CODE REUSE
&
TESTING / DEBUGGING

Marcos del Pozo Banos

CONTENT

Where to start?

Designing a program

Module programming

Functions

Fixing your program

Debugging

OBJECTIVES

Learn to think like a programmer

Learn to use flow charts

Learn to use modular programming

Learn to debug your code

Understand the jokes in these slides

What is the meaning of your ~~life~~ ? program

Always think before you program

Always think while you are programming

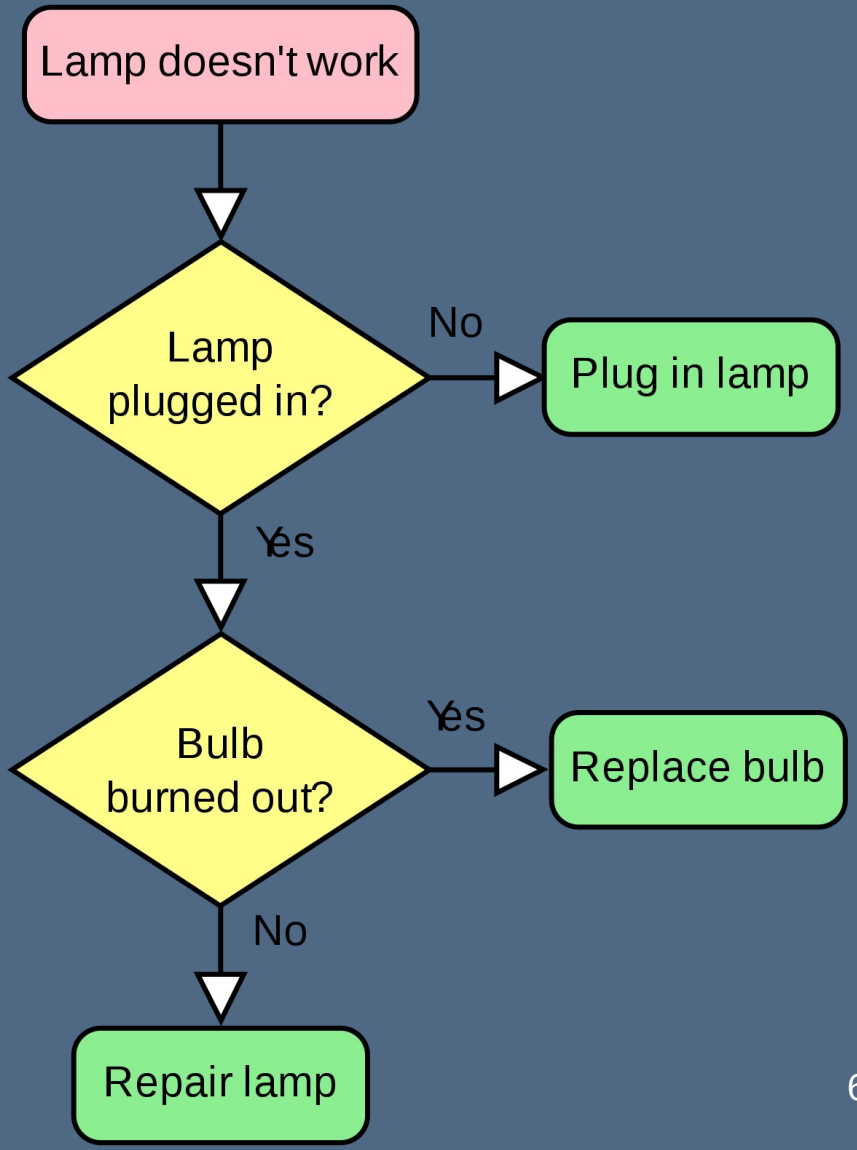
Always think after you have programmed

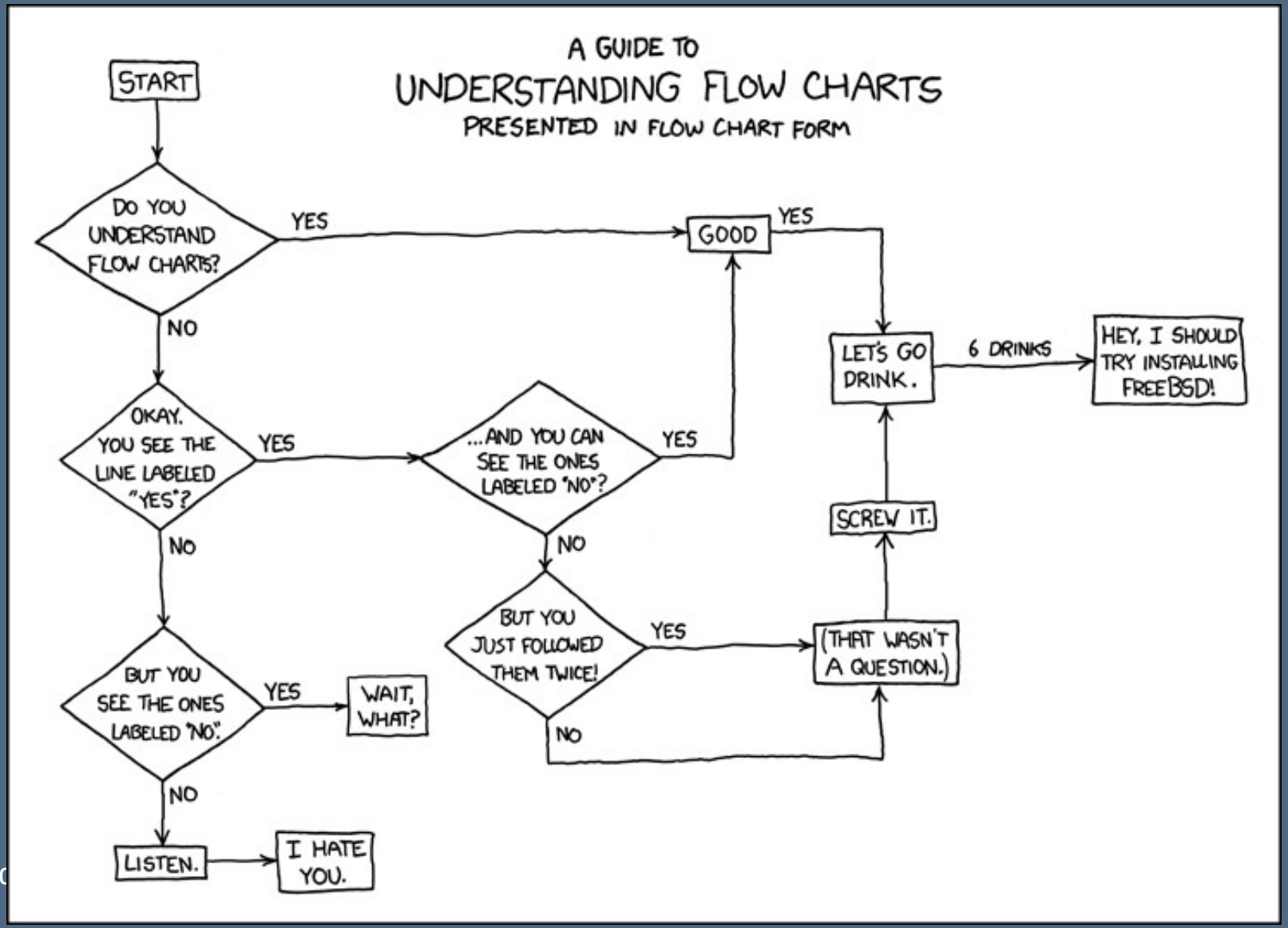
Marcos del Pozo Banos

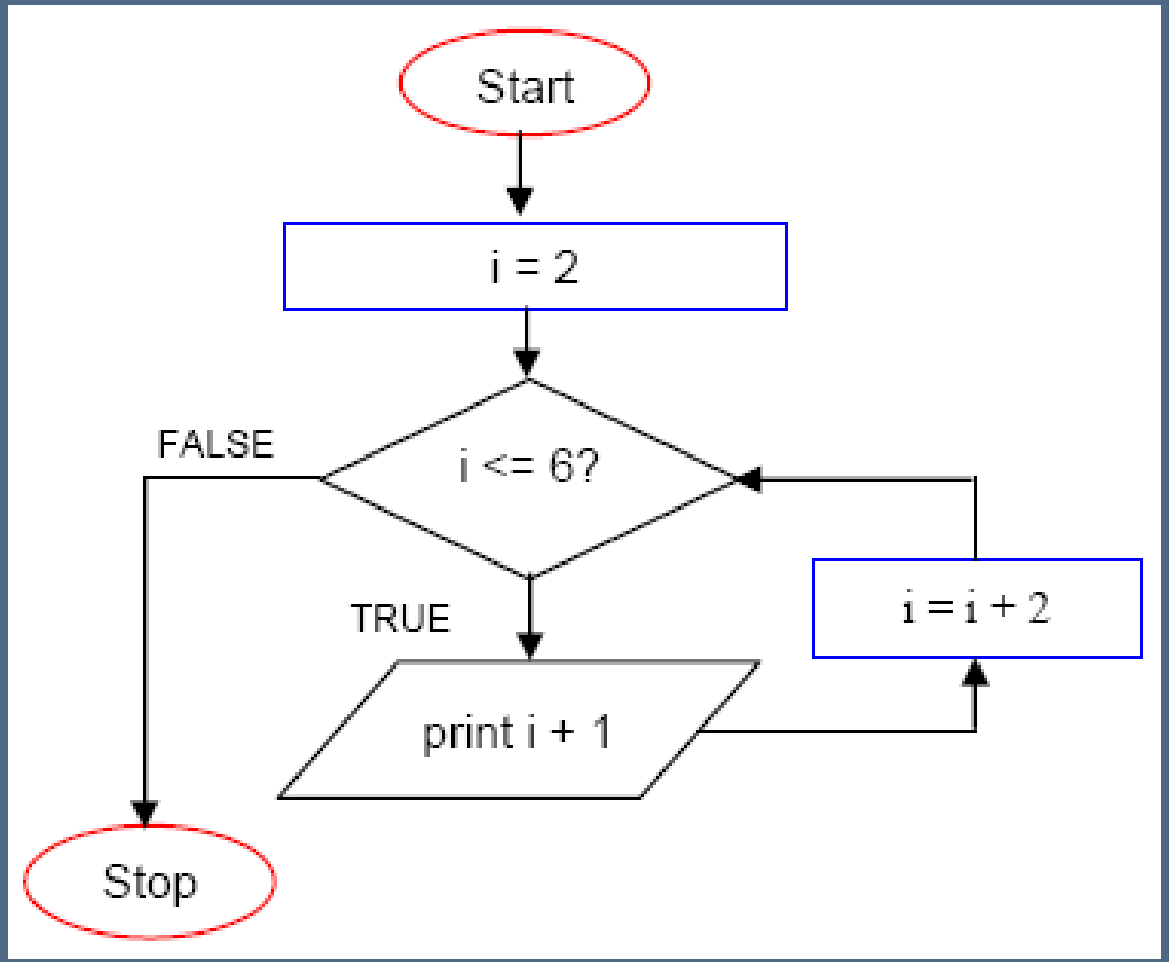
COMPUTERS OFF

Pen, paper

and FLOW CHARTS




A GUIDE TO
UNDERSTANDING FLOW CHARTS
PRESENTED IN FLOW CHART FORM
START
DO YOU UNDERSTAND FLOW CHARTS?
YES
GOOD
YES
NO
LET'S GO DRINK.
6 DRINKS
HEY, I SHOULD TRY INSTALLING FREEBSD!
OKAY. YOU SEE THE LINE LABELED "YES"?
YES
