Cmpe 230 Documentation

First of all, we wanted some structures that could represent data types of other languages like string, vector and hashmap. Therefore, we started our project with coding these structures. We tried to make them generic but we realized this topic was different in C from other languages. We searched and learned that we can do this by using macros and defines. These structures made the project much more easier and gave the opportunity of developing them as we want.

When we were finished with our structures, we started reading and parsing the input file. First the input file is separated into lines (func: read_line). Then these lines are splitted to words using special characters as seperators(func: parse_line). Special characters are defined as global variables thus we can access them everywhere. We also used our hashmap struct as a storage for the special characters.

After parsing the lines, we needed to adjust every line type differently so we implemented each of them their own function. We differentiate the lines by looking at their first word. We checked the syntax of lines in their handle functions. If there is an expression in line, handle function checks the syntax until nothing except the expression is left. (Expression is a sentence which evaluates to a value or a variable) We checked and evaluated these expressions in other functions(func: evaluateSyntax and evaluate). If there is no syntax error found, handle functions convert the matlang code to C code. We also used a varmap(which is constructed by hashmap struct) to keep types of variables.

Evaluate function first converts the statement to postfix form and then converts this postfix form to a suitable C code. Uses stack for conversions between infix to postfix and postfix to C code. For operations, evaluate function takes the type of input variables from varmap and then after applying the operations, puts the output variable to varmap. However, evaluate function assumes some preconditions hold and these preconditions are satisfied by evaluateSyntax function.

EvaluateSyntax function checks an expression’s syntax recursively. It starts by checking the first word and according to the first word it decides on the correctness of the other words. After checking the compulsory words it sends the smaller expressions to be divided again. If all of the expressions are successfully divided into simple values and no syntax error is detected, function sends this line to be evaluated by evaluate function.

Since we used sqrt function in the generated C file, it needs to be ran with “-lm” command.

Erkam Kavak - Emre Batuhan Göç 2020400174 - 2020400090