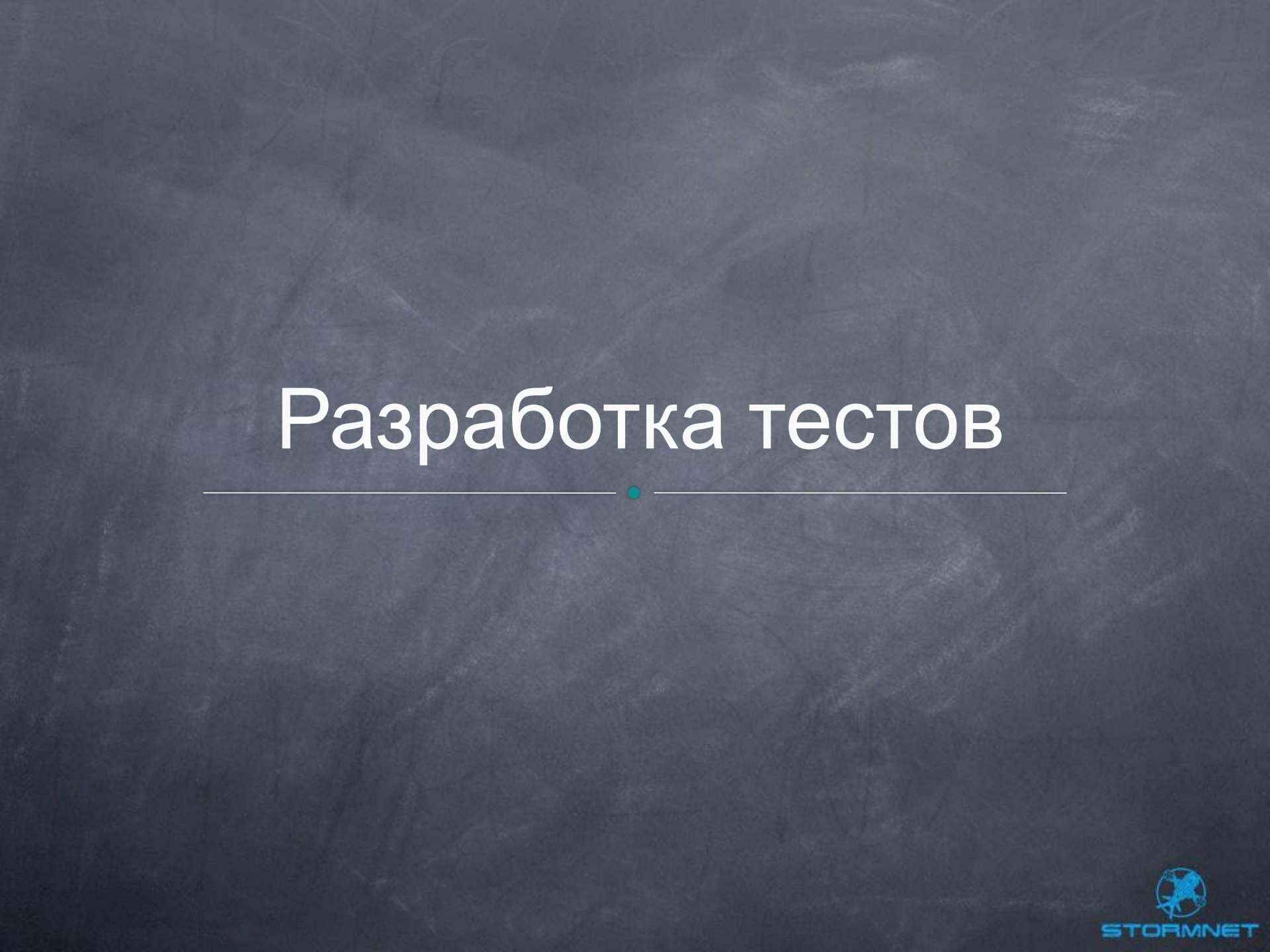
# Разработка тестов



# Какие бывают тесты

Перед вами обыкновенная ручка. Давайте подумаем, как её можно протестировать?

# Тестирование ручки

- Тесты на основе требований (requirements based tests)
  - Извлекается и вставляется ли в ручку стержень?
  - Присутствует ли держатель, позволяющий цеплять ручку за край кармана?
  - Переключается ли ручка из рабочего в нерабочее положение?

- Функциональные тесты (functional test)
  - Вставить в ручку стержень.
  - Переключить в рабочее положение.
  - Написать несколько слов.
  - Переключить в нерабочее положение.
  - Извлечь стержень.

# Тестирование ручки

- Сравнительные («параллельные») тесты (parallel testing)
  - Что мы можем сказать об этой ручке в сравнении с другими ручками, которые выпускает наша фирма?
  - Что мы можем сказать об этой ручке в сравнении с ручками, которые выпускают конкуренты?
  - В чём преимущества именно этой модели ручек?
- Сценарные тесты (scenario tests). Как ручку может использовать:
  - Секретарь.
  - Преподаватель.
  - Студент.
  - Школьник.
  - Прораб.
  - Сантехник.
  - Милиционер.
  - Моряк.
  - …

# Тестирование ручки

- Тесты ошибочных ситуаций (fault injection tests)
  - Что произойдёт, если препятствовать выходу стержня в рабочее положение?
  - Какое усилие и где надо приложить к ручке, чтобы её сломать?
  - Если стержень застрял, легко ли его извлечь?
  - Что произойдёт, если писать по стеклу, асфальту?
- Тесты интерфейса (interface tests, GUI tests)
  - Измерения: высота, ширина, длина, вес.
  - Цвет.
  - Читаемость логотипа фирмы-производителя.
- Тесты удобства использования (usability tests)
  - Есть ли у нас какие-либо замечания по юзабилити ручек от пользователей?
  - Как много времени у пользователя занимает переключение ручки из нерабочего положения в рабочее и обратно?
  - Как быстро пользователь понимает, как пользоваться ручкой?
  - Как быстро пользователь привыкает к этой ручке?
  - Легко ли понять, какие стержни подходят к ручке?
  - Легко ли заменить стержень?
  - Может ли ручкой пользоваться левша?
  - Не смазываются ли чернила, если пишет левша?

