COMPILER DESIGN FOR A NEW ESOTERIC LANGUAGE

Vinayak Kaushik (15BCB0028)

INTRODUCTION

The main aim of a compiler is to convert source code in a high-level language into code in a low-level language.

Due to the complexity of compiling languages, the process is split into several phases, which helps ensure that the overall process produces high-quality output and be easily maintained. The separation of stages helps ensure compiler correctness, which is extremely important due to compiler errors being so difficult to track down.

What I propose to do is to create a combination of two Esoteric Programming languages. “Piet” and “BrainFox”. These two language’s have their own syntax’s and different characteristics.

I would be using the syntax of the Brainfox language and then convert it into a graphic image for storing the program.

There are eight commands:

Symbol	Interpretation
+	Increments the value at the current cell by one.
-	Decrements the value at the current cell by one.
>	Moves the data pointer to the next cell (cell on the right).
<	Moves the data pointer to the previous cell (cell on the left).
.	Prints the ASCII value at the current cell (i.e. 65 = 'A').
,	Reads a single input character into the current cell.
[	If the value at the current cell is zero, skips to the corresponding] . Otherwise, move to the next instruction.
]	If the value at the current cell is zero, move to the next instruction. Otherwise, move backwards in the instructions to the corresponding [.

[and] form a while loop. Obviously, they must be balanced.

Proposed Color-Symbol Relation:

Hue\Darkness	Same	Darkness Change
1 Step	+	-
2 Step	>	<
3 Step	.	,
4 Step	[	]

Color Palette:

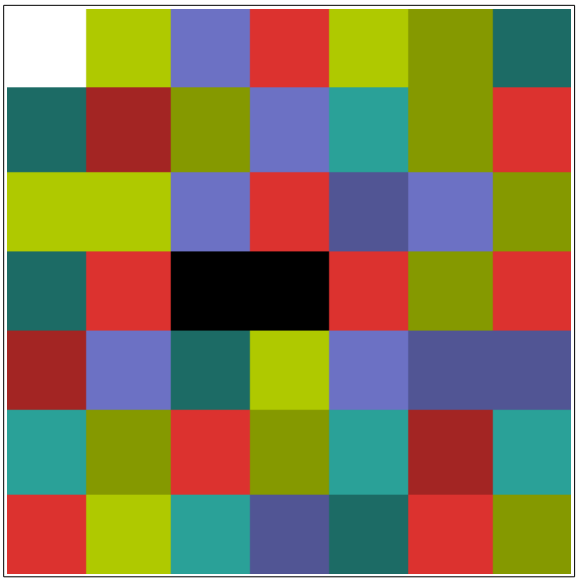
RED	#DC322F	#A32523
VIOLET	#6C71C4	#515594
CYAN	#2AA198	#1C6B65
GREEN	#859900	#AFC900

White: Program Start
Black: Program Ends

Steps to convert it into an image:

1) count the number of characters in the program.
2) Find the square root of the number and round it up to the nearest whole number.
3) Start with a white pixel in the top left corner.
4) Calculate the next hue and darkness change for the next symbol and fill the next pixel with that value.
5) Fill all the pixels in a clockwise direction.
6) When all the characters have been filled. Fill the remaining pixels with black.

Sample End Program:



Here, our program starts from the top-left corner, and then goes around in a clockwise direction.