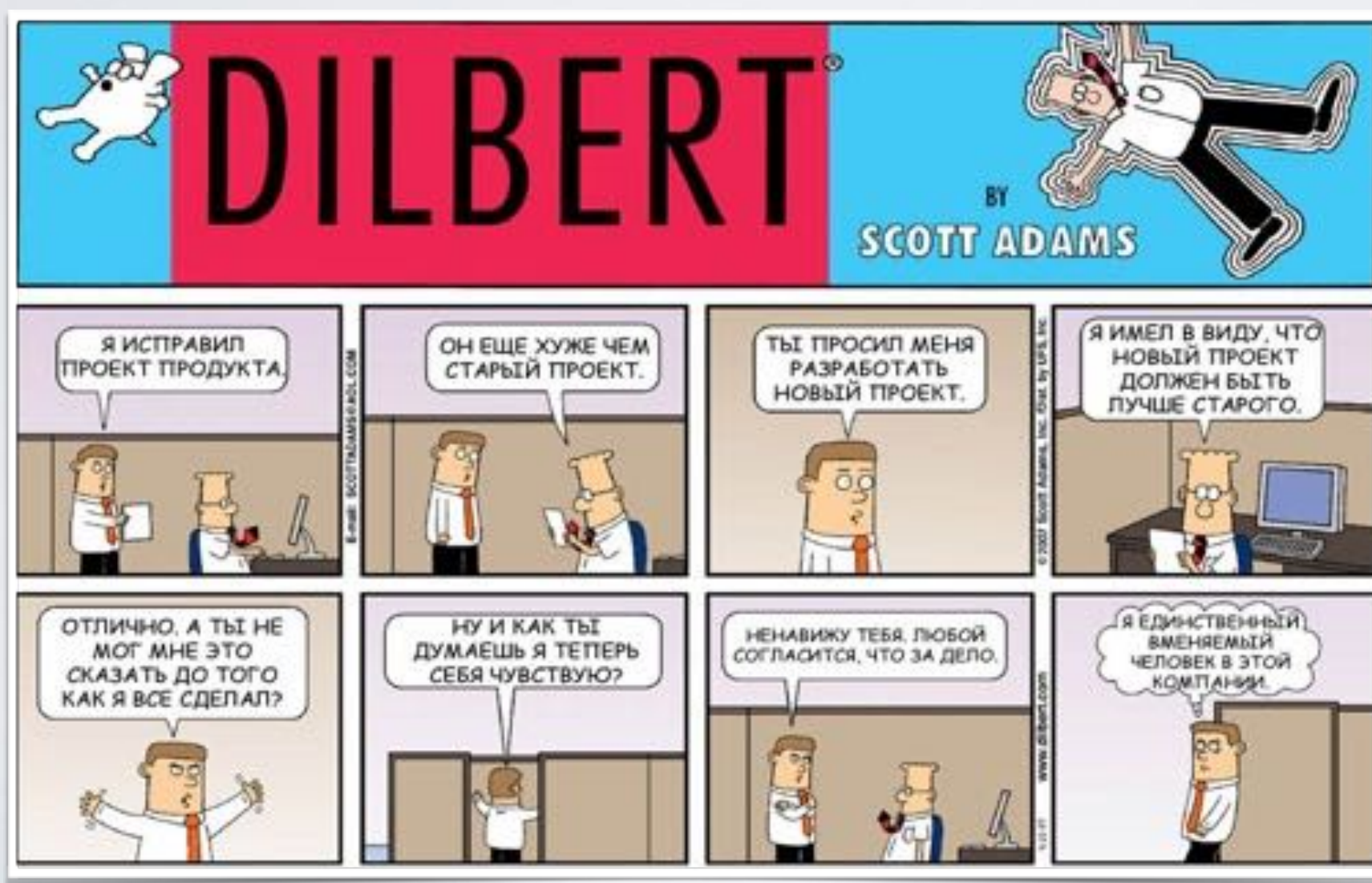


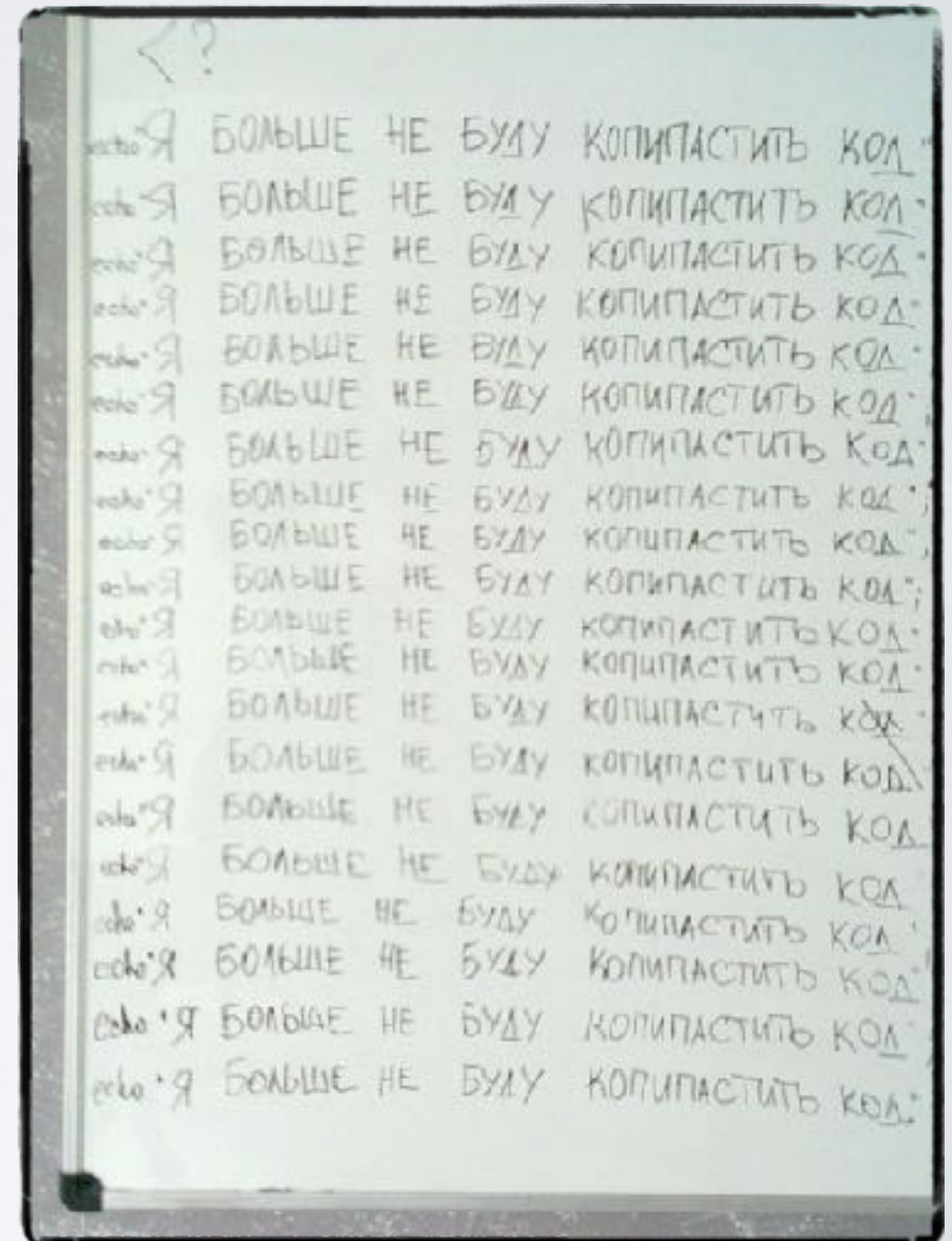
ОСНОВЫ ПРОГРАММНОГО КОНСТРУИРОВАНИЯ

Лекция № 15
11 декабря 2017 г.



