switch and arrays

# design: switch

Overall, the plan with switch statements is to reuse as much code as possible from the existing if statement system because they share very similar structures, with only slightly different syntax and evaluation methods.

Therefore, I am going to have a redesign of the if statement parsing system as well. However, this manly involves lots of moving of code into different methods, as overall the system will be the same, but it will become more generic in order to allow switch statements to also be parsed.

# development: switch

## A computer screen shot of a program code Description automatically generatedA screen shot of a computer program Description automatically generatedparser redesigns

As discussed in the plan, each of these methods will be called at each new case to parse it.

For the if statements, this code is very similar to previous: with if\_statement() basically being changed to if\_condition(), as well as now creating its own instance of Case rather than it being passed due to elseif now being a template keyword so that the Case class (the old IfCase class) can be reused.

Previously there was no parsing for the else condition, and I have managed to add some additional error checking, as previously if there were tokens after “else” they would be ignored rather than throwing an error, and now this check has been added so that this is the case.

The new switch equivalents are very similar, but instead of using “then” they use colons, as well as different error message to reflect their issues. The else, default case also requires a colon so that also must be added. When implemented, I will properly test them and add all the needed error handling.

### A screen shot of a computer program Description automatically generatedA screen shot of a computer program Description automatically generatednew build methods

The responsibility of these methods is to correctly parse everything before the contents of the first case. For the if statement, this is simple and involves parsing the if statement on the first line and advancing before calling the new method.

The switch statement has two lines before content, as the switch expression must also be evaluated before the first case, so that is why there is a lot more content. This is not yet properly error handled.

### generic building method

A screen shot of a computer program

Description automatically generatedThis will be called by both of the build() methods and will accept the main if or switch token as well as the initial case.

This is because after this point, both types of statement are parsed with the same structure, just using different keywords, which is why the corresponding keywords are all defined in the first line to reduce the number of checks.

I also managed to make this program a lot more readable. Originally, within the else section there was another loop which was essentially a replica of the main loop but without the if or else cases, which was a lot of repeated code.

Now the solution adds an additional variable that was not on the plan: allowCases, which is true by default, and is set to false when an else or default case are encountered.

This therefore no longer allows further cases to be parsed, which has the same effect of the inner loop but significantly reducing the amount of nesting and increasing how easy the program is to understand.

Overall, having this generic method reduces writing out an almost identical switch-building method so is a positive feature.

### debugging

A screen shot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generatedA computer screen with text

Description automatically generated