



Урок 4. C++: Типы файлов и чтение

Часть 1. Материалы ученика

Программирование на Python и C++ · 9 класс

Урок 4 из 34 · 45 минут · Модуль «Файлы»

Уровень: начинающий программист; C++17.

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

Файл заканчивается словом `blue` без перевода строки. Потеряется ли последняя запись? А пустая строка между двумя словами?

Результат урока: Прочитать все строки через `getline`, сохранить пустые записи и сравнить текстовое представление с исходными байтами.

01. Проблемная ситуация

0–5 минут. Нарисуйте записи файла `red\n\nblue`. Здесь `\n` означает настоящий перевод строки LF, а не два печатных знака. Предскажите число строк и число байтов ASCII.

02. Понятия и разобранный пример

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

Успешное чтение управляет циклом

Работаем в отдельной учебной папке: `input.txt` содержит до 8 строк по 40 знаков ASCII, окончания LF или CRLF. Путь относительно текущей рабочей папки запуска. `getline` извлекает строку до LF и удаляет разделитель; последняя строка без LF тоже успешно читается. Файл из одного LF содержит одну пустую строку; пустой файл не содержит строк.

Поток открыт с `ios::binary`: байты не преобразуются средой. После `getline` вручную убираем один завершающий CR, чтобы одинаково обработать LF и CRLF. Пробелы по краям остаются. В этом учебном формате CR служит только частью окончания строки. Для текста на других языках необходимо отдельно договориться о кодировке; здесь длина ASCII в байтах равна числу знаков.

`while(getline(f,line))` обрабатывает только успешно полученную запись. Проверка `while(!f.eof())` заранее ненадёжна: конец обнаруживается при чтении, и тело может обработать неудачное чтение. После цикла отличаем штатный конец от другого сбоя. Этот просмотрщик может успеть показать часть строк до ошибки; полноценный конвейер изучается в уроке 6.

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

Для `red\n\nblue` вывод: 1 [red], 2 [], 3 [blue], затем `count=3`, каждая запись на новой строке. Байтов $3+1+1+4=9$. Завершающего LF нет, однако `blue` не теряется.

```
#include <fstream>
#include <iostream>
#include <string>
int main() {
    std::ifstream f("input.txt",std::ios::binary);
    if (!f) { std::cout<<"READ_ERROR\n"; return 0; }
    std::string line;
    int n=0;
    while (std::getline(f,line)) {
        if (!line.empty() && line.back()=='\r') line.pop_back();
        std::cout<<"+n<<" ["<<line<<"]\n";
    }
    if (!f.eof()) { std::cout<<"READ_ERROR\n"; return 0; }
    std::cout<<"count="<<n<<'\n';
}
```



Байты LF-файла	red LF LF blue
getline 1	red → запись 1
getline 2	пустая строка → запись 2
getline 3	blue без LF → запись 3

Разделители создают границы записей; отсутствие последнего LF не отменяет последнюю запись.

03. Практика 1. Проверяем материал

Пустая запись тоже имеет номер

14–25 минут. Учитель выдаёт исходник и файлы из таблицы. Создайте таблицу «вход — прогноз — вывод — совпадение»; отметьте отдельно пробелы и пустые записи.

Файл	Число строк; содержимое
red\n\nblue	3; red, пустая, blue
Пустой файл / один LF	0 / 1 пустая
x \n	1; пробел, x, пробел
red\r\n\r\nblue	3; те же строки, 11 байтов
red\nblue\n	2; лишней пустой записи нет

2 минуты — прогнозы; 4 — запуски; 3 — объяснение каждого различия с подсчётом LF; 2 — вывод о последней строке. Критерии: 5 случаев, отсутствие лишней записи, сохранение пробелов, обработка CRLF.

04. Практика 2. Перенос способа

Читаем точные байты

25–33 минуты. В отдельный main.cpp вставьте второй исходник из разбора. За 2 минуты предскажите байты red\n\nblue, за 4 сравните бинарный вывод для LF и CRLF, за 2 объясните, почему число строк одинаково, а размер различается. Критерии: точные коды LF=10, CR=13, длины 9 и 11.

05. Обсуждение и границы вывода

33–40 минут. За 2 минуты выберите: считать переводы строк или выполнять успешные getline. За 3 приведите контрпример для первого подхода, за 2 сформулируйте границы ASCII-модели.

06. Индивидуальная проверка

40–45 минут. Для A\r\n\r\nB запишите три строки просмотра, число байтов и объясните назначение pop_back. Четыре пункта по баллу.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Числа строк: 3; 0 и 1; 1; 3; 2. Количество LF само по себе не даёт число строк, если последняя непустая строка не заканчивается LF. У `red\nblue` два LF? Нет: один LF, но две строки. Нельзя добавлять строку после каждого успешного завершения файла.

Практика 2: разбор и критерии

LF-файл: 114 101 100 10 10 98 108 117 101, затем `bytes=9`. CRLF-файл добавляет 13 перед каждым 10, всего 11 байтов. Пустой файл даёт `bytes=0`. Пробел имеет код 32. Приведение к `unsigned char` не даёт отрицательных значений байтов при выводе.

```
#include <fstream>
#include <iostream>
int main() {
    std::ifstream f("input.txt", std::ios::binary);
    if (!f) { std::cout<<"READ_ERROR\n"; return 0; }
    char c;
    int n=0;
    while (f.get(c)) {
        std::cout<<static_cast<int>(static_cast<unsigned char>(c))<<' ';
        ++n;
    }
    if (!f.eof()) { std::cout<<"READ_ERROR\n"; return 0; }
    std::cout<<"\nbytes="<<n<<"\n";
}
```

Обсуждение: возможный вывод

Файл `x` без LF имеет одну строку и ни одного перевода строки. Успех `getline` учитывает её. В ASCII один знак занимает байт, но это нельзя переносить на произвольный UTF-8-текст: русская буква обычно занимает несколько байтов. Бинарный режим считает байты, не смысловые символы.

Индивидуальная проверка: ответ и критерии

1 [A], 2 [], 3 [B]; `count=3`. Байтов 6: A, CR, LF, CR, LF, B. `pop_back` удаляет завершающий CR после `getline`; пустую запись сохраняет. Баллы: записи, число строк, размер, объяснение.

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: `string`, циклы `while`, условия, поток `cout`. 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	Прочитать все строки через <code>getline</code> , сохранить пустые записи и сравнить текстовое представление с исходными байтами.	Исходник, журнал тестов и объяснение результата
Типы файлов; чтение, запись и обработка данных	Читаем точные байты	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++: неформатированный ввод

<https://eel.is/c++draft/istream.unformatted>

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.