Пермский национальный исследовательский политехнический университет

**Лабораторная работа №10**

**Дисциплина: Основы алгоритмизации и**

**программирования**

**Вариант № 9**

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

Выполнил: Зайченко Никита Геннадьевич

Проверила: Доцент кафедры ИТАС Полякова О. А.

Пермь 2022

**Содержание отчета**

1) Постановка задачи (общая и конкретного варианта).

2) Описание пользовательского класса.

3) Перегруженные операции пользовательского класса.

4) Определение функций для работы с файлом (создание, вывод, удаление,

добавление, изменение).

5) Функция main().

6) Объяснение результатов работы программы.

7) Ответы на контрольные вопросы.

**Постановка задачи (общая и конкретного варианта)**

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Разработка программы, в которой данные сохраняются в файле, корректируются и выводятся из файла на печать. Работа с файлом осуществляется с использованием потоковых классов.3) 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:

A) Вычитание дробного числа из суммы

B) операции сравнения (==, !=). Задание:

C) Удалить все записи равные заданному значению.

D) Уменьшить все записи с заданным значением на 1 рубль 50 копеек. Значение интервала не должно быть меньше 0 рублей 0 копеек.

E) Добавить K записей после элемента с заданным значением.

**Описание пользовательского класса.**

class Money

{

long rub; int cop;

public:

Money() { rub = 0; cop = 0; }; //конструктор без параметров

Money(long r, int c) { rub = r; cop = c; } //конструктор с параметрами

Money(const Money& m) { rub = m.rub; cop = m.cop; } //конструктор копирования

~Money() {}; //деструктор

int get\_rub() { return rub; } //селектор

int get\_cop() { return cop; } //селектор

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

void set\_cop(int c) { cop = c; } //модификатор

//перегруженные операции

Money& operator=(const Money&); //перегрузка операции присваивания

Money& operator++(); //перегрузка префиксной операции инкремент

Money operator+(const Money&); //перегрузка бинарной операции сложения

Money operator-(const Money&); //перегрузка бинарной операции вычитания

bool operator==(const Money&); //перегрузка операции сравнения

bool operator!=(const Money&); //перегрузка операции сравнения

//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);

};

****

**Перегруженные операции пользовательского класса.**

//перегрузка операции присваивания

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

{

//проверка на самоприсваивание

if (&m == this) return \*this;

rub = m.rub;

cop = m.cop;

return \*this;

}

//перегрузка префиксной операции инкремент

Money& Money::operator++()

{

long e = rub + (cop / 100);

rub = e;

cop = cop % 100;

return \*this;

}

//перегрузка бинарной операции сложения

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

{

long rubs = rub + m.rub;

int cops = cop + m.cop;

if (cops >= 100) { // обработка случая, когда cop >= 100

rubs += 1;

cops -= 100;

}

Money p(rubs, cops);

return p;

}

//перегрузка бинарной операции вычитания

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

{

double total = rub + cop / 100.0 - (m.rub + m.cop / 100.0);

long rub = static\_cast<long>(total); // static\_cast: оператор преобразования типов данных

int cop = static\_cast<int>((total - rub) \* 100);

if (cop < 0) { // обработка случая, когда cop < 0

rub -= 1;

cop += 100;

}

if (rub < 0) { // обработка случая, когда rub < 0

rub = 0;

cop = 0;

}

Money p(rub, cop);

return p;

}

//перегрузка операции сравнения ==

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

{

if (rub == m.rub && cop == m.cop) { return true; }

else { return false; }

}

//перегрузка операции сравнения !=

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

{

if (rub != m.rub || cop != m.cop) { return true; }

else { return false; }

}

//перегрузка глобальной функции-операции ввода

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

{

cout << "Введите рубли = "; in >> m.rub;

cout << "Введите копейки = "; in >> m.cop;

cout << endl;

return in;

}

//перегрузка глобальной функции-операции вывода

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

{

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

}

//дружественные функции для работы с файловыми потоками

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

{

fin >> m.rub;

fin >> m.cop;

return fin;

}

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

{

fout << m.rub << "\n" << m.cop << "\n";

return fout;

}

**Определение функций для работы с файлом (создание, вывод, удаление,**

**добавление, изменение).**

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

{

fstream stream(f\_name, ios::out | ios::trunc);//открыть для записи

if (!stream) return -1;//ошибка открытия файла

int n;

Money m;

cout << "Введите количество объектов\n"; cin >> n;

