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

# По теме **«Сохранение данных в файле с использованием потоков»**

Вариант №10

Выполнял:

студент группы РИС-24-1б

Морозова Н.С.

Проверял:

Доцент кафедры ИТАС

Полякова О.А.

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.
5. Написать функцию для изменения объектов в потоке в соответствии с заданием варианта.
6. Написать функцию для добавления объектов в поток в соответствии с заданием варианта.
7. Для выполнения задания выполнить перегрузку необходимых операций.
8. Для вызова функций в основной программе предусмотреть меню.

Задание:

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:

* операции сравнения (==, !=).
* вычитание копеек (--) (постфиксная и префиксная формы)

Задание:

* Удалить все записи большие заданного значения.
* Увеличить все записи с заданным значением на 1 рубль 50 копеек.
* Добавить K записей после записи с номером N.

Анализ задачи:

1. Класс Money реализован как в лабораторной работе 7 (только без итератора).
2. Добавлены перегрузки операций сравнения и вычитания копеек в файле money.h. Функции для работы с потоками описаны в файле Lab\_10.cpp.
3. Также для работы с файлами были описаны функции добавления фиксированной суммы и метод для преобразования строки в Money.

Решение

Код

Файл Lab\_10.cpp

#include "money.h"

#include <Windows.h>

#include <fstream>

#include <iostream>

#include <string>

#include <sstream>

#include <cstdio>

using namespace std;

int make\_file(string filename)

{

// Открываем файл для записи

ofstream fout(filename);

if (!fout.is\_open()) {

cerr << "Ошибка открытия файла " << endl;

return -1;

}

int K;

Money m;

do {

std::cout << "Введите количество денежных сумм: ";

cin >> K;

} while (K <= 0);

// Запись сумм в файл

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

std::cout << "\nСумма #" << i + 1 << ":\n";

while (true) {

std::cout << "Введите сумму" << endl;

if (cin >> m) break;

cerr << "Неверный формат! Повторите ввод." << endl;

cin.clear();

cin.ignore(100000, '\n');

}

fout << m << endl;

}

fout.close();

return K;

}

int print\_file(string filename) {

ifstream fin(filename);

if (!fin.is\_open()) {

cerr << "Ошибка: не удалось открыть файл " << endl;

return -1;

}

string sum;

int K = 0;

std::cout << "\nСодержимое файла " << endl;

std::cout << "------------------------\n";

while (fin >> sum) {

cout << sum << endl;

K++;

}

fin.close();

std::cout << "------------------------\n";

std::cout << "Всего сумм: " << K << "\n\n";

return K;

}

int del\_greater(string filename)

{

// Создаем временный файл для результатов

ofstream temp\_fout("C:\\Users\\user\\Desktop\\C++\\Лаба\_10\\temp.txt");

if (!temp\_fout) {

cerr << "Ошибка создания временного файла!" << endl;

return -1;

}

// Открываем исходный файл

ifstream fin(filename);

if (!fin) {

cerr << "Ошибка открытия файла " << filename << endl;

temp\_fout.close();

return -1;

}

Money maxim;

std::cout << "Введите пороговую сумму:\n";

cin >> maxim;

string line;

int total = 0;

int kept = 0;

while (getline(fin, line)) {

total++;

try {

Money current = Money::from\_string(line);

if (!(current > maxim)) {

temp\_fout << line << endl;

kept++;

}

}

catch (...) { cerr << "Ошибка формата в строке: " << line << endl; }

}

temp\_fout.close();

fin.close();

// Заменяем исходный файл временным

if (remove(filename.c\_str()) != 0) {

cerr << "Ошибка удаления исходного файла!" << endl;

return -1;

}

if (rename("C:\\Users\\user\\Desktop\\C++\\Лаба\_10\\temp.txt", filename.c\_str()) != 0) {

cerr << "Ошибка переименования временного файла!" << endl;

return -1;

