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 ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Дисциплина: «Основы алгоритмизации и программирования»

Лабораторная работа №10

Тема : "Сохранение данных в файле с использованием потоков"

Выполнил работу

Студент группы РИС-22-1Б

Лихачев Д.А.

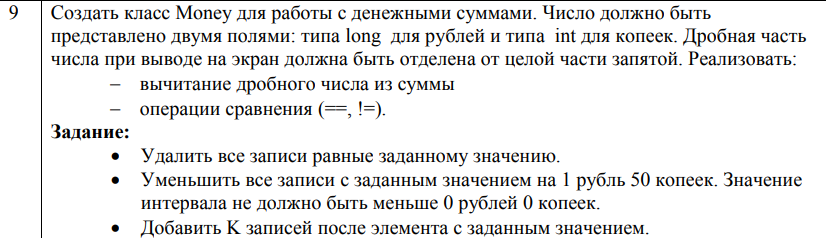
Проверила

Доцент кафедры ИТАС

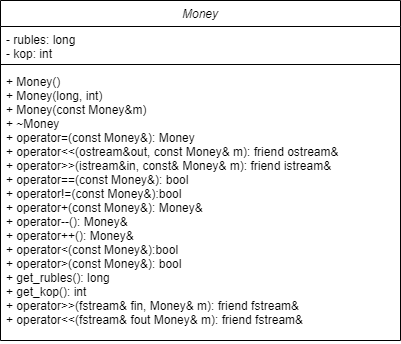
Полякова О.А.

Г. Пермь-2023

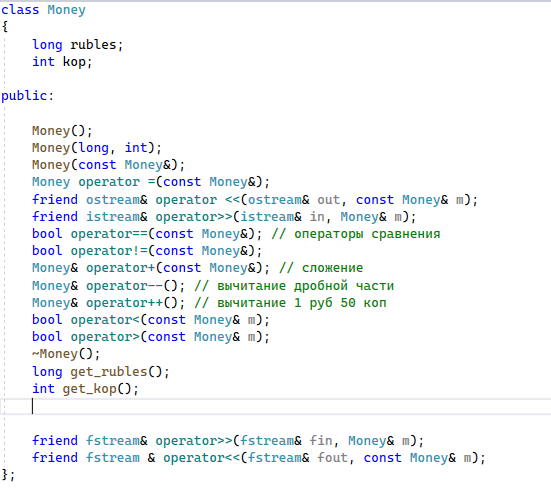
**Постановка задачи**



**Диаграмма классов**



**Описание пользовательского класса**



**Перегруженные операции пользовательского класса**

Money Money::operator=(const Money& m)

{

if (&m == this) return \*this;

rubles = m.rubles;

kop = m.kop;

return \*this;

}

Money::~Money() {}

ostream& operator<<(ostream& out, const Money& m)

{

out << m.rubles << "," << m.kop;

return out;

}

istream& operator>>(istream& in, Money& m)

{

cout << "rubles? "; in >> m.rubles;

cout << "kop? "; in >> m.kop;

return in;

}

fstream& operator>>(fstream& fin, Money& m)

{

fin >> m.rubles;

fin >> m.kop;

return fin;

}

fstream& operator<<(fstream& fout, const Money& m)

{

fout << m.rubles << "\n" << m.kop << "\n";

return fout;

}

bool Money::operator==(const Money& m)

{

if (rubles == m.rubles && kop == m.kop) return true;

return false;

}

bool Money::operator!=(const Money& m)

{

if (rubles != m.rubles || kop != m.kop) return true;

return false;

}

Money& Money::operator+(const Money& m)

{

rubles += m.rubles;

rubles += (kop + m.kop) / 100;

kop = (kop + m.kop) % 100;

return \*this;

}

Money& Money::operator--()

{

kop = 0;

return \*this;

}

bool Money::operator<(const Money& m)

{

if (rubles < m.rubles) return true;

else if (kop < m.kop) return true;

return false;

}

bool Money::operator>(const Money& m)

{

if (rubles > m.rubles) return true;

else if (kop > m.kop) return true;

return false;

}

Money& Money::operator++()

{

rubles -= 1;

if (kop / 50 == 0)

{

rubles--;

kop = kop + 50;

}

else kop = kop - 50;

return \*this;

}

**Определение функций для работы с файлом**

int make\_file(const char\* f\_name) // создать файл

{

fstream stream(f\_name, ios::out | ios::trunc); // открыть для записи

if (!stream) return -1; // ошибка открытия файла

int n;

Money m;

cout << "N?"; cin >> n;

for (int i = 0; i < n; i++)

