Kardelen Erdal 2018400024
Oğuzhan Tok 2019400267

CMPE 230 PROJECT 1 REPORT

1) Problem Definition:

The problem is developing a translator which translates a language called MyLang to LLVM (low-level virtual machine) language. MyLang is a basic, high level language that resembles Python. On the other hand, LLVM is a very low and complicated language for human beings. The aim is essentially creating our own language. Because in MyLang, we decided all the rules and grammar of the language. By translating our made-up language into LLVM, we provided that all the people around the world can use our language, because everyone can download LLVM and execute .ll files.

2) General Structure of the Program:

Our program interprets MyLang code line by line and specifies its type. Its type can be one of the followings: Assignment, Print Statement, If or While Statements or just an empty line.

After specifying its type, it sends the line to the appropriate function. The main functions of the program are:

expression: Every expression in the input file, enters this function. Here, the main thing is calculation. Given the postfix version of the line, this function writes the calculations in LLVM language one by one to the output file. If the given line is an assignment statement, then it stores the result of the right side to the left side.

expressionParser: Every line in the expression function enters this function. It basically parses the line and returns the postfix version of the line as a string vector. If there is a choose statement in the line, it calls choose function and recursively does the calculations and returns the line without any choose. If the line is an assignment statement, first element of the vector is the left side; else first element of vector is empty.

chooseParser: If there is any choose statement in the expression, the whole line enters into chooseParser. This function understands the first choose in the line and divides its parameters. It returns the four parameters of choose function as a string vector.

handleVariable: It takes a string as a parameter and if this string exists in the variable set, loads it; if not, allocates and puts it in variable set. If neither loading nor

allocating are required, it returns the variable as it is. If the parameter is none of them, then it is an error.

intToPost: It takes a string expression as infix and returns its postfix form. We use the code from "*https://www.tutorialspoint.com/Convert-Infix-to-Postfix-Expression#:~:text=To%20convert%20infix%20expression%20to,maintaining%20the%20precedence%20of%20them*.".

While allocating variables, we write allocation lines in output file. But the ones that are not allocation lines are written in a temporary file. After all the lines are executed, the content of temporary file is combined with the actual output file. At the end, the temporary file is removed.

3) What would we do if we have more time?

Since we solve the main problem easily, we would definitely spend more time on error situations. We would try more inputs and see if the output is true. Maybe we could optimize our program because it is over 700 lines now.

4) What are the situations that we had most trouble handling? And how do we handle?

The most difficult thing was parsing the lines, especially choose's parameters. It was hard but we spend time on it and get help from technologies of c++ like strtok. And the other thing we had trouble with is allocating the variables at the beginning of the file. But we handle it by creating a temporary file. And also, since error situations are not clear, we spend too much time thinking of any error situations, maybe we forgot couple of them, or maybe we probe the problem too much.

5) What are we proud of about our project?

First of all, Oğuzhan has never write c++ before, so it is really good for him to do such a difficult project with a language that is not familiar to him. Also, since we planned the architecture of the program at the early stages, it became really easy for us to distribute tasks. We decided not to write an object-oriented programming but functional programming. We use 20+ functions which dedicated to a particular mission. Each function calls another function so that our program looks neat. Even though we work together first time and even though we are in different cities, we did well job.

Kardelen Erdal 2018400024
Oğuzhan Tok 2019400267

- This is a graph of the program to visualize the travel of an input line in the functions. Note that, error cases are ignored in this graph.