}

std::cout << "Удалено сумм: " << (total - kept) << endl;

return kept; // Возвращаем количество оставшихся сумм

}

int add\_file(string filename)

{

//временный файл

ofstream temp\_fout("C:\\Users\\user\\Desktop\\C++\\Лаба\_10\\temp.txt");

if (!temp\_fout) {

cerr << "Ошибка создания временного файла!" << endl;

return -1;

}

ifstream fin(filename); // исходный файл

if (!fin) {

cerr << "Ошибка открытия файла " << filename << endl;

temp\_fout.close();

return -1;

}

int N, K;

std::cout << "Введите номер, после которого нужно вставить значения: ";

cin >> N;

std::cout << "Введите количество сумм для добавления: ";

cin >> K;

if (N < 1 || K < 1) {

cerr << "Некорректные параметры вставки!" << endl;

return -1;

}

string line;

int posit = 0;

int total\_added = 0;

// Чтение и обработка

while (getline(fin, line)) {

temp\_fout << line << endl;

posit++;

// Вставка после указанной позиции

if (posit == N) {

std::cout << "Введите " << K << " сумм для добавления:\n";

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

Money m;

cin >> m;

temp\_fout << m << endl;

total\_added++;

}

}

}

// Проверка существования указанной позиции

if (N > posit) {

cerr << "Файл содержит только " << posit << " записей!" << endl;

temp\_fout.close();

fin.close();

return -1;

}

temp\_fout.close();

fin.close();

if (remove(filename.c\_str()) != 0) { // Заменяем исходный файл

cerr << "Ошибка удаления исходного файла!" << endl;

return -1;

}

if (rename("C:\\Users\\user\\Desktop\\C++\\Лаба\_10\\temp.txt", filename.c\_str()) != 0) {

cerr << "Ошибка переименования временного файла!" << endl;

return -1;

}

std::cout << "Успешно добавлено " << total\_added << " сумм." << endl;

return posit + total\_added; // Общее количество записей

}

int add\_sum(string filename)

{

ifstream fin(filename);

ofstream temp("C:\\Users\\user\\Desktop\\C++\\Лаба\_10\\temp\_.txt");

if (!fin || !temp) {

cerr << "Ошибка открытия файлов!" << endl;

return -1;

}

string line;

int count = 0;

while (getline(fin, line)) {

if (line.empty()) continue; // Пропуск пустых строк

try {

Money m = Money::from\_string(line);

m.fixed\_sum();

temp << m << endl;

count++;

}

catch (...) {

cerr << "Ошибка формата в строке: " << line << endl;

temp << line << endl; // Сохраняем оригинал, если не удалось обработать

}

}

fin.close();

temp.close();

if (remove(filename.c\_str()) != 0) {

cerr << "Ошибка удаления исходного файла!" << endl;

return -1;

}

if (rename("C:\\Users\\user\\Desktop\\C++\\Лаба\_10\\temp\_.txt", filename.c\_str()) != 0) {

cerr << "Ошибка переименования временного файла!" << endl;

return -1;

}

std::cout << "Обновлено сумм: " << count << endl;

return count;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string filename = "C:\\Users\\user\\Desktop\\C++\\Лаба\_10\\MoneySumm.txt";

int code;

bool file\_exists = false;

do {

std::cout << "\nМеню:\n"

<< "1. Создать файл\n"

<< "2. Вывести содержимое файла\n"

<< "3. Удалить суммы БОЛЬШЕ заданного\n"

<< "4. Добавить 1 рубль 50 копеек ко всем суммам\n"

<< "5. Добавить K сумм после указанной позиции\n"

<< "0. Выход\n"

<< "> ";

cin >> code;

cin.ignore(1);

switch (code) {

case 1: {

std::cout << "Имя файла: " << filename << endl;

int result = make\_file(filename);

if (result > 0) {

file\_exists = true;

std::cout << "Файл создан. Записей: " << result << endl;

}

break;

