CmpE 230 Spring 2017

Project I

Mustafa Enes ÇAKIR **2013400105**

Problem

I think the biggest problem in this project is operator presence. You have to calculate inside of parentheses before all operation. In addition that multiplication and division have the higher presence over addition and subtraction.

The solution came from our *CmpE260 Principles of Programming Languages* course. While we were talking about the syntax of languages, Mr.Salah mentioned presence order. The operation that has the highest order should be the end of the *BNF* rules. So I decided to put parentheses handling end of my recursion. I put addition, subtraction, multiplication, division and parentheses in order that.

My Code

I didn't write an object oriented code. I wrote all of my codes in one file imperatively. My file 4 main section: *Global variables, Method declarations, Method implementations and Main method*. Also, my methods have 4 part: *Handling operations, Variable helpers, Error helpers, General helpers.* Detailed explanations of my methods are written in my code via comments.

I will briefly mention my architecture of my algorithm. My *main(,)* manages a general flow of my code. First of all, it reads input file line by line. After that *checkErrors(string)* methods checks basic syntax errors line by line. Such that, double equal signs, unmatched parenthesis. After basic error checks, it parses file line by line via *parse(line)* method. If the line has "=", *parse(line)* split it and store left side, then evaluate right part; else, evaluate line directly. Evaluating happens recursively via handle methods. First of all, I split line by “+” that not in parenthesis, and call again *handleAdd(string)* with two parts. When all “+” are parted, checks for “-”. Then it checks “”, “/”. When a string doesn’t have operator signs, it tries to handle parenthesis. It removes outer parenthesis and sends the inner part to *handleAdd(string)* . When all evaluations were done, *handleVariable(string)* methods decides is it variable or just a number. When it decides, return temp variable or variable. It prints to file in recursion. If it encounter an error, it print error description to console and abort application via *abortError(message)* method.

```
 void parse(string exp);

/* HANDLING OPERATIONS */
string handleAdd(string exp);
string handleSub(string exp);
string handleMul(string exp);
string handleDiv(string exp);
string handlePar(string exp);
string handleVariable(string exp);

/* VARIABLE HELPERS */
string allocateVarible(string var);
bool isValidVariable(string var);
bool isAllocatedVariable(string var);

/* ERROR HELPERS */
bool checkErrors(string line);
string abortError(string desc);

/* GENERAL HELPERS */
string trim(string str);
string left_trim(string str);
string right_trim(string str);
bool isNum(string str);
void checkEmptyStr(string str);
```