



Урок 27. C++: алгоритм Евклида

Часть 1. Материалы ученика

Программирование на Python и C++ · 7 класс

Урок 27 из 34 · 45 минут · Модуль «Циклы: НОД»

Уровень: начинающий программист; C++17.

Программа: <https://schoolotzyv.ru/blog/python-cpp-7-class-programma>

Ленты длиной 84 и 30 см нужно разделить на одинаковые максимально длинные части без остатка. Как избежать перебора всех возможных длин?

Результат урока: Выполнить алгоритм Евклида, проверить нулевые аргументы и применить НОД к сокращению дроби.

01. Проблемная ситуация

За 5 минут найдите общие делители 84 и 30 среди 1,2,3,6,7. Выберите наибольший подходящий и объясните проверку остатками. Предложите, чем заменить долгий перебор при очень больших длинах.

02. Понятия и разобранный пример

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

Почему переход к остатку сохраняет общие делители

Для $a = bq + r$ общие делители a и b совпадают с общими делителями b и r . Если число делит a и b , оно делит $r = a - bq$; обратное следует из $a = bq + r$. Поэтому $\text{НОД}(a, b) = \text{НОД}(b, a \% b)$ при $b > 0$.

Тело должно сначала сохранить r , затем заменить a прежним b , а b — r . После присваивания $a = b$ старое a утрачено, поэтому вычислять остаток позже нельзя. При положительном b новый остаток неотрицателен и меньше b , это обеспечивает завершение.

$\text{НОД}(a, 0) = a$ для $a > 0$. Для пары 0,0 в этой учебной программе явно принят вывод 0 как служебное соглашение, не утверждение о наибольшем положительном общем делителе. Отрицательные аргументы не входят в контракт. Для дроби исходный знаменатель всегда положителен, поэтому делитель сокращения не нулевой.

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

Пары: $(84, 30) \rightarrow (30, 24) \rightarrow (24, 6) \rightarrow (6, 0)$. Ответ 6; проверка $84 \% 6 = 0$ и $30 \% 6 = 0$. Равенство двух остатков нулю доказывает общий делитель, а максимальность следует из сохранения множества общих делителей на каждом шаге.

```
#include <iostream>
```

```
int main() {
    long long a, b;
    if (!(std::cin >> a >> b) || a < 0 || b < 0
        || a > 10000000000 || b > 10000000000) {
        std::cout << "error\n"; return 0;
    }
    while (b != 0) {
        long long r = a % b;
        a = b;
        b = r;
    }
    std::cout << a << "\n";
}
```



C++: алгоритм Евклида

Исходная пара	84,30
Остаток 24	30,24
Остаток 6	24,6
Остаток 0	6,0 → НОД 6

Алгоритм Евклида: $(84,30) \rightarrow (30,24) \rightarrow (24,6) \rightarrow (6,0)$; ответ 6.

03. Практика 1. Проверяем материал

Трассировка пар и проверка нулей

14–25 минут. Реализуйте Евклида и журнал пар до каждого тела цикла для 30 84.

a b	НОД / соглашение
84 30	6
30 84	6
17 13	1
0 9	9
9 0	9
0 0	0

2 минуты — ручные пары; 4 — код; 3 — шесть тестов и –1 5; 2 — объяснение порядка присваиваний. Продукт: код, таблица и довод об убывании b. Критерии: остаток сохранён до замены, нет деления на ноль, нулевые случаи соответствуют договорённости, 30 84 даёт тот же НОД.

04. Практика 2. Перенос способа

Сокращение дроби без потери исходных чисел

25–33 минуты. Ввод n/d: целые $0 \leq n \leq 10^9$, $1 \leq d \leq 10^9$. Выведите сокращённую дробь с символом /. За 2 минуты определите, что сохранить до Евклида; за 4 реализуйте; за 2 проверьте 84 30, 7 21, 0 15 и 5 0. Критерии: исходные n,d не утрачены, оба поделены на один НОД, знаменатель проверен.