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

{

cin >> m; ++m;//ввод атрибутов объекта из стандартного потока

stream << m << "\n";//запись объекта в файловый поток

}

stream.close();//закрыть поток

return n;//вернуть количество записанных объектов

}

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

{

fstream stream(f\_name, ios::in);//открыть для чтения

if (!stream)return -1;//ошибка открытия файла

Money m; int i = 0;

while (stream >> m)

{

cout << m << "\n";

i++;

}

stream.close();

return i;

}

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

{

fstream temp("temp", ios::out);//открыть для записи

fstream stream(f\_name, ios::in);//открыть для чтения

if (!stream)return -1;//ошибка открытия файла

int i = 0; Money m;

while (stream >> m)//пока нет конца файла выполняем чтение объекта

{

//если прочитан признак конца файла,то выход из цикла

if (stream.eof())break;

i++;

if (m.get\_rub() != r && m.get\_cop() != c)

temp << m;

}

//закрыть файлы

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

remove(f\_name);//удалить старый файл

rename("temp", f\_name);// переименовать temp

return i;//количество прочитанных

}

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

{

fstream temp("temp", ios::out);//открыть для записи

fstream stream(f\_name, ios::in);//открыть для чтения

if (!stream) return -1;//ошибка открытия файла

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

while (stream >> m)

{

if (stream.eof()) break;

i++;

temp << m; //записать текущую запись в temp

if (m.get\_rub() == r && m.get\_cop() == c)

{

temp << pp; //записать новую запись после текущей записи

l++;

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;//количество добавленных

}

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

{

fstream stream(f\_name, ios::app);//открыть для добавления

if (!stream)return -1;//ошибка открытия файла

stream << pp; //записать новую запись

return 1;

}

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

{

fstream temp("temp", ios::out);//открыть для записи

fstream stream(f\_name, ios::in);//открыть для чтения

if (!stream)return -1;//ошибка открытия файла

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

/\*char x;\*/

while (stream >> m)

{

if (stream.eof())break;

i++;

if (m.get\_rub() == r && m.get\_cop() == c)

{

m = m - pp;

temp << m;

if (m.get\_rub() < 0) m.set\_rub(0);

if (m.get\_cop() < 0) m.set\_cop(0);

l++;

}

else temp << m;

}

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

remove(f\_name);

rename("temp", f\_name);

return l;//количество измененных элементов

}

**Функция main().**

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Money m, m1;

int k, c;

char file\_name[30];

do

{ //Меню

cout << "\n1. Создание файла";

cout << "\n2. Печать файла";

cout << "\n3. Удалить запись из файла";

cout << "\n4. Добавить запись в файл";

cout << "\n5. Изменить запись в файле";

cout << "\n0. Выход\n";

cin >> c;

switch (c)

{

case 1: cout << "Введите название файла\n"; cin >> file\_name;

k = make\_file(file\_name);

if (k < 0)cout << "Не удается создать файл\n";

break;

case 2: cout << "Введите название файла\n"; cin >> file\_name;

k = print\_file(file\_name);

if (k == 0)cout << "Пустой файл\n";

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

break;

case 3: cout << "Введите название файла\n"; cin >> file\_name;

long rub; int cop; cout << "Введите значение рублей и копеек\n"; cin >> rub; cin >> cop;

cout << rub << "," << cop << endl;

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

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

else cout << "Запись удалена\n";

break;

case 4: cout << "Введите название файла\n";

cin >> file\_name;

long rubs; int cops; cout << "Введите значение рублей и копеек\n"; cin >> rubs; cin >> cops;

cout << rubs << "," << cops << endl;

cout << "Введите сколько элементов добавлять\n";

int s; cin >> s;

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

{

cin >> m1;

k = add\_file(file\_name, rubs, cops, m1);

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

if (k == 0) k = add\_end(file\_name, m1);

}

cout << "Добавлено " << s << " элементов\n";

break;

case 5: cout << "Введите название файла\n";

cin >> file\_name;

long rubss; int copss; cout << "Введите значение рублей и копеек\n"; cin >> rubss; cin >> copss;

cout << rubss << "," << copss << endl;

Money m2(1,50);

k = change\_file(file\_name, rubss, copss, m2);

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

if (k == 0) cout << "Пустой файл\n";

break;

}

} while (c != 0);

}

**Объяснение результатов работы программы.**

Данная программа демонстрирует, следующий функционал:

1) Создание файла;

2) Печать файла;

