Mitchell Coakley

mcoakley@uoguelph.ca

Student # :1013103

March 8th, 2021

CIS*4650 Checkpoint 1

Initial Analysis

Immediately after starting the project, I was rather confused. The file structure of the default parser is perplexing, and I have never used CUP before. My initial idea was to follow the steps described in the assignment description exactly. After quickly completing the JFlex scanner for C-, I attempted to get started on the parser. While the scanner took almost no time to understand, the parser was far more complex. After some time analyzing the tiny language parser given to us, I attempted to adapt the same structure to use on the C- language. After copying the BNF for C- from the C-Specification the CUP scanner appeared to be following the correct patterns, however it was not outputting an abstract syntax tree.

The actual abstract syntax generation was the hardest part of the assignment by a long margin. As I was already rushed for time due to a lengthy geography report due a few days prior, I neglected to see the part of the assignment description recommending me look to the lecture notes on “6-Syntax Trees”. As a result, I made the syntax tree generation almost 1:1 to the BNF specification described earlier. This may have to be corrected later, however, for the time being it appears to generate valid syntax trees. Unfortunately, at this point, I had run out of time, and the project demonstration was rapidly approaching.

Project Demonstration

As I was woefully underprepared for the demonstration, I only presented a version of my program that can generate a syntax tree, but not output it. The actual class generation was finished at this point, but the code to display it in ShowTreeVisitor.java was still missing. Thankfully, I received some advice on how to further improve my scanner/parser.

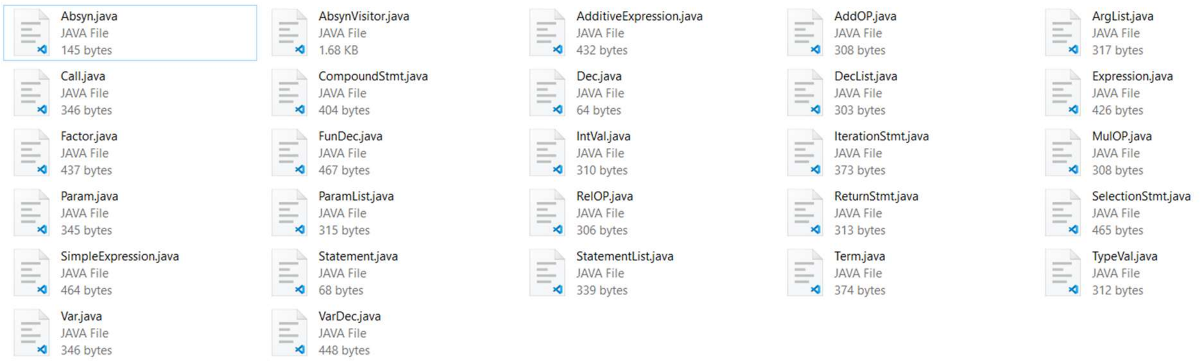
My scanner, which I thought was flawless, had an error in detecting C style multiline comments. I was redirected to lecture slide 35 in 2-Scanning. This slide described JFlex states, which are very useful in detecting comments. You can enter a “COMMENT” state and only exit when the proper “*/” characters are shown. This is difficult to do with regex as regex does not have a “not” operation for 2 characters in succession.

At this point I was also shown the simplified version of the AST for C- in slide 7 of 6-Syntax trees. The problem had already taken hold though. I had already designed a functioning system based on the BNF specification we were given. Also, the AST we were given is not the entire story. The CUP rules would also have to be described in terms of this new abstract syntax tree structure. Due to this, and the fact that the syntax tree I had already made was valid, I decided against converting my syntax tree to the form of the one described in the lecture.

At the end of this presentation, I felt as though I had a better understanding of what I had just worked on, and what I still had left to do.

Being Late

After a lengthy Geography related essay, I started work on the assignment on Saturday morning. This should have given me enough time to complete the assignment, if not for a series of mistakes made. Immediately I tried to “jump right in” to the assignment without looking at much documentation or code. This worked fine for the scanner, I had previously used JFlex before. When it came to the parser however, using the BNF directly from the C- specification may have been a mistake. There are a massive number of non-terminals in the BNF and this made it tedious to write the productions and generate the classes for each one.



Absyn.java JAVA File 145 bytes	AbsynVisitor.java JAVA File 1.68 KB	AdditiveExpression.java JAVA File 432 bytes	AddOP.java JAVA File 308 bytes	ArgList.java JAVA File 317 bytes
Call.java JAVA File 346 bytes	CompoundStmt.java JAVA File 404 bytes	Dec.java JAVA File 64 bytes	DecList.java JAVA File 303 bytes	Expression.java JAVA File 426 bytes
Factor.java JAVA File 437 bytes	FunDec.java JAVA File 467 bytes	IntVal.java JAVA File 310 bytes	IterationStmt.java JAVA File 373 bytes	MulOP.java JAVA File 308 bytes
Param.java JAVA File 345 bytes	ParamList.java JAVA File 315 bytes	RelOP.java JAVA File 306 bytes	ReturnStmt.java JAVA File 313 bytes	SelectionStmt.java JAVA File 465 bytes
SimpleExpression.java JAVA File 464 bytes	Statement.java JAVA File 68 bytes	StatementList.java JAVA File 339 bytes	Term.java JAVA File 374 bytes	TypeVal.java JAVA File 312 bytes
Var.java JAVA File 346 bytes	VarDec.java JAVA File 448 bytes			

The most detrimental error made during this assignment was almost certainly having a lapse of judgement and not realizing that any CUP production could be tested at any time instead of testing them all at once. Before the project demonstration, I made nearly all the abstract syntax tree classes and CUP productions at the same time and did not individually debug any of them except the most basic ones. Testing that much code at a single time was a nightmare I would not inflict on my worst enemy. Shortly after my demonstration, I had the abstract syntax tree printing code finished and I was ready to move on to testing.

Error Correction

For the most part, this was the easiest step. I identified common errors that could exist in a program, then used CUP error recovery to cover up these errors and continue generating the AST. The final AST is not accurate when there are errors because of the faked elements, however multiple syntax errors can be recognized because of this approach. Some errors I chose to look for include:

- Invalid function declaration types
- Invalid assignment statements
- Invalid function calls
- Unknown character in program

Any of these errors, including some combinations of them, can be successfully skipped over. Files [2345].cm include these errors and can be used for testing the error handling of the parser/scanner.

Conclusion

The most important thing I have learned from this assignment is that taking time at the beginning of a project to fully understand it is necessary, even if it seems like it is not. That might seem obvious but clearly me of a few days ago did not think so.