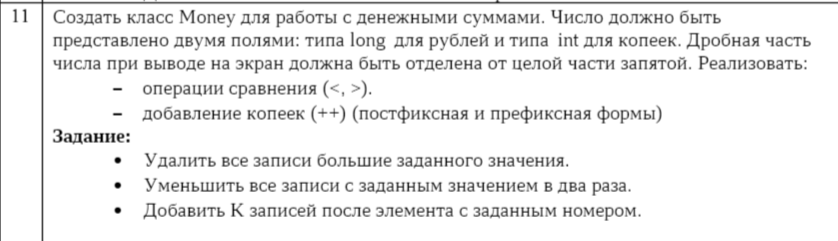
Доцент кафедры ИТАС

Полякова О.А.

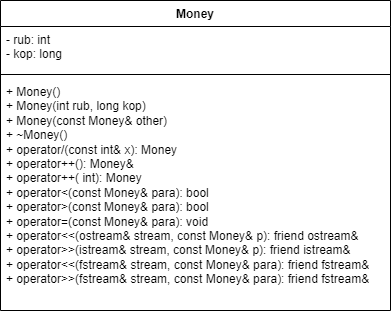
Г. Пермь-2024

**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.



**Uml диаграмма:**

****

**Код программы:**

**Money.h**

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Money {

int rub;

long kop;

public:

Money() { // конструктор по умолчанию

this->rub = 0;

this->kop = 0;

}

Money(int rub, long kop) { // конструктор с параметрами

this->rub = rub;

this->kop = kop;

}

Money(const Money& other) { // конструктор копирования

this->rub = other.rub;

this->kop = other.kop;

}

~Money() { } // деструктор

Money operator/(const int& x) {

return (Money(this->rub / x, this->kop));

}

Money& operator++() { // префиксная операция прибавления

Money x(this->rub++, this->kop);

return x;

}

Money operator ++(int) { // постфиксная операция прибавления

Money x(this->rub, this->kop++);

return x;

}

bool operator<(const Money& para) { // оператор сравнения

if (this->rub < para.rub) return true;

else if (this->rub == para.rub) {

if (this->kop < para.kop) return true;

else return false;

}

else return false;

}

bool operator>(const Money& para) { // оператор сравнения

if (this->rub < para.rub) return false;

else if (this->rub == para.rub) {

if (this->kop < para.kop) return false;

else return true;

}

else return true;

}

void operator=(const Money& para) { // оператор присваивания

this->rub = para.rub;

this->kop = para.kop;

}

friend ostream& operator <<(ostream& stream, const Money& p) {

stream << p.rub << " , " << p.kop;

return stream;

}

friend istream& operator >>(istream& stream, Money& p) {

cout << "Введите число rub и kop через пробел: ";

stream >> p.rub >> p.kop;

return stream;

}

friend fstream& operator<<(fstream& stream, Money& para) {

stream << para.rub << endl << para.kop << endl;

return stream;

}

friend fstream& operator>>(fstream& stream, Money& para) {

stream >> para.rub;

stream >> para.kop;

return stream;

}

};

**File.h**

#pragma once

#include <iostream>

#include <fstream>

#include"Money.h"

using namespace std;

void randomize(Money& p) {

int x = rand() % 100;

long y = (rand() % 10000) \* 0.01;

p = Money(x, y);

}

bool write(int N) {

fstream F1("file.txt", ios::out | ios::trunc);

if (!F1) {

cout << "Не удалось открыть файл!" << endl;

return 0;

}

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

Money p\_1;

randomize(p\_1);

F1 << p\_1 << endl;

}

F1.close();

return 1;

}

bool print\_file() {

fstream F1("file.txt", ios::in);

if (!F1) {

cout << "Не удалось открыть файл!" << endl;

return 0;

}

Money p\_1;

while (F1 >> p\_1) {

cout << p\_1 << endl;

}

return 1;

}

void error(bool N) {

if (!N) {

cout << "Файл не найден" << endl;

}

}

bool remove(Money& para) {

fstream F1("file.txt", ios::in);

if (!F1) {

cout << "Не удалось открыть файл!" << endl;

return 0;

}

fstream tmp("temp.txt", ios::out | ios::trunc);

Money p\_3;

while (F1 >> p\_3) {

if (p\_3 < para) {

tmp << p\_3 << endl;

}

}

F1.close();

tmp.close();

remove("file.txt");

rename("temp.txt", "file.txt");

return 1;