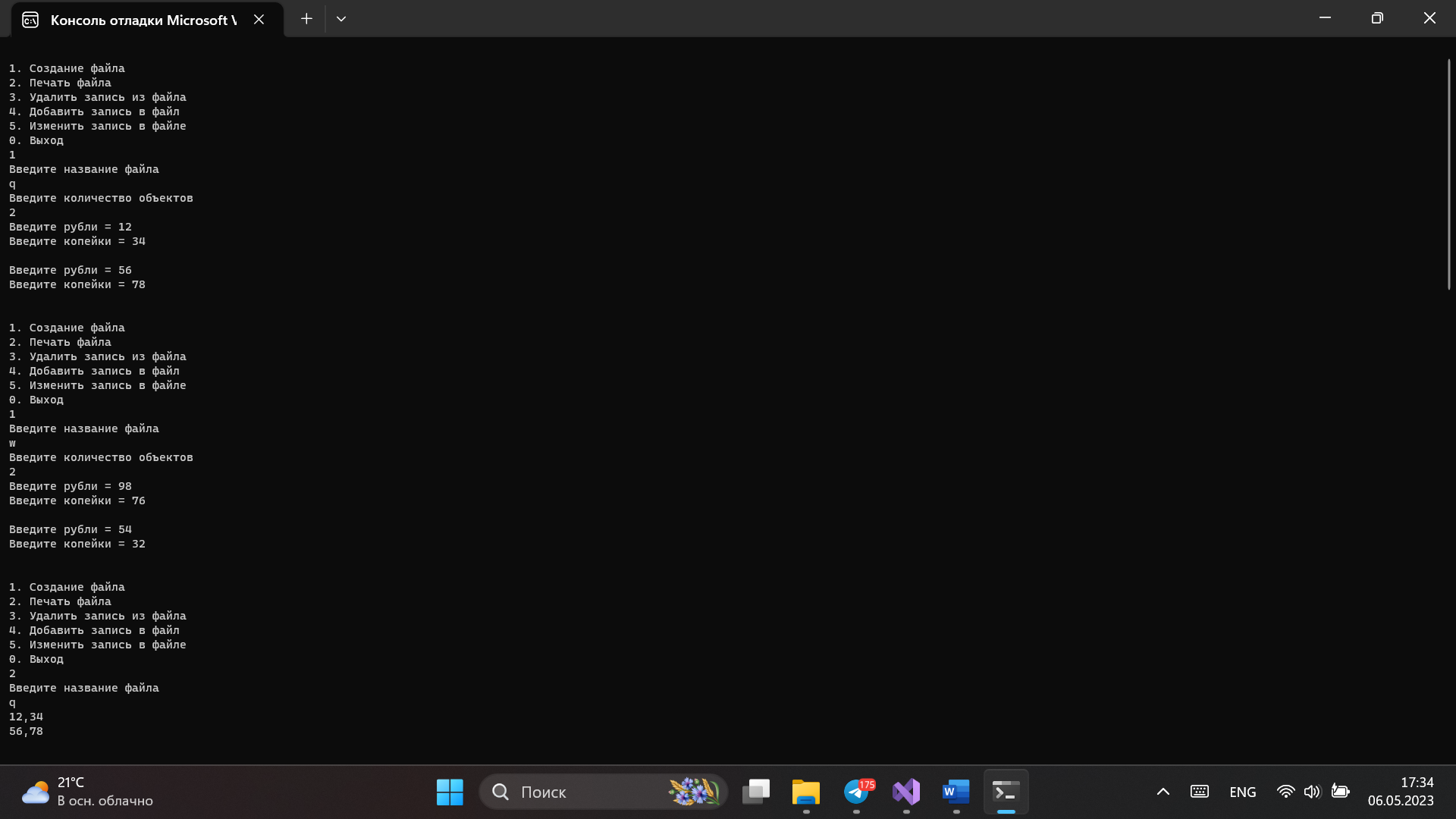
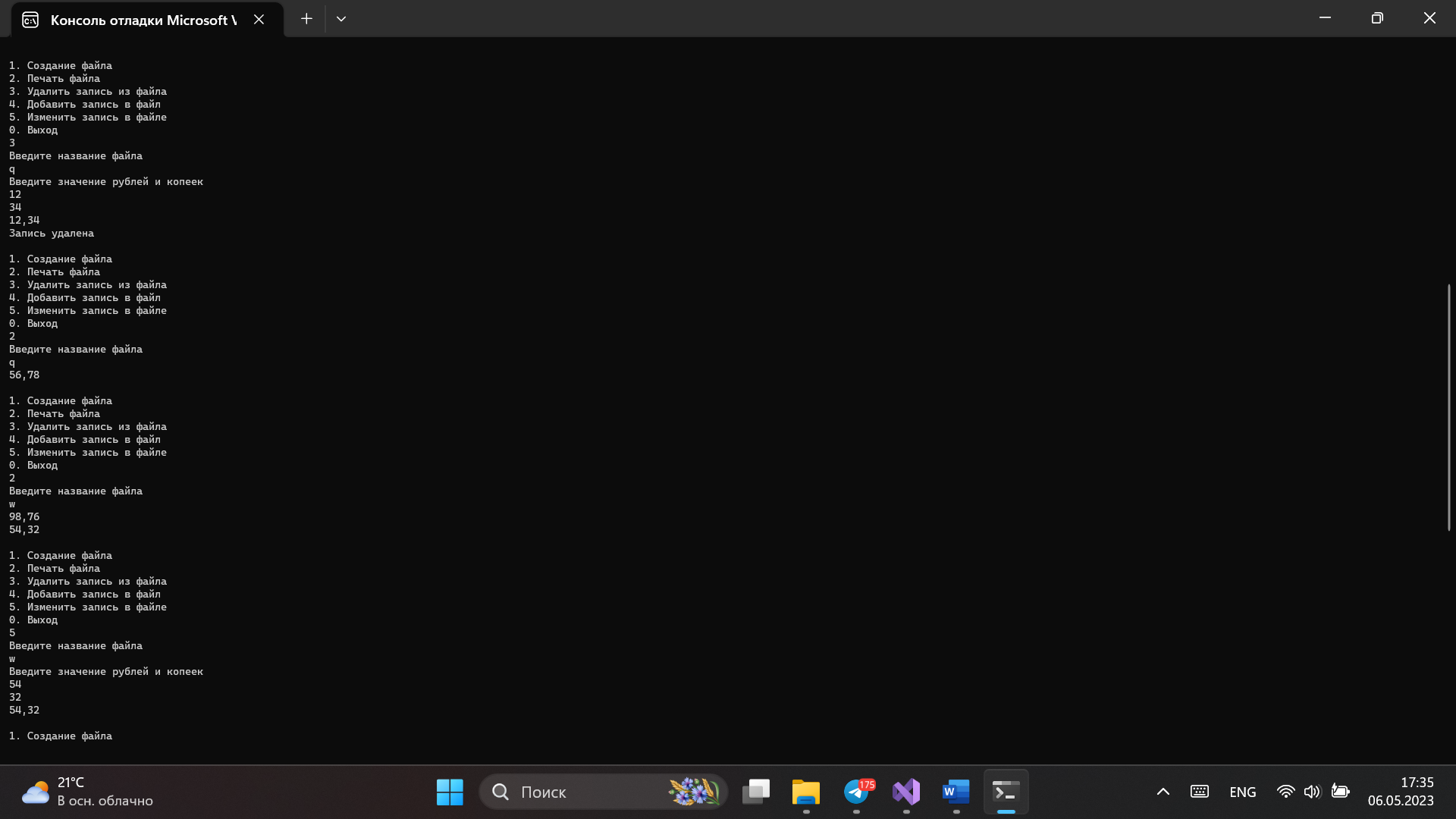