# Тестирование ручки

- Тесты упаковки и документации (packaging/documentation tests)
  - Вложена ли в упаковку копия текста о гарантийных обязательствах?
  - Ясно ли видно на упаковке, что внутри?
  - Легко ли открыть упаковку?
  - Насколько материалы упаковки вредны для окружающей среды?
  - Есть ли какие-то особые требования к упаковке?
  - На сайте, в каталоге, на упаковке написано и нарисовано одно и то же?
  - Текст на упаковке и в гарантийном обязательстве – на одном и том же языке?
  - На упаковке и в документации нет грамматических ошибок, опечаток и т.д.?

- Стрессовые тесты (stress tests)
  - При какой температуре расплавится пластиковая часть ручки?
  - При какой температуре потечёт стержень?
  - При какой температуре ручка перестаёт писать?
  - Какое воздействие необходимо применить к ручке, чтобы сломать её?
  - Пишет ли ручка под водой? А по мокрой бумаге?
  - Если ручку уронить в песок – что произойдёт?
  - А если уронить со стола?
  - А если из окна офиса?

# Тестирование ручки

- Тесты производительности (performance tests)
  - Сколько текста можно написать ручкой в единицу времени?
  - Как быстро ручку можно привести в рабочее положение?
  - Как много раз ручку можно переключить из нерабочего в рабочее положение, прежде чем её начнёт заедать?
- Конфигурационные тесты (configuration tests)
  - Какие стержни подходят к нашей ручке?
  - На каких поверхностях она может писать?
- Законодательные тесты (regulation tests)
  - Подлежит ли этот продукт какому-то виду лицензирования?
  - Необходима ли какая-то особая сопроводительная документация?
  - Ясно ли из документации ручки видно, в какой стране она произведена?
  - Существуют ли какие-то законодательные особенности, препятствующие распространению нашего продукта?

# Классы эквивалентности

- **Класс эквивалентности** (equivalence class) – набор тестов, полное выполнение которого является избыточным и не приводит к обнаружению новых дефектов.

# Признаки эквивалентности

- Несколько тестов эквивалентны, если:
  - Они направлены на поиск одной и той же ошибки
  - Если один из тестов обнаруживает ошибку, другие её тоже, скорее всего, обнаружат
  - Если один из тестов НЕ обнаруживает ошибку, другие её тоже, скорее всего, НЕ обнаружат
  - Тесты используют одни и те же наборы входных данных
  - Для выполнения тестов мы совершаем одни и те же операции
  - Тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние (например открытие валидных файлов одного типа и схожего объема, но с разным содержанием)
  - Все тесты приводят к срабатыванию одного и того же блока обработки ошибок («error handling block»)
  - Ни один из тестов не приводит к срабатыванию определенного блока обработки ошибок («error handling block») (например, блок обработки ошибок открытия файла)

# Граничные условия

- Граничные условия (или просто – границы) – это те места, в которых один класс эквивалентности переходит в другой. Например, одна группа тестов вызывает сообщение «вы ввели слишком маленькое число», а другая вызывает сообщение «вы ввели слишком большое число». Граница будет лежать где-то в районе чисел «самых больших из слишком маленьких» и «самых маленьких из слишком больших».

! Граничные условия очень важны, и их обязательно следует проверять в тестах, т.к. именно в этом месте чаще всего и обнаруживаются ошибки

# Граничные условия

- Пример: Необходимо проверить, как работает поле, в которое можно ввести целое число в диапазоне от 1 до 99

  Классы эквивалентности здесь:
  - Любое целое в диапазоне от 1 до 99. Как правило, это будет середина числового отрезка (Позитивный тест)
  - Любое число меньше 1 (Негативный тест)
  - Любое число больше 99 (Негативный тест)
  - Дробь и «не число» (буквы, спецсимволы) (Негативный тест)

# Граничные условия

- Тесты, которые мы выполним:
  - Ввести 1, 99, 50
  - Ввести 0
  - Ввести 100
  - Ввести 50.5
  - Ввести букву
  - Ввести спецсимвол: ~`!”@’#$;%:^&?*()[]{},.\/+=-_

# Пример для обсуждения

- Пример: Открытие файла. *Чтобы добавить файл в свою фотогалерею на сайте, пользователь должен кликнуть по кнопке Открыть, выбрать файл и кликнуть по кнопке OK*

  Какие случаи нам надо будет проверить?

# Пример для обсуждения

- «Корректный» файл
- Очень большой файл
- Несуществующий файл
- «Файл по сети» (Доступный. Недоступный будет эквивалентен несуществующему)
- Уже открытый файл (Нашим приложением и другим приложением)
- Файл неверного формата (По расширению и/или реальному содержимому)
- Пустой файл
- Повреждённый файл

# Выводы

- Классы эквивалентности не всегда очевидны.
- Как правило, негативных тестов получается больше, чем позитивных.
- Принадлежность теста к позитивным или негативным зависит от требований.