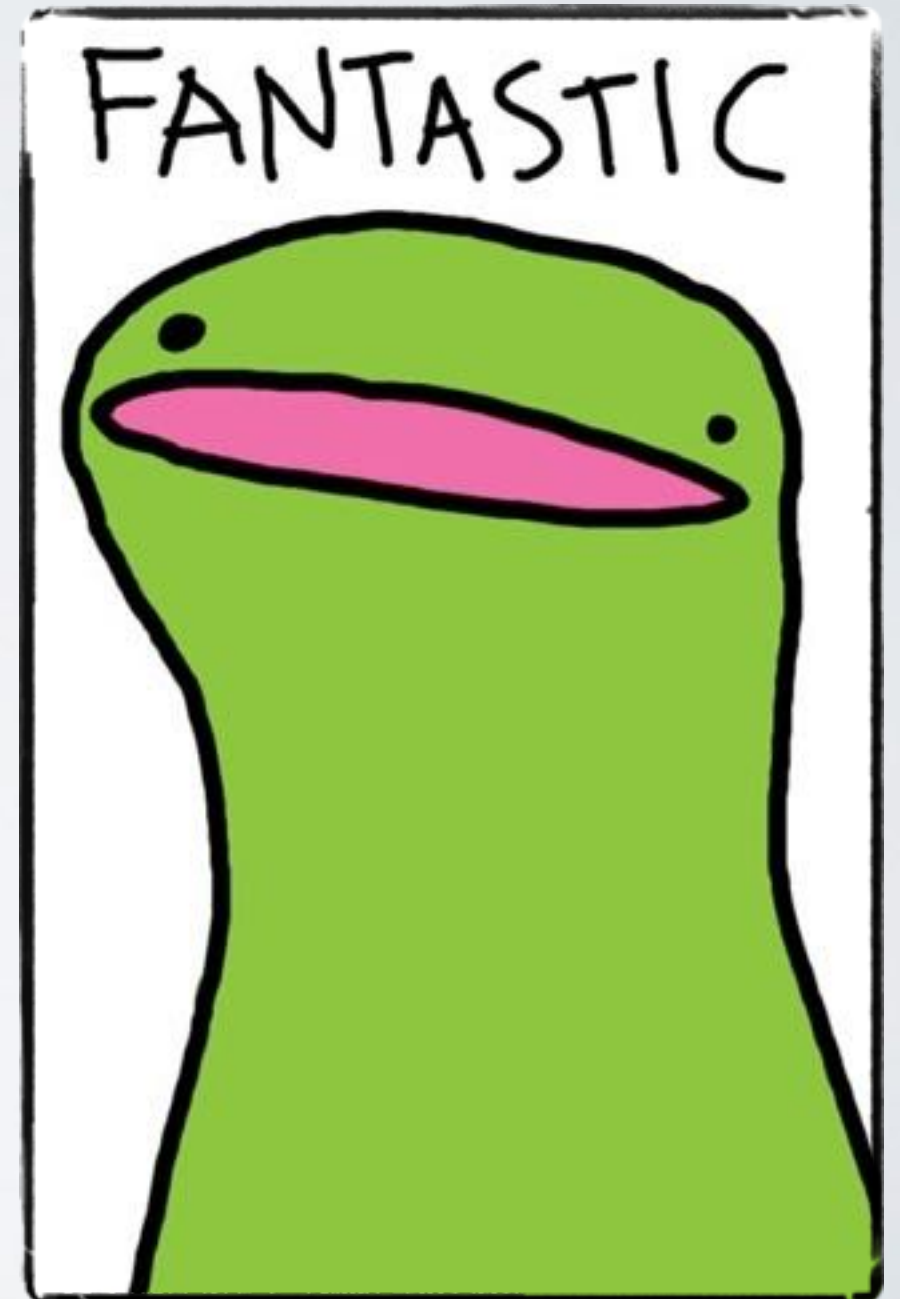
ПЛАН НА СЕГОДНЯ

- Что такое хороший проект?
- Отладка.
- Интерфейс пользователя.
- Документирование.
- Тестирование.
- Deja Vu.
- Формальности.
- Советы, хорошие и не очень.



ХОРОШИЙ ПРОЕКТ

- Функциональность (решение заявленной задачи).
- Отсутствие ошибок, устойчивость.
- Понятный интерфейс, наличие документации.
- Качество кода:
 - Функции и модули.
 - Отсутствие copy-paste.
 - Отступы и т. д.
- Наличие тестов.



ОТЛАДКА

Эксперимент. Профессиональных программистов попросили отладить программу с 12 дефектами.

