Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.10**

Дисциплина: «Информатика»

Тема: “ Объектно-ориентированное программирование.

Сохранение данных в файле с использованием потоком.”

Вариант 10

Выполнил:

Студент группы ИВТ-20-2Б Галинов О. Ю.

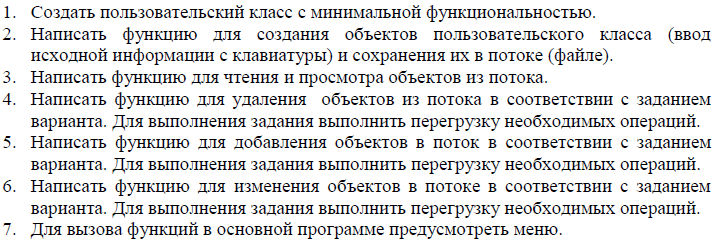
Проверила:

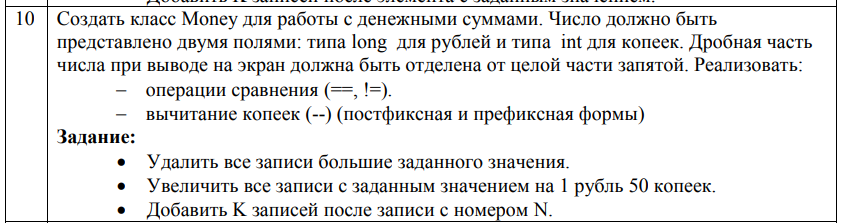
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**





**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Money с полями rub и kop для хранения числа в целочисленном типе и дробном типе данных.

**1.2.** Организовать необходимые методы для ввода данных в поля rub и kop: гетторы, сетторы, конструкторы, деструкторы.

**1.3.** Организовать перегрузку оператора >> дружественной классу Money.

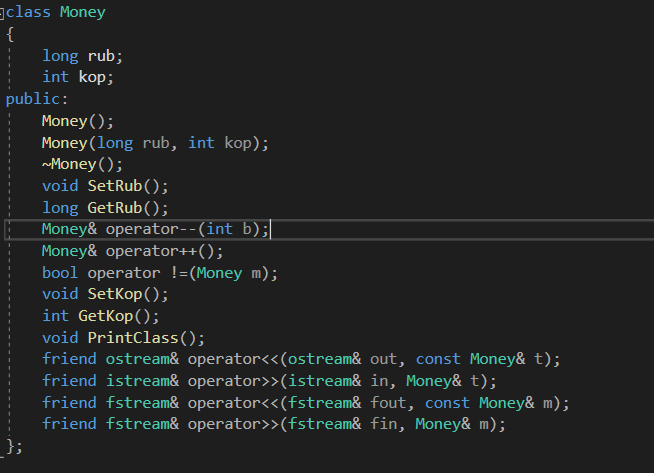
**1.4.** Организовать перегрузку оператора << дружественной классу Money.

**1.5.** Организовать перегрузку оператора ++ класса Money для сложения чисел.

**1.6.** Организовать перегрузку оператора != класса Money для сравнения экземпляров класса Money.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Организовать класс Money с полями rub и kop для хранения числа в целочисленном типе и дробном типе данных.



**2.2.** Переменные типа int c необходима для использования меню для выбора пользователем действий.

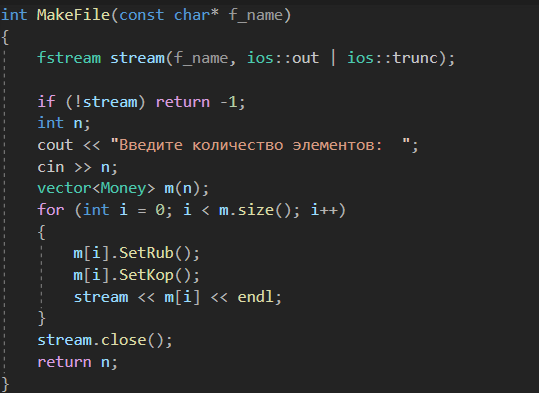
int c;

**2.2.** Массив типа char – file\_name, необходим для хранения называния файла, в котором будут храниться пары чисел.

char file\_name [] = "file\_name";

**3.** Для решения задачи данные были представлены в следующем виде:

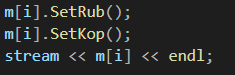
**3.1.** Пары вводятся самостоятельно и записываются в file\_name.



**4.** Для операций ввода и вывода использовались следующие операторы и функции:

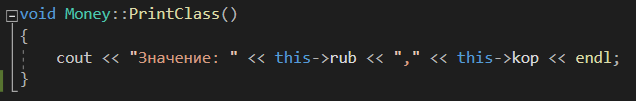
**4.1.** Открытие файла происходит в методе, созданном для изменения файла, в нем же происходит запись в файл.



