



Урок 6. C++: Ошибки и надёжный файловый конвейер

Часть 1. Материалы ученика

Программирование на Python и C++ · 9 класс

Урок 6 из 34 · 45 минут · Модуль «Файлы»

Уровень: начинающий программист; C++17.

Программа: <https://schoolotzyv.ru/blog/python-cpp-9-klass-programma>

Поток числового ввода способен прочитать 12 из начала строки 12x. Как проверить всю запись и сохранить старый отчёт, если хотя бы одна строка неверна?

Результат урока: Построить конвейер с проверкой полного числового формата до открытия выхода и различить ошибки чтения, данных и записи.

01. Проблемная ситуация

0-5 минут. input.txt: 4\n12x\n-4. Старый output.txt: OLD. Решите, можно ли выдать сумму 12; предложите проверку, отличающую 12 от 12x.

02. Понятия и разобранный пример

5-14 минут: 4 минуты на понятия, 3 на пример, 2 на объяснение способа партнёру.

Проверяется каждый символ записи

Контракт: от 0 до 8 строк, каждая — необязательный минус и 1-3 ASCII-цифры. Пробелы и плюс запрещены, значения от -999 до 999. Окончания LF или CRLF, последний перевод строки необязателен. Пустой файл допустим, пустая строка — ошибка. -000 означает ноль.

Функция parse возвращает bool: успех проверки отделён от самого числа, которое может равняться нулю. После необязательного минуса должно остаться от 1 до 3 цифр. Значение строится по формуле $v=10*v+цифра$; максимум 999, сумма восьми чисел по модулю не превосходит 7992. Перед каждым накоплением формат проверен.

convert различает READ_ERROR, DATA_ERROR, WRITE_ERROR и OK. До открытия output.txt весь вход проверен и сумма рассчитана. Поэтому неверная запись, девятая строка или ошибка чтения сохраняют OLD. После открытия с trunc сброс записи может оставить частичный файл: это не атомарная замена и не защита от отключения питания.

Отступы набирайте обычными пробелами: 4 на уровень. При копировании с веб-страницы замените неразрывные пробелы обычными; полный код с обычными пробелами есть в PDF.

4,12x,-4: вторая строка не проходит проверку x, итог DATA_ERROR, OLD сохранён. После исправления на 12 получаем count=3, sum=12. Для 0 parse возвращает true и записывает число 0, а не сообщает ошибку.

```
#include <fstream>
#include <iostream>
#include <string>
bool parse(const std::string& s,int& value) {
    std::size_t p=(!s.empty() && s[0]!='-')?1:0;
    if (s.size()-p<1 || s.size()-p>3) return false;
    int v=0;
    for (std::size_t i=p;i<s.size();++i) {
        if (s[i]<'0' || s[i]>'9') return false;
        v=10*v+(s[i]-'0');
    }
    value=p?-v:v;
    return true;
}
std::string convert() {
    std::ifstream f("input.txt",std::ios::binary);
    if (!f) return "READ_ERROR";
    std::string s;
    int n=0,total=0,v=0;
    while (std::getline(f,s)) {
        if (!s.empty() && s.back()=='\r') s.pop_back();
        if (++n>8 || !parse(s,v)) return "DATA_ERROR";
        total+=v;
    }
    if (!f.eof()) return "READ_ERROR";
    std::ofstream out("output.txt",std::ios::binary|std::ios::trunc);
```



```

if (!out) return "WRITE_ERROR";
out<<"count="<<n<<"\nsum="<<total<<"\n";
out.close();
return out?"OK":"WRITE_ERROR";
}
int main() { std::cout<<convert()<<"\n"; }

```

Разбор строки	знак? → 1-3 ASCII-цифры → число
Нулевая запись	0 и -000 допустимы; успех=true
Ошибка до записи	READ_ERROR / DATA_ERROR → OLD остаётся
Успешный вход	count и sum → запись → проверка close

Булев результат проверки не смешивается с числовым нулём; запись начинается после проверки всех строк.

03. Практика 1. Проверяем материал

Ноль является допустимым числом

14–25 минут. Для таблицы составьте журнал «формат — статус — отчёт». Перед каждым неверным входом записывайте OLD в output.txt, чтобы проверить отсутствие частичного результата.

Вход	Статус; count и sum
4\n12\n-4	OK; 3, 12
0\n-000\n-009	OK; 3, -9
Пустой файл	OK; 0, 0
12x / +1 / - / пробел1 / пустая строка	DATA_ERROR
8 строк по -999 / 9 строк по 1	OK; 8, -7992 / DATA_ERROR

2 минуты — ручные ответы; 4 — прогон данных; 3 — объяснение успеха для 0 и -000; 2 — проверка старого файла после отказа. Критерии: весь формат, обе границы количества строк, сумма, сохранность OLD.

