



Урок 24. C++: Встроенная сортировка и сравнение сложности

Часть 1. Материалы ученика

Программирование на Python и C++ · 9 класс

Урок 24 из 34 · 45 минут · Модуль «Сортировка»

Уровень: начинающий программист; C++17.

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

Стандартная сортировка принимает правило сравнения. Если правило говорит, что элемент меньше самого себя, контракт нарушен — это не способ «разрешить равенство».

Результат урока: Использовать `std::sort` и `std::stable_sort` с корректным компаратором, проверить ключи и сопоставить гарантии сложности.

01. Проблемная ситуация

0–5 минут. Записи (балл,метка): (2,B),(1,A),(2,C). Нужно по баллу и сохранить B перед C. Достаточно ли одного `std::sort` с сравнением только баллов, если такое поведение случайно получилось на компьютере?

02. Понятия и разобранный пример

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

Строгое сравнение и устойчивость

`std::sort` из `<algorithm>` переставляет элементы диапазона `[begin,end)` и не обещает сохранять порядок равных ключей. `std::stable_sort` сохраняет этот порядок. Компаратор должен задавать строгое слабое упорядочение: в частности `comp(x,x)` обязан быть `false`. Для числового ключа используем `<`, не `<=`.

Для `std::sort` стандарт задаёт $O(n \log n)$ сравнений. Для `stable_sort` число сравнений зависит от доступности дополнительной памяти: при достаточной памяти $O(n \log n)$, иначе допускается $O(n \log^2 n)$. Это не утверждение о точном времени или обязательном конкретном алгоритме реализации.

Пузырёк и выбор имеют $O(n^2)$ в худшем случае, поэтому рост объёма особенно важен. Для $n=8$ и $n=16$ оценки $n(n-1)/2$ равны 28 и 120, а $n \log_2 n$ равны 24 и 64. Последние числа — модель роста, не фактический счётчик `std::sort`. Контракт: 0...100 пар (целый балл, латинская метка); данные условные.

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

Копия `a` сортируется `stable_sort` только по `first`: получаем (1,A),(2,B),(2,C). Копия `b` сортируется обычным `sort` по всей паре: сначала балл, при равенстве метка. На исходных (2,C),(1,A),(2,B) устойчивый результат будет A,C,B, а сравнение всей пары — A,B,C.

```
#include <algorithm>
#include <iostream>
#include <vector>
#include <utility>
using Row=std::pair<int,char>;
void show(const std::vector<Row>& a) {
    for(auto [score,id]:a)
        std::cout<<score<<': '<<id<<' ';
    std::cout<<'\n';
}
int main() {
    std::vector<Row> rows{{2,'B'},{1,'A'},{2,'C'}};
    auto a=rows, b=rows;
    std::stable_sort(a.begin(),a.end(),
        [](const Row& x,const Row& y) {
            return x.first<y.first;
        });
    std::sort(b.begin(),b.end());
    show(a); show(b); show(rows);
}
```



Вход для проверки	$2:C \rightarrow 1:A \rightarrow 2:B$
stable_sort по баллу	$1:A \rightarrow 2:C \rightarrow 2:B$
sort по всей паре	$1:A \rightarrow 2:B \rightarrow 2:C$
Контракт сравнения	$\text{comp}(x,x) = \text{false}$

Сохранение входного порядка и вторичная сортировка по метке — разные требования.

03. Практика 1. Проверяем материал

Проверяем две библиотечные операции

14–25 минут. В C++ метки записываются в одинарных кавычках, как в полном коде. Таблица сокращает запись для чтения.

Вход rows	Ожидание
$\{\{2,B\},\{1,A\},\{2,C\}\}$	Обе копии A,B,C; исходник B,A,C
$\{\{2,C\},\{1,A\},\{2,B\}\}$	Устойчиво A,C,B; по паре A,B,C
$\{\}; \{\{0,Q\}\}$	Пустые строки; единственная Q
$n=32$	Выбор: 496 сравнений; $n \log_2 n = 160$

2 минуты — прогноз, 4 — четыре запуска, 3 — расчёт $n=32$ и проверка исходника, 2 — журнал. Критерии: равные ключи не спутаны с равными парами, исходник неизменен из-за копий, формула не названа точным числом `std::sort`.

04. Практика 2. Перенос способа

Два направления в одном ключе

25–33 минуты. Вход $\{\{2,C\},\{3,D\},\{2,B\},\{1,A\}\}$. Требуется больший балл раньше, а при равном балле меньшая метка раньше. За 2 минуты напишите правило словами, за 4 замените вызов `std::sort` для `b` на компаратор, за 2 проверьте $x=x$ и пару одинаковых баллов. Критерии: D,B,C,A и строгий компаратор.

05. Обсуждение и границы вывода

33–40 минут. Программа один раз сохранила порядок равных ключей после `sort`. За 2 минуты сформулируйте наблюдение, за 3 отделите его от гарантии, за 2 выберите решение для ведомости с порядком поступления.

06. Индивидуальная проверка

40–45 минут. Укажите библиотечную функцию для устойчивого порядка, правильный знак числового сравнения, значение `comp(x,x)` и верхнюю асимптотическую оценку сравнений `std::sort`. Четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Для второй серии `stable_sort` сохраняет `C` перед `B`, обычная сортировка всей пары ставит `B` перед `C`. `rows` не меняется, поскольку сортировались `a` и `b`. $32 \cdot 31 / 2 = 496$, $32 \cdot 5 = 160$. Пустой диапазон разрешён; стандартная функция не требует ручного обращения к первому элементу.

Практика 2: разбор и критерии

Полный заменяющий вызов приведён ниже. При равных баллах сравниваются метки через `<`. Для `x=x` обе проверки дают `false`. Нельзя использовать `x.first>=y.first`: оно возвращает `true` на равных ключах и нарушает контракт.

```
std::sort(b.begin(), b.end(),
    [](const Row& x, const Row& y) {
        if(x.first!=y.first) return x.first>y.first;
        return x.second<y.second;
    });
```

Обсуждение: возможный вывод

Наблюдение относится к конкретному входу и реализации; оно не добавляет гарантию стандарту. Для сохранения порядка равных баллов выбираем `stable_sort` по баллу. Если требование — метка по алфавиту, задаём вторичный ключ явно. Пробовать некорректный `<=` ради сравнения нельзя: результат не является допустимым тестом корректного алгоритма.

Индивидуальная проверка: ответ и критерии

`std::stable_sort`; знак `<` для возрастания; `false`; $O(n \log n)$. По одному баллу. Это четыре разных свойства контракта, а не результаты одного теста.

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: `vector`, пары, `lambda` и учебные алгоритмы сортировки. 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
Алгоритмы сортировки	Проверяем две библиотечные операции	Использовать <code>std::sort</code> и <code>std::stable_sort</code> с корректным компаратором, проверить ключи и сопоставить гарантии сложности.
Структурирование решения и отладка	Два направления в одном ключе	Исходник, журнал обычных и граничных тестов, самостоятельное объяснение

Источники и права на изображения

Исходная программа курса

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

Проект стандарта C++: `sort` и `stable_sort`

<https://eel.is/c++draft/alg.sort>

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.