



Урок 31. C++: рефакторинг рисунка

Часть 1. Материалы ученика

Программирование на Python и C++ · 7 класс

Урок 31 из 34 · 45 минут · Модуль «Процедуры: параметризованный SVG»

Уровень: начинающий программист; C++17.

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

Два домика имеют один принцип формы, но разные размеры. Как заменить два набора координат вызовами и доказать, что рисунок сохранился?

Результат урока: Собрать SVG из параметризованной процедуры и проверить рамки, цвета и отдельное расширение рисунка.

01. Проблемная ситуация

За 5 минут сравните два дома: первый опора (10,10), ширина 20; второй (60,20), ширина 30. В обоих крыша занимает верхнюю половину квадратной рамки, стены — нижнюю. Вычислите вершины крыш и рамки стен до чтения кода.

02. Понятия и разобранный пример

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

Относительные координаты вместо копий команд

Рефакторинг меняет организацию кода, сохраняя требуемый результат. Процедура `house` получает поток `out`, опоры `x,y`, чётную ширину `w` и цвет стен. Рамка `x...x+w,y...y+w`; верх стен `y+w/2`, высота стен `w/2`; вершина крыши `x+w/2,y`. Вызовы ниже используют только допустимые заранее заданные значения.

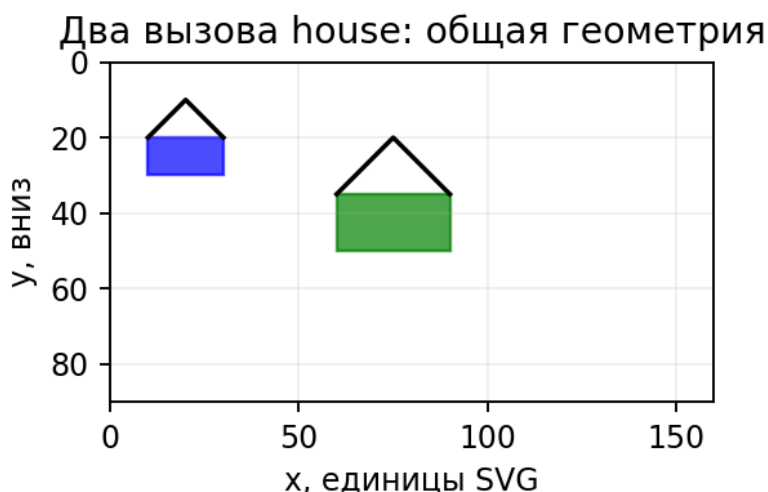
Ссылка `std::ostream&` передаёт в процедуру существующий поток файла: его нельзя просто копировать, как число. `const std::string&` позволяет читать строку цвета без её изменения. Эти записи даны как готовый интерфейс: для работы достаточно понимать, что процедура пишет в тот же открытый `out`.

SVG — текст с векторными фигурами, его открывают браузером. Файл сохраняется в рабочей папке, исходник `.cpp` открывают редактором. Цвета в уроке заданы константами, пользовательский текст в XML не вставляется. Наличие файла ещё не проверка рисунка: нужны рамки, цвета, число элементов и отсутствие обрезания.

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

Первый дом: стены (10,20), ширина 20, высота 10; вершина (20,10). Второй: стены (60,35), ширина 30, высота 15; вершина (75,20). Полная рамка второго заканчивается (90,50). Оба внутри 160×90. Код создаёт `houses.svg`.

```
#include <fstream>
#include <string>
void house(std::ostream& out, int x, int y, int w,
           const std::string& color) {
    out << "<rect x='" << x << "' y='" << y << " w='" << w << " height='" << w / 2
    << "' width='" << w << "' height='" << w / 2
    << "' fill='" << color << "'>\n";
    out << "<polyline points='" << x << "," << y << " w / 2
    << " " << x + w / 2 << "," << y << " " << x + w
    << "," << y << " w / 2
    << "' fill='none' stroke='black'/>\n";
}
int main() {
    std::ofstream out("houses.svg");
    out << "<svg xmlns='http://www.w3.org/2000/svg' "
    << "width='640' height='360' viewBox='0 0 160 90'>\n";
    house(out, 10, 10, 20, "blue");
    house(out, 60, 20, 30, "green");
    out << "</svg>\n";
}
```



SVG-домики: рамки (10,10)–(30,30) и (60,20)–(90,50). Крыша и стены масштабируются общей шириной.

03. Практика 1. Проверяем материал

Третий дом одним вызовом

14–25 минут. Добавьте дом В только вызовом процедуры. Сохраните прежние два вызова и одно определение house.

Дом	Опора, ширина, цвет
А	10,10,20,blue
Б	60,20,30,green
В	110,10,40,red

2 минуты — рамка В; 4 — программа; 3 — запуск и открытие SVG; 2 — сравнение с прогнозом. Продукт: .cpp, SVG и три строки проверки. Критерии: три дома, одно тело процедуры, красный В полностью внутри, старые дома не изменились.

04. Практика 2. Перенос способа

Новое окно — расширение, а не сохранение прежнего рисунка

25–33 минуты. Отдельный новый вариант: один зелёный дом с опорой (20,15), $w=40$. Добавьте белое квадратное окно: $x+w/4, y+5w/8$, сторона $w/4$. За 2 минуты найдите его координаты; за 4 измените только тело и вызов; за 2 откройте файл и проверьте окно внутри стен. Критерии: окно задаётся относительно опоры, ширина участвует в размере, старые две фигуры дома сохранены.

