Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Пермский национальный исследовательский политехнический университет»
Электротехнический факультет
Кафедра «Информационные технологии и автоматизированные системы»

ЛАБОРАТОРНАЯ РАБОТА №10

«Сохранение данных в файле с использованием потоков»

Выполнила:
студентка группы ИВТ-23-2б
Соловьева Екатерина
Александровна
Проверила:
доцент кафедры ИТАС
О. А. Полякова

2024 г.

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.

11 Вариант:

Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать:

- операции сравнения (<, >).
- добавление копеек (++) (постфиксная и префиксная формы) Задание:

- Удалить все записи большие заданного значения.
- Уменьшить все записи с заданным значением в два раза.
- Добавить K записей после элемента с заданным номером.

Код программы на C++

Money.h

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

using namespace std;
class Money {
private:
	long rub;
	int kop;
public:
	~Money();
public:
	Money();
	Money(long, int);
	Money(const Money&);
	Money operator =(const Money&);
	Money operator +(const Money&);
	bool operator ==(int val);
	void show();

	friend istream& operator >>(istream& in, Money& money);
	friend ostream& operator <<(ostream& out, const Money& money);

	friend fstream& operator >> (fstream& fin, Money& money);
	friend fstream& operator << (fstream& fout, const Money& money);
};
```

File.h

```
#pragma once
#include <iostream>
#include <fstream>
#include <string>
#include "Money.h"

using namespace std;

int makeFile(const char* fileName) {
	fstream file(fileName, ios::out | ios::trunc);
	if (!file) { return -1; }
	int n;
	Money m1;
	cout << "\nEnter count: "; cin >> n;
	for (int i = 0; i < n; i++) {
		cin >> m1;
		file << m1 << "\n";
	}
	file.close();
	return n;
}

int printFile(const char* fileName) {
	fstream file(fileName, ios::in);
	if (!file) { return -1; }
	Money m2; int i = 0;
	while (file >> m2) {
		cout << m2 << "\n";
		i++;
	}
	file.close();
	return i;
}

int delFile(const char* fileName, int index) {
	fstream temp("temp", ios::out);
	fstream file(fileName, ios::in);
	if (!file) { return -1; }
	int i = 0; Money m3;
	while (file >> m3) {
		if (file.eof()) { break; }
		i++;
		if (i != index) { temp << m3; }
	}
	file.close(); temp.close();
	remove(fileName);
	rename("temp", fileName);
	return i;
}

int addFile(const char* fileName, int index, Money m) {
	fstream temp("temp", ios::out);
	fstream file(fileName, ios::in);
	if (!file) { return -1; }
	Money m4; int i = 0, l = 0;
	while (file >> m4) {
		if (file.eof()) { break; }
		i++;
		if (i == index) {
			temp << m4;
			l++;
		}
		temp << m4;
	}
```

```
	file.close(); temp.close();
	remove(fileName);
	rename("temp", fileName);
	return l;
}

int addEnd(const char* fileName, Money m) {
	fstream file(fileName, ios::app);
	if (!file) { return -1; }
	file << m;
	return 1;
}

int changeFile(const char* fileName, Money m, int index) {
	fstream temp("temp", ios::out);
	fstream file(fileName, ios::in);
	if (!file) { return -1; }
	Money m5; int i = 0, l = 0; char x;
	while (file >> m5) {
		if (file.eof()) { break; }
		i++;
		if (i == index) {
			cout << m5 << " - is changing... Continue[y/n]?\n"; cin >> x;
			if (x == 'n' || x == 'N')break;
			temp << m;
			l++;
		}
		else temp << m5;
	}
	file.close(); temp.close();
	remove(fileName);
	rename("temp", fileName);
	return l;
}
```

Money.cpp

```
#include "Money.h"
#include <iostream>
#include <fstream>

using namespace std;

Money::Money() {
	rub = 32;
	kop = 45;
}
Money::Money(long R, int K) {
	rub = R;
	kop = K;
}
Money::~Money() {}

Money::Money(const Money& money) {
	rub = money.rub;
	kop = money.kop;
}

Money Money::operator =(const Money& money) {
	if (&money == this) { return *this; }
	rub = money.rub;
	kop = money.kop;
	return *this;
}
istream& operator >>(istream& in, Money& money) {
	cout << "\nEnter RUB: " << money.rub;
```

```
	cout << "\nEnter KOP: " << money.kop;
	cout << endl;
	return in;
}

ostream& operator <<(ostream& out, const Money& money) {
	out << "\nrub: " << money.rub;
	out << "\nkop: " << money.kop;
	cout << endl;
	return out;
}

fstream& operator >> (fstream& fin, Money& m) {
	fin >> m.rub;
	fin >> m.kop;
	return fin;
}
fstream& operator << (fstream& fout, const Money& m) {
	fout << m.rub << "\n" << m.kop << "\n";
	return fout;
}


void Money::show() {
	cout << "\nResult money: " << rub << ", " << kop;
}

Money Money::operator +(const Money& money) {
	int M = rub * 100 + kop;
	int monM = money.rub * 100 + money.kop;
	M += monM;