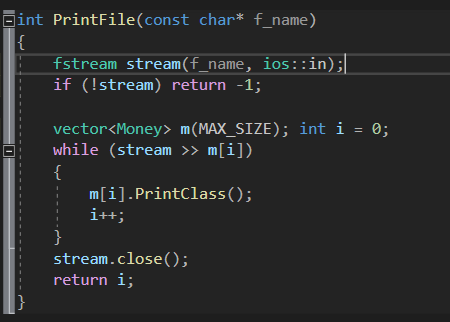




**4.2.** Для вывода пар чисел на консоль использовать функцию Print().



После того как все элементы были записаны в файл, происходит процесс их вывода в консоль при помощи прохода по всему файлу построчно и вывода каждого элемент вспомогательного вектора Money.



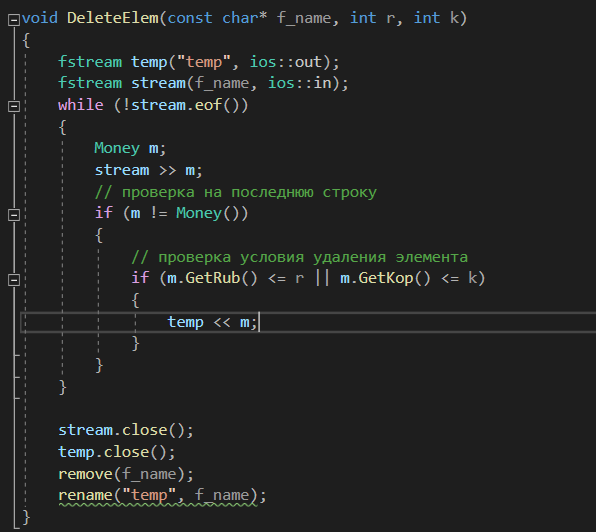
**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В функции DeleteElem () в качестве параметров передаётся название файла, и значения для сравнения с парами в самом файле. Таким образом, в цикле создаётся объект класса Money, который инициализируется в значениями из файла через перегрузку оператора >> .

Далее происходит проверка условия удаления элемента: если итерирование дошло до конца файла, то, чтобы не заносить в поток пустую строку объект m сравнивается с объектом класса Money, который инициализирован значениями INT\_MIN в конструкторе класса, что условно принимается как пустая строка.

Если объект m имеет значения, не равные INT\_MIN, то этот объект подходит и можно дальше проверять его: если значения пары меньше или равны, чем переданные как параметры в функцию, то они передаются в поток через перегрузку оператора << и отправляются в запасной файл.

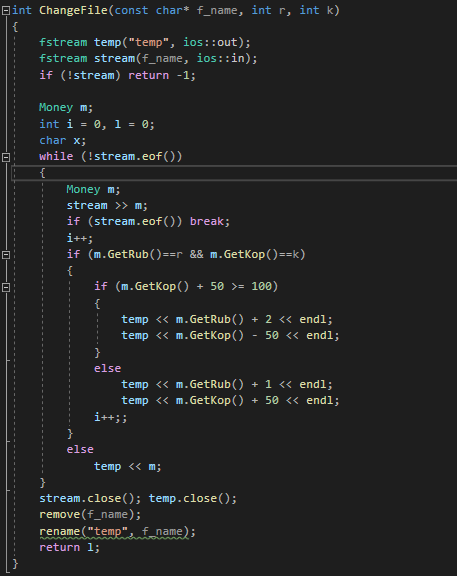
После цикла необходимо закрыть файлы, удалить файл изначальный, а запасному файлу дать имя изначального.



**5.2.** В функции ChangeFile () в качестве параметров передаётся название файла, пара чисел r, k, которые пользователь хочет увеличить. Таким образом, в цикле while() создаётся объект класса Money, который инициализируется в значениями из файла через перегрузку оператора >> .

В первом ветвлении происходит проверка на пустую строку путём сравнения потока c пустым (инициализированным INT\_MIN) объектом. Во втором ветвлении происходит сравнение параметров, переданных в функцию r и k c данной парой из объекта m.

В случае равенства введённых элементов с парой в объекте element с помощью метода GetRub() и GetKop() устанавливаются значения увеличенные на 1руб и 50коп.

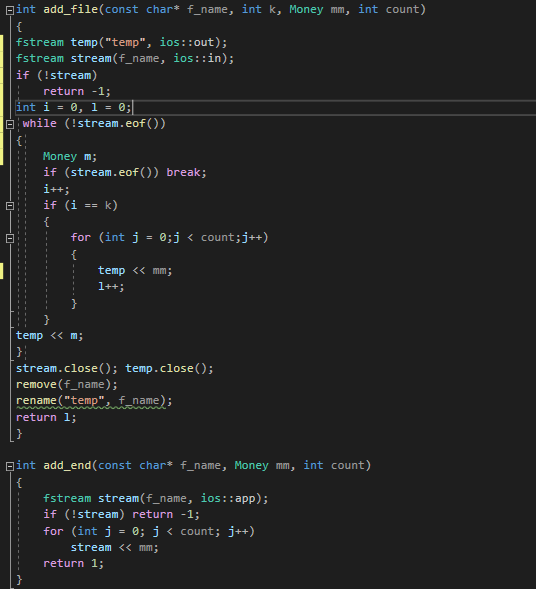


**5.3.** В функции DeleteElem () в качестве параметров передаётся название файла, позиция для новых элементов, кол-во новых элементов и значения для записи. Таким образом, в цикле создаётся объект класса Money, который инициализируется в значениями из файла через перегрузку оператора >> ..