}

template<typename T>

bool decrease(T) {

fstream F1("file.txt", ios::in);

if (!F1) {

cout << "Не удалось открыть файл!" << endl;

return 0;

}

fstream tmp("temp.txt", ios::out | ios::trunc);

Money p\_4;

while (F1 >> p\_4) {

p\_4 = p\_4 /2;

tmp << p\_4 << endl;

}

F1.close();

tmp.close();

remove("file.txt");

rename("temp.txt", "file.txt");

return 1;

}

bool add\_el(int number, int quantity) {

fstream F1("file.txt", ios::in);

if (!F1) {

cout << "Не удалось открыть файл!" << endl;

return 0;

}

fstream tmp("temp.txt", ios::out | ios::trunc);

Money p\_4;

int counter = 0;

while (F1 >> p\_4) {

counter++;

tmp << p\_4 << endl;

if (counter == number) {

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

Money new\_money;

randomize(new\_money);

tmp << new\_money << endl;

}

}

}

if (counter < number)

cout << "Элемент с номером " << number << " не найден!" << endl;

F1.close();

tmp.close();

remove("file.txt");

rename("temp.txt", "file.txt");

return 1;

}

int choosing() {

cout << "Выберите действие" << endl;

cout << "1 - Вывести содержимое файла" << endl;//+

cout << "2 - Перезаписать содержимое файла" << endl;//+

cout << "3 - Удалить элементы, большие заданного значения" << endl;//+

cout << "4 - Уменьшить все записи с заданным значением в 2 раза" << endl;

cout << "5 - Добавить K записей после элемента с номером N" << endl;

cout << "0 - Завершение работы" << endl;//+

int choice;

do {

cout << "> ";

cin >> choice;

} while (choice < 0 || choice > 6);

cout << endl;

return choice;

}

**Lab10main.cpp**

#include<iostream>

#include <fstream>

#include"Money.h"

#include"file.h"

using namespace std;

int main() {

system("chcp 1251 > Null");

srand(time(0));

int n;

do {

cout << "Введите количество элементов: ";

cin >> n;

} while (n < 1);

cout << endl;

error(write(n));

bool flag = true;

while (flag) {

int current\_action = choosing();

int tmp\_1;

double tmp\_2;

Money p\_2;

switch (current\_action) {

case 1:

cout << "Текущее содержимое файла" << endl;

error(print\_file());

cout << endl;

break;

case 2:

do {

cout << "Введите количество элементов: ";

cin >> tmp\_1;

} while (tmp\_1 < 1);

error(write(tmp\_1));

break;

case 3:

cout << "Введите значение, значения больше которого будут удалены: " << endl;

cin >> p\_2;

error(remove(p\_2));

break;

case 4:

cout << "Уменьшим все записи в 2 раза: " << endl;

error(decrease(2));

cout << endl;

break;

case 5:

int N;

do {

cout << "Введите номер, после которого необходимо ввести K элементов" << endl;

cin >> N;

} while (N < 1);

do {

cout << "Введите количество элементов, которые необходимо добавить: " << endl;

cin >> tmp\_1;

} while (tmp\_1 < 1);

error(add\_el(N, tmp\_1));

break;

default:

flag = false;

break;