	Money temp(M / 100, M % 100);
	return temp;
}
bool Money::operator==(int val) {
	int all = rub * 100 + kop;
	return all == val;
}
```

Main.cpp

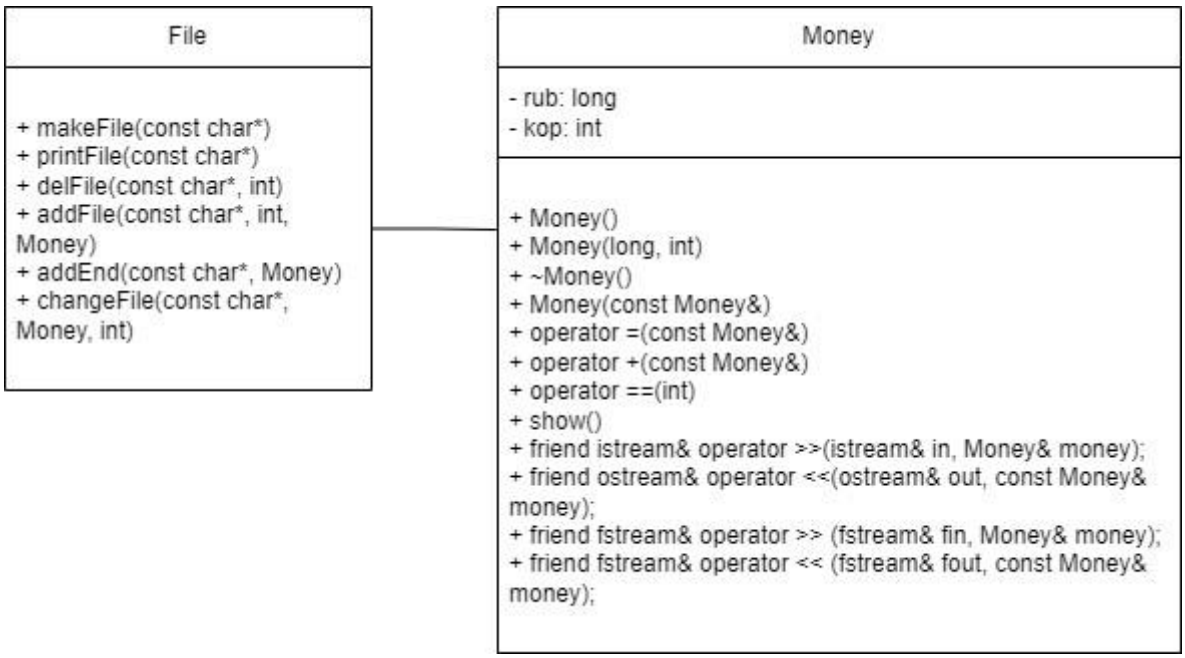
```
#include "Money.h"
#include <iostream>
#include <fstream>
#include "File.h"

using namespace std;

int main() {
	Money A;
	cin >> A;
	cout << "A:  " << A << endl;

	Money B;
	B = A;
	cout << endl;
	cout << "B: " << B << endl;

	cout << "\n----------FILE----------" << endl;
	Money M;
	int tmp, choise;
	char fileName[30];

	do {
		cout << "\n1. Make file";
```

```
                cout << "\n2. Print file";
                cout << "\n3. Delete record from file";
                cout << "\n4. Add record to file";
                cout << "\n5. Change record in file";
                cout << "\n0. Exit\n";
                cin >> choise;
                switch (choise) {
                case 1:
                        cout << "File name: "; cin >> fileName;
                        tmp = makeFile(fileName);
                        if (tmp < 0) { cout << "Can't make file" << endl; }
                        break;
                case 2:
                        cout << "File name: "; cin >> fileName;
                        tmp = makeFile(fileName);
                        if (tmp == 0) { cout << "Empty file\n"; }
                        if (tmp < 0) { cout << "Can't read file\n"; }
                        break;
                case 3:
                        cout << "File name: "; cin >> fileName;
                        int num; cout << "Number: "; cin >> num;
                        tmp = delFile(fileName, num);
                        if (tmp < 0) { cout << "Can't read file" << endl; }
                        break;
                case 4: {
                        Money m1;
                        cout << "File name: "; cin >> fileName;
                        cout << "Number: "; cin >> num;
                        cout << "New money sum: "; cin >> m1;
                        tmp = addFile(fileName, num, m1);
                        if (tmp < 0) { cout << "Can't read file" << endl; }
                        if (tmp == 0) { tmp = addEnd(fileName, m1); }
                        break;
                }
                case 5: {
                        Money m2;
                        cout << "File name: "; cin >> fileName;
                        cout << "\nNumber: "; cin >> num;
                        cout << "\nNew money sum: "; cin >> m2;
                        tmp = changeFile(fileName, m2, num);
                        if (tmp < 0) { cout << "Can't read file" << endl; }
                        if (tmp == 0) { cout << "\nNot such record" << endl; }
                        break;
                }
                }
        } while (choise != 0);
        return 0;
}
```

Вывод:

```
Enter RUB: 32
Enter KOP: 45
A:
rub: 32
kop: 45


B:
rub: 32
kop: 45


----------FILE----------

1. Make file
2. Print file
3. Delete record from file
4. Add record to file
5. Change record in file
0. Exit
1
File name: Name

Enter count: 2

Enter RUB: 32
Enter KOP: 45

Enter RUB: 32
Enter KOP: 45
```

UML-диаграмма



Анализ результатов

Программа сработала корректно и вывела необходимые результаты.