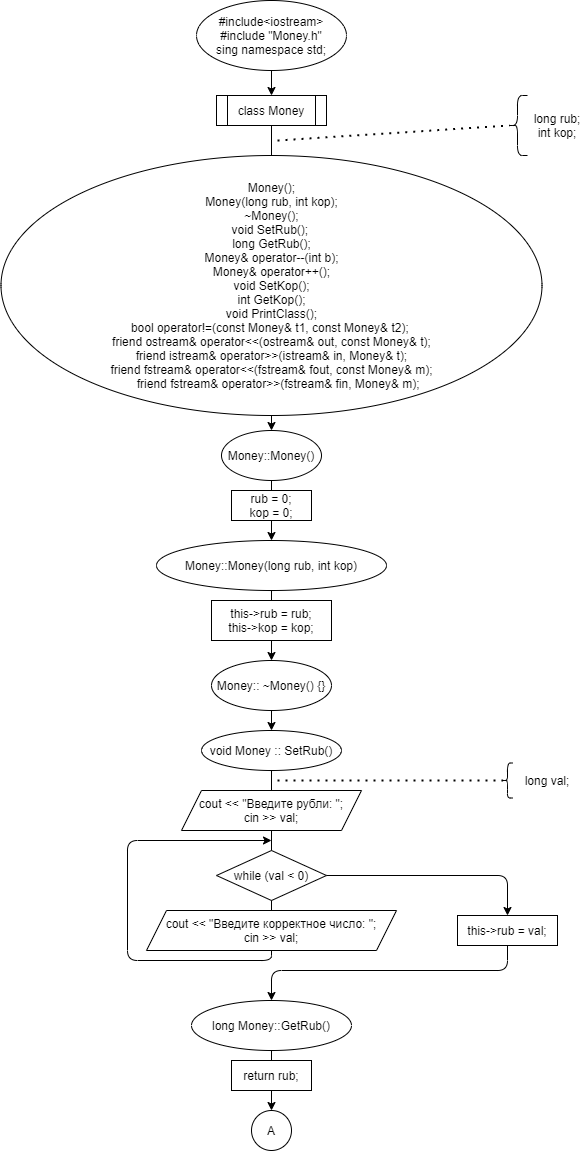
В цикле while() создаётся объект класса Money - m, который инициализируется в значениями из файла через перегрузку оператора >> .

Далее происходит проход по всему файлу, до той позиции, которой передается как параметр (кроме случая, когда это последняя позиция) и начиная с этой позиции до кол-ва новых элементов происходит запись в ново созданный файл.

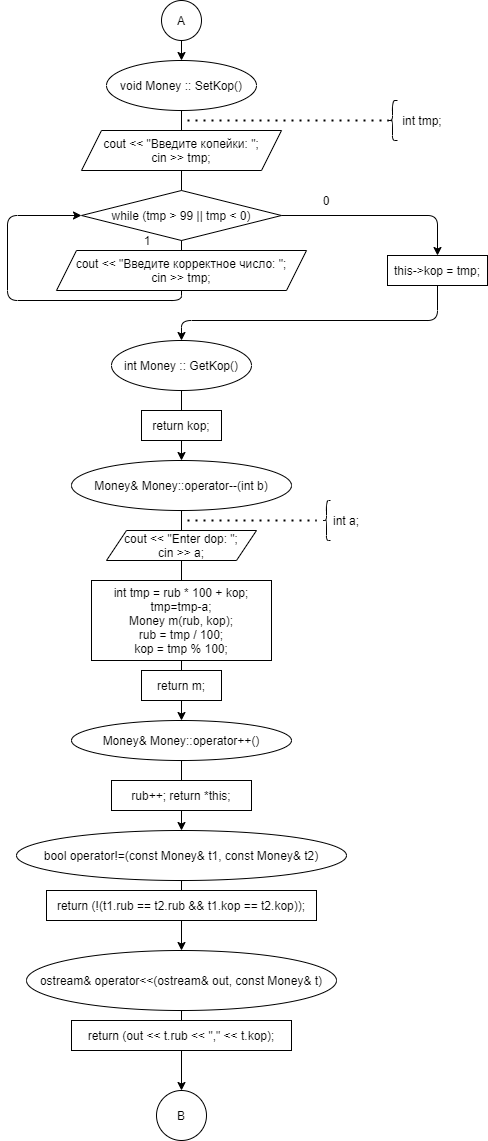
Если же позиция последняя, то в исходный файл просто записывается кол-во которое так же передавалось в функцию, новых элементов.

