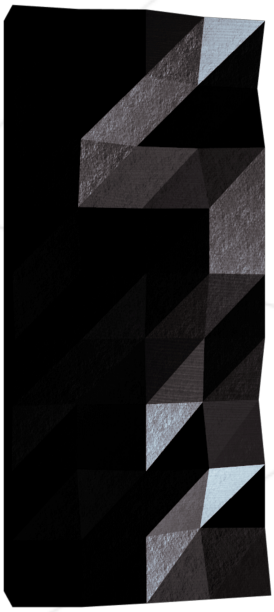


APCSP - Part04

Abstraction and Keyboard Control

Kai kai@42.us.org

Summary:

Contents

Chapter I

The Concept of Abstraction

For APCSP you must form some understanding of how `abstraction` is used in computer science, and be able to identify and use it. Within the umbrella of abstraction you should learn about `functions` (AKA procedures) and `objects` (next PDF).

Abstraction is all about simplifying the amount of mental work we have to do to understand or write the code. We create generalizations and compartmentalize.

Here are some examples of abstraction in everyday life:

- A map of your city is an abstraction of the actual land. It uses symbols to represent where streets, buildings and parks are, and it's simpler to look at than a sattelite photo.

- Teen movies use stereotypes as an abstraction to represent an average high school campus. The "geeks", "jocks", and "goths" are abstract representations of people who are, if you get to know them, complex human individuals.

- If you ask a little kid to draw someone they will probably draw a stick figure with a smiley face. It represents traits that most humans have in common such as having a head, eyes, mouth, nose, arms, and legs, and standing on two feet. The kid will be able to modify their abstract "person" figure to represent some basic information such as how tall people are relative to each other, or if someone is missing a limb, or has long vs short hair.

- Whether your walk, bike, take transit, or drive to and from school, you probably memorize the route after many repetitions. The memorized actions are an abstraction that approximates your actual actions. You have formed a 'procedure' such as: go out the front door, take a left, keep going until the church, turn right, stop at the light, continue for three miles and then you're there. In general you do the same thing each time, but you have other strategies that might modify your routine - such as what to do if it rains, or if you need to carry more than usual, or if a street is closed.

- A bank account is an abstraction which appears on the surface to be a container that you put money into and take money out of. In reality, it is a complex legal

relationship between the bank and its customers, and the money is stored as bits on a computer - they don't actually keep a box with cash in it for you.

- An atom is an abstraction that represents several protons, neutrons and electrons interacting with each other. A molecule is an abstraction that represents many atoms interacting with each other. It's too complex for us to "see" the landscape of physics as one electron at a time, so we simplify our perception by categorizing them into larger units.

- In any programming language that you use, the keywords and functions used are abstractions for `machine code` that is much harder for humans to understand. When you run your code, the computer translates it into its own machine code language, and then follows those commands.

`Procedures (AKA functions)` are abstractions because they provide a general script for how to complete a task, while allowing the script to be customized in predefined ways.

`Objects` are abstractions because they create a "black box" that a programmer or user can interact with, without understanding every detail about how it works.

Chapter II

Appendix on using p5js Without Setup & Draw

You can write code in p5js outside of the setup() and draw() functions. If you want to use it as a space to play with coding logic instead of drawing figures, just replace every print() statement with console.log() instead.

You can also add "new p5js()" to the first line to make sure that functions like random() still work.

sketch.js • Saved: 4 hours ago

```
new p5();

// Example from Part 04
function randomPowerOfTwo(){
  var exponent = round(random(12));
  return pow(2, exponent);
}

for(var i = 0; i< 10; i++){
  number = randomPowerOfTwo();
  console.log(number);
}


// Examples from Part 01
var numturns = 2;
var points = 100;
var easy_mode = false;
var single_player = true;
if ((single_player && easy_mode) || (numturns < 2 && points < 99)) {
    print("Take another turn");
}

var number = 7; if (number < 0) {
print("Your number is negative.") } else if (number === 0) {
print("Your number is zero.") } else {
    print("Your number is positive.")
}
```

