



Урок 26. C++: Создание списка или массива

Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 26 из 33 · 45 минут · Модуль «Массивы и списки»

Уровень: начинающий программист; C++17.

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

Нужно создать таблицу квадратов индексов. Чем отличаются `vector<int>(4,0)` и `vector<int>{4,0}`, хотя в обоих выражениях встречаются 4 и 0?

Результат урока: Создать `vector` нужного размера, заполнить его по индексам и получить независимую копию с дополнительным элементом.

01. Проблемная ситуация

0–5 минут. Для четырёх мест нарисуйте индексы 0,1,2,3 и запишите их квадраты. Предскажите результат для нуля мест. Объясните, почему число элементов и последнее значение здесь различаются.

02. Понятия и разобранный пример

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

Размер конструктора и список значений

`std::vector<int>` — изменяемая последовательность целых с размером, определяемым во время выполнения. `vector<int> a(n,0)` создаёт `n` нулей. Запись `vector<int> a{4,0}` задаёт два элемента 4 и 0. Скобки здесь меняют смысл конструкции.

`a.size()` — число элементов; для содержимого допустимы индексы от 0 до `size()-1`. `operator[]` не выполняет обычную проверку границ в C++17: выход за границы недопустим. Цикл `i<a.size()` не обращается к пустому вектору. Создание ёмкости `reserve` не создаёт элементы; здесь сразу задаём размер.

Присваивание `b=a` копирует значения `vector<int>`: изменение `b[0]` не изменит `a[0]`. `push_back` добавляет новый элемент, в отличие от присваивания существующему `a[i]`. В этом уроке `n` ограничено 0..8, поэтому квадраты индексов не превышают 49.

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

`n=4`: после создания `[0,0,0,0]`, затем записи `i*i` дают `[0,1,4,9]`. При `n=0` результат пуст. Сначала проверяем `n`, чтобы отрицательное число не превратилось в огромный беззнаковый размер.

```
#include <iostream>
#include <vector>
std::vector<int> squares(int n) {
    std::vector<int> a(n, 0);
    for (std::size_t i = 0; i < a.size(); ++i) {
        a[i] = static_cast<int>(i * i);
    }
    return a;
}
int main() {
    int n;
    if (!(std::cin >> n) || n < 0 || n > 8) {
        std::cout << "error\n"; return 0;
    }
    auto a = squares(n);
    std::cout << a.size() << " ";
    for (int x : a) { std::cout << " " << x; }
    std::cout << "\n";
}
```



Размер $n = 4$	0 1 2 3 — индексы
После создания	0 0 0 0
После заполнения	0 1 4 9
Список {4,0}	всего 2 элемента: 4 и 0

Два способа создания vector: размер с начальным значением и явный список элементов.

03. Практика 1. Проверяем материал

Таблица квадратов по индексам

14–25 минут. Выполните восемь отдельных запусков с таблицей. До запуска нарисуйте вектор для $n=2$ и $n=4$; запишите допустимые индексы.

Ввод n	Вывод: размер и значения
0; 1	0: и 1: 0
2; 4	2: 0 1 и 4: 0 1 4 9
8	8: 0 1 4 9 16 25 36 49
-1; 9; x	error для каждого

2 минуты — прогноз, 4 — сборка и запуск, 3 — замените круглые скобки создания на $\{n,0\}$ и найдите контрпример, 2 — верните правильный код. Критерии: размер, значения, пустой случай и объяснение различия конструкторов.

04. Практика 2. Перенос способа

Копия и контрольная сумма в конце

25–33 минуты. Функция `with_checksum(a)` должна вернуть копию исходного `vector<int>` с суммой его элементов в конце. Исходник не менять. Вход содержит не более 8 целых от -9 до 9. За 2 минуты разберите $[2,5,1]$, $[], [-2,2]$; за 4 реализуйте, за 2 проверьте. Критерии: копия, сумма, новая длина $len+1$ и пустой вход.

05. Обсуждение и границы вывода

33–40 минут. За 2 минуты сравните копию `vector<int>` с $b=a$ у списка Python. За 3 объясните, почему одинаковый знак `=` не гарантирует одинаковую семантику в языках. За 2 выберите тест, который обнаружит случайное изменение исходника.

06. Индивидуальная проверка

40–45 минут. Для $n=3$ укажите `squares`, последний индекс, `with_checksum` от этого вектора и длину нового результата. Четыре пункта — четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

После `{n,0}` размер всегда 2. Например, `n=4` даст после цикла `[0,1]`, а не `[0,1,4,9]`; `n=0` тоже создаст два элемента. Проверка `n` остаётся в `main`. Функция `squares` предполагает уже проверенное `0<=n<=8`.

Практика 2: разбор и критерии

Результаты `[2,5,1,8]`, `[0]`, `[-2,2,0]`. Исходные векторы не меняются. Даже у пустого вектора сумма 0, поэтому результат содержит один элемент. Сумма по заданному контракту лежит в `-72..72`.

```
#include <iostream>
#include <vector>
std::vector<int> with_checksum(const std::vector<int>& a) {
    auto b = a;
    int total = 0;
    for (int x : a) { total += x; }
    b.push_back(total);
    return b;
}
int main() {
    for (auto a : std::vector<std::vector<int>>{{2,5,1},{},{-2,2}}) {
        auto b = with_checksum(a);
        for (int x : b) { std::cout << x << " "; }
        std::cout << "| original-size=" << a.size() << "\n";
    }
}
```

Обсуждение: возможный вывод

У `vector<int>` копируются элементы, у списка Python присваивание второго имени само по себе не копирует список. Проверяйте не только возвращаемое значение, но и исходный объект после вызова. Для контрольной суммы полезны непустой и пустой входы.

Индивидуальная проверка: ответ и критерии

`[0,1,4]`; индекс 2; `[0,1,4,5]`; длина 4. Каждое значение подтверждается индексом или суммой `0+1+4=5`.

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: целые числа, `for`, индексы и функции C++. 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 участием 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
Массивы и списки	Создать vector нужного размера, заполнить его по индексам и получить независимую копию с дополнительным элементом.	Исходник, трассировка и таблица проверок
Создание, ввод и обработка массивов; перебор элементов	Копия и контрольная сумма в конце	Самостоятельный алгоритм, ключ и индивидуальная проверка

Источники и права на изображения

Исходная программа курса

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

Рабочий текст стандарта C++: vector

<https://eel.is/c++draft/vector.overview>

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.