...AND YOU CAN SEE THE ONES LABELED "NO"?
YES
NO
SCREW IT.
NO
BUT YOU JUST FOLLOWED THEM TWICE!
YES
(THAT WASN'T A QUESTION.)
NO
BUT YOU SEE THE ONES LABELED "NO".
YES
WAIT, WHAT?
NO
LISTEN.
I HATE YOU.


Start
i = 2
i <= 6?
FALSE
TRUE
print i + 1
i = i + 2
Stop

Terminator
Indicates the beginning or end of a program flow in your diagram.

Process
Indicates any processing function.

Decision
Indicates a decision point between two or more paths in a flowchart.

Delay
Indicates a delay in the process.

Data
Can represents any type of data in a flowchart.

Document
Indicates data that can be read by people, such as printed output.

Multiple documents
Indicates multiple documents.

Subroutine
Indicates a predefined (named) process, such as a subroutine or a module.

Preparation
Indicates a modification to a process, such as setting a switch or initializing a routine.

Display
Indicates data that is displayed for people to read, such as data on a monitor or projector screen.

Manual input
Indicates any operation that is performed manually (by a person).

Manual loop
Indicates a sequence of commands that will continue to repeat until stopped manually.

Loop limit
Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.

Stored data
Indicates any type of stored data.

Connector
Indicates an inspection point.

Off-page connector
Use this shape to create a cross-reference and hyperlink from a process on one page to a process on another page.

Off-page connector

Off-page connector

Off-page connector

Or
Logical OR

Summing junction
Logical AND

Collate
Indicates a step that organizes data into a standard format.

Sort
Indicates a step that organizes items list sequentially.

Merge
Indicates a step that combines multiple sets into one.

Database
Indicates a list of information with a standard structure that allows for searching and sorting.