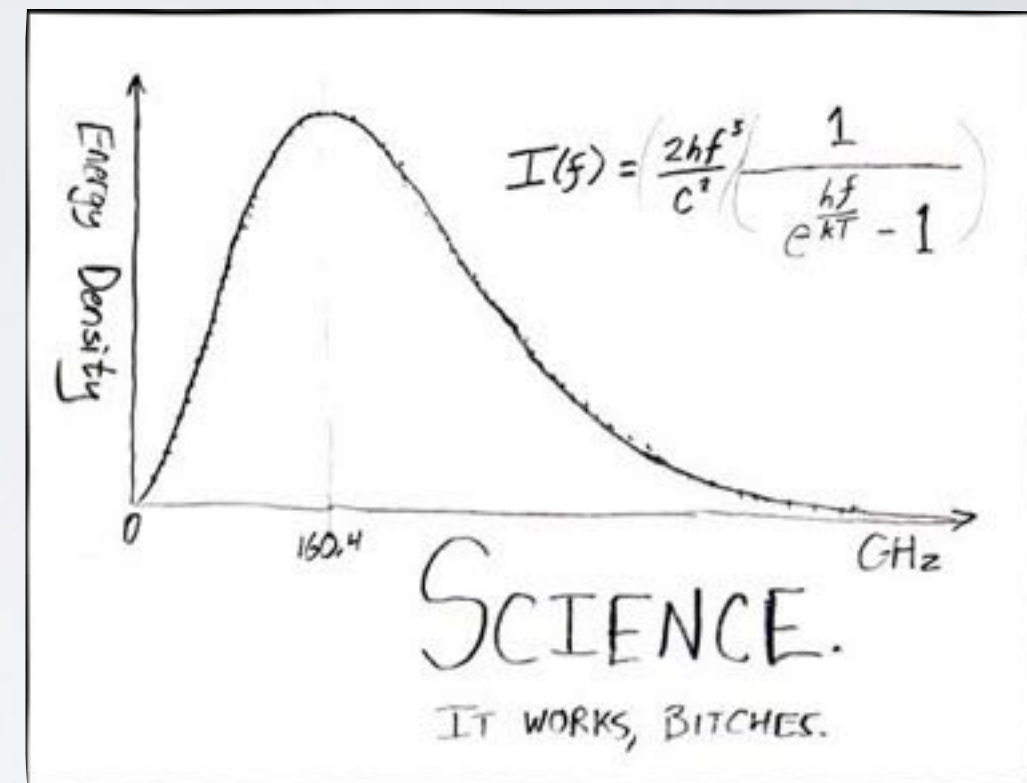
	Трое лучших	Трое худших
Среднее время отладки (мин)	5	14,1
Среднее число не обнаруж. дефектов	0,7	1,7
Среднее число новых дефектов	3	7,7

Самый лучший: нашел всё, ничего не поломал.

Самый худший: не нашел 4, внес 11 новых дефектов.

НАУЧНЫЙ МЕТОД ОТЛАДКИ (!)

- Стабилизация ошибки.
- Определение источника ошибки.
 - Сбор данных, приводящих к дефекту.
 - Анализ собранных данных, формулирование гипотезы.
 - Как можно подтвердить или опровергнуть гипотезу?
 - Подтверждение или опровержение.
- Исправление дефекта.
- Тестирование исправления.
- Поиск и устранение похожих ошибок.



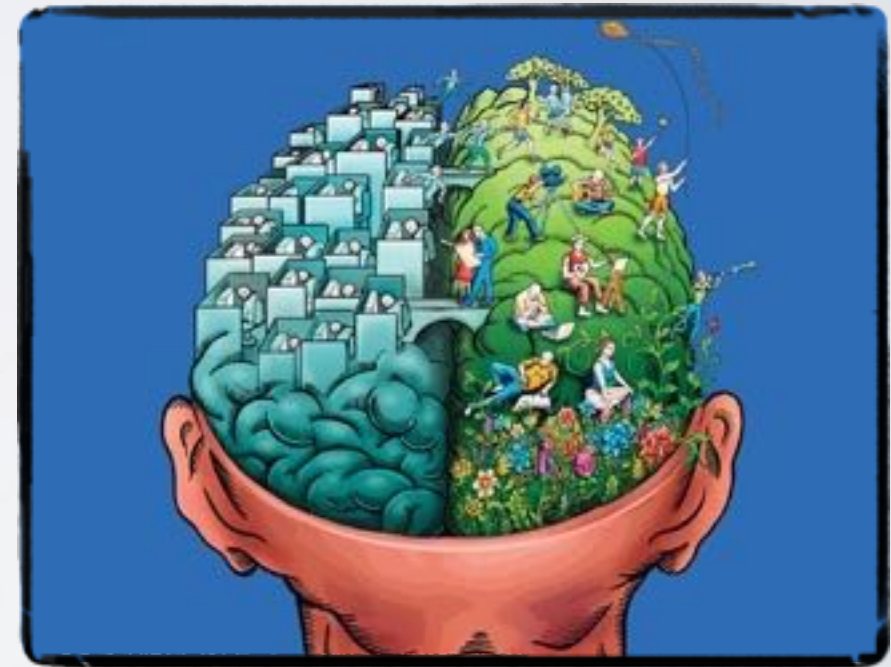
СТАБИЛИЗАЦИЯ ОШИБКИ

- **Задача:** построить самый простой тест, который будет проявлять ошибки в 100% случаев.
- Минимальный размер структуры данных.
- Желательно отказаться от ввода данных пользователем.