05. Обсуждение и границы вывода

33–40 минут. Две программы выдали 6 для 84 30. Одна только перебрала делители, другая использовала остатки. Что подтверждает правильность, а что эффективность? Нужно ли объявлять перебор всегда неправильным?

06. Индивидуальная проверка

40–45 минут. Выпишите пары для 48 18 и сократите 48/18. Назовите ответ программы для 0 12 и смысл ответа для 0 0. Четыре критерия.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Для 30 84: (30,84),(84,30),(30,24),(24,6),(6,0). Первый остаток 30, хотя первый аргумент меньше второго; отдельная сортировка не требуется. $-1 \ 5 \rightarrow \text{error}$. Полный рабочий код ниже.

```
#include <iostream>

int main() {
    long long a, b;
    if (!(std::cin >> a >> b) || a < 0 || b < 0
        || a > 1000000000 || b > 1000000000) {
        std::cout << "error\n"; return 0;
    }
    while (b != 0) {
        long long r = a % b;
        a = b;
        b = r;
    }
    std::cout << a << "\n";
}
```

Практика 2: разбор и критерии

Ответы: 14/5; 1/3; 0/1; error. Для 0/15 НОД равен 15, деление определено. Если запускать Евклида прямо на n,d, после выхода d станет нулём и исходная дробь потеряется. Поэтому используют копии a,b.

```
#include <iostream>

int main() {
    long long n, d;
    if (!(std::cin >> n >> d) || n < 0 || n > 1000000000
        || d < 1 || d > 1000000000) {
        std::cout << "error\n"; return 0;
    }
    long long a = n, b = d;
    while (b != 0) {
        long long r = a % b;
        a = b;
        b = r;
    }
    std::cout << n / a << "/" << d / a << "\n";
}
```

Обсуждение: возможный вывод

Перебор может быть правильным при корректных границах, но делает больше проверок. Тест совпадения одного ответа не доказывает общий алгоритм. Для Евклида нужны инвариант общих делителей, убывание b и корректный выход. Число шагов и доказательство — разные свидетельства.

Индивидуальная проверка: ответ и критерии

(48,18)→(18,12)→(12,6)→(6,0); дробь 8/3. Для 0 12 ответ 12. Для 0 0 вывод 0 принят контрактом, а не как наибольший положительный общий делитель.

Учителю: подготовка и план на 45 минут

Подготовьте компьютер, рабочую папку с правом записи, редактор, распечатку ученической части PDF, бумагу и ручку. Локальный компилятор G++ с C++17 или настроенная учителем IDE; сборка g++ -std=c++17 -Wall -Wextra -pedantic main.cpp -o lesson. В Windows задайте lesson.exe. Запуск .\lesson в Linux/macOS, .\lesson.exe в Windows. Для урока 31 дополнительно обычный браузер: открывать локальный SVG, не C++-исходник. Все входы и ожидаемые результаты даны. Установку и проверку запуска учитель выполняет до занятия; детские аккаунты и интернет не нужны.

Уровень — начинающий программист, 7 класс. Предпосылки: while, остаток, сохранение промежуточного значения. В паре автор вводит и объясняет, проверяющий сопоставляет с прогнозом, затем роли меняются. Без компьютера трассируйте полный код на бумаге, отмечайте координаты по



схеме и проверяйте готовые результаты; учитель демонстрирует заранее собранный пример. Бумажная трассировка не выдаётся за реальную компиляцию или запуск ученика. План 45 минут включает чтение, ввод, сборку и обсуждение. В графических задачах сначала прогноз, затем сравнение. 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
Алгоритм Евклида	Выполнить алгоритм Евклида, проверить нулевые аргументы и применить НОД к сокращению дроби.	Самостоятельный продукт и тесты, описанные в практиках
Составлять и отлаживать простые диалоговые программы	Сверяет прогноз с фактическим запуском; объясняет ошибочный вариант	Журнал тестов и индивидуальный ответ по критериям

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-7-klass-programma>

C++: остаток целочисленного деления

<https://eel.is/c++draft/expr.mul>

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: 17.10.2026.