code editor

Console Clear

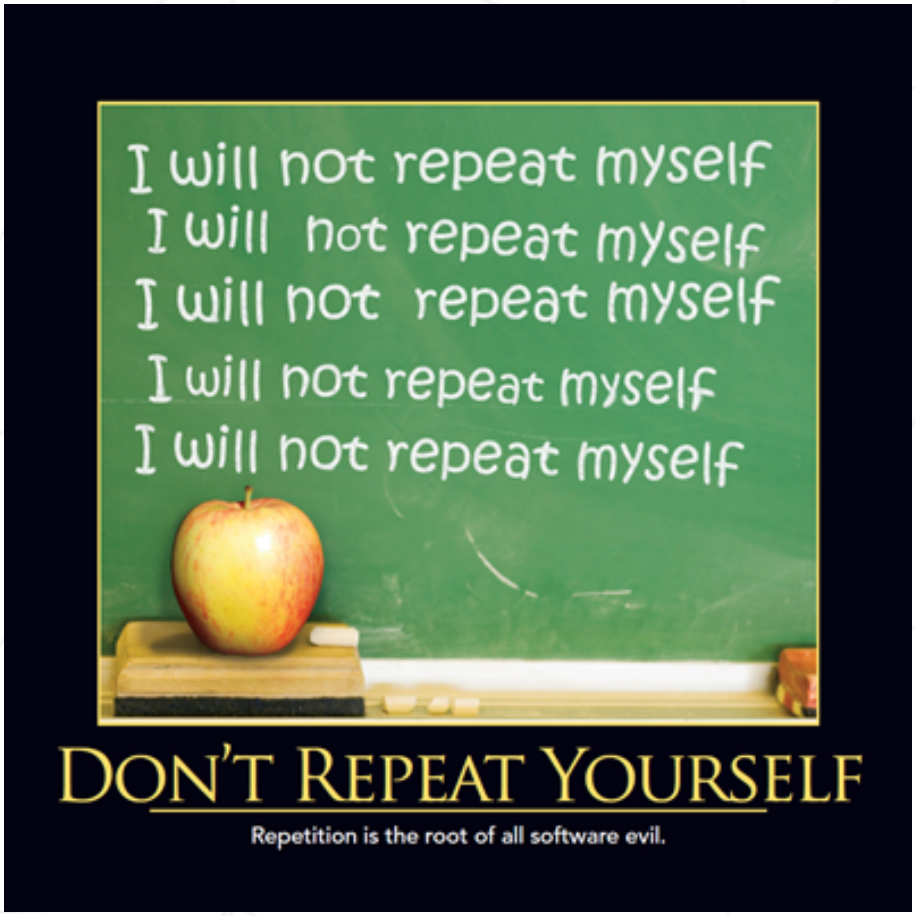
```
4
64
128
2
8
32
2
32
4
256
Your number is positive.
```


Chapter III

Write your own Functions

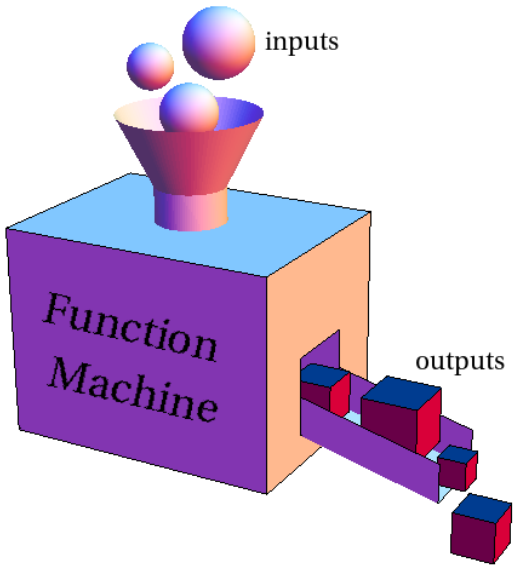
III.1 Discussion

In computer science there is an important motto called DRY or DIE, which stands for: "Don't Repeat Yourself", and "Duplication Is Evil." Following the DRY principle will make you a faster coder and make your code easier to read. How is it possible to avoid repeating yourself?... Build your own functions!

