



Урок 13. C++: Логические функции и контракт

Часть 1. Материалы ученика

Программирование на Python и C++ · 9 класс

Урок 13 из 34 · 45 минут · Модуль «Функции»

Уровень: начинающий программист; C++17.

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

Ряд 2,2,5 уже упорядочен по неубыванию, хотя два значения равны. Проверка должна точно соответствовать выбранному определению порядка.

Результат урока: Написать предикат неубывания, определить поведение для пустого и одиночного вектора и сравнить со строгим возрастанием.

01. Проблемная ситуация

0–5 минут. Для {}, {4}, {2,2,5}, {2,5,3} решите, нарушен ли порядок. Укажите конкретную соседнюю пару в каждом нарушающем примере.

02. Понятия и разобранный пример

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

Контракт и ранний возврат

Контракт `nondecreasing(a)`: вектор до 100 целых от -1000 до 1000; результат `true` тогда и только тогда, когда каждая соседняя пара `a[i-1] ≤ a[i]`. Функция не изменяет вектор: параметр передаётся по константной ссылке. Ограничения размера и значений считаются выполненными вызывающим кодом.

Одна нарушающая пара достаточна для `false`. Если цикл завершился без нарушения, возвращаем `true`. Возврат `true` внутри первой итерации ошибочен: он пропускает последующие пары. Пустой и одиночные векторы не содержат нарушающих соседей, поэтому по этому контракту возвращают `true`.

В C++ запись `a ≤ b ≤ c` не означает математическую цепочку. Используйте отдельные сравнения и `&&`. Здесь проверяется одна пара за шаг. `std::boolalpha` печатает `true/false` вместо `1/0`, но тип результата функции остаётся `bool`.

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 {2,5,3}: пара `2 ≤ 5` проходит, затем `5 > 3` вызывает `false`. Для {2,2,5} обе пары допустимы. Равенство запрещает другой контракт — строгое возрастание.

```
#include <iostream>
#include <vector>
bool nondecreasing(const std::vector<int>& a) {
    for(std::size_t i=1;i<a.size();++i) {
        if(a[i-1]>a[i]) return false;
    }
    return true;
}
int main() {
    std::cout<<std::boolalpha;
    for(const auto& a:std::vector<std::vector<int>>
        {{},{4},{2,2,5},{2,5,3},{-3,-2,-2}}) {
        std::cout<<nondecreasing(a)<<"\n";
    }
}
```



Пары {2,5,3}	2 <= 5: да; 5 <= 3: нет
Первое нарушение	Индексы 1 и 2
Результат	false
Нет нарушений	Проверить все пары → true

Истина требует проверки всех соседей; ложь подтверждается одной нарушающей парой.

03. Практика 1. Проверяем материал

Ищем контрпример среди соседей

14–25 минут. В журнале для каждой серии укажите результат и первую нарушающую пару либо «нет».

Вектор	Неубывание
{ }; {4}	true; true
{2,2,5}	true
{2,5,3}	false
{-3,-2,-2}; {0,-1}	true; false

2 минуты — прогноз; 4 — сборка и запуск; 3 — проверка ошибочной версии с `return true` внутри цикла; 2 — восстановление. Критерии: пустота, равные соседи, позднее нарушение, неизменность входа.

04. Практика 2. Перенос способа

Строгое и нестрогое сравнение

25–33 минуты. Реализуйте `strictly_increasing(a)`. За 2 минуты измените определение, за 4 код, за 2 проверьте { }, {2}, {2,2}, {-2,0,3}. Критерии: отказ при равенстве, прежнее поведение пустоты и одиночного входа, четыре теста.

05. Обсуждение и границы вывода

33–40 минут. Один ученик предлагает возвращать false для пустоты, другой true. За 2 минуты приведите формулировки обоих контрактов, за 3 обсудите правило всех соседей, за 2 выберите поведение текущей функции.

06. Индивидуальная проверка

40–45 минут. Для {1,1,4} назовите оба результата, число соседних пар и результат предиката непустоты с неубыванием для { }. Четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Для {2,5,3} первое нарушение — 5 и 3; для {0,-1} — 0 и -1. Ошибочная ранняя истина примет {2,5,3} после проверки первой пары. Правильный цикл стартует с 1, поэтому `a[i-1]` не выходит за левую границу.

Практика 2: разбор и критерии

Результаты true,true,false,true. Отказ теперь при `a[i-1]>=a[i]`. Полная программа:

```
#include <iostream>
#include <vector>
bool strictly_increasing(const std::vector<int>& a) {
    for(std::size_t i=1;i<a.size();++i) {
        if(a[i-1]>=a[i]) return false;
    }
    return true;
}
int main() {
    std::cout<<std::boolalpha;
    for(const auto& a:std::vector<std::vector<int>>
        {{},{2},{2,2},{-2,0,3}}) {
        std::cout<<strictly_increasing(a)<<"\n";
    }
}
```

Обсуждение: возможный вывод

По условию «нет нарушающей соседней пары» пустота истинна. Другой предикат «непустая упорядоченная серия» может вернуть false: `!a.empty() && nondecreasing(a)`. Ошибки нет, если смысл явно другой; нельзя менять его незаметно для вызывающего кода.

Индивидуальная проверка: ответ и критерии

`nondecreasing=true; strictly_increasing=false; 2 пары; !a.empty() && nondecreasing(a)` для {} равно false.

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: `bool`, `vector`, цикл, индексы и 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.	Ищем контрпример среди соседей	Написать предикат неубывания, определить поведение для пустого и одиночного вектора и сравнить со строгим возрастанием.
Структурирование решения и отладка	Строгое и нестрогое сравнение	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.