



Урок 34. C++: Графический мини-проект

Часть 1. Материалы ученика

Программирование на Python и C++ · 7 класс

Урок 34 из 34 · 45 минут · Модуль «Повторение и обобщение: марафон»

Уровень: начинающий программист; C++17.

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

Три кружка записали 2, 5 и 3 выполненных задания. Построим диаграмму, у которой высота действительно передаёт количество, а не украшает страницу.

Результат урока: Создать SVG-диаграмму через процедуру, проверить масштаб и границы, защитить проект по тестам.

01. Проблемная ситуация

0–5 минут. Даны учебные количества 2, 5, 3. Автор рисует столбики высотой 40, 50, 60. Найдите нарушение: какое количество должно иметь самый высокий столбик? Выберите единый масштаб 20 единиц SVG на задание и запишите три высоты. Обсудите, почему красивый рисунок ещё не доказывает правильность.

02. Понятия и разобранный пример

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

Один масштаб и общий нижний край

SVG — текстовое описание векторного рисунка. Координата x растёт вправо, y вниз. Нижний край всех столбиков равен 110, поэтому высота $h=20\cdot\text{value}$, а верх $y=110-h$. При $\text{value}=0$ прямоугольник имеет нулевую высоту: это допустимый нулевой результат, не потерянные данные.

Процедура `bar` получает поток вывода, x и value . Она описывает один прямоугольник и не читает вход. Главная программа проверяет данные, открывает файл и вызывает процедуру. Передавать поток по ссылке позволяет писать в один общий файл. Цвет и ширина одинаковы, поэтому сравнение зависит только от значения.

Работающий проект включает контракт, код и доказательства проверки. Контракт здесь: три целых от 0 до 5; выход `chart.svg` размером 180×130 , столбцы шириной 30 на $x=20, 70, 120$. При ошибке данных — `error`, запись нового файла не начинается. Старый `chart.svg` после ошибки не является результатом нового запуска: проверяйте в отдельной чистой папке.

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 высоты равны 40, 100, 60; $y=70, 10, 50$. Нижние края: $70+40=110, 10+100=110, 50+60=110$.

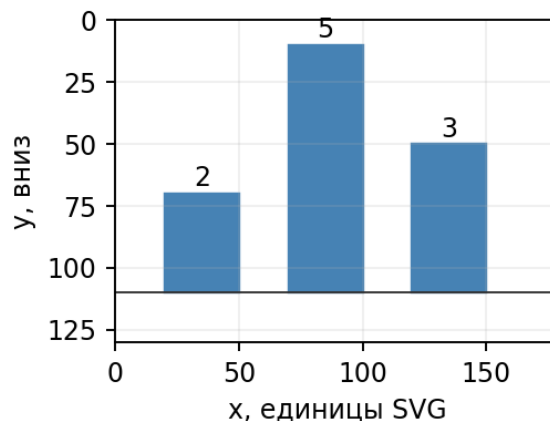
Компилируйте `main.cpp`, введите три числа, затем откройте `chart.svg` в браузере. Программа сообщает имя файла только после успешного закрытия.

```
#include <iostream>
#include <fstream>
void bar(std::ostream& out, int x, int value) {
    int h = 20 * value;
    out << "<rect x='" << x << "' y='" << 110-h
        << "' width='30' height='" << h
        << "' fill='steelblue'/>\n";
}

int main() {
    int a, b, c;
    if (!(std::cin >> a >> b >> c) || a < 0 || a > 5
        || b < 0 || b > 5 || c < 0 || c > 5) {
        std::cout << "error\n"; return 0;
    }
    std::ofstream out("chart.svg");
    if (!out) { std::cout << "file error\n"; return 0; }
    out << "<svg xmlns='http://www.w3.org/2000/svg' "
        "width='180' height='130'>\n";
    bar(out, 20, a); bar(out, 70, b); bar(out, 120, c);
    out << "</svg>\n";
}
```



```
out.close();  
std::cout << (out ? "chart.svg\n" : "file error\n");  
}
```



Учебные данные 2, 5, 3: высоты 40, 100, 60; общий нижний край $y=110$, ось y направлена вниз.

03. Практика 1. Проверяем материал

Три входа и протокол проекта

14–25 минут. Соберите проект из примера. Переименуйте рабочую папку, чтобы не спутать старый результат с новым.