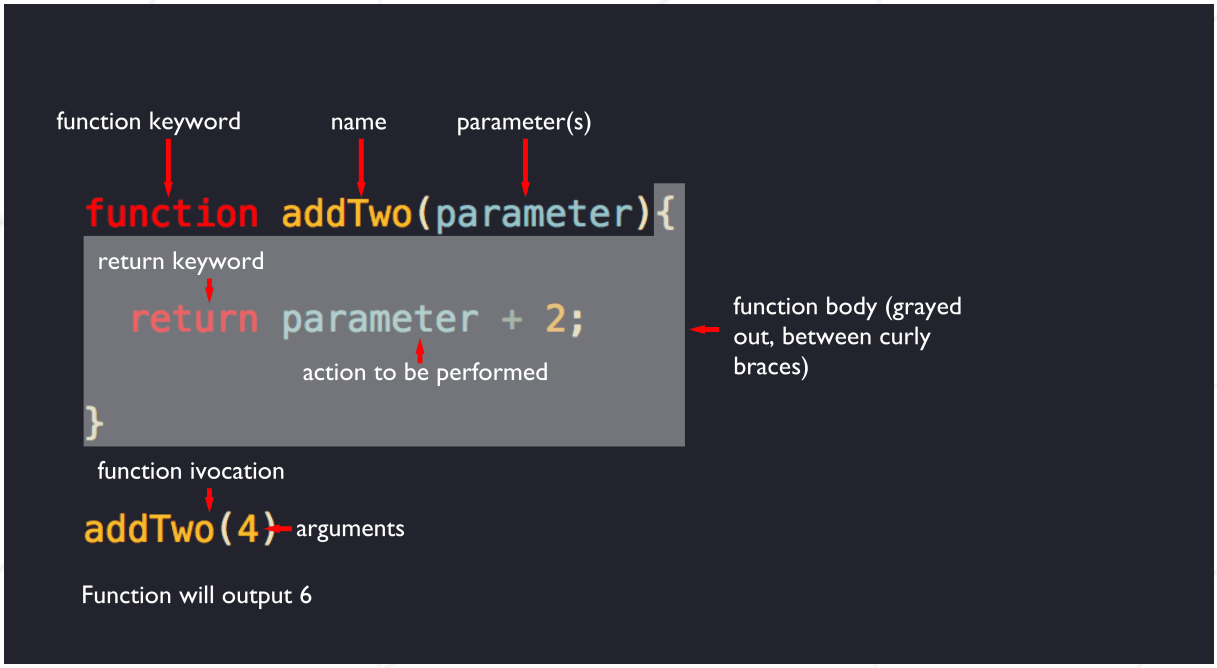


(image from DevIQ)

As we discussed earlier, `functions` (AKA `procedures`) are blocks of code that can be re-used. They are able to take in parameters and return a result. Whenever you are writing some code and find yourself writing the same or similar logic more than once, you should think about taking the repetitive code and making a function out of it.



(image from MathInsight.org)



(image from Bryan Smith, Frontamentals.com)

```
Please don't put functions inside of functions!  Just don't.  It
may be a thing in some programming language somewhere but it is not
standard.  Let them each have their own space.
```

III.2 Example: One-Line Text Box

Here's an example using the most common repetitive code we have seen so far: drawing a text box.

It's repetitive, because we are using the same `algorithm` three times with slightly different settings. Each time we set the fill color, draw a rectangle, change the fill color, and then write some text.

Think about the ways each text box is similar and different:

- If it's only one line of text, they can all have the same height.
- They all have different widths, but it should be scaled by the length of the word.
- The coordinates given to the text() function can be calculated relative to the coordinates used for the rect() function.
- They might use different background colors and text colors.
- Different text boxes hold different text
- They are located at different x, y coordinates.

We will create a `parameter` for everything that changes from one text box to another. Then, we need to write a general algorithm which is useful for drawing text boxes with different locations, contents and colors. We'll write that algorithm inside a new function, which we get to name: I'm calling it "textBox()".

III.3 Functions with Return Values

The `return` keyword allows you to specify an `output` from your functions. If you do not tell the function to return something, it will return the symbol `undefined`. To make use of a return value, you can assign it to a variable with the syntax "var myvar = myFunction();". In the example below, I wrote a program that prints out a random power of 2, ten times. The variable called "number" gets assigned to the return value of my "randomPowerOfTwo" function. The for-loop runs ten times, so "number" receives ten different values, one at a time.

You can also use a function return value by calling the function anywhere in your code. In this example, I created a function that returns true or false, and call the function inside of an if statement. In this other example, I created a function that returns a string, and used that as a parameter for my textBox() function.

III.4 Homework Problems

III.4.1 Fix repetitive code by relying on variables

Sometimes a new function isn't necessary, but by thinking intelligently about how to use variables, you can reduce code repetition.

The sketch Typer uses a built-in function, keyTyped(), which works similarly to mouseClicked(). In keyTyped, the p5js fills in the "key" variable with the letter that was typed. Make a copy of this sketch and fix the repetitive code by adding the contents of the "key" variable to "message" regardless of which key is typed.

III.4.2 Write a function for rolling two dice

The sketch Game of Chance is a dice-rolling game. On each turn, Player A and Player B both roll two dice and add the total to their score. Can you fix the repetition by replacing lines 29 through 43 with a function? The function should return a number representing the number rolled by two dice. Then the draw function should use those return values to increase each score.

III.4.3 Add parameters to make the text box more adaptable

Modify the textBox() function so that it takes an additional parameter, text_size, which will change both the height of the text box and the size of the text.

III.4.4 Remix code by adding another function

Do something interesting with the Flower Power code written by Kelsey Mason on OpenProcessing. Requirements: There should be at least three shapes, and a random number should determine which one is drawn in each box. Use a different function for drawing each shape. Suggestion: Just add a third function which draws a peace sign, and modify the draw function so that you get a random number between 0 and 3, and there is another if-else section in the middle of the the if statement which will call the peace sign function if your random number is larger than 2.

III.4.5 Practice with keyboard control

Work with a partner and complete one but not both of these for credit:

1. Create the game mechanic where there is a character (a circle or whatever) which you can control with the arrow keys or w,a,s,d. You should be able to move the

character up, down, left and right in a convenient way using the keyboard. The character should collide (not be able to move) if it is at the edges of the canvas. (Hint: Check the reference for windowWidth, windowHeight, and the built-in width and height variables.)

2. Create a simple typing game where a string of random letters is shown on the canvas, and the program acts as a stopwatch to record how long it takes you to type those words or letters. (the built in variable frameCount can act as a timer).

Are they difficult? Get a P2P discussion going on Slack!