# Чек-лист

- Упрощенная форма тест-кейса
- Главный принцип чек-листов заключается в том, что каждый тестировщик по-своему проходит их, расширяя тестовый набор своей экспертизой

| Проверка | Результат | Комментарии |
|---|---|---|
| **Операции с файлами** | ok | |
| Создание файла | ok | |
| Открытие файла | ok | |
| Сохранение документа | ok | |
| Печать | ok | |
| **Редактирование файлов** | bugs | |
| Отмена | ok | |
| Копирование | ok | |
| Вырезание | ok | |
| Вставка | ok | |
| Удаление | ok | |
| Поиск | fail | bug #123 |
| Поиск с заменой | fail | bug #126 |
| Вставка даты | ok | |
| **Форматирование** | ok | |
| Перенос строки | ok | |
| Изменение шрифта | ok | |
| **Справка** | ok | |

# Чек-лист

- Разбивайте приложение на модули (модуль авторизации, модуль настроек и т.д.)
- Используйте «косметику»
- Используйте техники ускорения написания (copy-paste)

# Документирование тестов

- Результатом документирования тестов является **тест-кейс**.
- Набор тест-кейсов – **Test Suite.**

# Определения тест-кейсов

- IEEE Std 610-1990:
  - «A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.»
  («Набор тестовых входных данных, условий выполнения и ожидаемых результатов, разработанных с конкретной целью, такой как проверка некоторого пути выполнения программы или проверка соответствия некоторому требованию.»)

- IEEE Std 829-1983:
  - «Documentation specifying inputs, predicted results, and a set of execution conditions for a test item.»
  («Документ, определяющий набор входных данных, ожидаемых результатов и условий выполнения теста.»)

  - Далее в нашем курсе мы будем рассматривать слова «тест-кейс» и «тест» как синонимы (в случаях, если не оговорено обратное).

# Зачем нужны тест-кейсы

- «Планирование, и только потом – выполнение!» Тест-кейсы дают нам структурированный системный подход, что снижает вероятность пропуска ошибки.
- Тест-кейсы – хороший способ хранения части проектной информации.
- Написание тест-кейсов – один из способов протестировать проектную документацию ещё до выхода первого билда.
- Наличие тест-кейсов значительно ускоряет регрессионное тестирование.
- Тест-кейсы – прекрасный способ быстро ввести в курс дела новичка или сотрудника, только что подключившегося к проекту.
- Имея тест-кейсы, мы можем в любой момент «вспомнить», что мы делали месяц, полгода, год назад.
- Мы можем обмениваться тест-кейсами (и «чек-листами») между проектами.
- Тест-кейсы позволяют легко отслеживать прогресс (X% тестов выполнено, Y% тестов прошло (завалилось), Z% требований покрыто тестами).

STORMNET

# Свойства тест-кейсов

- Тест-кейсы могут быть:
  - Специфичными или общими.
  - Простыми или сложными.
  - Независимыми или связанными друг с другом.
  - Позитивными или негативными.

# Специфичность или общность

- Когда все детали прописаны до мелочей, при повторных выполнениях теста всегда будут выполняться строго одни и те же действия, что снижает вероятность обнаружить ошибку.
- Слишком общий тест-кейс сложно выполнять по многим объективным и субъективным причинам, а потому он вполне может остаться невыполненным.
- Однако интеграционные тесты, как правило, бывают более общими, чем иные. Это связано со спецификой интеграционного тестирования.
- Если в тесте прописано много мелких деталей, возрастает время его создания и поддержки.
- Однако недостаток деталей может усложнить работу новичка.

# Простота или сложность

- Рассмотрим на примере. Где в ниже перечисленном простые тест-кейсы, а где – сложные?
- **Набор 1:**
- 1. Откройте файл «1.txt». Файл открыт.
  2. Введите слово «Дом». Появляется слово «Дом.
  3. Сохраните файл. Кнопка «Сохранить» становится неактивной.
- **Набор 2:**
- 1. В документе размером более 100 Мб создайте таблицу 100×100, в ячейку 50×50 вставьте картинку размером 30 Мб, применив к ней функцию «Авторасположение». Проверьте результат.

# Простота или сложность

- Простые тесты оперируют за раз одним объектом.
- Каковы *преимущества* простых тест-кейсов?
  - Их легко выполнять.
  - Они понятны новичкам.
  - Они упрощают диагностику ошибки.
  - Они делают наличие ошибки очевидным.
- Каковы *преимущества* сложных тест-кейсов?
  - Больше шансов что-то сломать.
  - Пользователи, как правило, используют сложные сценарии.
  - Программисты сами редко проверяют такие варианты.
- Следует постепенно повышать сложность тестов.

# Независимость или связанность

- Каковы *преимущества* *независимого самостоятельного тест-кейса*?
  - Его легко и просто выполнить.
  - Такие тесты могут работать даже после краха приложения на других тестах.
  - Такие тесты можно группировать любым образом и выполнять в любом порядке.
- Каковы *преимущества* наборов *тесно связанных тестов*?
  - Они имитируют работу реальных пользователей.
  - Они удобны для интеграционного тестирования.
  - Они удобны для разбиения на части тестов с большим количеством шагов.
  - Следующий в наборе тест использует данные и состояние приложения, подготовленные предыдущим.
- Промышленным стандартом являются независимые тесты. Использование сценариев не запрещено, но не следует делать их слишком длинными.

# Позитивность или негативность

- **Позитивные тесты** проверяют, что приложение делает то, на что оно рассчитано (т.е. такие тесты используют корректные данные и условия выполнения).
  **Негативные тесты** проверяют работу приложения в нестандартных условиях (при получении некорректных данных или команд или при работе в некорректных условиях).
- Обе разновидности тестов важны и нужны, однако следует помнить последовательность их разработки и выполнения:
  1. Простые позитивные.
  2. Простые негативные.
  3. Сложные позитивные.
  4. Сложные негативные.