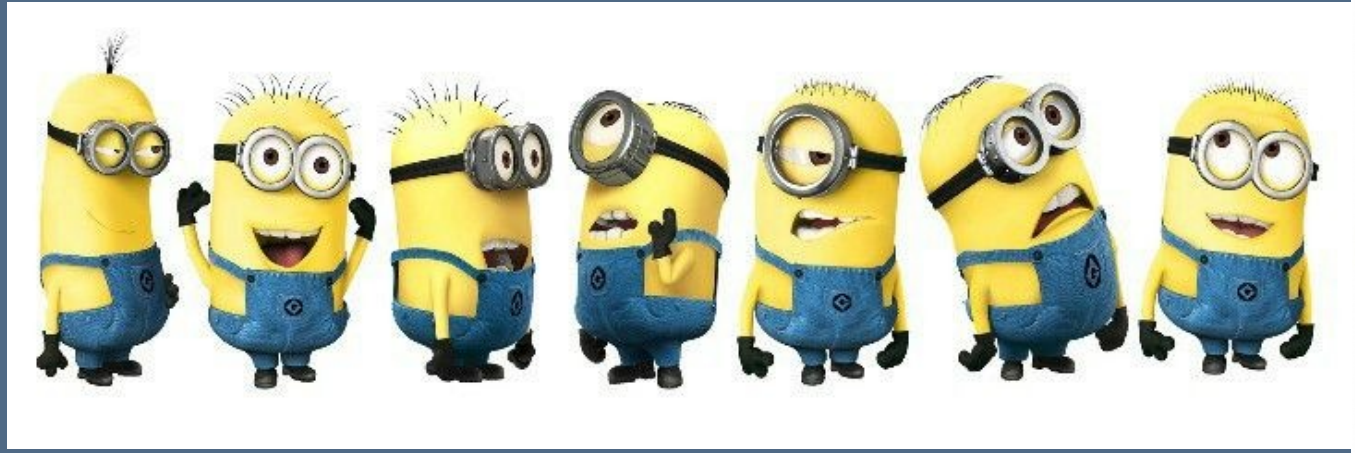
Internal storage
Indicates an internal storage device.

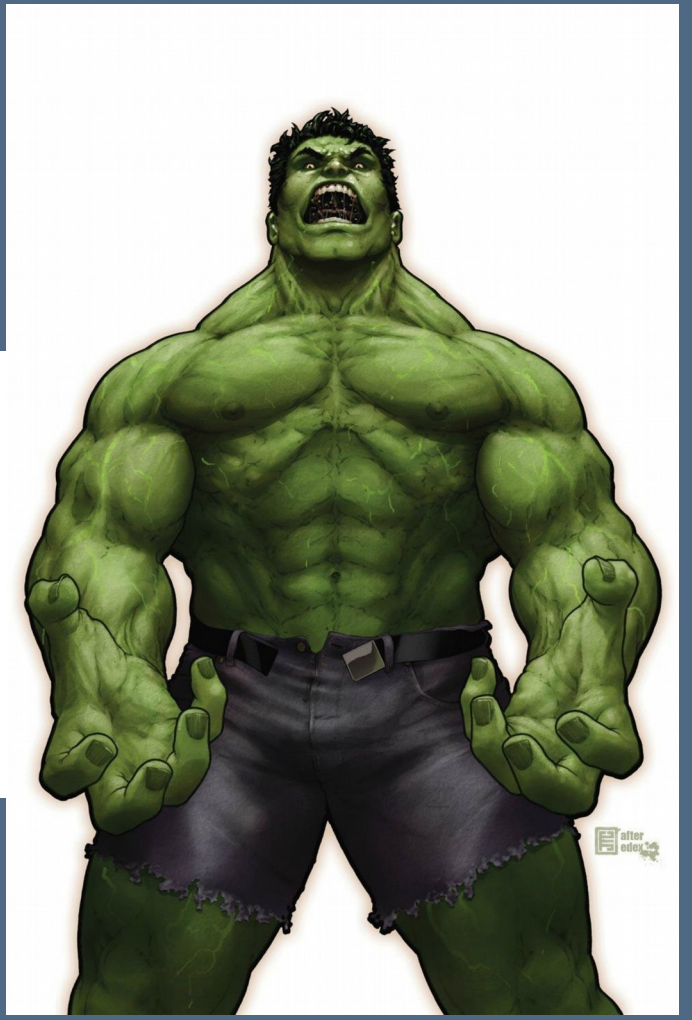
EXERCISE 1

Design a program to control a fridge. Your fridge has:

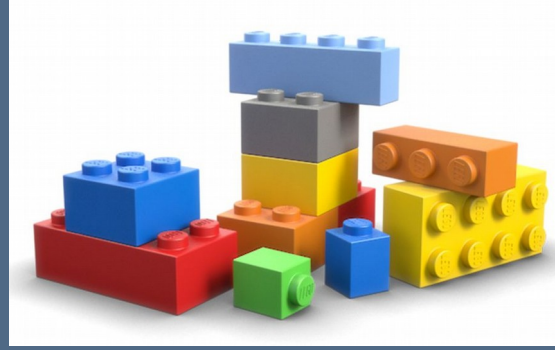
+ A thermostat to read the fridge's temperature.
+ Cooling compressor to cool down the fridge.