{

cin >> m; // ввод атрибутов объекта из стандартного потока

stream << m << "\n"; // запись объекта в файловый поток

}

stream.close(); // закрыть поток

return n; // вернуть количество записанных объектов

}

int print\_file(const char\* f\_name) // вывод файла

{

fstream stream(f\_name, ios::in);

if (!stream) return -1;

Money m; int i = 0; string n;

while (stream>>m)

{

cout << m << " ";

++i;

}

stream.close();

return i;

}

int del\_file(const char\* f\_name, Money t) // удаление записей, равных введенному значению

{

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

fstream stream(f\_name, ios::in);

if (!stream)return -1;

int i = 0;

long c = t.get\_rubles();

int d = t.get\_kop();

Money m;

while (stream >> m) // пока нет конца файла выполняем чтение файла

{

// если прочитан признак конца файла, то выход из цикла

if (stream.eof()) break;

i++;

// если номер объекта не равен к, то записываем его в вспомогательный файл

long a = m.get\_rubles();

int b = m.get\_kop();

if ((a != c) || (b != d))

{

temp << m;

}

}

stream.close();

temp.close();

remove(f\_name); // удалить старый файл

rename("temp", f\_name);

return i; // количество прочитанных

}

int change\_file(const char\* f\_name, Money& t) // уменьшение всех записей, равных t, на 1 руб 50 коп

{

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

fstream stream(f\_name, ios::in);

if (!stream) return -1;

Money m; int i = 0, l = 0;

while (stream >> m)

{

if (stream.eof()) break;

i++;

if (m != t) temp << m;

else temp << ++m;

}

stream.close(); temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;

}

int add\_file(const char\* f\_name, int k, Money& t) // добавление к элементов после t

{

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

fstream stream(f\_name, ios::in);

if (!stream) return -1;

Money m; int i = 0;

long c = t.get\_rubles();

int d = t.get\_kop();

while (stream >> m)

{

long a = m.get\_rubles();

int b = m.get\_kop();

if ((a != c) || (b != d))

{

temp << m;

i++;

}

else

{

temp << m;

for (int j = 0; j < k; j++)

{

cout << "введите money: ";

Money n;

cin >> n;

temp << n;

i++;

}

}

}

stream.close(); temp.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

**Функция main()**

void main()

{

setlocale(0, "Rus");

Money m,temp;

int k, c, num, cnt;

char file\_name[30];

do

{ // меню

cout << "\n5. Добавить к записей после элемента с заданным значением";

cout << "\n4. Уменьшить все записи с заданным значением на 1 руб 50 коп";

cout << "\n3. Удалить все записи, равные заданному значению";

cout << "\n2. Вывести файл"; // печать файла

cout << "\n1. Создать файл"; // создание файла

cout << "\n0. Выход\n";

cin >> c;

switch (c)

{

case 1: cout << "file name? "; cin >> file\_name; // задаем имя файла

k = make\_file(file\_name); // вызов функции для записи в файл

if (k < 0) cout << "can't make file\n"; // вывод сообщения об ошибке

break;

case 2: cout << "file name? "; cin >> file\_name;

k = print\_file(file\_name);

if (k == 0)cout << "Empty file\n";

if (k < 0)cout << "can't read file\n";

break;

case 3: cout << "file name? "; cin >> file\_name;

cout << "введите money";

cin >> temp;

k = del\_file(file\_name, temp);

if (k < 0)cout << "can't read file\n";

break;

case 4: cout << "file name? "; cin >> file\_name;

cout << "введите money";

cin >> temp;

k = change\_file(file\_name,temp );

if (k < 0) cout << "can't read file";

break;

case 5: cout << "file name? "; cin >> file\_name;

cout << "k?"; cin >> k;

cout << "введите money: "; cin >> temp;

int a = add\_file(file\_name, k,temp);