# Что должен содержать тест-кейс

- Идентификатор теста (id)
- Связанное с тестом требование (related requirement)
- Краткое заглавие теста (title)
- Модуль и подмодуль приложения, к которым относится тест (module, submodule)
- Приоритет теста (priority: smoke, critical, extended; A, B, C, D)
- Исходные данные, необходимые для теста (initial data) (обычно включается в шаги выполнения)
- Шаги для выполнения теста (steps)
- Ожидаемые результаты (expected results)
- Поле для пометки, прошёл тест или нет (status)
- Последний полученный актуальный результат (actual result), связанный с тестом баг (если есть) (related bug)
- Указать автора теста (author), время последнего выполнения теста (last time run) (часто эта информация указывается в заголовке файла)

# Документирование тестов


| | | | | | | |
|---|---|---|---|---|---|---|
| UG_U 1.12 | A | R9 | Галер ея | Загрузка файла | **Галерея, загрузка файла, имя со спецсимволами**<br>Приготовления: создать непустой файл с именем #$%^&.jpg<br>1. Нажать кноп картинку»<br>2. Нажать кноп<br>3. Выбрать из с приготовленны<br>4. Нажать кноп<br>5. Нажать кноп в гал ерею» | 1. Появляется окно загрузки картинки<br>2. Появляется диалоговое окно брау ера выбора файла дл я загрузки<br>3. Им я выбранного файла появляется в поле «Файл»<br>4. Д алоговое окно файла закр ывается, в поле «Файл» появляется полное имя файла<br>5. Выбранный файл появляется в списке файлов галереи |

STORMNET

# Рекомендации по написанию тест-кейсов

- Начинайте с простых очевидных тестов.
- Затем переходите к более сложным тестам.
- Помните о граничных условиях.
- Если остаётся время, занимайтесь исследовательским тестированием.

# Язык написания тест-кейсов

- Используйте активный залог: («open», «paste», «click»). В русском языке используйте безличную форму: «открыть» (вместо «откройте»)
- Описывайте поведение системы: «появляется окно...», «приложение закрывается»
- Используйте простой технический стиль
- ОБЯЗАТЕЛЬНО указывайте ТОЧНЫЕ названия всех элементов приложения
- Не объясняйте базовые понятия работы с ОС

# Критерии хорошего тест-кейса

- Обладает высокой вероятностью обнаружения ошибки.
- Не выполняет ненужных действий.
- Не является избыточным по отношению к другим тестам.
- Исследует соответствующую («ту, которую надо») область приложения.
- Позволяет легко диагностировать ошибку.
- Делает обнаруженную ошибку очевидной.
- Независим (каждый тест-кейс – это индивидуальный сценарий с точкой входа и точкой выхода).

# Тестовые набор (Test Suite)

- **Тестовый набор** – набор тестов (тест-кейсов), собранных в последовательность для достижения некоторой цели.
  Хороший тестовый сценарий всегда следует некоторой логике, например: типичному использованию приложения, удобству тестирования, распределению функций по модулям и т.д.

# Рекомендации по написанию тестовых наборов

- Пишите набор для отдельной части приложения.
- Пишите отдельно набор для Smoke и Critical Path тестов.
- Постепенно повышайте сложность тестов.
- Организуйте сценарий логично.
- Используйте один тест для ОДНОЙ проверки.
- Помните, что заголовки тестов отражают их суть. Правильно формулируйте и оформляйте заголовки.
- Помните о необходимых приготовлениях к тесту. Описывайте их.
- Не повторяйте в нескольких тестах одни и те же шаги.
- Старайтесь избегать похожих тестов (таких, в которых набор шагов и ожидаемых результатов визуально кажется одинаковым).

# Техники ускорения написания тестов

- Copy-paste.
- Если по ходу разработки тестов возникают вопросы, пишите их прямо в документ с тестами, помечая красным цветом.
- Используйте т.н. «косметику» (жирный, подчёркнутый, наклонный шрифт, разные цвета т.д.) Это значительно повышает читаемость документа.
- По-максимуму используйте возможности ПО, в котором вы разрабатываете тесты (группировки, фильтры, ссылки и т.д.)
- Если вы пишете тесты в файле, обязательно прописывайте в самом файле историю его изменения.

# Шаги разработки тестовых сценариев

- Сбор информации (требования, мок-апы и т. д.)
- Разделение приложения на модули
- Написание чек-листов
- Написание тест-кейсов

# Шаги разработки тестовых сценариев

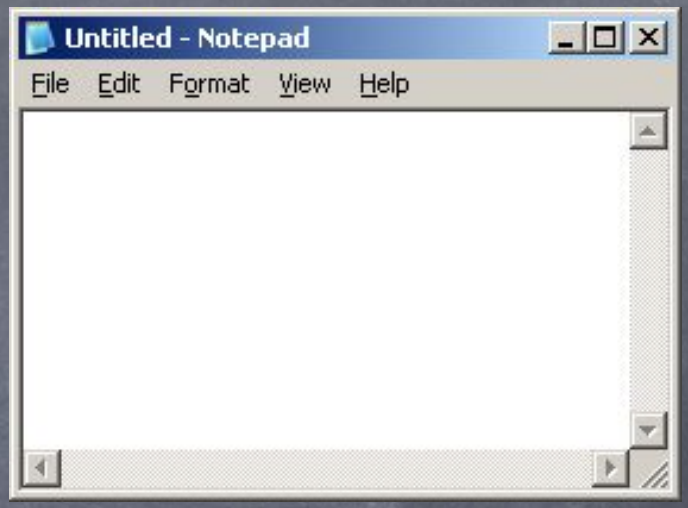
- Начинайте как можно раньше, ещё до выхода первого билда.
- Разбивайте приложение на отдельные части/модули.
- Для каждой области/модуля пишите чек-лист.
- Пишите вопросы, уточняйте детали, добавляйте «косметику», используйте copy-paste.
- Получите рецензию коллег-тестировщиков, разработчиков, заказчиков.
- Обновляйте тесты, как только обнаружили ошибку или изменилась функциональность.