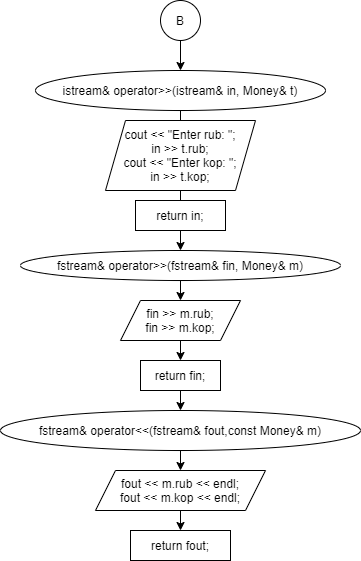


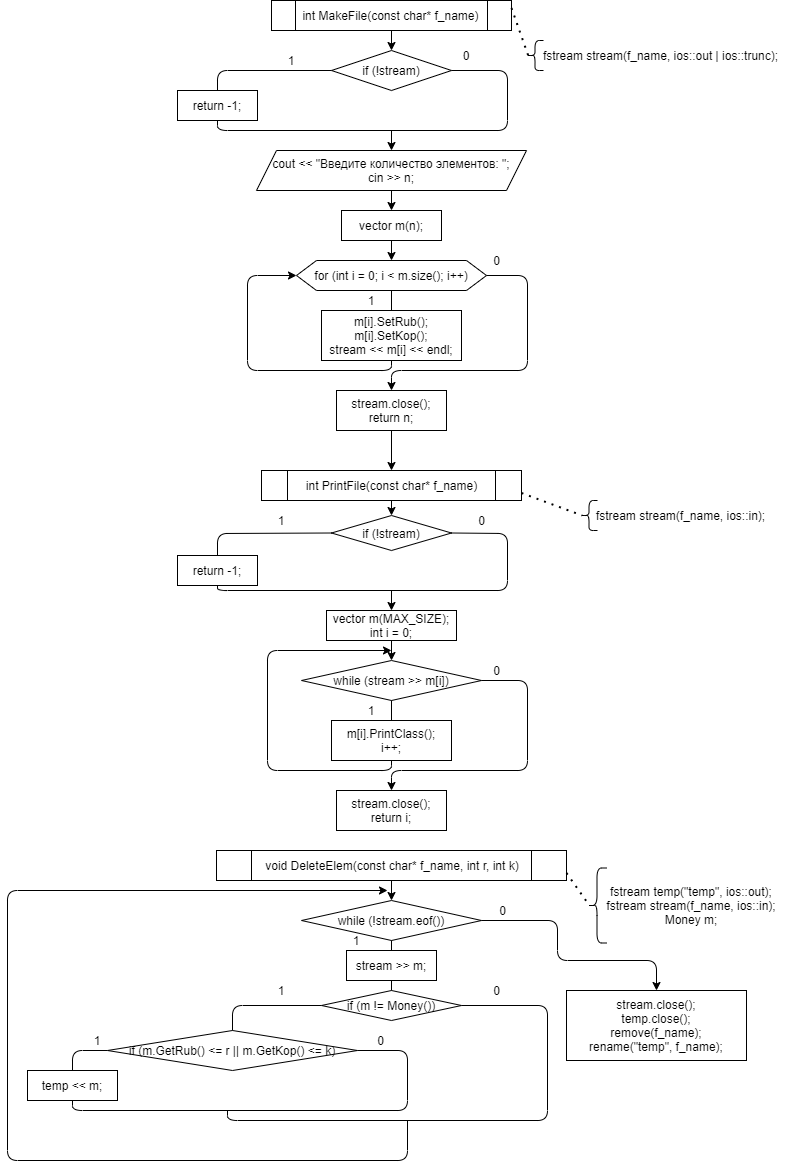
**Блок-схема**

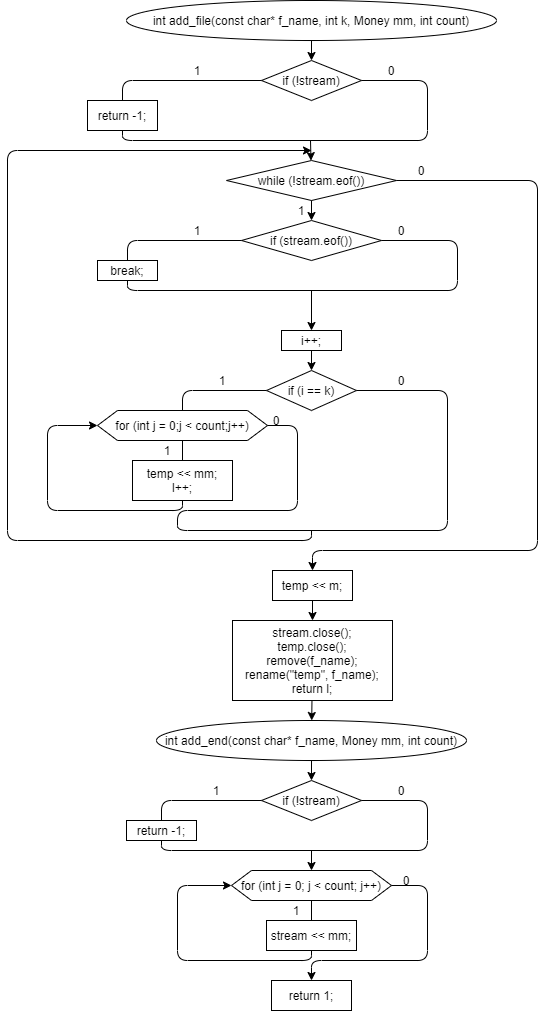


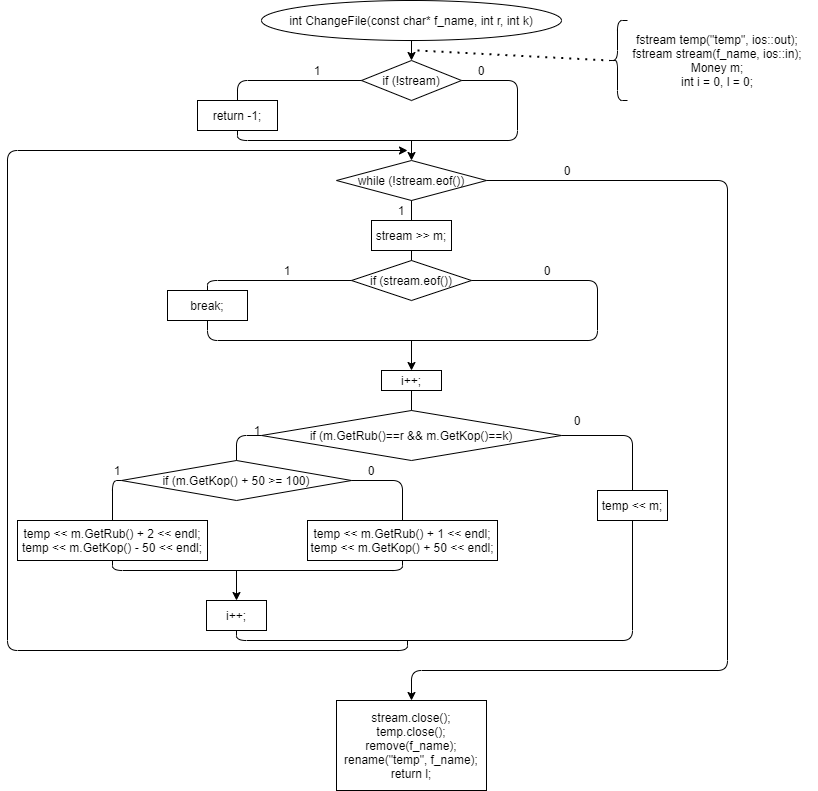
10

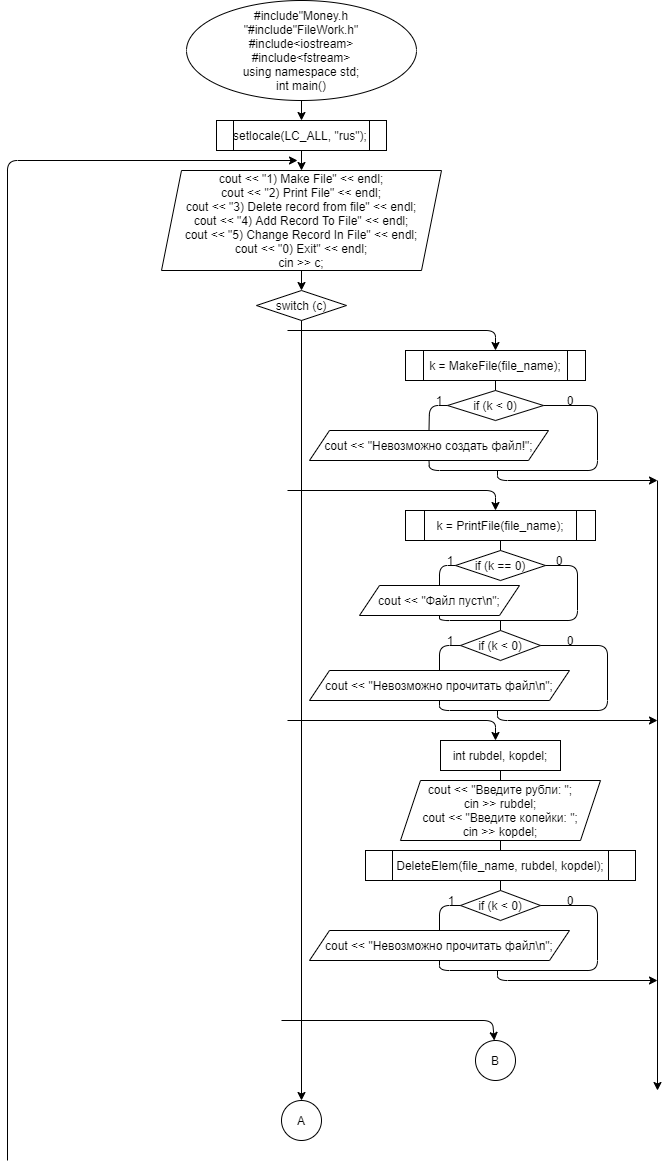


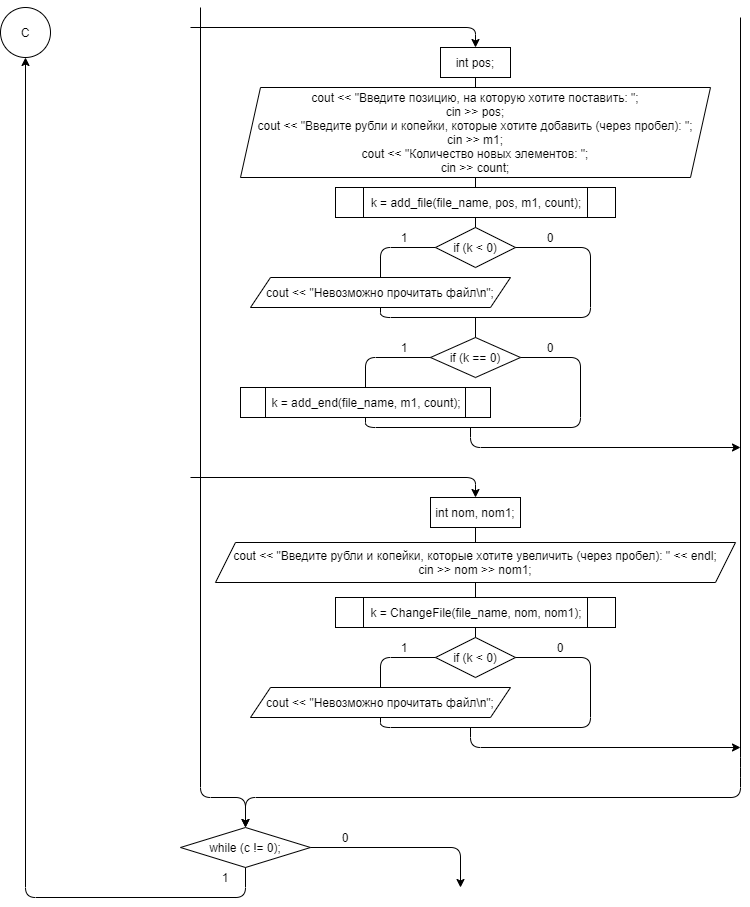












**Код**

#pragma once

#include"Money.h"

#include<fstream>

#include<iostream>

#include<string>

#include<vector>

using namespace std;

const int MAX\_SIZE = 20;

int MakeFile(const char\* f\_name)

{

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

if (!stream) return -1;

int n;

cout << "Введите количество элементов: ";

cin >> n;

vector<Money> m(n);

for (int i = 0; i < m.size(); i++)

{

m[i].SetRub();

m[i].SetKop();

stream << m[i] << endl;

}

stream.close();

return n;

}

int PrintFile(const char\* f\_name)

{

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

if (!stream) return -1;

vector<Money> m(MAX\_SIZE); int i = 0;

while (stream >> m[i])

{

m[i].PrintClass();

i++;

}

stream.close();

return i;

}

void DeleteElem(const char\* f\_name, int r, int k)

{

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

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

while (!stream.eof())

{

Money m;

