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 №10**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Сохранение данных в файле с использованием потоков

Вариант 6.

Выполнил работу

Студент группы РИС-22-1Б

Поважный В. Е.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

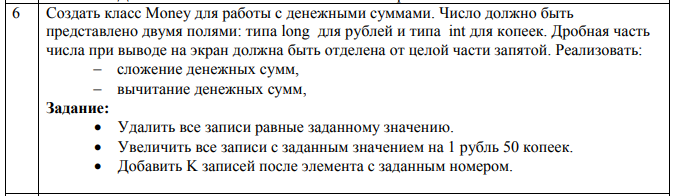
г. Пермь-2023

**Введение**

Для выполнения лабораторной работы требуется сделать отчет программы. Создание программ – отличный способ практики программирования.

**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.



**Код программы**

**ооп 10.cpp**

#include "Money.h"

#include <iostream>

#include <fstream>

#include "fire\_work.h"

#include <windows.h>

using namespace std;

void main()

{

Money p, p1, p2;

int k, c;

char file\_name[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 >> c;

switch (c)

{

case 1:

cout << "file name ";

cin >> file\_name;

k = make\_file(file\_name);

if (k < 0)

cout << "Can't make file";

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;

int nom;

cout << "nom ";

cin >> nom;

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

if (k < 0)

cout << "Can't read file";

break;

case 4:

cout << "file name ";

cin >> file\_name;

cout << "nom ";

cin >> nom;

cout << "New Money:";

k = add\_file(file\_name, nom, p1);

break;

case 5:

cout << "file name ";

cin >> file\_name;

cout << "nom ";

cin >> nom;

cout << "New Money: ";

cin >> p2;

k = change\_file(file\_name, nom, p2);

if (k < 0)

cout << "\nCan't read file";

if (k == 0)

cout << "\nNot such record";

break;

}

} while (c != 0);

}

**Money.h**

**#include** <iostream>

#include <fstream>

#include <string>

using namespace std;

class Money

{

public:

int rub;

int kop;

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

Money(int m, int s) { rub = m; kop = s; }

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

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

~Money() {};

void clear() { kop = 0; };

int get\_rub() { return rub; }

int get\_kop() { return kop; }

void set\_rub(int m) { rub = m; }

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

Money& operator=(const Money&);

Money& operator++();

Money operator++(int);

Money operator+(const Money&);

Money operator-(const Money&);

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

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

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

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

};

**fire\_work.h**

**#pragma** once

#include "Money.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

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

{

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

if (!stream)

return -1;

int n;

Money p;

cout << "Kolvo ";

cin >> n;

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

{

cin >> p;

stream << p << "\n";

}

stream.close();

return n;

}

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

{

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

if (!stream)

return -1;

Money p;

int i = 0;

while (stream >> p)

{

cout << p << "\n";

i++;

}

stream.close();

return i;

}

int del\_file(const char\* f\_name, int k)

{

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

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

if (!stream)

return -1;

int i = 0;

Money p;

while (stream >> p)

{

if (stream.eof())

break;

i++;

if (i != k)

temp << p;

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

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

{

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

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

if (!stream)

return -1;

Money p;

int i = 0, l = 0;

while (stream >> p)

{

if (stream.eof())break;

i++;

if (i == k)

{

temp << pp;

l++;

}

temp << p;

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;

}

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

{

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

if (!stream)

return -1;

stream << pp;

return 1;

}

int change\_file(const char\* f\_name, int k, Money pp)

{

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

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

if (!stream)

return -1;

Money p;

int i = 0, l = 0;

char x;

while (stream >> p)

{

if (stream.eof())

break;

i++;

if (i == k)

{

cout << p << " - is changing... Continue[y/n]?\n";

cin >> x;

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

break;

temp << pp;

l++;

}

else temp << p;

}

stream.close();

temp.close();

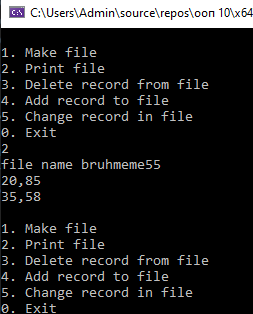
remove(f\_name);

rename("temp", f\_name);

return l;

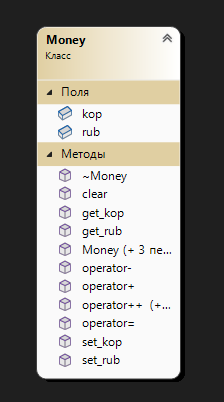
}

**Вывод программы**

****

***Рисунок 1 – вывод программы***

**UML диаграмма**

****

***Рисунок 2 – UML диаграмма***

**Вывод**

Программа выполняет свою задачу.

**Заключение**

Для решения задачи потребовались знания языка программирования, а конкретнее C++. Программа выполняет те условия, что были указаны в постановке задачи и работает без проблем.