МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Факультет: Электротехнический

Кафедра: «Информационные технологии и автоматизированные системы» (ИТАС)

Направление: Разработка информационных систем (РИС)

**О Т Ч Ё Т**   
**о лабораторной работе №10**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Сохранение данных в файл с использованием потоков»

Выполнил:

Студент группы РИС-24-3б

Суяргулов А.О.

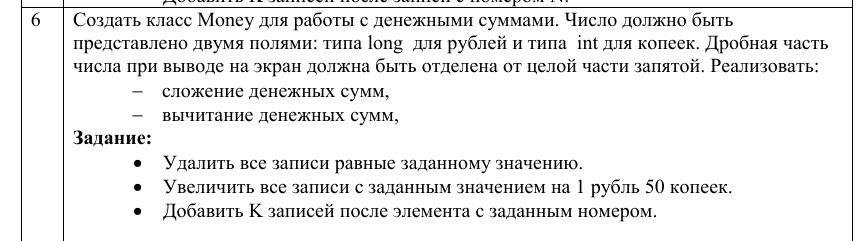
Проверил:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2025

ВВЕДЕНИЕ



ОСНОВНАЯ ЧАСТЬ

**Диаграмма классов:**

**Код:**

**Main:**

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

class Money {

public:

long rubles;

int kopecks;

// Конструктор

Money(long r = 0, int k = 0) {

rubles = r;

kopecks = k;

normalize();

}

// Функция для корректировки рублей и копеек

void normalize() {

while (kopecks >= 100) {

rubles++;

kopecks -= 100;

}

while (kopecks < 0) {

rubles--;

kopecks += 100;

}

}

// Вывод на экран

void print() {

cout << rubles << ",";

if (kopecks < 10) cout << "0";

cout << kopecks << " руб." << endl;

}

// Ввод с клавиатуры

void input() {

cout << "Введите рубли: ";

cin >> rubles;

cout << "Введите копейки: ";

cin >> kopecks;

normalize();

}

};

// Функция для сложения денег

Money addMoney(Money a, Money b) {

Money result;

result.rubles = a.rubles + b.rubles;

result.kopecks = a.kopecks + b.kopecks;

result.normalize();

return result;

}

// Функция для вычитания денег

Money subtractMoney(Money a, Money b) {

Money result;

result.rubles = a.rubles - b.rubles;

result.kopecks = a.kopecks - b.kopecks;

result.normalize();

return result;

}

// Функция для сохранения в файл

void saveToFile(vector<Money> moneyList, string filename) {

ofstream file(filename);

for (Money m : moneyList) {

file << m.rubles << " " << m.kopecks << endl;

}

file.close();

cout << "Данные сохранены в файл!" << endl;

}

// Функция для чтения из файла

vector<Money> readFromFile(string filename) {

vector<Money> moneyList;

ifstream file(filename);

long r;

int k;

while (file >> r >> k) {

moneyList.push\_back(Money(r, k));

}

file.close();

return moneyList;

}

// Главное меню

void showMenu() {

cout << "\n=== Меню ===" << endl;

cout << "1. Добавить новую запись" << endl;

cout << "2. Показать все записи" << endl;

cout << "3. Удалить записи по значению" << endl;

cout << "4. Увеличить записи на 1.50 руб." << endl;

cout << "5. Добавить записи после позиции" << endl;

cout << "6. Выход" << endl;

cout << "Выберите действие: ";

}

int main() {

setlocale(LC\_ALL, "RU");

vector<Money> moneyList;

string filename = "money.txt";

int choice;

do {

showMenu();

cin >> choice;

switch (choice) {

case 1: {

Money m;

m.input();

moneyList.push\_back(m);

saveToFile(moneyList, filename);

break;

}

case 2: {

moneyList = readFromFile(filename);

cout << "\nВсе записи:" << endl;

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

cout << i + 1 << ". ";

moneyList[i].print();

}

break;

}

case 3: {

moneyList = readFromFile(filename);

Money toDelete;

cout << "Введите значение для удаления:" << endl;

toDelete.input();

for (int i = 0; i < moneyList.size(); ) {

if (moneyList[i].rubles == toDelete.rubles &&

moneyList[i].kopecks == toDelete.kopecks) {

moneyList.erase(moneyList.begin() + i);

}

else {

i++;

}

}

saveToFile(moneyList, filename);

cout << "Удаление завершено!" << endl;

break;

}

case 4: {

moneyList = readFromFile(filename);

Money toIncrease;

cout << "Введите значение для увеличения:" << endl;

toIncrease.input();

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

if (moneyList[i].rubles == toIncrease.rubles &&

moneyList[i].kopecks == toIncrease.kopecks) {

moneyList[i].rubles += 1;

moneyList[i].kopecks += 50;

moneyList[i].normalize();