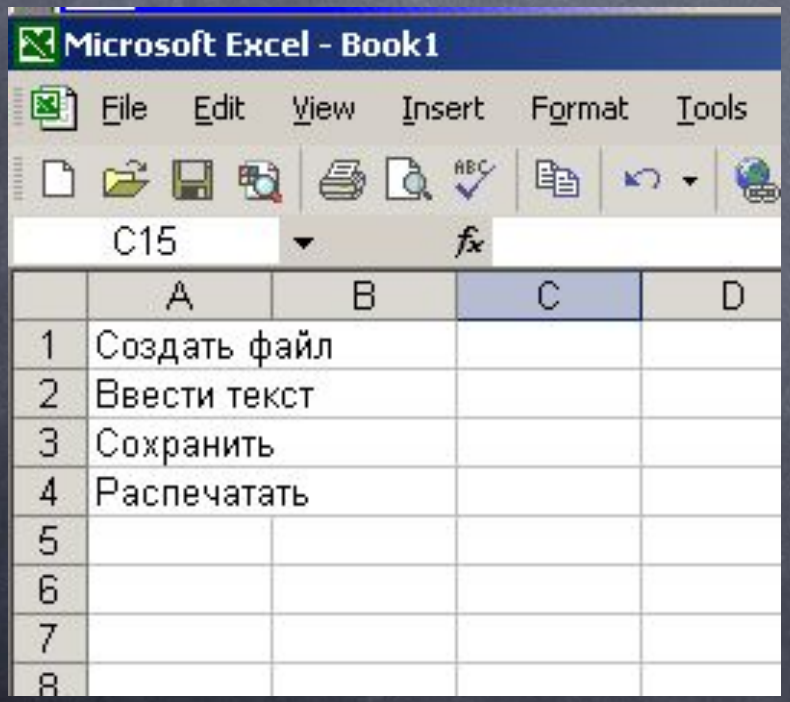
# Последовательность разработки и выполнения тестов

- Простые позитивные.
- Простые негативные.
- Сложные позитивные.
- Сложные негативные.

# Учимся составлять первый тест-кейс

- Что такое notepad?
- Какие функции для него важны?

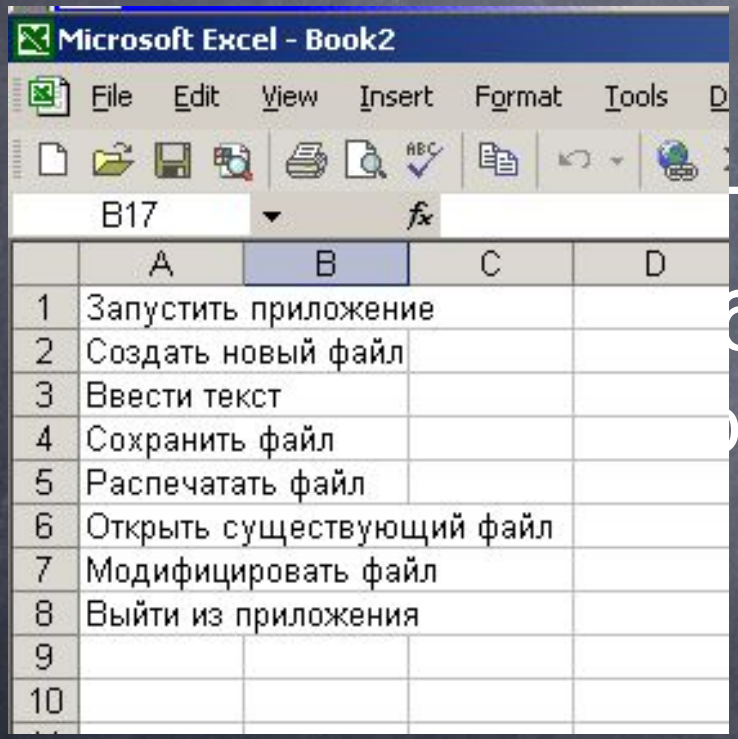



STORMNET

# Учимся составлять первый тест-кейс

- Итак, вот наш Smoke test




Перенесём его в блон для аботки тестов.

# Учимся составлять первый тест-кейс

- Фактически, это – чек-лист. И сами пункты грамотно сформированного чек-листа – готовые заголовки тест-кейсов.

