



Урок 5. C++: Документирование и читаемость кода

Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 5 из 33 · 45 минут · Модуль «Документирование программы. Процедуры с параметрами»

Уровень: начинающий программист; C++17.

Программа: <https://schoolotzyv.ru/blog/python-cpp-8-klass-programma>

В программе соревнования встретилось выражение $x \cdot 3 + y$. Сделаем правило очков видимым и проверим, что переименование не изменило результат.

Результат урока: Заменить неясные имена и необъяснённые константы, документировать ограничение и подтвердить сохранение результатов.

01. Проблемная ситуация

0–5 минут. Код `int f(int x,int y){return x*3+y;}` выдаёт 14 для 4 и 2. Сформулируйте две возможные трактовки x и y . Для нашего учебного турнира x — победы, y — ничьи; за победу 3 очка, за ничью 1. Запишите смысл числа 3.

02. Понятия и разобранный пример

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

Название раскрывает назначение

Имена `total_points`, `won`, `tied` связывают код с задачей. Константы `points_per_win` и `points_per_tie` отделяют правила подсчёта от структуры выражения. Это учебное правило данного задания, не утверждение о регламенте реального соревнования.

Комментарий описывает единицы, диапазон и результат; не нужно комментировать каждый знак. `const` запрещает присваивать новое значение этой переменной после инициализации. Умеренные отступы и отдельная функция помогают увидеть, что вывод выполняется в `main`, а вычисление возвращает `int`.

При рефакторинге результат сохраняют на всех допустимых входах. Здесь `won, tied` целые $0 \dots 20$, максимум $20 \cdot 3 + 20 = 80$, он безопасно помещается в `int`. Замена коэффициента 3 на 5 меняет правило задачи: это уже изменение поведения, даже если новое имя стало понятнее.

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

Четыре победы и две ничьи: $4 \cdot 3 + 2 \cdot 1 = 14$. Нулевые входы дают 0. Для 20,20 результат 80. Полная программа ниже реализует тот же подсчёт, что старое `f`, но явно показывает назначение коэффициентов.

```
#include <iostream>
// won и tied – целые 0...20; учебное правило: 3 и 1 очко.
int total_points(int won, int tied) {
    const int points_per_win=3;
    const int points_per_tie=1;
    return won*points_per_win + tied*points_per_tie;
}

int main() {
    std::cout << total_points(4,2) << "\n";
}
```



Победы	$4 \times 3 = 12$
Ничьи	$2 \times 1 = 2$
Сумма	14 очков
Контроль максимума	$20 \times 3 + 20 = 80$

Учебное правило: победы $\times 3$, ничьи $\times 1$; для 4 и 2 получаем $12+2=14$ очков.

03. Практика 1. Проверяем материал

Сравнение старой и новой версий

14–25 минут. Перепишите `f` в понятном виде и сравните обе функции на пяти входах. Не меняйте учебное правило.

won, tied	Очки
0,0	0
1,0	3
0,1	1
4,2	14
20,20	80

2 минуты — смысл параметров; 4 — код; 3 — сравнение; 2 — взаимное чтение. Продукт: исходник с двумя функциями и журнал совпадений. Критерии: одинаковые результаты, осмысленные имена, `const`, точный комментарий о диапазоне. Для компактного тестового драйвера здесь перебираются все пары $0\dots 2$.

04. Практика 2. Перенос способа

Другое правило — другой контракт

25–33 минуты. Для другого учебного состязания дают 5 очков за завершённый этап и 2 за частично завершённый; каждый счётчик целый $0\dots 10$. Создайте `stage_points(done, partial)`, не меняя `total_points`. За 2 минуты напишите контракт, за 4 код, за 2 проверьте $(0,0), (2,3), (10,10)$. Критерии: отдельное имя, новое правило, три результата и максимум.

05. Обсуждение и границы вывода

33–40 минут. Нужно ли объявлять константой каждое число, даже 0 в начале цикла? Сравните границу счётчика и коэффициент очков. Предложите, какие имена действительно помогают понять задачу и когда комментарий становится избыточным.

06. Индивидуальная проверка

40–45 минут. Найдите `total_points(3,1)` и `stage_points(3,1)`. Объясните, почему ответы различаются, и напишите полезный комментарий к одному параметру. По баллу за четыре элемента.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

В драйвере ниже для каждой пары совпадают два последних числа. Обязательно отдельно проверьте также 4,2 и 20,20 из таблицы. Имена сами не гарантируют верный коэффициент: 0,1 отличает ничью от победы.

```
#include <iostream>
int f(int x,int y) { return x*3+y; }
// Аргументы – числа побед и ничьих, целые 0...20.
int total_points(int won,int tied) {
    const int points_per_win=3, points_per_tie=1;
    return won*points_per_win+tied*points_per_tie;
}

int main() {
    for (int w=0; w<=2; ++w)
        for (int t=0; t<=2; ++t)
            std::cout << w << " " << t << " " << f(w,t)
                        << " " << total_points(w,t) << "\n";
    std::cout << f(4,2) << " " << total_points(4,2) << "\n";
    std::cout << f(20,20) << " " << total_points(20,20) << "\n";
}
```

Практика 2: разбор и критерии

Ответы 0,16,70; максимум 70. Это новая задача, не рефакторинг прежней. Полный тестовый файл выводит три строки.

```
#include <iostream>
// Целые done,partial в 0...10; вернуть учебные очки.
int stage_points(int done,int partial) {
    const int full_reward=5, partial_reward=2;
    return done*full_reward+partial*partial_reward;
}

int main() {
    std::cout << stage_points(0,0) << "\n";
    std::cout << stage_points(2,3) << "\n";
    std::cout << stage_points(10,10) << "\n";
}
```

Обсуждение: возможный вывод

Имена особенно полезны для предметных правил, например награды за этап. Ноль как начало счётчика часто понятен без отдельной константы. Избыточные имена могут скрыть простой алгоритм; полезность проверяем чтением кода другим человеком.

Индивидуальная проверка: ответ и критерии

10 и 17; разные учебные правила (3/1 и 5/2); например «partial — число частично завершённых этапов, целое 0...10». 4 балла.

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, 8 класс. Предпосылки: функции int, параметры, const и арифметика. 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
Документирование программы. Процедуры с параметрами	Разбирает пример и создаёт две подпрограммы	Исходники, прогнозы и протокол проверки
Составление и отладка программ	Сравнивает фактический результат с контрактом	Объяснение границ и самостоятельный ответ

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-8-klass-programma>

Рабочий текст стандарта C++: условные операторы

<https://eel.is/c++draft/stmt.if>

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.