TODOSKOVA Darya

INDIVIDUAL CSE201 REPORT

Our goal was to make a tool that would allow code visualization and debugging. The project team is divided into three subgroups : the parser team, the AST team and the GUI team. I personally worked in the AST team, which is an essentiel link between the backend and the graphical elements we will use for the visualization cell on our graphic interface.

The beginning of the project was mainly research. I had a general idea of what we were going to implement but didn't know what tools we were going to use to make it happen. I thus started by looking up ASTs, and understood the different steps that were involved (tokenization, parsing then code generation). I also got informed on building a parser/tokenizer from scratch with an online course - until we noticed, with my team, that it would already be generated by the library we chose (ANTLR). I had first read a lot of documentation on the library and wanted to install it to know what output we would be receiving from Team 1. I spent hours trying to install the ANTLR library, after realizing that Team 1 was doing it as well.

Hence, I had continued working with my team on defining the AST. I wrote all the AST-inherited getters and setters in the ast.hpp file. I then had to focus on the transition from Team 2 to the GUI, all the while refining our AST classes. Indeed, I added basic methods to our classes in ast.hpp and coded a type_check for every enumeration type in our classes.

I also managed to finally set up ANTLR on macOS (which was apparently easier than on Windows) and visualize the files it generated for us from Team 1 (lexer and parser), so that at least one member of Team 2 had access to it, and so that we could freely do some testing.

In addition, with my teammates, we worked on looking for a concrete method to walk through the AST (useful for the transition to the GUI team), and the classes we were going to use to make that happen. I worked on constructors for our Statement classes (for when a dictionary is passed by reference as an argument to it) with Milena. I also modified certain classes (binop and unop) to get the values of expressions.

Moreover, my team and I had spent weeks looking up ideas of how to refine the walker and the different techniques we were going to use to make it efficient. I then started implementing the walker, as did Matea, and looking for errors in a few classes. In addition, I started working on the counter for the lists we already had, but the task was then taken over by Milena since I had technical issues. I also had to get in touch with the GUI team to know how they wanted their cells to look like to draw the flowchart (shapes, i.e. Block classes).

Besides that, what we needed and didn't have yet, were tests for the expression evaluator - which is what I did. However, due to changes in the implementation of our interpreter (we had decided to make templates for every variable, because it would help us create the necessary arrays), they were obsolete and needed to be re-done.

Finally, with the help of Elsa, I did the presentation slides for our project defense. I also gathered all the necessary information to trace back the whole timeline of our project in order to sum it up for the defense.

Throughout the project, I wasn't in charge of one fixed area of the backend (ex: variable tracking, or the cache, etc), but I was rather given many different tasks (installing antlr, writing tests, refining AST, designing the walker, project presentation) - which allowed me to truly understand the project in its depth and do the connection between different sectors of Algorithm Visualizer as a whole.