Microsoft Excel - Шаблон для разработки тестов v2

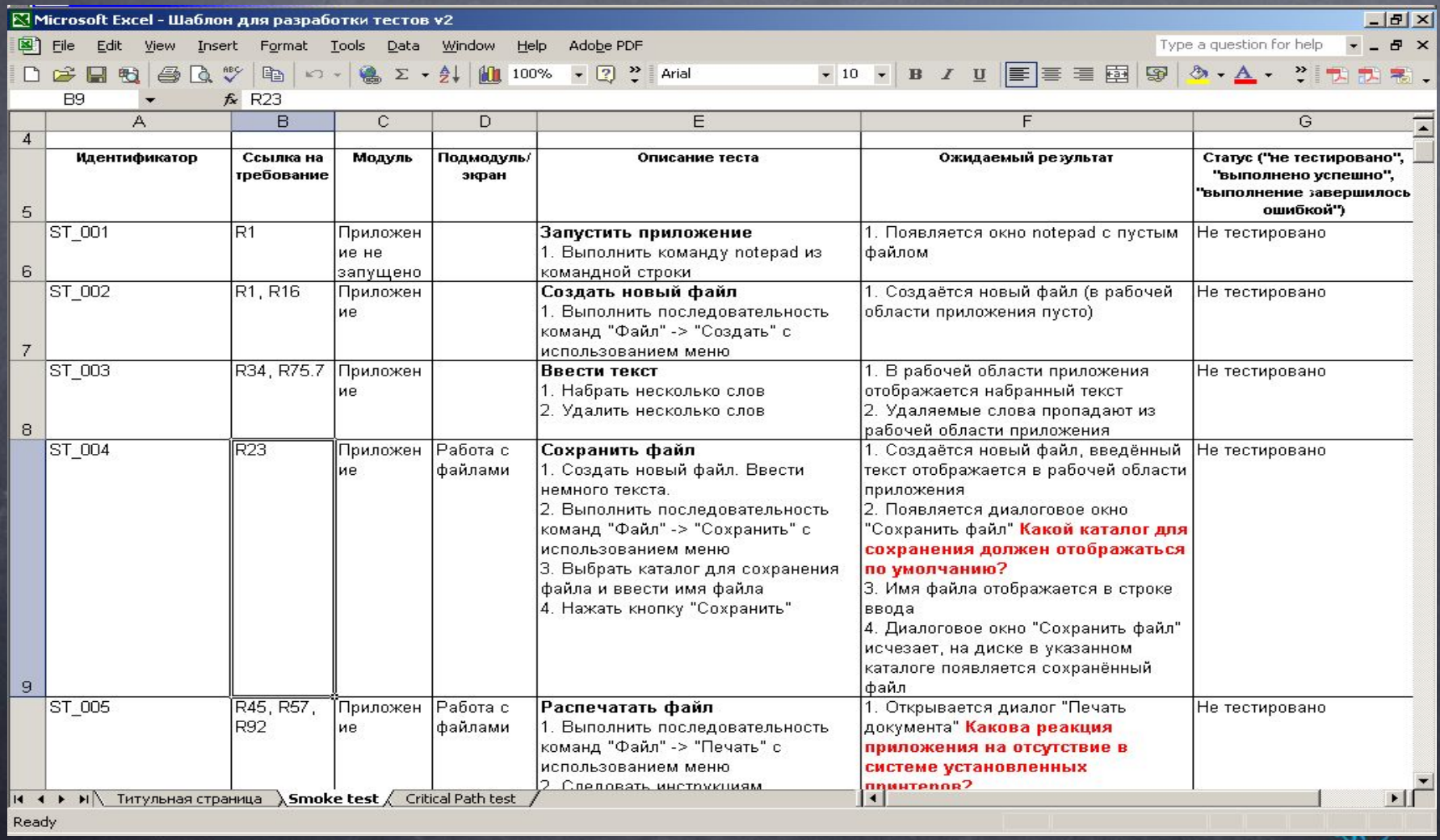
File Edit View Insert Format Tools Data Window Help Adobe PDF

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | | | **"Notepad"**<br>**Функциональные тест-кейсы** | |
| 2 | **Тестировал(и):** | | | | | |
| 3 | **Дата(даты) тестирования:** | | | **ОС:** | | |
| 4 | | | | | | |
| 5 | **Идентификатор** | **Ссылка на требование** | **Модуль** | **Подмодуль/ экран** | **Описание теста** | **Ожидаемый результат** |
| 6 | | | | | Запустить приложение | |
| 7 | | | | | Создать новый файл | |
| 8 | | | | | Ввести текст | |
| 9 | | | | | Сохранить файл | |
| 10 | | | | | Распечатать файл | |
| 11 | | | | | Открыть существующий файл | |
| 12 | | | | | Модифицировать файл | |
| 13 | | | | | Выйти из приложения | |
| 14 | | | | | | |
| 15 | | | | | | |

STORMNET

# Учимся составлять первый тест-кейс

- Когда мы распишем наши тесты по правилам, Smoke Test примет следующий вид:



Microsoft Excel - Шаблон для разработки тестов v2

File Edit View Insert Format Tools Data Window Help Adobe PDF

B9 R23

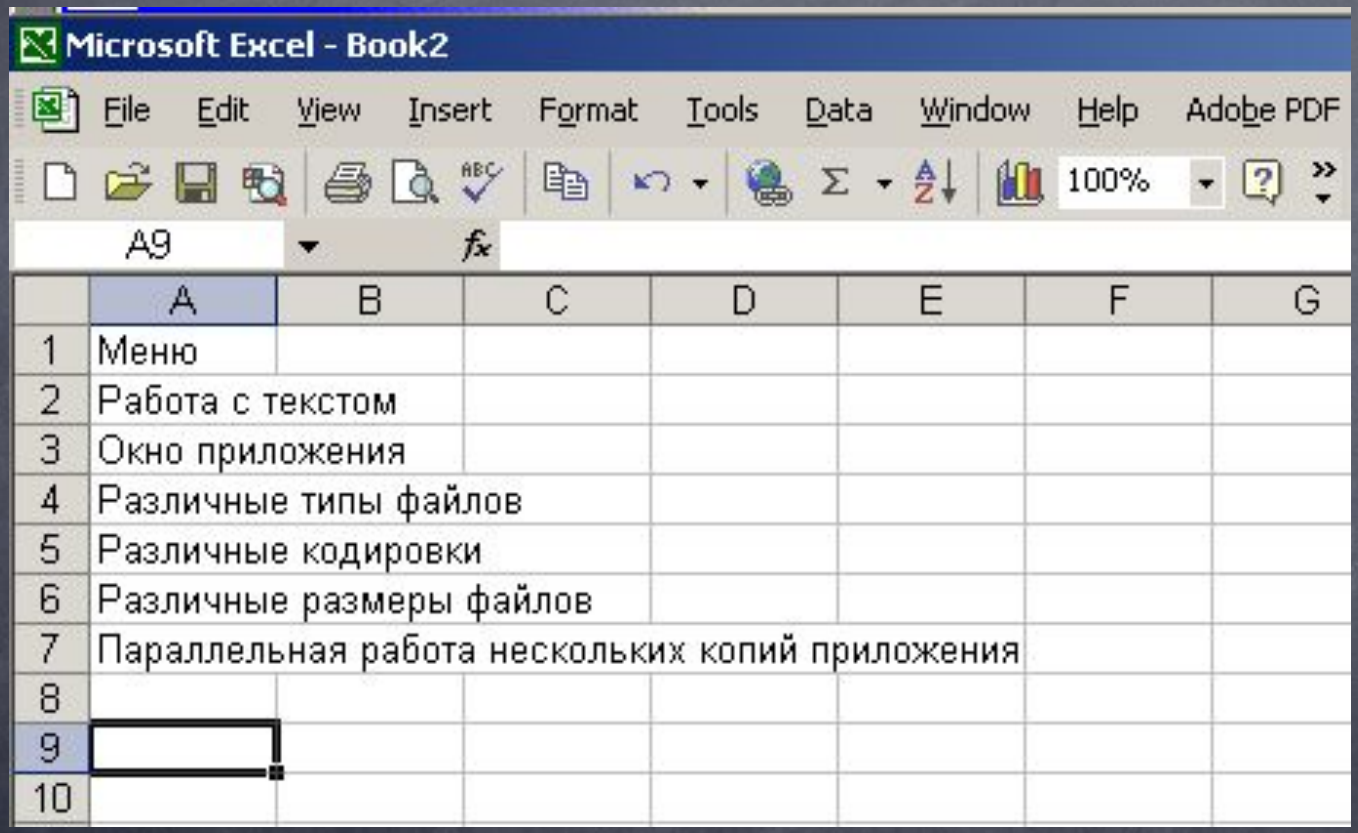
| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 4 | | | | | | | |
| 5 | **Идентификатор** | **Ссылка на требование** | **Модуль** | **Подмодуль/ экран** | **Описание теста** | **Ожидаемый результат** | **Статус ("не тестировано", "выполнено успешно", "выполнение завершилось ошибкой")** |
| 6 | ST_001 | R1 | Приложение не запущено | | **Запустить приложение**<br>1. Выполнить команду notepad из командной строки | 1. Появляется окно notepad с пустым файлом | Не тестировано |
| 7 | ST_002 | R1, R16 | Приложение | | **Создать новый файл**<br>1. Выполнить последовательность команд "Файл" -> "Создать" с использованием меню | 1. Создаётся новый файл (в рабочей области приложения пусто) | Не тестировано |
| 8 | ST_003 | R34, R75.7 | Приложение | | **Ввести текст**<br>1. Набрать несколько слов<br>2. Удалить несколько слов | 1. В рабочей области приложения отображается набранный текст<br>2. Удаляемые слова пропадают из рабочей области приложения | Не тестировано |
| 9 | ST_004 | R23 | Приложение | Работа с файлами | **Сохранить файл**<br>1. Создать новый файл. Ввести немного текста.<br>2. Выполнить последовательность команд "Файл" -> "Сохранить" с использованием меню<br>3. Выбрать каталог для сохранения файла и ввести имя файла<br>4. Нажать кнопку "Сохранить" | 1. Создаётся новый файл, введённый текст отображается в рабочей области приложения<br>2. Появляется диалоговое окно "Сохранить файл" **Какой каталог для сохранения должен отображаться по умолчанию?**<br>3. Имя файла отображается в строке ввода<br>4. Диалоговое окно "Сохранить файл" исчезает, на диске в указанном каталоге появляется сохранённый файл | Не тестировано |
| | ST_005 | R45, R57, R92 | Приложение | Работа с файлами | **Распечатать файл**<br>1. Выполнить последовательность команд "Файл" -> "Печать" с использованием меню<br>2. Следовать инструкциям | 1. Открывается диалог "Печать документа" **Какова реакция приложения на отсутствие в системе установленных принтеров?** | Не тестировано |

