Writedown Documentation

Paul Joshy

March 21, 2020

Contents

1 Introduction

This is a format that I've devised for jotting down pseudocode. I think it is better because

- It forces us to think like a compiler and stick to rigid rules
- It's top to bottom and we don't have to go back and edit what we've written
- It's equally easy to write it down or type it out in a regular computer
- It is engineered for repetitive output so it's easier for our brain to recognize patterns easily

Idk if it's useful or not but if I'm being ambitious I think it can

- Work as an export format for logging
- Can automate test cases by checking congruency of diff writedown files
- Can be used to teach introductory CS to almost anyone without the need for an actual computer

2 Basic rules

- All variables are functions
- All functions should return output even when piped
- if you're in a function (f prefixed) block or an argument (<>) block, you don't have to give output
- deferred output (written in o block) should give outputs with variables or expressions. eg `variable[output]` or `?[output]`
- all outputs except in <> should only be from one variable at a time
- functions have to be rewritten from scratch when they are changed or they can be piped to include changes

3 Syntax

3.1 Basic output

- Outputs are usually enclosed within square brackets `[out]`
- Outputs returned by functions are enclosed inside angular brackets seperated by argumenets `<arg1[out],arg2[out]>`
- Outputs returned by constants are enclosed inside double square brackets `[[const output]]`
- We can append outputs inside outputs using curly braces `["Hi, My name is (name)"]`

3.2 Variables and constants

- Variables are denoted by regular non spaced words eg: `var1`
- They can be assigned values using the `=` sign eg: `var1 = "val"`
- constants are denoted by prefixing c before the name eg: `cpi=3.14`

3.3 if block

- An if block expression ends with `?`
- it has a true `t` block and a false `f` block
- they both are ended by a `et` block or a `ef` block respectively

3.4 functions

- Functions are denoted by prefixing f before the function name eg: `fprime`
- They are ended by prefixing e before the function name eg: `eprime`

3.5 Output block

Outputs blocks are called when deferred output exists. Deferred outputs are outputs that are written long after the expression is written. for eg, calling a function or a loop. They are called using `o` and `eo` blocks

3.6 Comments

You can use Single line `#` comments

4 Code example

Here is the code for doing the following algorithm

- Step 1: set loop = 0 and doubler = 0
- Step 2: Add 1 to loop
- Step 3: Multiply doubler by two
- Step 4: if Loop $<$ 4 go to step 2
- Step 5: Return doubler
- Step 6: End

eg code

```
# simple looping function
loop[0], doubler[1]

fforloop<loop,doubler>
loop++
doubler*2
loop<4?
t
=forloop<loop,doubler>
et

f
r<doubler>
ef

eforloop

=forloop<loop[0],doubler[1]>
o
# 1st iteration
loop[2]
doubler[2]
?[true]
# 2nd iteration
loop[2]
doubler[4]
?[true]
# 3rd iteration
loop[3]
doubler[8]
?[true]
# 4th iteration
loop[4]
doubler[16]
?[false]
<doubler[16]>
#
eo
```