ГИПОТЕЗЫ

- Сбор данных и формулирование гипотезы:
 - Отладочная печать.
 - Участки Кода Которые Менялись Совсем Недавно!
- Частые проблемы в С:
 - «Плюс один» и «минус один» (индексы и размеры массивов).
 - Ошибки выделения и освобождения памяти.
- Проверка гипотез.
 - Найти другой тест, который не должен падать.



УСТОЙЧИВОСТЬ

- **Аксиома.** Входным данным нельзя доверять.
- **Следствие.** Данные нужно проверять и либо заменять некорректные значения корректными, либо сигнализировать об ошибке.
- Чем проще программа, тем она, как правило, устойчивее.
- Выбор правильного типа данных: **unsigned** для неотрицательных чисел.

ИНТЕРФЕЙС МОДУЛЯ (БИБЛИОТЕКИ)

- Вопросы, на которые ответ должен быть ясен:
 - Кто создает и освобождает структуру данных?
 - Как возвращаются ошибки?
- Самый простой способ ответить на вопросы: поместить в документацию небольшой пример.

ИНТЕРФЕЙС КОМАНДНОЙ СТРОКИ

- Программа, запущенная без аргументов (или с ключами `-h`, `--help`, `/?`), выдает справку: «я такая-то программа, меня запускать так-то».
- Использование кода возврата из `main()` для сигнализации об ошибках и результатах.
- Текстовые сообщения об ошибках на разные случаи.

ПРИМЕР СПРАВКИ (ZIP)

\$ zip

Copyright (c) 1990-2008 Info-ZIP - Type 'zip "-L"' for software license.

Zip 3.0 (July 5th 2008). Usage:

zip [-options] [-b path] [-t mmddyyyy] [-n suffixes] [zipfile list] [-xi list]

The default action is to add or replace zipfile entries from list, which can include the special name - to compress standard input.

If zipfile and list are omitted, zip compresses stdin to stdout.

-f	freshen: only changed files	-u	update: only changed or new files
-d	delete entries in zipfile	-m	move into zipfile (delete OS files)
-r	recurse into directories	-j	junk (don't record) directory names
-0	store only	-l	convert LF to CR LF (-ll CR LF to LF)
-1	compress faster	-9	compress better
-q	quiet operation	-v	verbose operation/print version info
-c	add one-line comments	-z	add zipfile comment
-@	read names from stdin	-o	make zipfile as old as latest entry
-x	exclude the following names	-i	include only the following names
-F	fix zipfile (-FF try harder)	-D	do not add directory entries
-A	adjust self-extracting exe	-J	junk zipfile prefix (unzipsfx)
-T	test zipfile integrity	-X	exclude extra file attributes
-y	store symbolic links as the link		instead of the referenced file
-e	encrypt	-n	don't compress these suffixes
-h2	show more help		

ПРИМЕР ВЫВОДА ZIP И UNZIP

```
$ zip src *.c *.h
```

```
adding: assoc_list.c (deflated 68%)  
adding: phonebook.c (deflated 62%)  
adding: reader.c (deflated 67%)  
adding: assoc_list.h (deflated 57%)  
adding: reader.h (deflated 39%)
```

```
$ unzip -t src.zip
```

```
Archive:  src.zip  
  testing: assoc_list.c          OK  
  testing: phonebook.c          OK  
  testing: reader.c             OK  
  testing: assoc_list.h         OK  
  testing: reader.h             OK
```