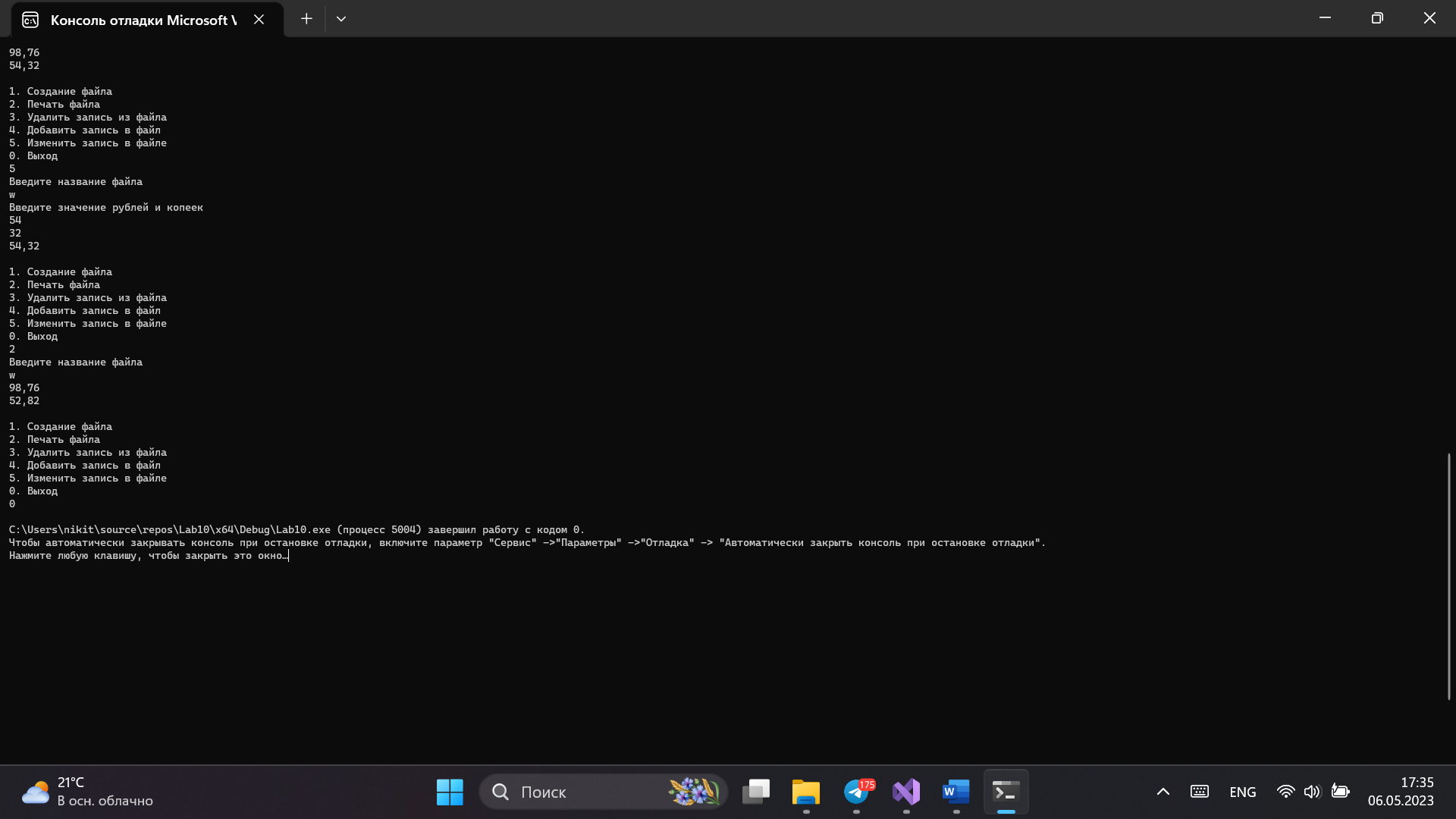
3) Удалить запись из файла;