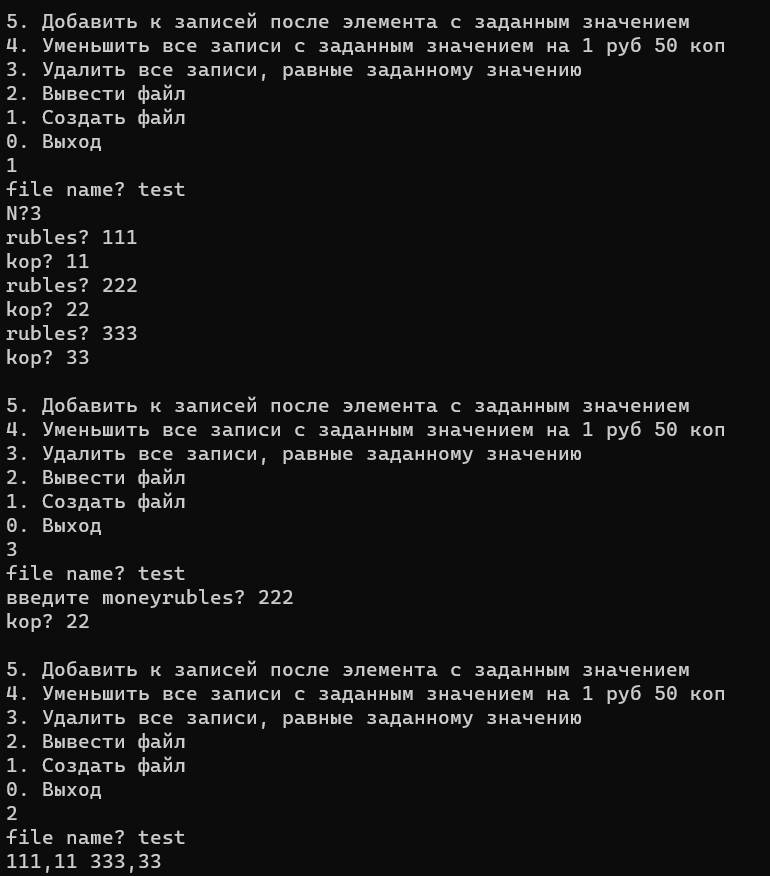
if (a < 0) cout << "can't read file";

}

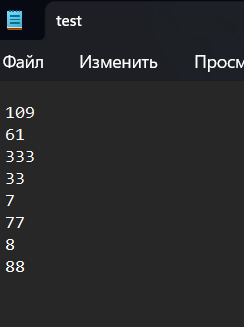
} while (c != 0);

}

**Результаты работы программы**





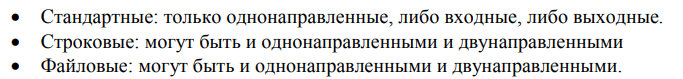


**Ответы на контрольные вопросы**

**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

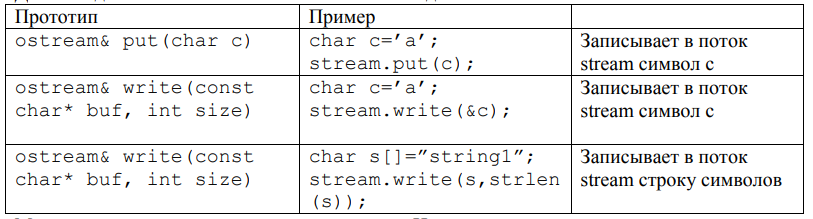
**6. Какая операция используется при выводе в форматированный поток?**

<< (бинарный оператор вставки) 

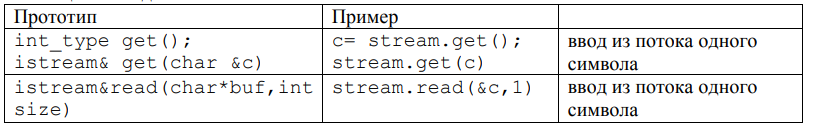
**7. Какая операция используется при вводе из форматированных потоков?**

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.

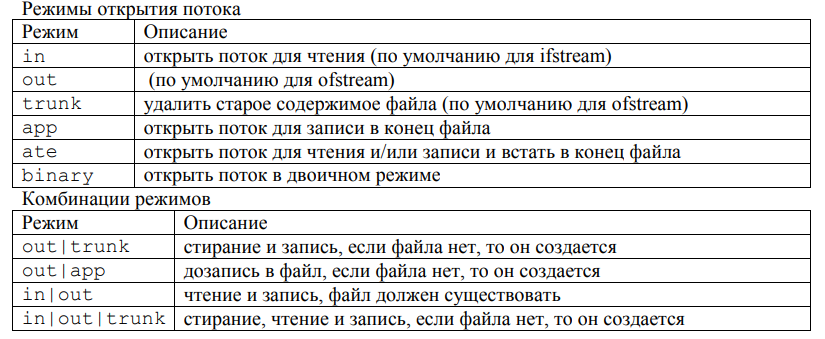
**9. Какие методы используется при вводе из форматированного потока?**



- 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



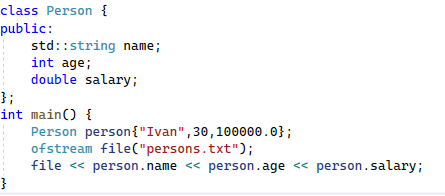
**-** открыт для чтения и записи



**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