Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 - «Программная инженерия»

**Лабораторная работа №10**

**"Сохранение данных в файле с использованием потоков"**

Выполнил студент гр. РИС-24-3б Носков Егор Михайлович

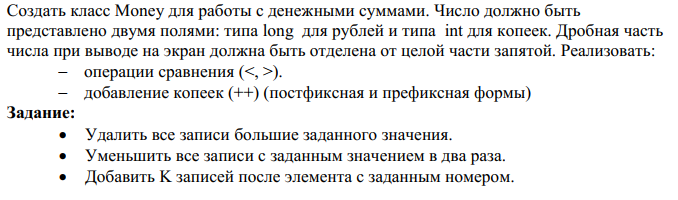
Проверил: Доцент кафедры ИТАС Ольга Андреевна Полякова

|  |  |
| --- | --- |
|  | |
| (оценка) (подпись) | |
|  |
| (дата) |

Г. Пермь, 2024

**Постановка задачи**

**Вариант 1:**

****

**Программа на языке C++**

Money.h:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Money

{

long rub;

int kop;

private:

void recount();

public:

Money() { rub = 0; kop = 0; };

Money(long r, int k) { rub = r; kop = k; recount(); };

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

~Money() {};

long get\_rub() { return rub; }

int get\_kop() { return kop; }

void set\_rub(long r) { rub = r; }

void set\_kop(int k) { kop = k; }

//перегруженные операции

Money& operator=(const Money&);

Money operator-(float value);

bool operator==(const Money& m);

bool operator!=(const Money& m);

//глобальные функции ввода-вывода

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

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

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

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

};

Money.cpp:

#include <iostream>

#include "Money.h"

using namespace std;

void Money::recount()

{

if (kop >= 100) {

rub += kop / 100;

kop %= 100;

}

else if (kop < 0) {

rub -= (abs(kop) / 100 + 1);

kop = 100 - abs(kop) % 100;

}

}

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

{

if (this != &m)

{

rub = m.rub;

kop = m.kop;

}

return \*this;

}

Money Money::operator-(float value)

{

long r = static\_cast<long>(value);

int k = static\_cast<int>((value - r) \* 100);

return Money(rub - r, kop - k);

}

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

{

return rub == m.rub && kop == m.kop;

}

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

{

return rub != m.rub || kop != m.kop;

}

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

{

bool fail = false;

cout << "rub: "; in >> m.rub;

do

{

cout << "kop(0-99): "; in >> m.kop;

if (m.kop > 99 || m.kop < 0)

{

cout << "Error! Enter correct value.\n";

fail = true;

}

else fail = false;

} while (fail);

return in;

}

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

{

return (out << m.rub << "," << m.kop);

}

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;

}

file\_work.h:

#include "Money.h"

int make\_file(const char\* f\_name)

{

fstream stream(f\_name, ios::out | ios::trunc);

if (!stream)return -1;

int n;

Money m;

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

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

{

cout << "#" << i+1 << "\n";

cin >> m;

stream << m << "\n";

}

stream.close();

return n;

}

int print\_file(const char\* f\_name)

{

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

if (!stream)return -1;

Money m; int i = 0;

while (stream >> m)

{

cout << "#" << i+1 << "\n";

cout << m << "\n";

i++;

}

stream.close();

return i;

}

int del\_file(const char\* f\_name, long rub, int kop)

{

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

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

if (!stream)return -1;

int i = 0; Money m;

while (stream >> m)

{

if (stream.eof())break;

i++;

if (m.get\_rub() != rub || m.get\_kop() != kop) temp << m;

}

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

remove(f\_name);

rename("temp", f\_name);

return i;

}

int add\_file(const char\* f\_name, long rub, int kop)

{

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

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

if (!stream)return -1;

Money m; int n, l = 0;

while (stream >> m)

{

if (stream.eof())break;

if (m.get\_rub() == rub && m.get\_kop() == kop)

{

temp << m;

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

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

{

cout << "#" << i + 1 << "\n";

cin >> m;

temp << m << "\n";

}

l++;

}

else temp << m;

}

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

remove(f\_name);

rename("temp", f\_name);

return l;

}

int add\_end(const char\* f\_name)

{

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

if (!stream)return -1;

int n;

Money m;

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

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

{

cout << "#" << i + 1 << "\n";

cin >> m;

stream << m << "\n";

}

stream.close();

return 1;

}

int change\_file(const char\* f\_name, long rub, int kop)

{

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

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

if (!stream)return -1;

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

char x;

while (stream >> m)

{

if (stream.eof())break;

i++;

if (m.get\_rub() == rub && m.get\_kop() == kop)

{

cout << "#" << i << ": " << m << " - is changing(" << m << "-1,50)... Continue[y/n]?\n";

cin >> x;

if (x == 'n' || x == 'N')break;

if (m.get\_rub() < 2 && m.get\_kop() < 50)

{

cout << "Error. Out of range!";

temp << m;

}

else

{

m = m - 1.50;

temp << m;

}

l++;

}

else temp << m;

}

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

remove(f\_name);

rename("temp", f\_name);

return l;

}

Money\_main.cpp:

#include "file\_work.h"

int main()

{

int k, c;

char file\_name[30] = "";

do

{

cout << "\nFile [" << file\_name << "]";

cout << "\n1. Select file name";

cout << "\n2. Make file";

cout << "\n3. Print file";

cout << "\n4. Delete record from file";

cout << "\n5. Add record to file";

cout << "\n6. Change record in file";

cout << "\n0. Exit\n";

cout << ">>"; cin >> c;

switch (c)