04. Практика 2. Перенос способа

Выход открывается после всех проверок

25–33 минуты. Используйте тот же полный исходник. За 2 минуты предскажите три входа: 999\r\n-999 (CRLF, без последнего LF), восемь строк по 999, девять строк по 999. За 4 выполните, восстанавливая OLD перед каждым; за 2 сравните результат. Затем в отдельной учебной папке проверьте отсутствие input.txt и каталог с именем output.txt при корректном входе. Критерии: суммы 0 и 7992, DATA_ERROR с OLD, READ_ERROR, WRITE_ERROR.

05. Обсуждение и границы вывода

33–40 минут. За 2 минуты сравните отказ в начале и в конце обработки; за 3 укажите, что именно доказали тесты с OLD; за 2 сформулируйте честную гарантию результата.

06. Индивидуальная проверка

40–45 минут. Для -001\n001\n0 укажите статус и отчёт; для 1\n2x — статус и судьбу OLD; объясните, почему parse возвращает bool отдельно от числа. Четыре балла: отчёт, отказ, сохранность, ноль.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Ожидания приведены в таблице. Пустой файл даёт нулевой отчёт, но не ошибку; пустая строка отклоняется. Проверка `if(value)` была бы неверна: допустимый ноль ложен как условие. Проверка только начального числового фрагмента пропустила бы 12х. Девятая запись отклоняет весь вход до открытия выхода.

Практика 2: разбор и критерии

Первый вход: OK, count=2, sum=0. Второй: OK, count=8, sum=7992. Третий: DATA_ERROR, OLD сохранён. Отсутствие входа даёт READ_ERROR до открытия выхода. Каталог `output.txt` нельзя открыть как обычный файл, получаем WRITE_ERROR. Не меняйте системные права и чужие папки; используйте отдельные каталоги для каждого опыта.

Обсуждение: возможный вывод

Тесты подтверждают сохранность старого файла при исследованных ошибках до записи. Они не доказывают атомарность и защиту от всех аппаратных сбоев. Для открытого выхода проверяется закрытие потока; WRITE_ERROR не обещает сохранность его прежнего содержимого. Лимиты учебного формата делают арифметику обозримой.

Индивидуальная проверка: ответ и критерии

OK, count=3 и sum=0 с LF. Второй вход: DATA_ERROR, OLD не меняется. bool сообщает успех синтаксической проверки независимо от численного значения: ноль допустим. По баллу за каждый пункт.

Учителю: подготовка и план на 45 минут

Подготовьте компьютер, редактор, папку с правом записи, распечатку ученической части PDF, бумагу и ручку. Компилятор G++ с поддержкой C++17 и редактор. Сборка: `g++ -std=c++17 -Wall -Wextra -pedantic main.cpp -o lesson`. Запуск `./lesson` в Linux/macOS или `./lesson.exe` в Windows (при сборке задайте `-o lesson.exe`). Установку и проверку запуска учитель выполняет заранее. Все программы, данные и ожидаемые ответы даны; регистрация детей и интернет на занятии не нужны.

Уровень — начинающий программист, 9 класс. Предпосылки: файловые потоки, строки, функции, проверки условий. В паре автор прогнозирует и вводит код, проверяющий ведёт журнал, затем роли меняются. Без компьютера выполните трассировку полного кода на бумаге и сопоставьте с готовыми результатами; учитель демонстрирует подготовленный запуск. Бумажный вариант не считается фактическим запуском ученика. Перед практикой учитель выдаёт готовый файл разобранного примера, чтобы дети тратили время на изменение и проверку, а не длинное перепечатывание. Матрица программы предназначена учителю.

Минуты	Работа	Результат
0–5	Проблема	Вопрос и первое решение
5–14	Понятия и пример	Способ проверки
14–25	Практика 1	Готовый продукт и проверка
25–33	Практика 2	Перенос способа
33–40	Обсуждение	Аргумент и ограничение
40–45	Индивидуальная проверка	Самостоятельный ответ



07. Соответствие исходной программе

Пункт программы	Действие ученика	Проверяемый результат
Файлы	Построить конвейер с проверкой полного числового формата до открытия выхода и различить ошибки чтения, данных и записи.	Исходник, журнал тестов и объяснение результата
Типы файлов; чтение, запись и обработка данных	Выход открывается после всех проверок	Самостоятельная доработка, ключ и индивидуальная проверка

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-9-klass-programma>

C++: файловые потоки

<https://eel.is/c++draft/fstream.syn>

C++: состояние выходного потока

<https://eel.is/c++draft/ostream.members>

Тексты, учебные данные, реплики, программы и схемы подготовлены для уроков портала «Школы России». Числовые серии — учебные модели, не результаты проведённых классом опытов. Схемы авторские, сторонние фотографии не используются. Методические ссылки приведены для учителя; ученик получает все материалы здесь. Проверка первичных источников: 18.10.2026.