4) Добавить запись в файл;

5) Изменить запись в файле;

6) Выход;

Так же представлен пользовательский класс Money, для записи его в файлы, все функции работы с файлами представлены отдельным заголовочным файлом.

**Ответы на контрольные вопросы**

**1. Что такое поток?**

**2. Какие типы потоков существуют?**

**3. Какую библиотеку надо подключить при использовании стандартных потоков?**

**4. Какую библиотеку надо подключить при использовании файловых потоков?**

**5. Какую библиотеку надо подключить при использовании строковых потоков?**

**6. Какая операция используется при выводе в форматированный поток?**

**7. Какая операция используется при вводе из форматированных потоков?**

**8. Какие методы используются при выводе в форматированный поток?**

**9. Какие методы используется при вводе из форматированного потока?**

**10. Какие режимы для открытия файловых потоков существуют?**

**11. Какой режим используется для добавления записей в файл?**

**12. Какой режим (комбинация режимов) используется в конструкторе**

**ifstream file(“f.txt”)?**

**13. Какой режим (комбинация режимов) используется в конструкторе**

**fstream file(“f.txt”)?**

**14. Какой режим (комбинация режимов) используется в конструкторе**

**ofstream file(“f.txt”)?**

**15. Каким образом открывается поток в режиме ios::out|ios::app?**

**16. Каким образом открывается поток в режиме ios::out |ios::trunc?**

**17. Каким образом открывается поток в режиме ios::out |ios::in|ios::trunk?**

**18. Каким образом можно открыть файл для чтения?**

**19. Каким образом можно открыть файл для записи?**

**20. Привести примеры открытия файловых потоков в различных режимах.**

**21. Привести примеры чтения объектов из потока.**

**22. Привести примеры записи объектов в поток.**

**23. Сформулировать алгоритм удаления записей из файла.**

**24. Сформулировать алгоритм добавления записей в файл.**

**25. Сформулировать алгоритм изменения записей в файле.**

1. Поток (Stream) - это последовательный набор данных.