}

case 2: {

if (!file\_exists) { cerr << "Файл не создан! Сначала выберите пункт 1." << endl; break; }

print\_file(filename); break;

}

case 3: {

if (!file\_exists) { cerr << "Файл не создан! Сначала выберите пункт 1." << endl; break; }

del\_greater(filename); break;

}

case 4: {

if (!file\_exists) { cerr << "Файл не создан! Сначала выберите пункт 1." << endl; break; }

add\_sum(filename); break;

}

case 5: {

if (!file\_exists) { cerr << "Файл не создан! Сначала выберите пункт 1." << endl; break; }

add\_file(filename); break;

}

case 0: {

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

break;

}

default: { std::cout << "Неверный код операции!" << endl; }

}

} while (code != 0);

// реализация остальных перегруженных функций

Money M1;

cout << "Введите 1 денежную сумму" << endl;

cin >> M1;

cout << M1 << endl;

Money M2;

cout << "\nВведите 2 денежную сумму" << endl;

cin >> M2;

cout << M2 << endl;

if (M1 == M2)

cout << "Значения равны" << endl;

if (M1 != M2)

cout << "Значения различны" << endl;

--M1;

cout << "\nПрефиксное вычитание: " << M1 << endl;

Money M3;

M3 = M2--;

cout << "Постфиксное вычитание: " << M3 << " -> " << M2 << endl;

return 0;

}

Файл money.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Money

{

private:

long rub;

int kop;

public:

Money(void);

Money(long, int);

Money(const Money&);

virtual ~Money(void) {};

Money& operator=(const Money&);

Money& operator--();

Money operator--(int);

bool operator>(const Money& other) const {

if (rub > other.rub) return true;

if (rub == other.rub && kop > other.kop) return true;

return false;

}

//операции сравнения

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

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

}

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

return (rub != m.rub) or (kop != m.kop);

}

// Добавление фиксированной суммы

void fixed\_sum() {

kop += 50;

rub += 1;

if (kop >= 100) {

rub += kop / 100;

kop %= 100;

}

}

//перегруженные операции ввода-вывода

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

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

// Метод для преобразования строки в Money

static Money from\_string(const string& str)

{

size\_t comma\_pos = str.find(',');

if (comma\_pos == string::npos) {

return Money(stol(str), 0);

}

long r = stol(str.substr(0, comma\_pos));

int k = stoi(str.substr(comma\_pos + 1));

return Money(r, k);

}

};

Файл money.cpp

#include "money.h"

#include <iostream>

using namespace std;

Money::Money(void) {

rub = 0;

kop = 0;

}

Money::Money(long R, int K) {

rub = R;

kop = K;

}

Money::Money(const Money& m) {

rub = m.rub;

kop = m.kop;

}

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

{

if (this == &m)

return \*this;

rub = m.rub;

kop = m.kop;

return\*this;

}

Money& Money::operator--()

{

if (--kop < 0) { //если вычитаем из 0 копеек

kop += 100;

rub--;

}

return \*this;

}

Money Money::operator--(int)

{

Money tmp = \*this;

--(\*this); // используем префиксную версию

return tmp;

}

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

{

if (m.kop < 10) out << m.rub << ",0" << m.kop;

else out << m.rub << "," << m.kop;

return out;

}

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

{

do {

cout << "рубли: ";

in >> m.rub;

} while (m.rub < 0);

do {

cout << "копейки: ";

in >> m.kop;

} while (m.kop < 0);

if (m.kop >= 100) {

m.rub += m.kop / 100;