Pick a 1vs1 fight





Think MODULAR

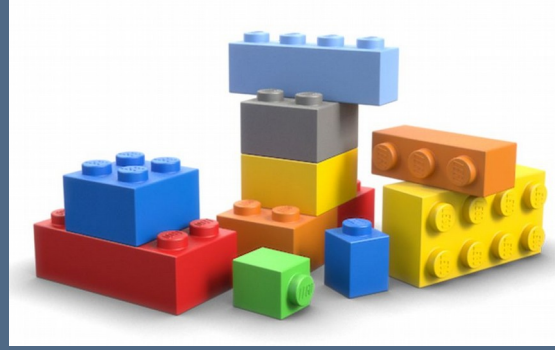


+ Divide and conquer.

  One big, difficult problem vs many small, easy problems

+ Small, simple programs are easier to understand & maintain.

Think MODULAR

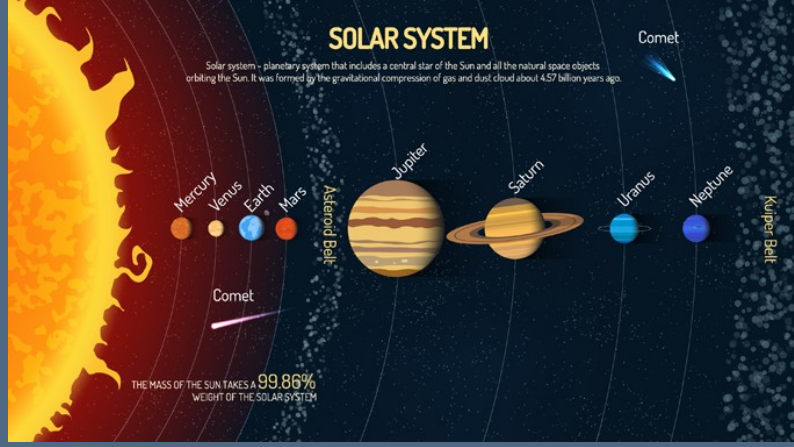


+ Program = multiple modules.

+ Each module is a self-contained task.

