Checkpoint One Cis*4650 Compilers

Ryan Taylor, Jackson McAvoy

Design Process

For this assignment we started off by trying to find the problems with our first checkpoint. We solved a few problems, and discovered a few work arounds that allowed us to move on to the next stage. From here we started on the hash table. We decided to have it traverse through the AST made in checkpoint one, adding variables and types into it as it went. Next we assigned different variables to function blocks, this allows us to know the scope of each variable, and keep track of which instance of a variable is relevant to an assignment or evaluation.

From there we created our symbol table that would be displayed to the user. Each block is considered a sort of separate entity, so when it's left, local variables to that block or scope are no longer included in the table of symbols.

At this point we started error checking. The first step was to go through opExp, a tree structure and ensure that all participant variables were ints, and that none were void, or used a void function for a return. At this point we created a few test files in the form of the 1-5.cm files. After a few rounds of testing we moved onto return types within functions to make sure voids weren't returning anything, and ints weren't returning voids. At this stage we went through a few quick rounds of error checking before finalizing everything.

Limitations

From the parser in the last assignment we had an issue or two we could not find resolutions for. The program can't handle more than one mulop in a single equation, so in some cases it needs to be split up.

```
int x; int y;
y=1;
x=13;
y = x*x+y*x;
```

becomes

```
Int x; int y; int temp;
Y=1;
X=3;
Temp=x*x;
y = temp+y*x;
```

the program will also require something to be inside a function. Prototypes don't seem to be working.

If you use an if else statement, there must be a trailing else.

There are some known issues with some tokens being evaluated before others such as == being seen as an = token, and then a lone unused = bungs up the code. This is resolvable through a precedence change, however by the time we discovered it we deemed it not worth the risk of introducing more bugs into the program.

Improvements

There are a few improvements we could have made, as well as a couple that we should have made. Firstly, at times it can be a bit difficult to understand what’s happening in the symbol table, its possible we could have cleaned that up as a quality of life improvement. We also had some known issues with the grammar that should be cleaned up before checkpoint three. However we didn’t deem it an efficient use of time to fix and then test the changes when they didn’t impact the checkpoint 2 marking scheme. Error checking in the past checkpoint could also be improved, but since this checkpoint works under the assumption of a syntactically clean c minus file we decided to focus our efforts forward.

Lessons

We learned from this checkpoint that past mistakes have a tendency to pile up on you. For every minor error in checkpoint one there was a need to fix, or create and find a work around for it in checkpoint two. It’s likely that we have some similar problems in this assignment, but overall we feel we gave ourselves enough time to do a good job with what we had available.

As a pair, we’re learning how to communicate better. We both had different ideas on code implementation and design, so finding ways to give succinct explanations on how something works, and what it’s doing has been valuable for this checkpoint, and beyond. We’ve started to see where we have strengths and weaknesses and we’re building off of those to more effectively use time, since we both have other important things to do, as well as activities outside of school.

Moving forward, we both hope to enter a more collaborative stream of progress, instead of a back and forth ‘I work on this chunk then you get that slice’ as it sometimes ended up having to be the same person building on their own portion since they understand it better.

Work Allocation (Contributions)

Task in %	Jackson	Ryan
Symbol tables	20	80
Type Checking	40	60
Error Checking	80	20
Documentation	90	10