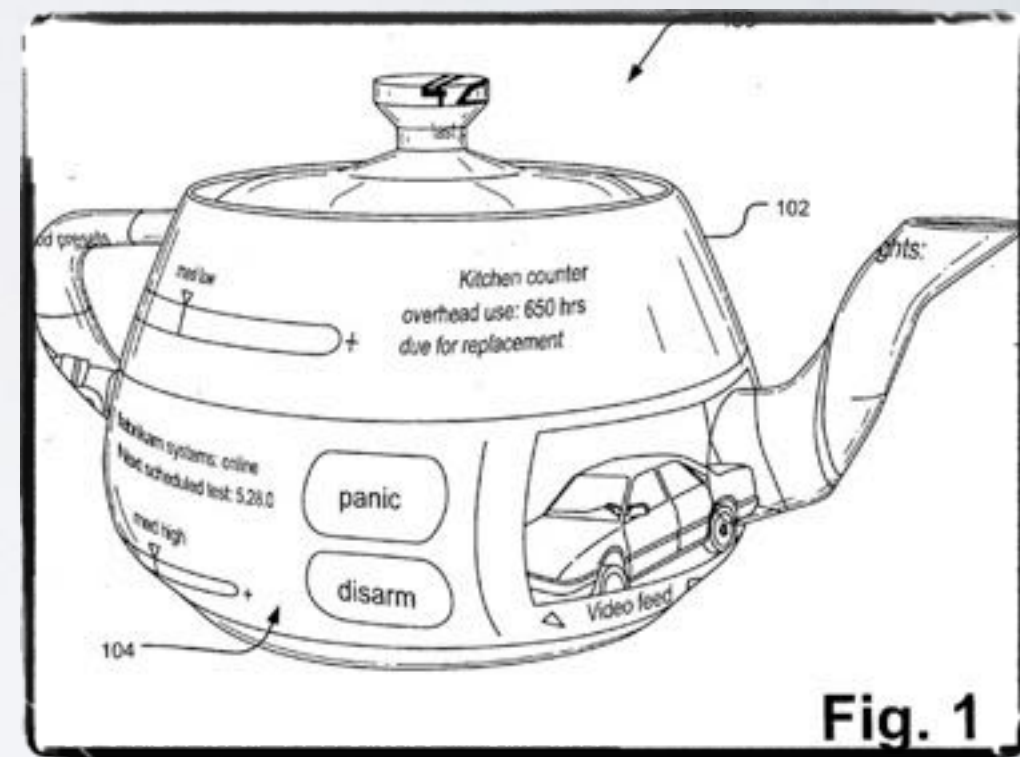
```
No errors detected in compressed data of src.zip.
```

```
$ unzip -t notfound.zip
```

```
unzip:  cannot find or open notfound.zip, notfound.zip.zip or  
notfound.zip.ZIP.
```


ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

- Должно быть понятно, как выйти из программы!
- Подтверждение деструктивных операций.
- **Диалоговый режим:** приглашения ко вводу.
- **Консольный интерфейс:** меню, справка по горячим клавишам.
- Самостоятельная документация.



ДОКУМЕНТИРОВАНИЕ

- **Аксиома.** Ваша программа – на Земле не первая.
- **Следствие.** Пользователь, скорее всего, уже имеет опыт взаимодействия с другими программами.
- **Следствие.** Если ваша программа ведет себя так же, как уже известная, пользователь сразу сможет с ней работать, ничего не читая.
- Главные вопросы, на которые отвечает документация:
 - «Что это? На что это похоже из того, что я уже знаю?»
 - «Какие здесь особенности?»

ОСНОВНЫЕ ВИДЫ ДОКУМЕНТАЦИИ

- Файл README (что это, зачем нужно и на что похоже).
- Инструкция для пользователя (детальное описание функций и режимов работы).
- Инструкция для разработчиков (внутреннее устройство, модули, соглашения, используемые алгоритмы и т.п.)

АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ

- **Аксиома.** Все не протестируешь.
- **Следствие.** Тестировать нужно там, где появление ошибок наиболее вероятно и где ошибки будут наиболее критичны.
- «Бизнес-логика» vs. «Интерфейс».