05. Обсуждение и границы вывода

33–40 минут. Достаточно ли меньшего количества строк для удачного рефакторинга? Защитите проверку по старым домам; возразите против «похожего» рисунка без координат; назовите новый тест для будущей ширины 40.

06. Индивидуальная проверка

40–45 минут. Для `house(out,80,5,40,"blue")` назовите вершину крыши, верх стен, правый и нижний края. По одному баллу за величину; исходный дом без окна.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

У В стены $x=110\dots150, y=30\dots50$; вершина (130,10). В XML три rect и три polyline. Правый край $150<160$, низ $50<90$. Просто уменьшить масштаб просмотра не исправляет обрезание координатами SVG.

```
#include <fstream>
#include <string>
void house(std::ostream& out, int x, int y, int w,
           const std::string& color) {
    out << "<rect x='" << x << "' y='" << y + w / 2
        << "' width='" << w << "' height='" << w / 2
        << "' fill='" << color << "'/>\n";
    out << "<polyline points='" << x << "," << y + w / 2
        << " " << x + w / 2 << "," << y << " " << x + w
        << "," << y + w / 2
        << "' fill='none' stroke='black'/>\n";
}
int main() {
    std::ofstream out("houses.svg");
    out << "<svg xmlns='http://www.w3.org/2000/svg' "
        << "width='640' height='360' viewBox='0 0 160 90'>\n";
    house(out, 10, 10, 20, "blue");
    house(out, 60, 20, 30, "green");
    house(out, 110, 10, 40, "red");
    out << "</svg>\n";
}
```

Практика 2: разбор и критерии

Стены $x20\dots60, y35\dots55$; окно $x30\dots40, y40\dots50$, сторона 10. Оно полностью внутри стен. Код p2 — новый полный исходник, заменяющий прежний, не приписка после второго main. Добавление окна меняет требуемый рисунок, значит это расширение после проверенного рефакторинга.

```
#include <fstream>
#include <string>
void house(std::ostream& out, int x, int y, int w,
           const std::string& color) {
    out << "<rect x='" << x << "' y='" << y + w / 2
        << "' width='" << w << "' height='" << w / 2
        << "' fill='" << color << "'/>\n";
    out << "<polyline points='" << x << "," << y + w / 2
        << " " << x + w / 2 << "," << y << " " << x + w
        << "," << y + w / 2
        << "' fill='none' stroke='black'/>\n";
    out << "<rect x='" << x + w / 4 << "' y='" << y + 5 * w / 8
        << "' width='" << w / 4 << "' height='" << w / 4
        << "' fill='white'/>\n";
}
int main() {
    std::ofstream out("houses.svg");
    out << "<svg xmlns='http://www.w3.org/2000/svg' "
        << "width='640' height='360' viewBox='0 0 160 90'>\n";
    house(out, 20, 15, 40, "green");
    out << "</svg>\n";
}
```

Обсуждение: возможный вывод

Проверяют сохранение рамок, цветов и порядка примитивов старых вызовов. Новое окно отдельно проверяется на вложенность. Копирование только двух готовых SVG-картинок не демонстрирует параметризованную процедуру. Команды остаются в одном теле, данные вызовов различаются.

Индивидуальная проверка: ответ и критерии

Вершина (100,5), верх стен $y=25$, правый край $x=120$, нижний $y=45$. Четыре верных элемента — 4 балла.



Учителю: подготовка и план на 45 минут

Подготовьте компьютер, рабочую папку с правом записи, редактор, распечатку ученической части PDF, бумагу и ручку. Локальный компилятор G++ с C++17 или настроенная учителем IDE; сборка `g++ -std=c++17 -Wall -Wextra -pedantic main.cpp -o lesson`. В Windows задайте `lesson.exe`. Запуск `./lesson` в Linux/macOS, `.\lesson.exe` в Windows. Для урока 31 дополнительно обычный браузер: открывать локальный SVG, не C++-исходник. Все входы и ожидаемые результаты даны. Установку и проверку запуска учитель выполняет до занятия; детские аккаунты и интернет не нужны.

Уровень — начинающий программист, 7 класс. Предпосылки: void-процедуры, координаты SVG, строки и ofstream. В паре автор вводит и объясняет, проверяющий сопоставляет с прогнозом, затем роли меняются. Без компьютера трассируйте полный код на бумаге, отмечайте координаты по схеме и проверяйте готовые результаты; учитель демонстрирует заранее собранный пример. Бумажная трассировка не выдаётся за реальную компиляцию или запуск ученика. План 45 минут включает чтение, ввод, сборку и обсуждение. В графических задачах сначала прогноз, затем сравнение. 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
Процедуры с параметрами. Рефакторинг	Собрать SVG из параметризованной процедуры и проверить рамки, цвета и отдельное расширение рисунка.	Самостоятельный продукт и тесты, описанные в практиках
Составлять и отлаживать простые диалоговые программы	Сверяет прогноз с фактическим запуском; объясняет ошибочный вариант	Журнал тестов и индивидуальный ответ по критериям

Источники и права на изображения

Исходная программа курса

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

W3C SVG: прямоугольники и ломаные

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

C++: определения функций

<https://eel.is/c++draft/dcl.fct.def.general>

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: 17.10.2026.