Ввод	Что проверить
2 5 3	h: 40,100,60; y: 70,10,50
0 0 0	Три элемента rect высотой 0
5 5 5	Три равных столбца высотой 100
-1 2 3; 6 2 3; x	error; новый файл не создаётся

2 минуты — прогноз таблицы; 4 — ввод и сборка; 3 — тесты и просмотр SVG как изображения и текста; 2 — взаимная защита. Продукт: исходник, три корректных SVG в разных папках и журнал четырёх групп тестов. Критерии: верные координаты, общий масштаб, обработка ошибки, объяснение роли процедуры.

04. Практика 2. Перенос способа

Цикл вместо повторных вызовов

25–33 минуты. Перенесите проект на четыре столбца: данные 1,4,0,2, масштаб 15, нижняя линия 90, $x=15+40 \cdot i$ при $i=0 \dots 3$, ширина 25, поле 180×110 . Сначала прочитайте все четыре числа и проверьте диапазон $0 \dots 5$; только затем создайте файл. Используйте цикл и массив из четырёх элементов. За 2 минуты вычислите координаты, за 4 доработайте код, за 2 проверьте. Критерии: 4 элемента, цикл, новая геометрия и защита ввода.

05. Обсуждение и границы вывода

33–40 минут. Первая пара защищает тезис «Достаточно открыть красивую диаграмму». Вторая предъявляет нулевой тест и текст SVG. За 2 минуты подготовьте доказательство, за 3 обменяйтесь возражениями, за 2 сформулируйте критерий готовности проекта. Можно ли сравнивать столбики, если каждому назначен свой масштаб?

06. Индивидуальная проверка

40–45 минут. Для масштаба 12, нижнего края 80 и $value=4$ найдите h и y . Для $x=145$ и ширины 25 проверьте размещение в поле шириной 180. Назовите один граничный тест и объясните его цель. По баллу за каждый из четырёх элементов.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

В корректных файлах ровно три rect. Нулевая высота не видна, но элемент есть в тексте. Ошибочные входы печатают error до открытия файла. Если столбцы перевёрнуты, проверьте $110-h$, а не $110+h$. Ввод считается последовательностью трёх целых; дополнительные слова после них этот контракт не анализирует. Полный код находится в разобранном примере.

Практика 2: разбор и критерии

Высоты 15,60,0,30; $y=75,30,90,60$; $x=15,55,95,135$. Максимальный правый край 160 меньше 180. Максимальная высота 75 оставляет верхний отступ 15. Полный вариант ниже; массив здесь служит для сохранения уже проверенных входов.

```
#include <iostream>
#include <fstream>
void bar(std::ostream& out, int x, int value) {
    int h=15*value;
    out << "<rect x='" << x << "' y='" << 90-h
        << "' width='25' height='" << h
        << "' fill='teal'/>\n";
}

int main() {
    int values[4];
    for (int i=0; i<4; ++i) {
        if (!(std::cin >> values[i]) || values[i]<0
            || values[i]>5) {
            std::cout << "error\n"; return 0;
        }
    }
    std::ofstream out("chart.svg");
    if (!out) { std::cout << "file error\n"; return 0; }
    out << "<svg xmlns='http://www.w3.org/2000/svg' "
        "width='180' height='110'>\n";
    for (int i=0; i<4; ++i) bar(out,15+40*i,values[i]);
    out << "</svg>\n"; out.close();
    std::cout << (out ? "chart.svg\n" : "file error\n");
}
```

Обсуждение: возможный вывод

Внешний вид не выявляет отсутствующий нулевой элемент и не гарантирует точный масштаб. Нужны изображение, координаты и тесты. Разные масштабы делают прямое сравнение высот неверным. Защита проекта опирается на контракт и проверку, а не на количество строк кода.

Индивидуальная проверка: ответ и критерии

$h=48$; $y=32$; правый край $170 \leq 180$, помещается; $value=0$ проверяет нулевую высоту (или $value=5$ — максимальную). 4 балла. При ошибке знака вернитесь к направлению оси y .

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`). Для SVG нужен обычный браузер и отдельная папка с правом записи для каждого теста. 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, 7 класс. Предпосылки: ввод, if, цикл for, процедура с параметрами и SVG. 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
Повторение и обобщение: марафон	Разбирает пример и создаёт две подпрограммы	Исходники, прогнозы и протокол проверки
Составление и отладка программ	Сравнивает фактический результат с контрактом	Объяснение границ и самостоятельный ответ

Источники и права на изображения

Исходная программа курса

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

W3C SVG 2: геометрия прямоугольника

<https://www.w3.org/TR/SVG2/shapes.html#RectElement>

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.