stream >> m;

// проверка на последнюю строку

if (m != Money())

{

// проверка условия удаления элемента

if (m.GetRub() <= r || m.GetKop() <= k)

{

temp << m;

}

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

}

int add\_file(const char\* f\_name, int k, Money mm, int count)

{

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

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

if (!stream)

return -1;

int i = 0, l = 0;

while (!stream.eof())

{

Money m;

if (stream.eof()) break;

i++;

if (i == k)

{

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

{

temp << mm;

l++;

}

}

temp << m;

}

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

remove(f\_name);

rename("temp", f\_name);

return l;

}

int add\_end(const char\* f\_name, Money mm, int count)

{

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

if (!stream) return -1;

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

stream << mm;

return 1;

}

int ChangeFile(const char\* f\_name, int r, int k)

{

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

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

if (!stream)

return -1;

Money m;

int i = 0, l = 0;

while (!stream.eof())

{

Money m;

stream >> m;

if (stream.eof()) break;

i++;

if (m.GetRub()==r && m.GetKop()==k)

{

if (m.GetKop() + 50 >= 100)

{

temp << m.GetRub() + 2 << endl;

temp << m.GetKop() - 50 << endl;

}

else

temp << m.GetRub() + 1 << endl;

temp << m.GetKop() + 50 << endl;

i++;

}

else

temp << m;

}

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

remove(f\_name);

rename("temp", f\_name);

return l;

}

#include"Money.h"

#include"FileWork.h"

#include <iostream>

#include<fstream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

//system("chcp>1251");

Money m1;

int k, c, count;

char file\_name[] = "file\_name";

do

{

cout << "1) Make File" << endl;

cout << "2) Print File" << endl;

cout << "3) Delete record from file" << endl;

cout << "4) Add Record To File" << endl;

cout << "5) Change Record In File" << endl;

cout << "0) Exit" << endl;

cin >> c;

switch (c)

{

case 1:

k = MakeFile(file\_name);

if (k < 0)

cout << "Невозможно создать файл!";

break;

case 2:

k = PrintFile(file\_name);

if (k == 0) cout << "Файл пуст\n";

if (k < 0) cout << "Невозможно прочитать файл\n";

break;

case 3:

int rubdel, kopdel;

cout << "Введите рубли: ";

cin >> rubdel;

cout << "Введите копейки: ";

cin >> kopdel;