Титульная страница | Smoke test | Critical Path test

Ready

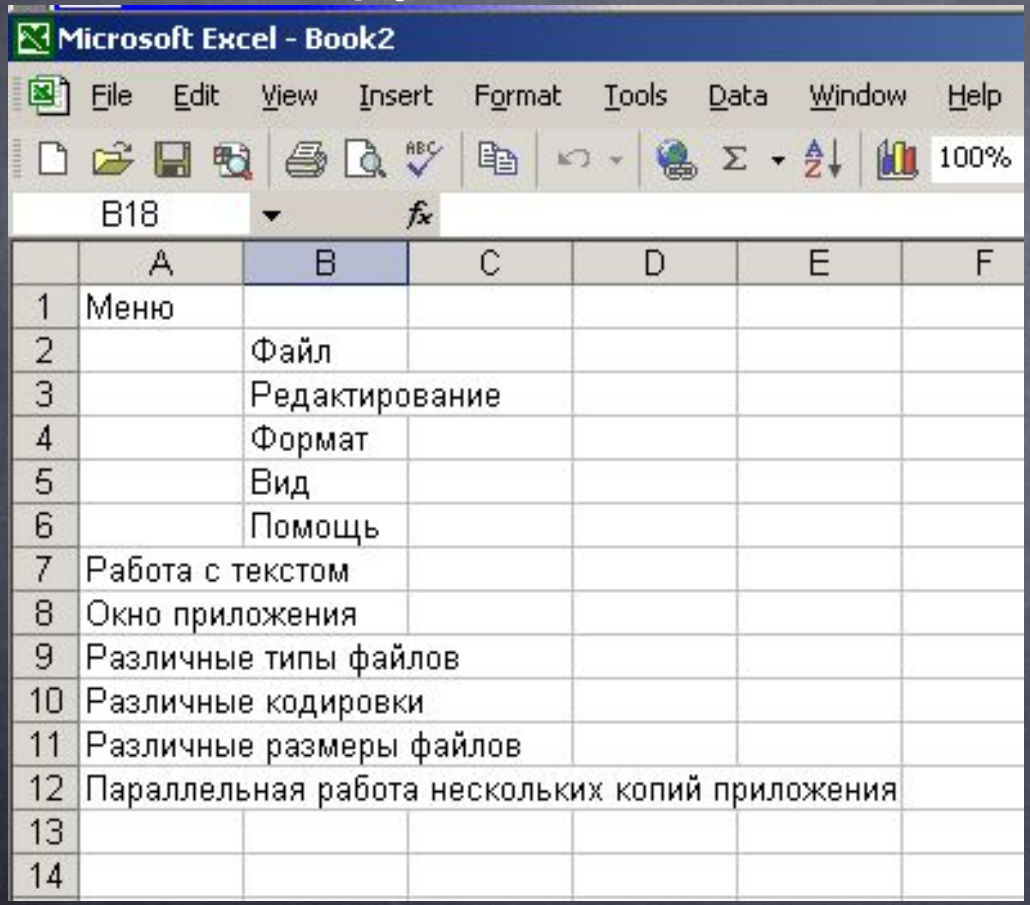
# Учимся составлять первый тест-кейс

- Аналогичным образом начинаем и продолжаем работать с тестом критического пути:



# Учимся составлять первый тест-кейс

- Детализируем чек-лист:

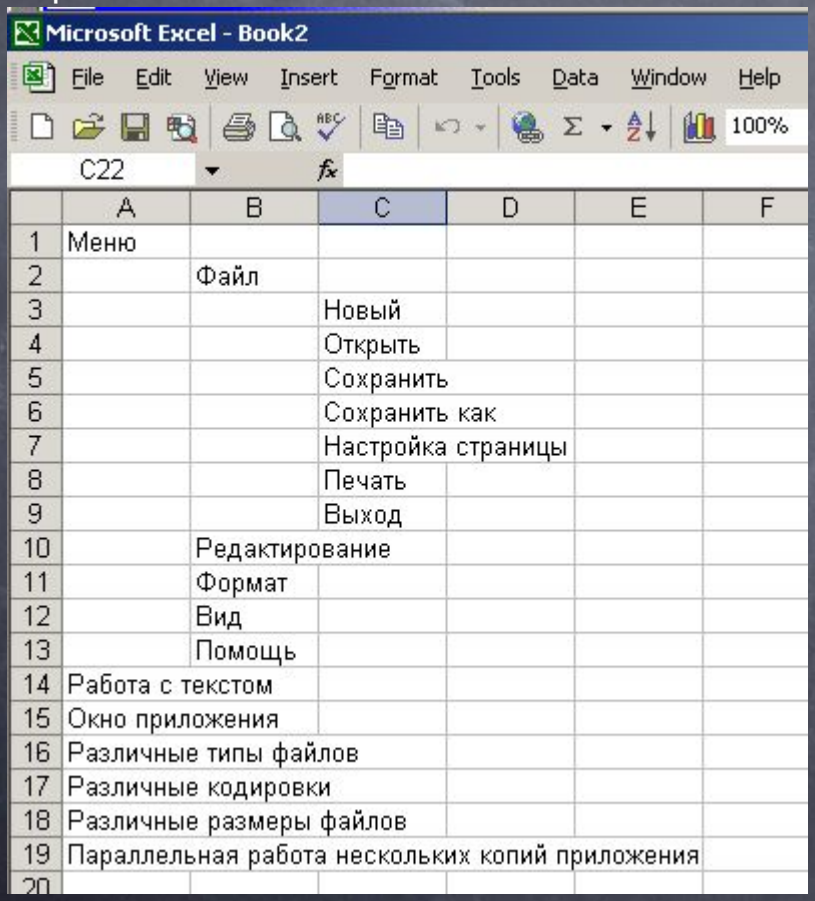



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Меню | | | | | |
| 2 | | Файл | | | | |
| 3 | | Редактирование | | | | |
| 4 | | Формат | | | | |
| 5 | | Вид | | | | |
| 6 | | Помощь | | | | |
| 7 | Работа с текстом | | | | | |
| 8 | Окно приложения | | | | | |
| 9 | Различные типы файлов | | | | | |
| 10 | Различные кодировки | | | | | |
| 11 | Различные размеры файлов | | | | | |
| 12 | Параллельная работа нескольких копий приложения | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |

STORMNET

# Учимся составлять первый тест-кейс

- Продолжаем детализацию до тех пор, пока не получим логичный и достаточный набор тестов. После этого переносим его в шаблон и работаем аналогично тому, как мы делали это при разработке Smoke Test.




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Меню | | | | | |
| 2 | | Файл | | | | |
| 3 | | | Новый | | | |
| 4 | | | Открыть | | | |
| 5 | | | Сохранить | | | |
| 6 | | | Сохранить как | | | |
| 7 | | | Настройка страницы | | | |
| 8 | | | Печать | | | |
| 9 | | | Выход | | | |
| 10 | | Редактирование | | | | |
| 11 | | Формат | | | | |
| 12 | | Вид | | | | |
| 13 | | Помощь | | | | |
| 14 | Работа с текстом | | | | | |
| 15 | Окно приложения | | | | | |
| 16 | Различные типы файлов | | | | | |
| 17 | Различные кодировки | | | | | |
| 18 | Различные размеры файлов | | | | | |
| 19 | Параллельная работа нескольких копий приложения | | | | | |
| 20 | | | | | | |

STORMNET

# Задача о треугольнике

- Эту задачу предложил в 1979 году Гленфорд Майерс в своей книге «Искусство тестирования программ» («The Art Of Software Testing»). С тех пор она известна как «задача о треугольнике» и является своего рода классическим вопросом на множестве собеседований на должность тестировщика

- Программа производит чтение с перфокарты трёх целых чисел, которые интерпретируются как длины сторон треугольника. Далее программа печатает сообщение о том, является ли треугольник неравносторонним, равнобедренным или равносторонним