+ Modules are combined by a main program

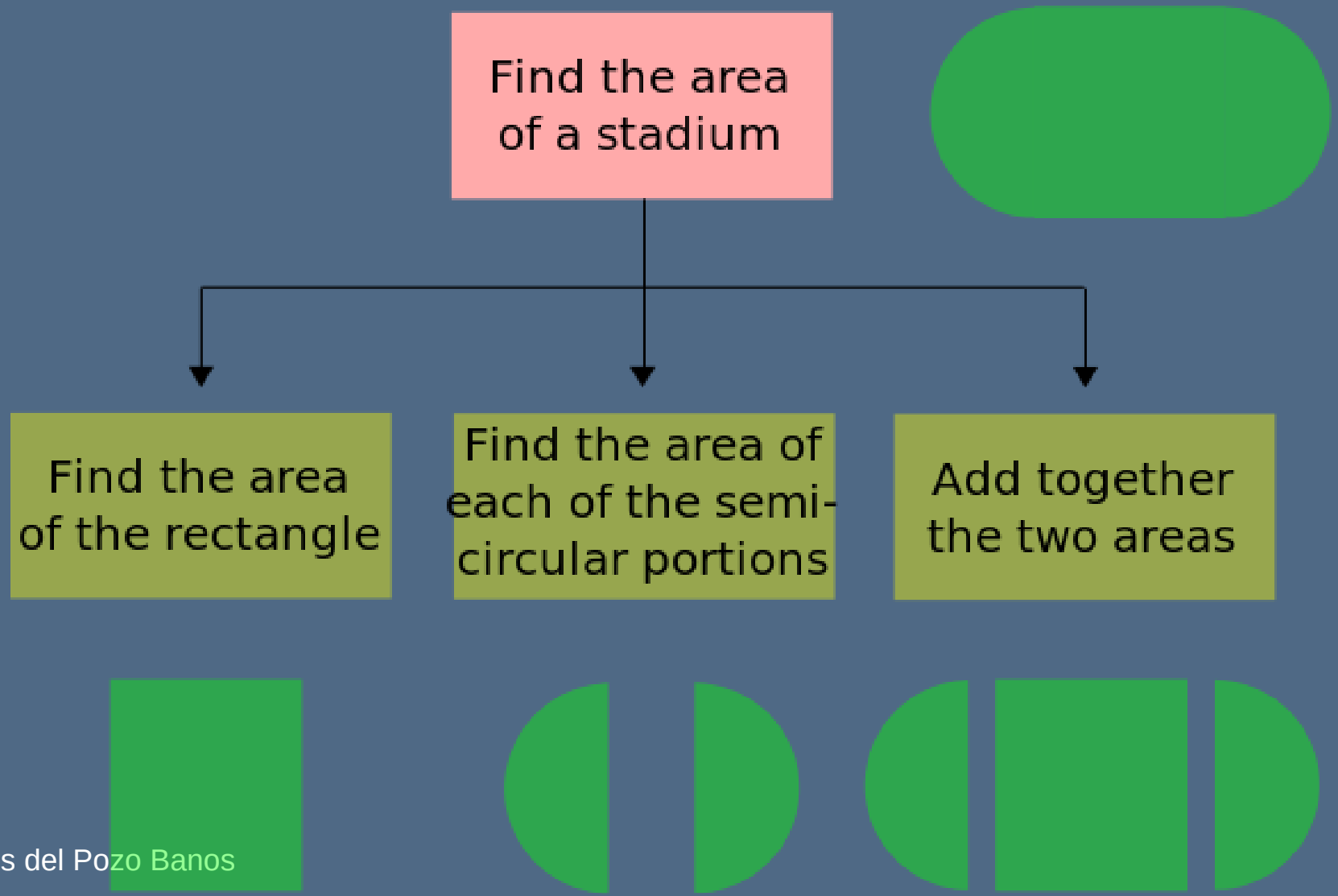
Think MODULAR



+ Planetary system = multiple planets.

+ Each planet is a self-contained world.

+ Planets are bound by a star.


Find the area of a stadium
Find the area of the rectangle
Find the area of each of the semi-circular portions
Add together the two areas

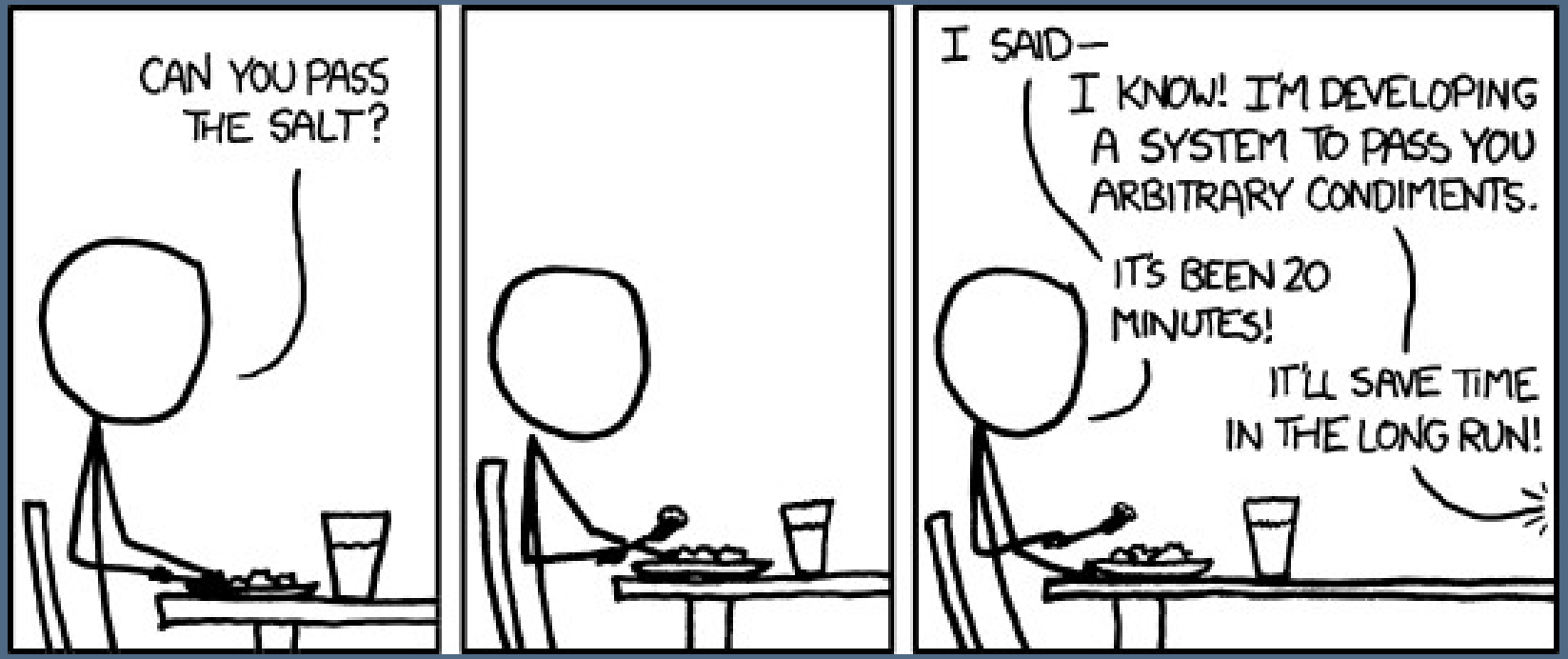
EXERCISE 2

Design a program to keep a car between the lines in a motorway. You have:

+ Two sensors (left and right) to read distance to the lines.
+ Steering wheel.