DeleteElem(file\_name, rubdel, kopdel);

if(k < 0)

cout << "Невозможно прочитать файл\n";

break;

case 4 :

int pos;

cout << "Введите позицию, на которую хотите поставить: ";

cin >> pos;

cout << "Введите рубли и копейки, которые хотите добавить (через пробел): ";

cin >> m1;

cout << "Количество новых элементов: ";

cin >> count;

k = add\_file(file\_name, pos, m1, count);

if (k < 0)

cout << "Невозможно прочитать файл\n";

if (k == 0)

k = add\_end(file\_name, m1, count);

break;

case 5:

int nom, nom1;

cout << "Введите рубли и копейки, которые хотите увеличить (через пробел): " << endl;

cin >> nom >> nom1;

k = ChangeFile(file\_name, nom, nom1);

if (k < 0)

cout << "Невозможно прочитать файл\n" << endl;

break;

}

} while (c != 0);

}

#pragma once

#include<iostream>

#include<fstream>

#pragma once

#include<iostream>

using namespace std;

class Money

{

long rub;

int kop;

public:

Money();

Money(long rub, int kop);

~Money();

void SetRub();

long GetRub();

void SetKop();

int GetKop();

Money& operator--(int b);

Money& operator++();

bool operator !=(Money m);

void PrintClass();

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

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

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

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

};

#include "Money.h"

#include<iostream>

using namespace std;

Money::Money()

{

rub = 0;

kop = 0;

}

Money::Money(long rub, int kop)

{

this->rub = rub;

this->kop = kop;

}

Money:: ~Money() {}

void Money::SetRub()

{

long val;

cout << "Введите рубли: ";

cin >> val;

while (val < 0)

{

cout << "Введите корректное число: ";

cin >> val;

}

this->rub = val;

}

long Money::GetRub() { return rub; }

void Money::SetKop()

{

int tmp;

cout << "Введите копейки: ";

cin >> tmp;

while (tmp > 99 || tmp < 0)

{

cout << "Введите корректное число: ";

cin >> tmp;

}

this->kop = tmp;

}

int Money::GetKop() { return kop; }

void Money::PrintClass()

{

cout << "Значение: " << this->rub << "," << this->kop << endl;

}

Money& Money::operator++()

{

rub++;

return \*this;

}

Money& Money::operator--(int b)

{

int a;

cout << "Enter dop: ";

cin >> a;

int tmp = rub \* 100 + kop;

tmp = tmp - a;

Money m(rub, kop);

rub = tmp / 100;

kop = tmp % 100;

return m;

}

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

{

out << m.rub;

out << ",";

out << m.kop;

return out;

}

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

{

in >> m.rub;

in >> m.kop;

return in;

}

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

{

fin >> m.rub;

fin >> m.kop;

return fin;

}

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

{

fout << m.rub << endl;

fout << m.kop << endl;

return fout;

}

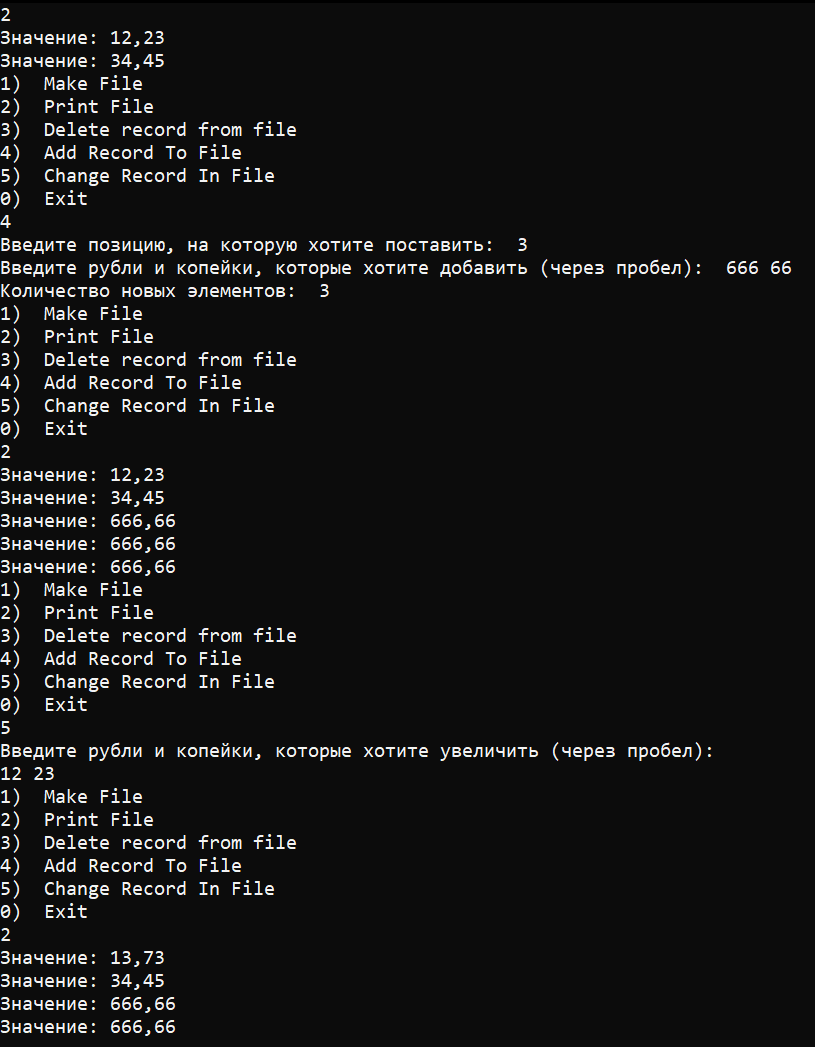
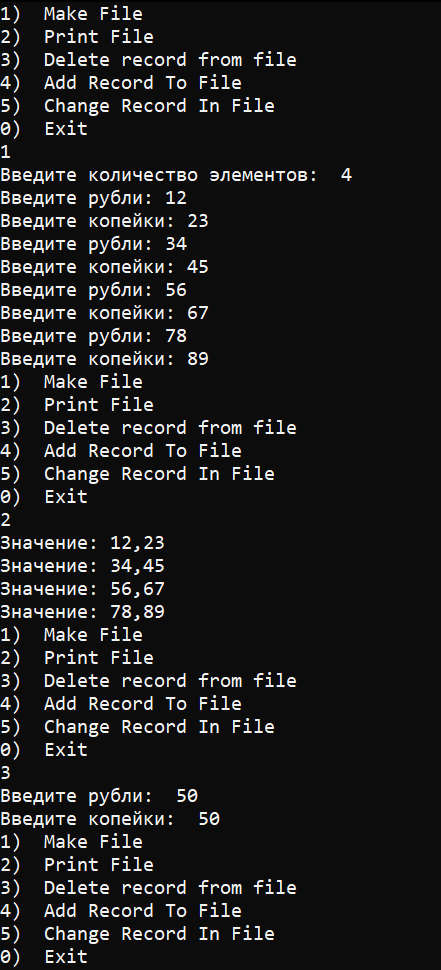
bool Money::operator!=(Money secondOperand)