СОДЕРЖАНИЕ КУРСА

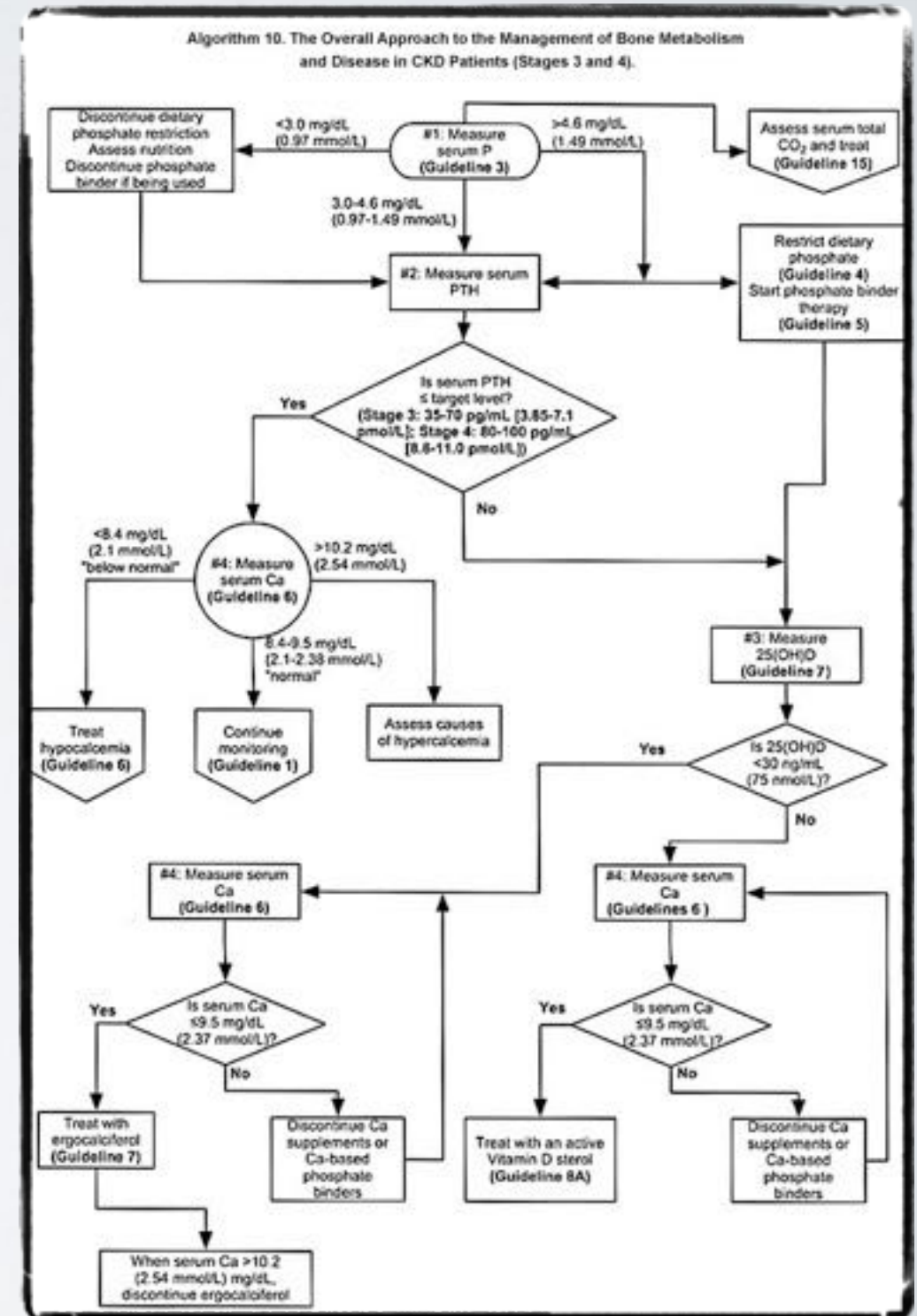
ДААННЫЕ

- Машинное представление.
- Прimitives, составные и сложные типы данных.
- Абстрактные типы данных (стеки, очереди, списки, деревья, хеш-таблицы).



АЛГОРИТМЫ И ЗАДАЧИ

- Принцип «разделяй и властвуй».
- Алгоритмы сортировки.
- Алгоритмы поиска (в массиве, в строке).
- Алгоритмы архивации.
- Задача трансляции.



ИНСТРУМЕНТЫ

- Языки и парадигмы программирования.
- Язык C.
- Модули.
- Способы отладки и тестирования программ.



ДИФ. ЗАЧЕТ

- 28 декабря, четверг, 10:00.
- Теоретическая часть: два вопроса по билетам.
 - Методичка: bit.ly/1jNb0ac.
- Практическая часть: ваш проект.

Экзамен для меня
всегда праздник,
профессор!