Do not repeat yourself
Do not repeat yourself
Do not repeat yourself

Be clever...



… but not too clever!

R FUNCTIONS

```
# Function's documentation
#   - What it does
#   - How to use it
#   - Examples
[Name of function] <- function([arguments]) {

    [function's body]

    return([result])
}
```

R FUNCTIONS

```
# Return sum of squares
#
# Input arguments:
#       a, b: Numeric values
#
# Returns: a^2 + b^2
#
# Example:
#       c = sum_of_squares(2, 3)
sum_of_squares <- function(a, b) {
    c = a^2 + b^2
    return(c)
}
```

EXERCISE 3

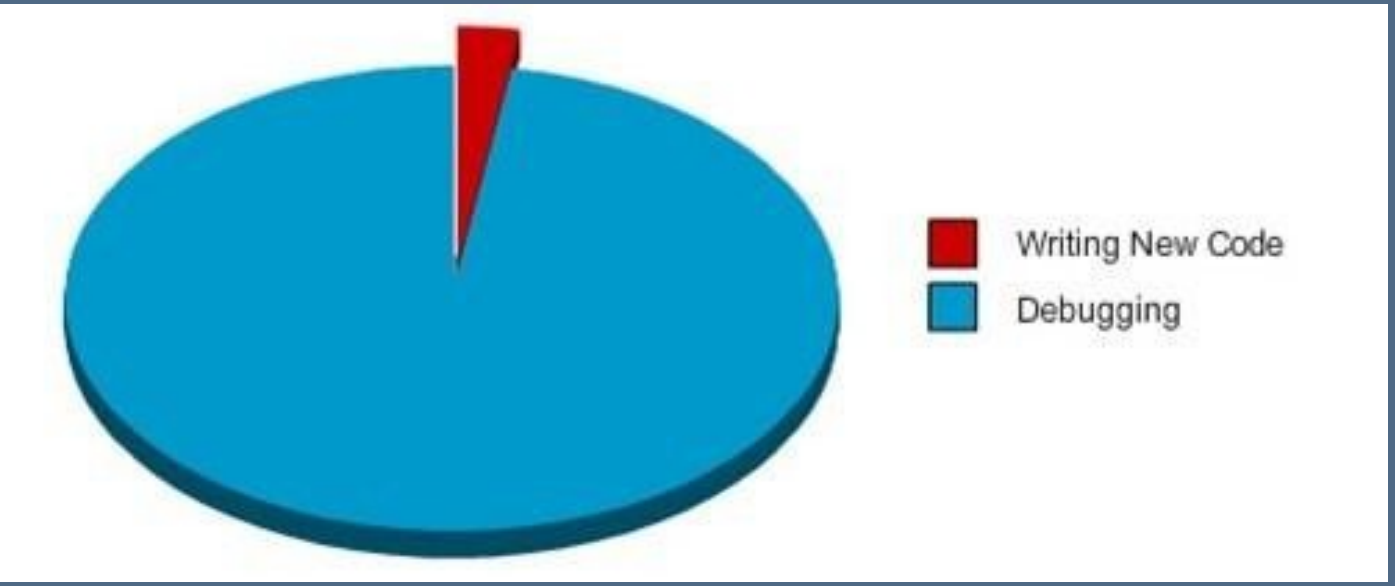
Write a program on R that computes the area of any rectangle.