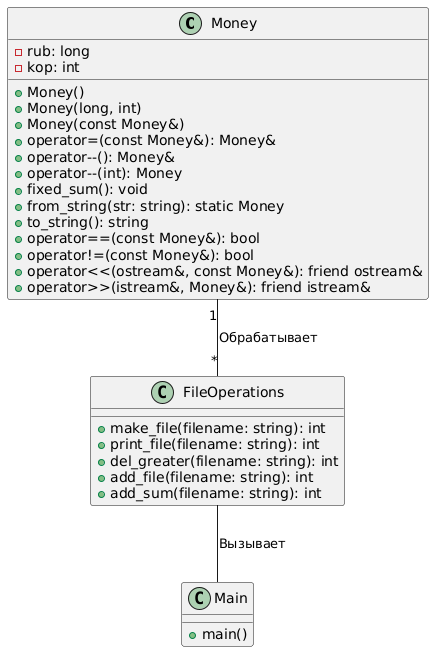
m.kop = m.kop % 100;

}

return in;

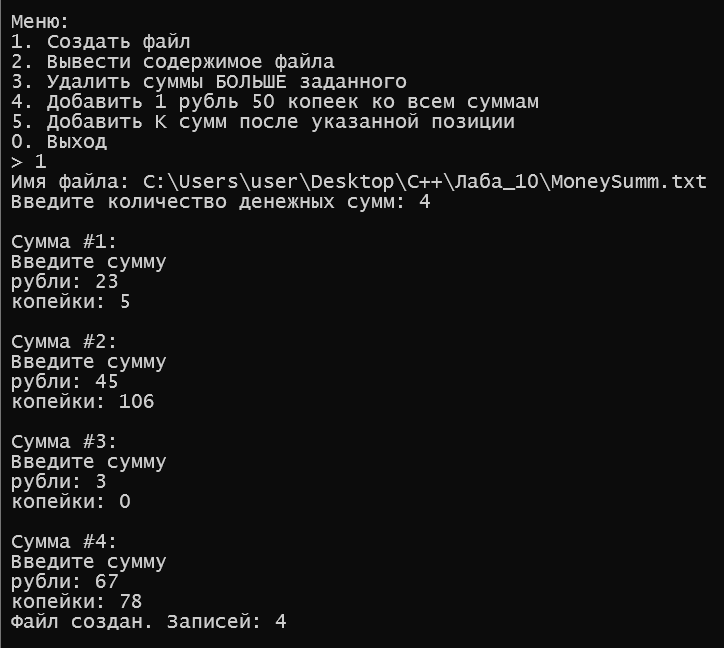
}

UML-диаграмма

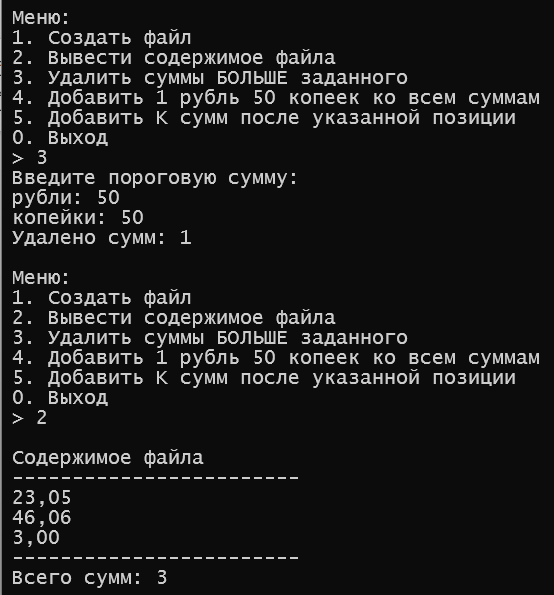


Результат работы программы:

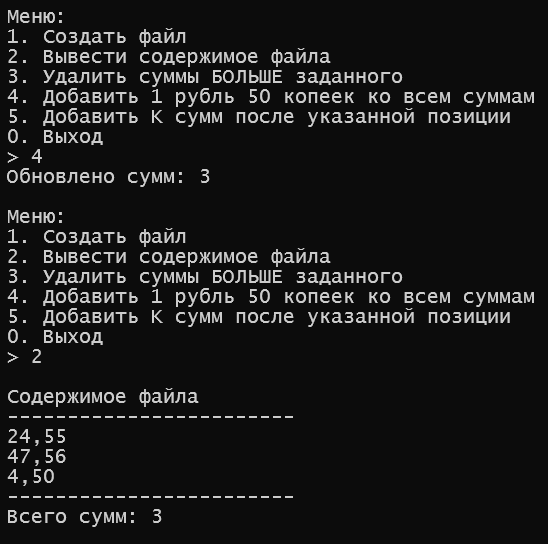
Создание файла



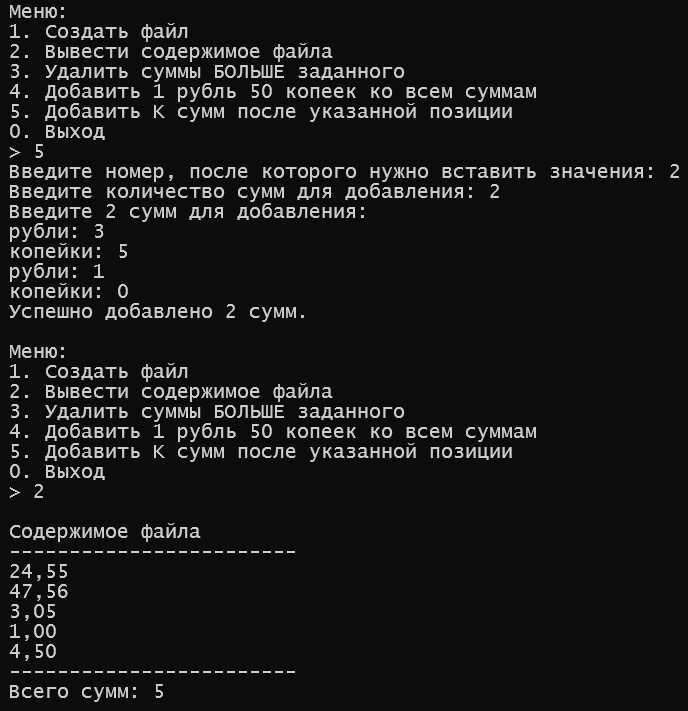
Удаление и вывод элементов



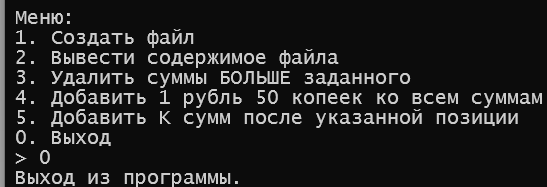
Добавление 1 рубля и 50 копеек



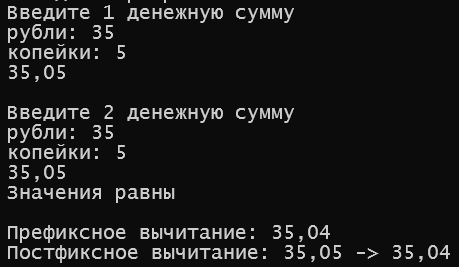
Добавление элементов после заданного номера

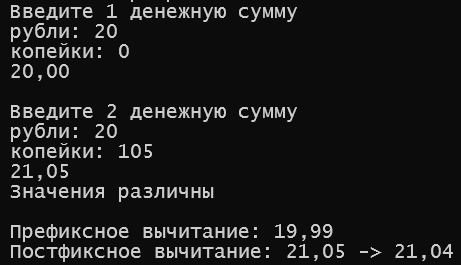


Выход из программы



Реализация остальных перегрузок





Контрольные вопросы:

1. Что такое поток?

Поток (stream) — это абстракция для последовательного ввода/вывода данных, представляющая источник или приёмник информации (консоль, файл, строка и т.д.).

1. Какие типы потоков существуют?

Стандартные (cin, cout, cerr, clog);

Файловые (ifstream, ofstream, fstream);

Строковые (istringstream, ostringstream, stringstream).

1. Какую библиотеку надо подключить при использовании стандартных потоков?