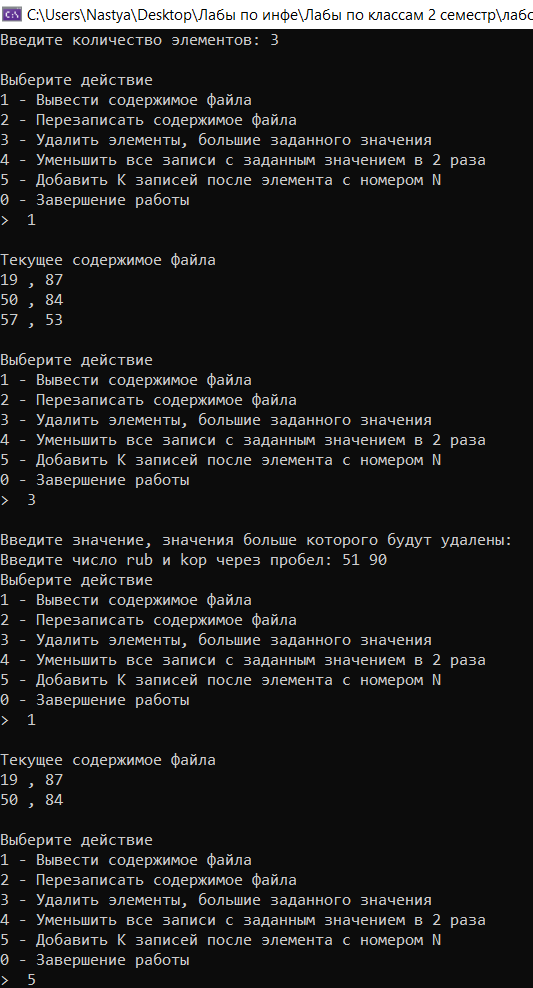
}

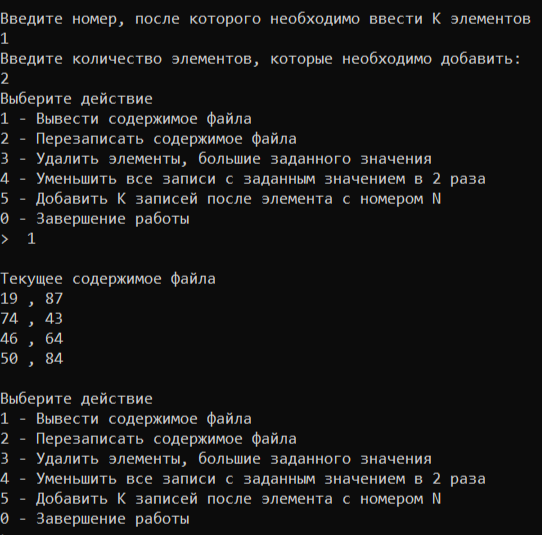
}

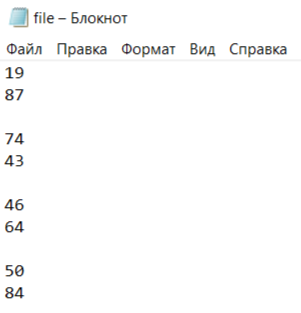
return 0;

}

**Результат работы программы:**

****

****

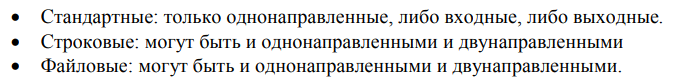
****

**Ответы на контрольные вопросы:**

**1. Что такое поток?**

Поток - это абстрактный объект, который представляет собой последовательность байтов, которые могут быть считаны из устройства ввода или записаны в устройство вывода. Потоки представляют собой абстракцию для ввода и вывода данных и используются для чтения или записи данных в файлы, сетевые соединения, память и т.д.

**2. Какие типы потоков существуют?**



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

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

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

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

<< (бинарный оператор вставки)



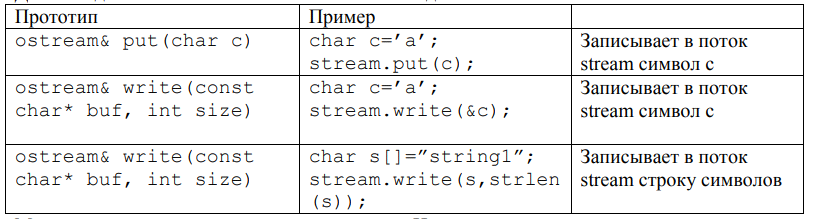
**7. Какая операция используется при вводе из форматированных потоков?**

операция извлечения из потока >>

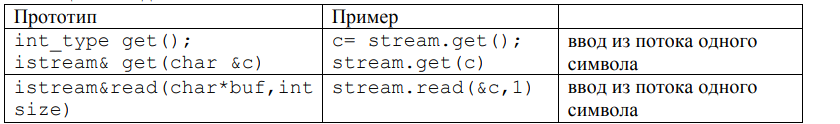


**8. Какие методы используются при выводе в форматированный поток?**

operator<< - оператор вставки, который позволяет добавить значение в поток. Он перегружен для разных типов данных и может быть использован для вывода числовых значений, строк, символов и других типов данных.