}

}

saveToFile(moneyList, filename);

cout << "Увеличение завершено!" << endl;

break;

}

case 5: {

moneyList = readFromFile(filename);

int pos, k;

cout << "Введите позицию (1-" << moneyList.size() << "): ";

cin >> pos;

cout << "Сколько записей добавить? ";

cin >> k;

if (pos < 1 || pos > moneyList.size()) {

cout << "Неверная позиция!" << endl;

break;

}

for (int i = 0; i < k; i++) {

Money m;

cout << "Введите запись " << i + 1 << ":" << endl;

m.input();

moneyList.insert(moneyList.begin() + pos + i, m);

}

saveToFile(moneyList, filename);

cout << "Добавление завершено!" << endl;

break;

}

case 6:

cout << "Выход из программы." << endl;

break;

default:

cout << "Неверный выбор!" << endl;

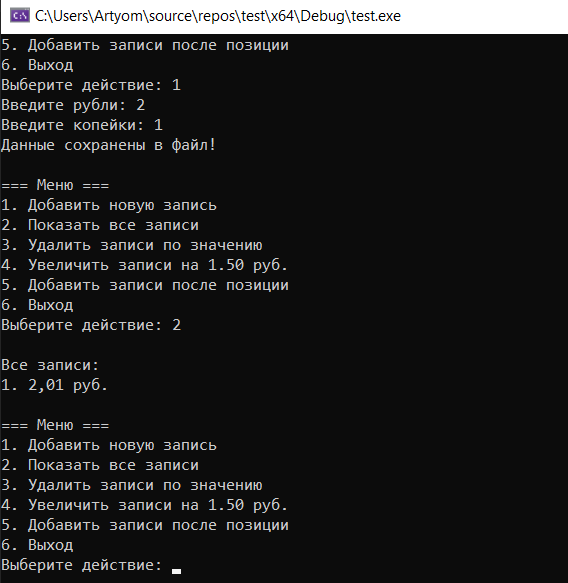
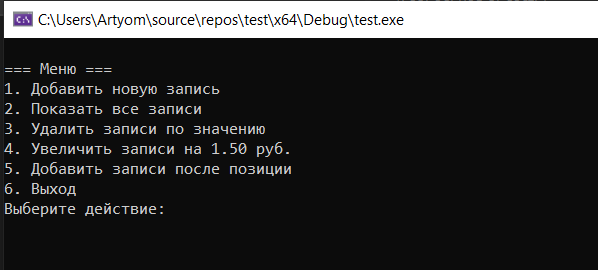
}

} while (choice != 6);

return 0;

}

**Результат работы кода:**

****

**Контрольные вопросы:**

1. Что такое поток? Поток (stream) - это абстракция, представляющая собой последовательность данных, которая может быть прочитана или записана. Он предоставляет унифицированный способ работы с различными источниками и приемниками данных, такими как файлы, консоль, память, сетевые соединения и т.д.

2. Какие типы потоков существуют? В C++ существуют следующие основные типы потоков:

\* istream - для ввода данных (например, cin, ifstream).

\* ostream - для вывода данных (например, cout, ofstream).

\* iostream - для одновременного ввода и вывода данных (например, fstream).

\* stringstream, wstringstream - для работы со строками в памяти.

\* fstream, wfstream - для работы с файлами.

3. Какую библиотеку надо подключить при использовании стандартных потоков? Библиотеку <iostream>.

4. Какую библиотеку надо подключить при использовании файловых потоков? Библиотеку <fstream>.

5. Какую библиотеку надо подключить при использовании строковых потоков? Библиотеку <sstream>.

6. Какая операция используется при выводе в форматированный поток? Операция << (оператор вставки).

7. Какая операция используется при вводе из форматированных потоков? Операция >> (оператор извлечения).

8. Какие методы используются при выводе в форматированный поток?

\* operator<<() (перегруженные операторы вставки для различных типов данных)

\* put() - для вывода одного символа.

\* write() - для вывода блока данных.

\* flush() - для принудительной записи данных в поток.

\* width(), precision(), fill(), setf(), unsetf() - для форматирования вывода.

9. Какие методы используются при вводе из форматированного потока?

\* operator>>() (перегруженные операторы извлечения для различных типов данных)

\* get() - для чтения одного символа или строки.

\* getline() - для чтения строки до указанного разделителя.

\* read() - для чтения блока данных.

\* ignore() - для пропуска указанного количества символов.

\* peek() - для просмотра следующего символа без его извлечения.

10. Какие режимы для открытия файловых потоков существуют?

\* ios::in - Открыть файл для чтения (по умолчанию для ifstream).