#include <iostream>

1. Какую библиотеку надо подключить при использовании файловых потоков?

#include <fstream>

1. Какую библиотеку надо подключить при использовании строковых потоков?

#include <sstream>

1. Какая операция используется при выводе в форматированный поток?

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

1. Какая операция используется при вводе из форматированных потоков?

Оператор >> (например, cin >> x;).

1. Какие методы используются при выводе в форматированный поток?

put() — вывод одного символа;

write() — вывод блока данных;

flush() — принудительная очистка буфера.

1. Какие методы используется при вводе из форматированного потока?

get() — чтение одного символа;

getline() — чтение строки;

read() — чтение блока данных;

ignore() — пропуск символов.

1. Какие режимы для открытия файловых потоков существуют?

|  |  |
| --- | --- |
| ios::in | Чтение |
| ios::out | Запись (с удалением содержимого) |
| ios::app | Добавление в конец файла |
| ios::ate | Открытие с переходом в конец файла |
| ios::binary | Бинарный режим |
| ios::trunc | Очистка файла при открытии |

1. Какой режим используется для добавления записей в файл?

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

1. Какой режим используется в конструкторе ifstream file(“f.txt”)?

ifstream file("f.txt", ios::in);

1. Какие режимы используются в конструкторе fstream file(“f.txt”)?

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

1. Какие режимы используются в конструкторе ofstream file(“f.txt”)?

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

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

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

Открывает файл для записи в конец.

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

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

Открывает файл для записи, предварительно очищая его содержимое.

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

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

Открывает файл для чтения и записи, очищая его при открытии (аналог создания нового файла).

1. Каким образом можно открыть файл для чтения?

ifstream input("data.txt"); // Режим по умолчанию: ios::in

// Или явно:

ifstream input("data.txt", ios::in);

1. Каким образом можно открыть файл для записи?

ofstream output("data.txt"); // Режим по умолчанию: ios::out | ios::trunc

// Или для добавления:

ofstream output("data.txt", ios::app);

1. Привести примеры открытия файловых потоков в различных режимах.

// 1. Чтение

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

// 2. Запись с очисткой

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

// 3. Чтение и запись (без очистки)

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

// 4. Бинарный режим

fstream bin\_file("data.bin", ios::in | ios::out | ios::binary);

1. Привести примеры чтения объектов из потока.

ifstream fin("data.txt");

Money m;

// 1. Чтение через оператор >>

while (fin >> m) { cout << "Прочитано: " << m << endl; }

//2. Построчное чтение

string line;

while (getline(fin, line)) {

Money m = Money::from\_string(line);

cout << "Прочитано: " << m << endl;

}

1. Привести примеры записи объектов в поток.

ofstream fout("data.txt");

Money m(100, 50);

// 1. Через оператор <<

fout << m << endl;

// 2. Через метод to\_string()

fout << m.to\_string() << endl;

// 3. Запись массива объектов

vector<Money> money\_list = { Money(10, 20), Money(30, 40) };

for (const auto& item : money\_list) { fout << item << endl; }

1. Сформулировать алгоритм удаления записей из файла.
2. Открыть исходный файл для чтения (ifstream).
3. Создать временный файл для записи (ofstream).
4. Построчно читать исходный файл, копируя во временный только нужные записи.
5. Закрыть оба файла.
6. Удалить исходный файл (remove()).
7. Переименовать временный файл в исходный (rename()).
8. Сформулировать алгоритм добавления записей в файл.
9. Открыть файл в режиме добавления (ios::app):

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

1. Записать новые данные: file << new\_data << endl;
2. Закрыть файл: file.close();
3. Сформулировать алгоритм изменения записей в файле.
4. Открыть исходный файл для чтения (ifstream).
5. Создать временный файл для записи (ofstream).
6. Построчно читать исходный файл, изменяя нужные записи перед записью во временный файл.
7. Закрыть оба файла.
8. Заменить исходный файл временным (аналогично удалению).