2. Существует много типов потоков в зависимости от их назначения и свойств. Например: стандартные потоки (stdin, stdout, stderr), файловые потоки, сетевые потоки, строковые потоки, бинарные потоки и т.д.

3. Для использования стандартных потоков не нужно подключать дополнительную библиотеку, так как они уже определены в стандартной библиотеке языка.

4. Для использования файловых потоков, нужно подключить библиотеку <fstream>.

5. Для использования строковых потоков, нужно подключить библиотеку <sstream>.

6. Для вывода в форматированный поток используется операция "<<".

7. Для ввода из форматированных потоков используется операция ">>".

8. При выводе в форматированный поток, часто используются методы, такие как setprecision(), setw(), setfill() и др.

9. При вводе из форматированного потока, также могут использоваться методы, такие как getline(), ignore(), peek() и др.

10. Существует несколько режимов для открытия файловых потоков:

- std::ios::in - открытие файла для чтения

- std::ios::out - открытие файла для записи

- std::ios::binary - открытие файла в бинарном режиме

- std::ios::app - добавление данных в конец файла (append)

- std::ios::trunc - очистка файла при его открытии.

Можно комбинировать эти режимы, например, std::ios::in | std::ios::binary для открытия файла в двоичном режиме для чтения.

11. Режим ios::app используется для добавления записей в файл, т.е. новые данные будут добавляться в конец файла, не удаляя старые записи.

12. В конструкторе ifstream file("f.txt"), используется режим ios::in. Он открывает файл только для чтения.

13. В конструкторе fstream file("f.txt"), используется режим ios::in | ios::out, который открывает файл и для чтения и для записи.

14. В конструкторе ofstream file("f.txt"), используется режим ios::out, который открывает файл только для записи. Если файл существует, то он будет перезаписан новыми данными, если нет, то он будет создан.

15. Для открытия потока в режиме ios::out|ios::app нужно выполнить следующий код:

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

Это откроет файл для записи и добавит новые данные в конец файла, не удаляя старые записи.

16. Для открытия потока в режиме ios::out|ios::trunc нужно выполнить следующий код:

ofstream file("f.txt", ios::out|ios::trunc);

Это откроет файл для записи и удалит все старые данные в файле.

17. Для открытия потока в режиме ios::out|ios::in|ios::trunc нужно выполнить следующий код:

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

Это откроет файл для записи, чтения и удаления всех старых данных.

18. Для открытия файла для чтения нужно создать объект ifstream с именем файла и передать имя файла в качестве параметра. Например:

ifstream file("input.txt");

19. Для открытия файла для записи нужно создать объект ofstream с именем файла и передать имя файла в качестве параметра. Например:

ofstream file("output.txt");

20. Примеры открытия файловых потоков в различных режимах:

ifstream file("input.txt"); // для чтения

ofstream file("output.txt"); // для записи

fstream file("file.txt", ios::in); // для чтения

fstream file("file.txt", ios::out|ios::app); // для добавления записей в конец файла

fstream file("file.txt", ios::in|ios::out); // для чтения и записи

21. Примеры чтения объектов из потока:

string str;

ifstream file("input.txt");

file >> str; // чтение первого слова из файла

int x;

file >> x; // чтение следующего целого числа из файла

22. Примеры записи объектов в поток:

ofstream file("output.txt");

file << "Hello world!"; // запись строки в файл

int x = 42;

file << "The answer is " << x; // запись нескольких элементов в файл

23. Алгоритм удаления записей из файла:

1. Открыть файл в режиме ios::in|ios::out.

2. Проверить, было ли закрытие файла успешным.

3. Создать временный файл или использовать буфер в памяти для хранения новых данных.

4. Считать данные из исходного файла и записывать их в новый файл/буфер, пропустив записи, которые нужно удалить.

5. Закрыть исходный файл и временный файл/буфер.

6. Удалить исходный файл.

7. Переименовать временный файл/буфер в имя исходного файла.

24. Алгоритм добавления записей в файл:

1. Открыть файл в режиме ios::out | ios::app.

2. Проверить, было ли закрытие файла успешным.

3. Дописать новые записи в конец файла.

4. Закрыть файл.