Введение в тестирование
Конспект

Яндекс Практикум

Принципы тестирования и отладки кода

Внимательно читайте задание.
Убедитесь, что поняли, как должна работать программа.

Проверьте поведение программы на примерах из условия.
Если решение даёт не тот результат, который ожидается, в коде ошибка. Если результат ожидаемый, это тоже не гарантия верного решения. Тестирование показывает наличие ошибок, но не доказывает, что их нет.

Вносите изменения в код маленькими порциями.
И тестируйте после каждого изменения. Так вы быстрее обнаружите и исправите ошибку.

Составьте план тестирования.
- Проанализируйте условия задачи и выделите классы эквивалентности. Класс эквивалентности — одно или несколько значений входных данных, к которым программа применяет одинаковую логику.
- Проверьте работу программы на небольшом наборе входных данных из каждого класса эквивалентности.

Проверьте, как программа работает на граничных условиях.
Если значение целочисленного параметра должно быть в диапазоне от 0 до 100 включительно, проверьте поведение программы при значениях вблизи границ: 0, 1, 99, 100.
Если параметр строковый, протестируйте код на пустых строках и строках с одним символом.
Если имеете дело с контейнером типа vector, set и map, проверьте, как программа себя ведёт с пустыми контейнерами.

Юнит-тестирование функции

Юнит-тестирование (модульное тестирование) автоматизирует проверку отдельных модулей программы — классов и функций. Дополнительно к основному коду пишут модульные тесты — вспомогательный код, проверяющий поведение классов и функций.

Чтобы написать юнит-тесты для программы, её сначала декомпозируют — выделяют функции в коде и дают им понятное название. Затем для функции пишут модульный тест. Для проверок применяют макрос assert из библиотеки <cassert>.

Макросы — специальные конструкции языка C++. Они выполняются на этапе предварительной обработки исходного текста программы.

Макрос assert вставляет в код проверку логического условия, нарушение которого говорит об ошибках в коде. Если во время выполнения программы условие истинно, она продолжит работать. Если нет — программа выведет сообщение об ошибке и аварийно завершит работу. То есть упадёт юнит-тест.

Сообщение об ошибке, выведенное макросом assert, содержит имя файла и номер строки с условием, не прошедшим проверку. Это помогает проанализировать причины ошибки и исправить её.

Юнит-тестирование класса

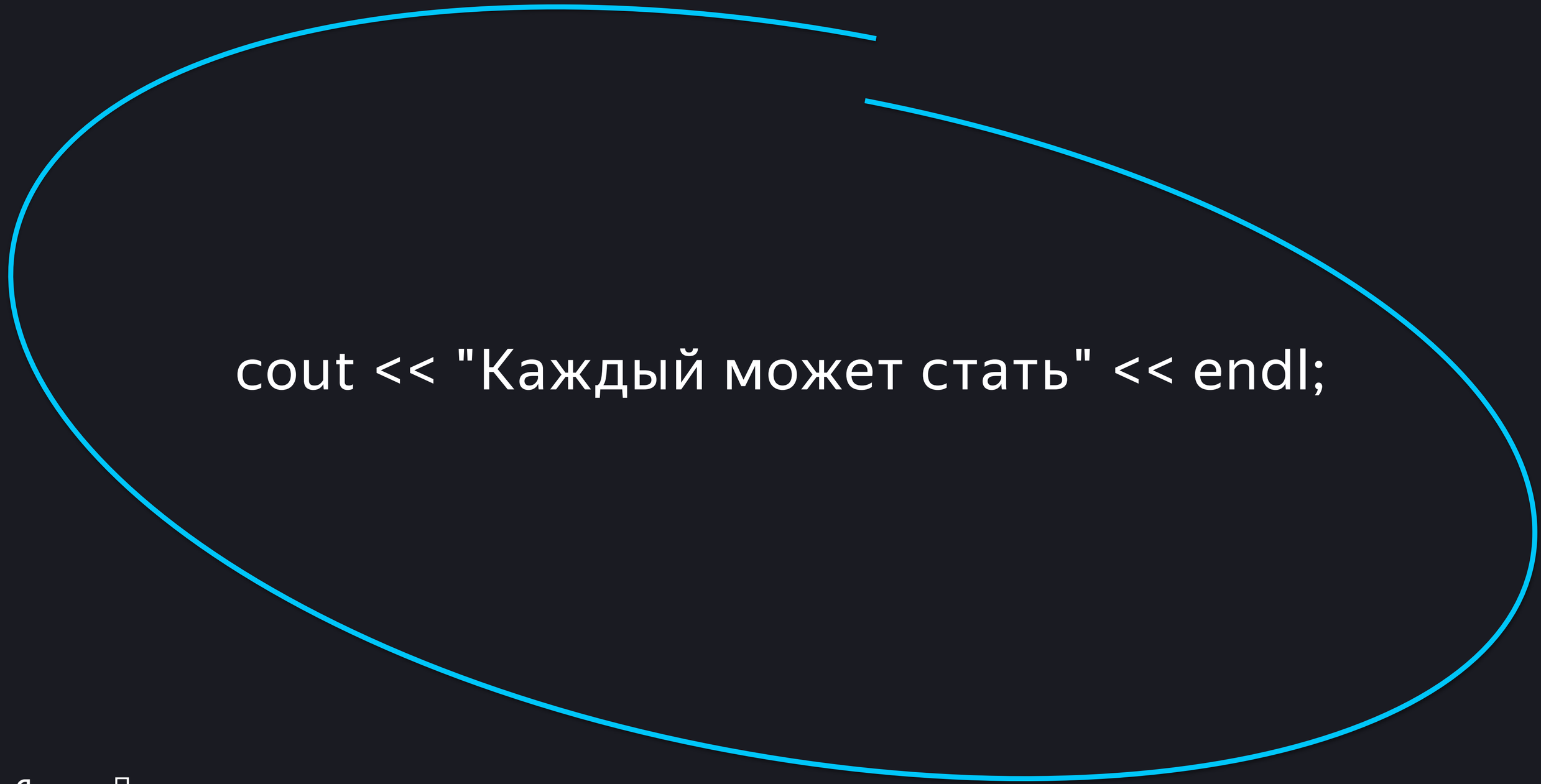
sstream (от англ. string stream) — заголовочный файл (библиотека), который содержит переменные, функции и классы и позволяет считывать данные из строк и выводить данные в строки. Входит в стандартную библиотеку C++.

Класс istringstream — это строковый поток ввода,
Класс ostringstream — строковый поток вывода.

Разработка через тестирование (test-driven development, TDD) — написание теста до написания кода:
- пишем небольшой тест, предъявляющий к тестируемому классу или функции новые требования;
- запускаем тест, чтобы убедиться: работает и проверяет ещё не реализованный функционал;
- дорабатываем модуль так, чтобы успешно пройти новый тест и все ранее написанные;
- пишем новый тест, и процесс начинается сначала.

TDD стимулирует декомпозицию программу на независимые функциональные модули, которые легче тестировать.

Юнит-тестирование класса похоже на тестирование функции: тест выполняет некоторые операции над экземпляром класса, а затем применением assert сравнивает состояния класса с ожидаемым.


cout << "Каждый может стать" << endl;