

Тестирование и отладка

Юрий Литвинов
y.litvinov@spbu.ru

15.09.2023

Тестирование

- ▶ Любая программа содержит ошибки
- ▶ Если программа не содержит ошибок, их содержит алгоритм, который реализует эта программа
- ▶ Если ни программа, ни алгоритм ошибок не содержат, такая программа даром никому не нужна

Тестирование не позволяет доказать отсутствие ошибок, оно позволяет лишь найти ошибки, которые в программе присутствуют

Виды тестирования

- ▶ По уровню тестируемых компонент
 - ▶ Модульное
 - ▶ Интеграционное
 - ▶ Системное
- ▶ По целям
 - ▶ Функциональное
 - ▶ Нагрузочное
 - ▶ Удобства использования
 - ▶ Смоук-тестирование
 - ▶ Регрессионное
 - ▶ Приёмочное тестирование

Тестирование, выполняемое программистами

- ▶ Тестирование типичного сценария работы
- ▶ Тестирование граничных случаев
- ▶ Тестирование некорректных входных данных
 - ▶ Программа должна адекватно себя вести и сообщать об ошибках ввода
- ▶ Тестирование должно быть по возможности автоматическим
 - ▶ На самом деле, используются модульные тесты
 - ▶ Можно писать модульные тесты вручную, как функции, возвращающие true/false

Пример типичного теста

```
bool balanceOfParentheses(const char* parentheses) {  
    ...  
}  
  
bool testCorrectCase() {  
    return balanceOfParentheses("(");  
}  
  
bool testIncorrectCases() {  
    return !balanceOfParentheses("(") && !balanceOfParentheses(")");  
}  
  
void main() {  
    if (!testCorrectCase() || !testIncorrectCases()) {  
        printf("Tests failed\n");  
        return;  
    }  
    printf("Enter string\n");  
    ...  
}
```

Отладка

- ▶ Устойчивое воспроизведение ошибки
 - ▶ Вместо `srand(time(nullptr))` – `srand(<какое-то фиксированное значение>)`
 - ▶ Ошибка должна воспроизводиться быстро
- ▶ Локализация ошибки
 - ▶ Аналитически
 - ▶ Отладка
- ▶ Отладочная гипотеза
 - ▶ Похоже на научный подход — гипотеза, эксперимент, уточнение, эксперимент и т.д.
 - ▶ Тестовый прогон с отладочной печатью
 - ▶ Тестовый прогон под отладчиком

Демонстрация