{

case 1:

cout << "Enter file name: "; cin >> file\_name;

break;

case 2:

k = make\_file(file\_name);

if (k < 0)cout << "Can't make file";

break;

case 3:

k = print\_file(file\_name);

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

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

break;

case 4:

long rub; cout << "rub: "; cin >> rub;

int kop; cout << "kop: "; cin >> kop;

k = del\_file(file\_name, rub, kop);

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

break;

case 5:

cout << "rub: "; cin >> rub;

cout << "kop: "; cin >> kop;

k = add\_file(file\_name, rub, kop);

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

if (k == 0) k = add\_end(file\_name);

break;

case 6:

cout << "rub: "; cin >> rub;

cout << "kop: "; cin >> kop;

k = change\_file(file\_name, rub, kop);

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

if (k == 0) cout << "\nNot such record";

break;

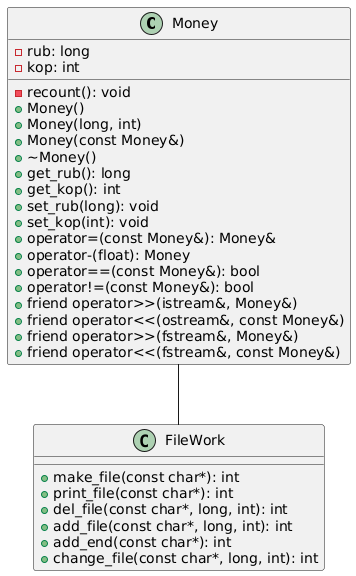
}

}

while (c != 0);

}

**UML-диаграмма классов**

****

**Контрольные вопросы**

**1. Что такое поток?**

Поток (stream) — это абстракция для работы с последовательным вводом-выводом данных. Он обеспечивает унифицированный интерфейс для взаимодействия с различными устройствами (консоль, файлы, строки).

**2. Типы потоков:**

* **Стандартные**: cin, cout, cerr, clog (консоль).
* **Файловые**: ifstream (ввод), ofstream (вывод), fstream (ввод/вывод).
* **Строковые**: istringstream (ввод из строки), ostringstream (вывод в строку), stringstream (ввод/вывод).

**3. Библиотека для стандартных потоков:**

#include <iostream>

**4. Библиотека для файловых потоков:**

#include <fstream>

**5. Библиотека для строковых потоков:**

#include <sstream>

**6. Операция для вывода в форматированный поток:**

Оператор << (например, cout << "Hello";).

**7. Операция для ввода из форматированного потока:**

Оператор >> (например, cin >> x;).

**8. Методы для вывода в форматированный поток:**

* put() — вывод одного символа.
* write() — вывод блока данных.
* Форматирующие методы: width(), precision(), setf().

**9. Методы для ввода из форматированного потока:**

* get() — чтение символа.
* getline() — чтение строки.
* read() — чтение блока данных.
* ignore() — пропуск символов.

**10. Режимы открытия файловых потоков:**

* ios::in — чтение.
* ios::out — запись (с удалением содержимого).
* ios::app — добавление в конец файла.
* ios::ate — открытие с перемещением в конец.
* ios::trunc — удаление содержимого при открытии.
* ios::binary — бинарный режим.

**11. Режим для добавления записей:**

ios::app (например, ofstream file("f.txt", ios::app);).

**12. Режим в**ifstream file("f.txt");**:**

ios::in (только чтение).

**13. Режим в**fstream file("f.txt");**:**

По умолчанию — ios::in | ios::out (чтение и запись, файл должен существовать).

**14. Режим в**ofstream file("f.txt");**:**

ios::out | ios::trunc (запись с удалением содержимого).

**15. Открытие в режиме**ios::out | ios::app**:**

ofstream file("f.txt", ios::out | ios::app); *// Добавление в конец.*

**16. Открытие в режиме**ios::out | ios::trunc**:**

ofstream file("f.txt", ios::out | ios::trunc); *// Перезапись файла.*

**17. Открытие в режиме**ios::out | ios::in | ios::trunc**:**

fstream file("f.txt", ios::out | ios::in | ios::trunc); *// Чтение/запись с очисткой.*

**18. Открытие файла для чтения:**

ifstream file("f.txt"); *// или с явным режимом: ios::in.*

**19. Открытие файла для записи:**

ofstream file("f.txt"); *// или с режимами: ios::out, ios::app.*

**20. Примеры открытия файлов:**

*// Чтение*

ifstream fin("input.txt", ios::binary);

*// Запись с очисткой*

ofstream fout("output.txt", ios::trunc);

*// Чтение и запись (файл должен существовать)*

fstream file("data.txt", ios::in | ios::out);

**21. Пример чтения объектов:**

ifstream fin("data.txt");

Money m;

while (fin >> m) { *// Используется перегруженный operator>>*

cout << m << endl;

}

**22. Пример записи объектов:**

ofstream fout("data.txt");

Money m(100, 50);

fout << m; *// Используется перегруженный operator<<*

**23. Алгоритм удаления записей:**

1. Открыть исходный файл для чтения.
2. Создать временный файл для записи.
3. Копировать записи, кроме удаляемых.
4. Удалить исходный файл, переименовать временный.

**24. Алгоритм добавления записей:**

* **В конец**: открыть файл в режиме ios::app.
* **В середину**: аналогично удалению, с вставкой новых данных.

**25. Алгоритм изменения записей:**

1. Открыть исходный файл для чтения.
2. Создать временный файл для записи.
3. Копировать записи, изменяя нужные.
4. Заменить исходный файл временным.

**Ссылка на репозиторий с отчетом Github**

****