НЕСКОЛЬКО ПЛОХИХ СОВЕТОВ

ВЫКИНУЛ ASSERT'Ы



**Программа
заработала!**

DIYLOL.COM

**Решай 100000
задач по 2
балла**



Получай пятерку!

Решил не париться

**Начал проект
в ночь перед
зачетом**

DIYLOL.COM

Дерево поиска?

**Господи,
как скучно.
Уж лучше массив
пузырьком**

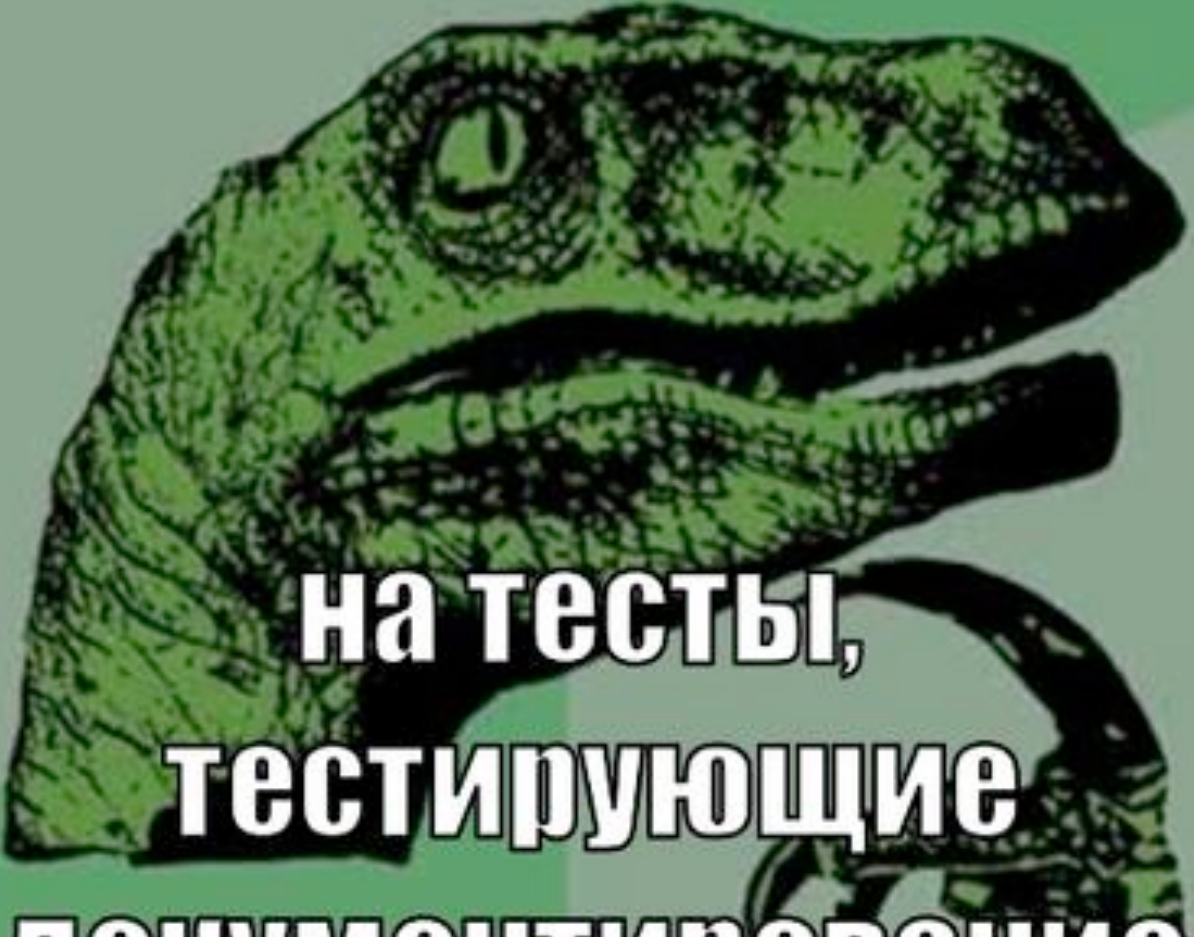
**Оформлять
код?**

**Оставим это
для перфокарт**

**Не надо
документации**

**Лучше лежать
и смеяться**

**Как писать
тесты**



**на тесты,
тестирующие
документирование
тестов?**



КОНЕЦ ПОСЛЕДНЕЙ ЛЕКЦИИ