



Урок 14. C++: Рекурсия и базовый случай

Часть 1. Материалы ученика

Программирование на Python и C++ · 9 класс

Урок 14 из 34 · 45 минут · Модуль «Функции»

Уровень: начинающий программист; C++17.

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

Наибольший общий делитель можно найти, каждый раз заменяя пару меньшей задачей. Остановка происходит до деления на ноль.

Результат урока: Реализовать рекурсивный алгоритм Евклида, объяснить сохранение общих делителей и проверить нулевые аргументы.

01. Проблемная ситуация

0–5 минут. Выпишите общие положительные делители 48 и 18. Затем замените пару на (18, 48%18). Сравните списки делителей.

02. Понятия и разобранный пример

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

Остаток уменьшает задачу

Для $a \geq 0$, $b > 0$ общие делители a и b совпадают с общими делителями b и $a \% b$: из $a = q \cdot b + r$ следует, что общий делитель a, b делит r , а общий делитель b, r делит a . Поэтому $\text{gcd}(a, b) = \text{gcd}(b, a \% b)$.

База $b == 0$ возвращает a до вычисления остатка. При каждом небазовом шаге второй аргумент становится меньше прежнего положительного b , поэтому достигает нуля. Для $(0, b)$, $b > 0$, первый шаг даёт $(b, 0)$.

Контракт учебной функции: целые $0 \leq a, b \leq 1000$. В этом коде $\text{gcd}(0, 0) = 0$ — явно выбранное программное соглашение; положительного наибольшего общего делителя у пары $(0, 0)$ нет. Для отрицательных входов проверка выполняется в `main` до вызова; функция предполагает соблюдение контракта.

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

$(48, 18) \rightarrow (18, 12) \rightarrow (12, 6) \rightarrow (6, 0)$. Все вызовы на возврате дают 6. Четыре вызова включают базовый. Если перенести $a \% b$ перед проверкой $b == 0$, последний шаг приведёт к недопустимому делению на ноль.

```
#include <iostream>
int gcd_value(int a, int b) {
    if (b == 0) return a;
    return gcd_value(b, a % b);
}
int main() {
    int a = 48, b = 18;
    if (a < 0 || a > 1000 || b < 0 || b > 1000) {
        std::cout << "invalid\n"; return 1;
    }
    std::cout << gcd_value(a, b) << "\n";
}
```

Спуск	$(48, 18) \rightarrow (18, 12)$
Продолжение	$(12, 6) \rightarrow (6, 0)$
Базовый случай	$b = 0$: вернуть $a = 6$
Возврат	6 во всех ожидающих вызовах

Алгоритм Евклида: второй аргумент уменьшается; база проверяется до остатка.



03. Практика 1. Проверяем материал

Проверяем спуск к нулю

14–25 минут. Для шести пар выпишите цепочки вызовов, затем меняйте a, b в `main` и запускайте.

a, b	Результат; число вызовов
48,18	6; 4
0,7; 7,0	7; 2 и 7; 1 соответственно
7,7; 13,5	7; 2 и 1; 5 соответственно
0,0	0; 1 — соглашение функции

2 минуты — прогноз; 4 — запуск; 3 — сравнение цепочек, тест -1,5; 2 — объяснение остановки.

Критерии: результат, число вызовов, отдельный смысл (0,0), отказ вне диапазона.

04. Практика 2. Перенос способа

Используем НОД в сокращении дроби

25–33 минуты. Сократите дробь a/b с $0 \leq a \leq 1000$, $1 \leq b \leq 1000$. За 2 минуты рассчитайте $18/48$ и $0/7$, за 4 используйте `gcd_value`, за 2 проверьте $13/5$ и отказ при $b=0$. Критерии: деление обеих частей на один d , ненулевой знаменатель, целые результаты.

05. Обсуждение и границы вывода

33–40 минут. За 2 минуты предложите проверку ответа d , за 3 обсудите, почему одной делимости недостаточно, за 2 сформулируйте независимую проверку для малых чисел.

06. Индивидуальная проверка

40–45 минут. Для (21,15) запишите цепочку, НОД, число вызовов и сокращение $21/15$. Четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Для (13,5): (13,5)→(5,3)→(3,2)→(2,1)→(1,0), пять вызовов и результат 1. Отрицательный a отклоняется `main` с `invalid`. При $b=0$ остаток не вычисляется. Для (0,0) возвращаем 0 по договорённости, не называя его положительным НОД.

Практика 2: разбор и критерии

18/48→3/8; 0/7→0/1; 13/5 остаётся 13/5. В этом контракте d всегда положителен, поэтому деление безопасно. Полная программа:

```
#include <iostream>
int gcd_value(int a,int b) {
    if(b==0) return a;
    return gcd_value(b,a%b);
}
int main() {
    int a=18,b=48;
    if(a<0 || a>1000 || b<1 || b>1000) {
        std::cout<<"invalid\n"; return 1;
    }
    int d=gcd_value(a,b);
    std::cout<<a/d<<"/"<<b/d<<"\n";
}
```

Обсуждение: возможный вывод

d должен делить оба числа, но 1 тоже делит 48 и 18 и не является их НОД. Для малых входов перебираем положительные общие делители и выбираем максимальный. Пара (0,0) исключается из такого определения и проверяется отдельно по соглашению.

Индивидуальная проверка: ответ и критерии

(21,15)→(15,6)→(6,3)→(3,0); 3; 4 вызова; 7/5. Каждый пункт — один балл.

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: функции, `return`, остаток, сравнения. 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
Функции. Логические функции. Рекурсивные функции.	Проверяем спуск к нулю	Реализовать рекурсивный алгоритм Евклида, объяснить сохранение общих делителей и проверить нулевые аргументы.
Структурирование решения и отладка	Используем НОД в сокращении дроби	Исходник, журнал обычных и граничных тестов; индивидуальный ответ

Источники и права на изображения

Исходная программа курса

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

C++: возврат из функции

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

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.