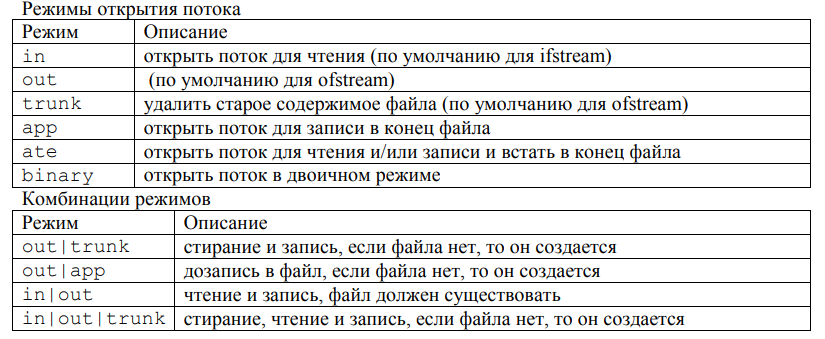
**9. Какие методы используется при вводе из форматированного потока?**



- operator>>: основной метод, используемый для чтения из потока. Он позволяет считывать данные из потока и сохранять их в переменной или объекте.

- getline: метод, используемый для чтения строки из потока. Он позволяет считать всю строку, включая пробелы и символы новой строки.

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



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

out

fstream temp("temp", ios::out);

**12. Какой режим (комбинация режимов) используется в конструкторе**

**ifstream file("f.txt")?**

ios::in

**13. Какой режим (комбинация режимов) используется в конструкторе**

**fstream file("f.txt")?**

ios::in | ios::out

**14. Какой режим (комбинация режимов) используется в конструкторе**

**ofstream file("f.txt")?**

ios::out

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

ios::out указывает на то, что файл будет открыт для записи, а ios::app гарантирует, что при каждой записи данные будут добавляться в конец файла. Если файл не существует, он будет создан. Если файл уже существует, то данные будут добавляться в конец файла, не перезаписывая его содержимое.

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

ios::out - файл будет открыт в режиме вывода и при этом, если файл уже существует, его содержимое будет удалено (ios::trunc). Если же файл не существует, он будет создан.

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

ios::out указывает на то, что файл будет использоваться для записи, ios::in указывает на то, что файл будет использоваться для чтения, а ios::trunc указывает на то, что содержимое файла будет очищено перед записью в него.

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



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



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

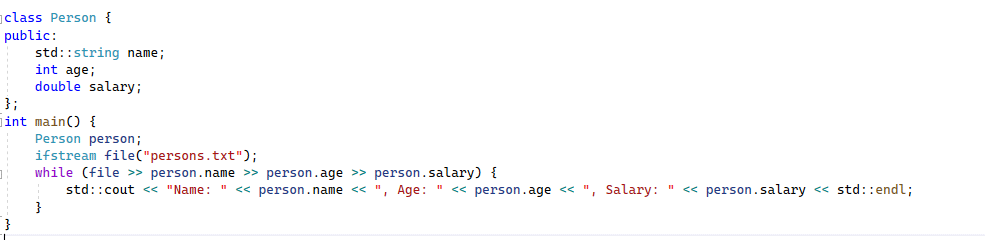
- добавляет новое содержимое в конец файла, не удаляя предыдущее содержимое



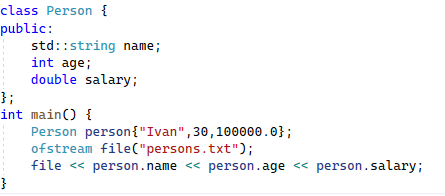
**-** открыт для чтения и записи



**21. Привести примеры чтения объектов из потока.**



**22. Привести примеры записи объектов в поток.**



**23. Сформулировать алгоритм удаления записей из файла.**

1) открыть файл в режиме чтения

2) открыть вспомогательный файл в режиме записи

3) пока не конец файла, копировать записи из файла в вспомогательный, пропуская те, которые нужно удалить

4) удалить изначальный файл

5) переименовать вспомогательный файл в изначальный

**24. Сформулировать алгоритм добавления записей в файл.**

1) открыть файл в режиме чтения

2) открыть вспомогательный файл в режиме записи

3) пока не конец файла, копировать записи из файла в вспомогательный, если встретится запись, после которой нужно добавить новую, остановиться, начать добавлять новые, затем продолжить копирование старых записей из изначального файла

4) удалить изначальный файл

5) переименовать вспомогательный файл в изначальный

**25. Сформулировать алгоритм изменения записей в файле.**

1) открыть файл в режиме чтения

2) открыть вспомогательный файл в режиме записи

3) пока не конец файла, копировать записи из файла в вспомогательный, пропуская те, которые нужно удалить, если встретится запись, после которой нужно добавить новую, остановиться, начать добавлять новые, затем продолжить копирование старых записей из изначального файла

4) удалить изначальный файл

5) переименовать вспомогательный файл в изначальный