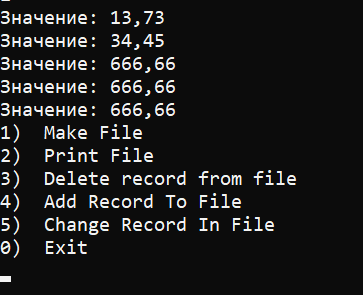
{

return rub != secondOperand.rub && kop != secondOperand.kop;

}

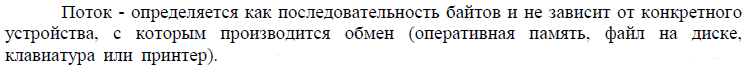
**Скриншоты**





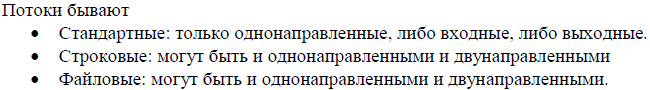
**Контрольные вопросы**



1. 



2.





3. 



4. 



5. 

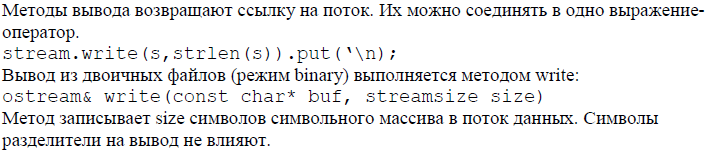


6. 

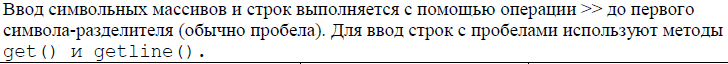


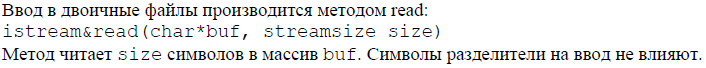
7. operator >>

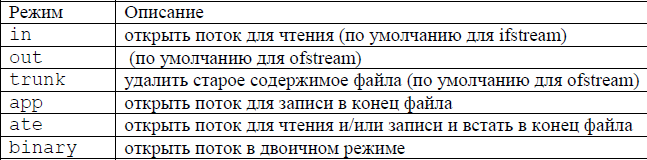


8. 



9.



10. 



11. 

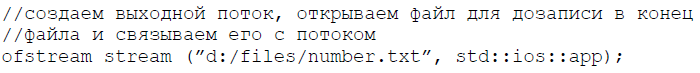


12. 



13. fstream file("f.txt", ios\_base::in | ios\_base::out)



14. 



15. 



16. 



17. 

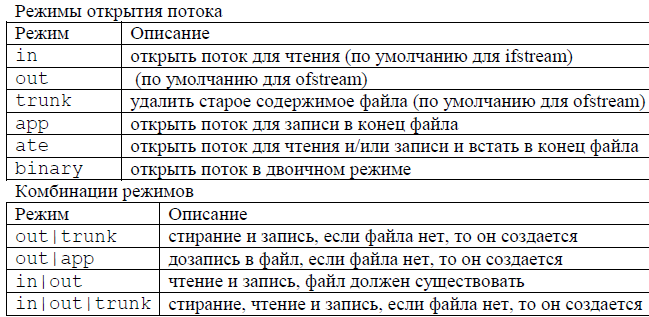


18. 



19. 



20. 



21. Перегрузка оператора в классе.



22. 



23. Перенесение в файл элементов, удовлетворяющих условию.



24. Запись в файл из массива элементов между записью элементов из входного файла.



25. Перезаписывание всех элементов в выходной файл с исправлением тех, что удовлетворяют условию.