\* ios::out - Открыть файл для записи (по умолчанию для ofstream). Если файл существует, его содержимое будет удалено.

\* ios::app - Открыть файл для добавления данных в конец файла.

\* ios::ate - Открыть файл и установить указатель в конец файла.

\* ios::trunc - Если файл существует, его содержимое будет удалено.

\* ios::binary - Открыть файл в двоичном режиме.

\* ios::noreplace - Открытие завершится ошибкой, если файл уже существует.

11. Какой режим используется для добавления записей в файл? ios::app (append).

12. Какой режим (комбинация режимов) используется в конструкторе ifstream file("f.txt")? ios::in (только чтение)

13. Какой режим (комбинация режимов) используется в конструкторе fstream file("f.txt")? ios::in | ios::out (чтение и запись).

14. Какой режим (комбинация режимов) используется в конструкторе ofstream file("f.txt")? ios::out (только запись, с удалением содержимого, если файл существует).

15. Каким образом открывается поток в режиме ios::out | ios::app? ofstream file("f.txt", ios::out | ios::app);

16. Каким образом открывается поток в режиме ios::out | ios::trunc? ofstream file("f.txt", ios::out | ios::trunc); Или просто ofstream file("f.txt"); так как ios::trunc подразумевается по умолчанию для ofstream.

17. Каким образом открывается поток в режиме ios::out | ios::in | ios::trunc? Некорректный пример. ios::out и ios::in не могут быть использованы вместе с ios::trunc. Более подходящий вариант для чтения и записи с удалением существующего содержимого: fstream file("f.txt", ios::out | ios::trunc); или, для того чтобы иметь возможность читать, но при этом изначально очистить файл: fstream file("f.txt", ios::out | ios::in | ios::trunc);

18. Каким образом можно открыть файл для чтения? ifstream file("f.txt"); или ifstream file("f.txt", ios::in);

19. Каким образом можно открыть файл для записи? ofstream file("f.txt"); или ofstream file("f.txt", ios::out);

20. Привести примеры открытия файловых потоков в различных режимах.

\* ifstream inputFile("input.txt"); (чтение)

\* ofstream outputFile("output.txt"); (запись с перезаписью)

\* ofstream appendFile("log.txt", ios::app); (добавление в конец файла)

\* fstream ioFile("data.txt", ios::in | ios::out); (чтение и запись)

\* fstream binaryFile("image.dat", ios::in | ios::out | ios::binary); (чтение и запись в двоичном режиме)

21. Привести примеры чтения объектов из потока.

#include <iostream>

#include <fstream>

#include <string>

struct Person {

std::string name;

int age;

};

int main() {

std::ifstream inputFile("people.txt");

if (inputFile.is\_open()) {

Person person;

while (inputFile >> person.name >> person.age) {

std::cout << "Name: " << person.name << ", Age: " << person.age << std::endl;

}

inputFile.close();

} else {

std::cerr << "Unable to open file for reading" << std::endl;

}

return 0;

}

22. Привести примеры записи объектов в поток.

#include <iostream>

#include <fstream>

#include <string>

struct Person {

std::string name;

int age;

};

int main() {

std::ofstream outputFile("people.txt");

if (outputFile.is\_open()) {

Person person1 = {"Alice", 30};

Person person2 = {"Bob", 25};

outputFile << person1.name << " " << person1.age << std::endl;

outputFile << person2.name << " " << person2.age << std::endl;

outputFile.close();

} else {

std::cerr << "Unable to open file for writing" << std::endl;

}

return 0;

}

23. Сформулировать алгоритм удаления записей из файла.

1. Открыть файл для чтения.

2. Прочитать все записи из файла и сохранить их в структуру данных (например, вектор).

3. Закрыть файл.

4. Определить, какие записи нужно удалить, и удалить их из структуры данных.

5. Открыть файл для записи в режиме ios::trunc (чтобы очистить его).

6. Записать оставшиеся записи из структуры данных обратно в файл.

7. Закрыть файл.

24. Сформулировать алгоритм добавления записей в файл.

1. Открыть файл для записи в режиме ios::app (append).

2. Записать новую запись в файл.

3. Закрыть файл.

25. Сформулировать алгоритм изменения записей в файле.

1. Открыть файл для чтения.

2. Прочитать все записи из файла и сохранить их в структуру данных (например, вектор).

3. Закрыть файл.

4. Определить запись, которую нужно изменить, и внести изменения в соответствующую запись в структуре данных.

5. Открыть файл для записи в режиме ios::trunc (чтобы очистить его).

6. Записать все записи (включая измененную) из структуры данных обратно в файл.

7. Закрыть файл.

**Ссылка на GitHub:**