+ Input arguments: side1, side2

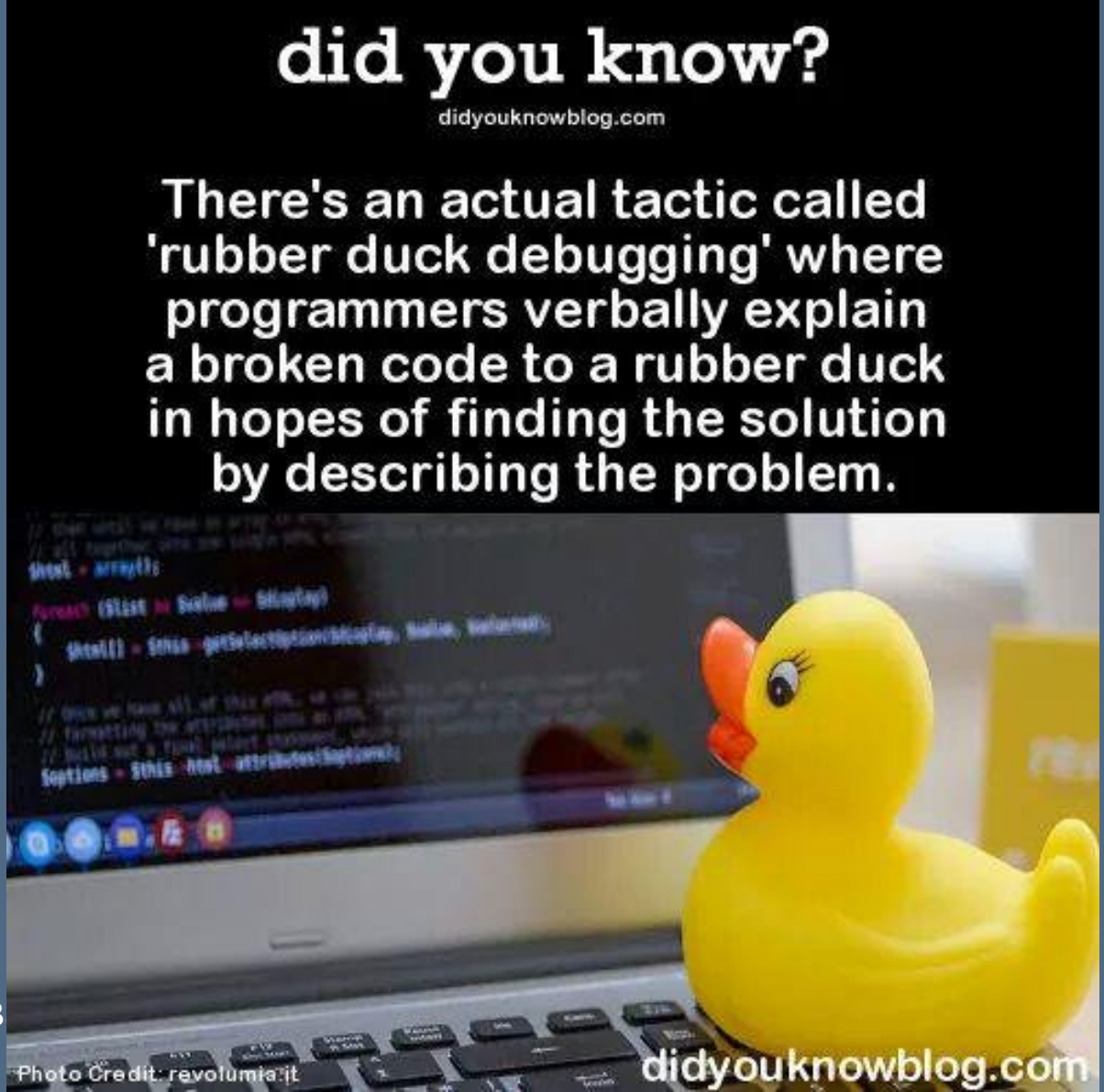
+ Returns: area of the rectangle with dimensions side1 x side2.

When things go wrong…

A programmer's life



Debug


did you know?
didyouknowblog.com
There's an actual tactic called 'rubber duck debugging' where programmers verbally explain a broken code to a rubber duck in hopes of finding the solution by describing the problem.
Photo Credit: revolumia.it
didyouknowblog.com

DEBUGGING

+ Add temporary “print” lines to find where the error is.

+ Print variable values.

LOGGING

Use a DEBUGGER

EXERCISE 4

Locate the errors in the R script provided.

UNIT TESTING

Code testing code

Do not reinvent the wheel

+ CRAN package repository.

+ Google it.

Marcos del Pozo Banos

Reinventing the wheel.
Knowing *when* and *how*.

Make your code reusable for you and for others

EXERCISE 5

Write the following function in R:

```
# Count of elements in a list.
#
# Input arguments:
#    l: (list of int) List of integers
#    v: (int) Value to count in l
#
# Returns: (int) Count of v values in l
#
count_event <- function(l, v){
    …
}
```

EXERCISE 6

Write the following function in R:

```
# Finds the list of unique events
#
# Input arguments:
#    l: (list of int) List of integers
#
# Returns: (list of int) List of unique events
#
unique_events <- function(l, v){
    ...
}
```

EXERCISE 7

Write the following function in R:

```
# Finds the list of unique events in a list and count their occurrence.
#
# Input arguments:
#    l: (list of int) List of integers
#
# Returns: (data.frame) A data.frame with columns “event” and
“count”.
#
count_all_events <- function(l, v){
    …
}
```

CHALLENGE

Design a program to control an elevator.

+ Sensor measuring current floor.

+ Control panel inside elevator.

+ Control panel on each floor.