CmpE 230 System Programming Spring 2023
Project 1 - Adv Calc

Ahmet Ertuğrul Hacıoğlu – Ceyhun Sonyürek

2020400132 - 2020400258

Introduction

In this project, an advanced calculator is implemented in C programming language. The purpose of the project is generating a calculator that can compute complicated expressions which may include many functions and nested operations and store many variables assigned to use them in calculations.

Approach and Solution

1- Taking an input from command.

2- Checking whether this is an assignment or an expression.

3- Iterating over input and creating a node for every meaningful token.

4- Checking if the lexeme has an invalid expression and printing “Error!” if detected.

5- Applying necessary operations and functions in prder of precedence and removing handled nodes in linked list and adding a single handled node instead of removed ones.

6- Returning last node after shortened linked list.

7- Assigning the value to variable if there is a ‘=’, if not, printing the result of expression.

Implementation Summary

- Libraries

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <ctype.h>
```

Libraries used

- Structs

1-) Node

A struct which has 5 members to store every token in a lexeme in a linked list format for every lexeme in inputs. “data” member stores the smallest token which has a meaning in lexemes. “next” and “prev” members helps accessing elements in appropriate order. “func” and “var” members are used for checking the token contains alphabetic character if it is a function or a variable.

```c
struct Node{
    char data[256+1];
    struct Node* next;
    struct Node* prev;
    bool func ;
    bool var;
};

struct Variable{
    char data[24];
    struct Variable* next;
    struct Variable* prev;
    long long int val;
};
```

2-) Variable

A struct which has 4 members to store every variable name and its value with “data” and “val” members. “next” and “prev” members again helps reaching the variable while iterating in a linked list format storing.

- Functions

1-) void operations(struct Node** head)

Function performs mathematical operations inside the innermost paranthesis ofthe nested parentheses.

2-) void paren(struct Node** head)

Aim is that find the number of paranthesis and iteratively do operations in paranthesis starting from the innermost one.

3-) void inside_func(struct Node** head,bool flag)

In this function, we do all operations and removes paranthesis before or after comma in function. There can be functions in functions so we need to get innermost function and iteratively to uppermost function.

4-) long long int terminate(struct Node** head)

The function that traverse all linkedlist and do all requirements with using other functions. The reason for changing the data of the node before the operation node is because that I want to delete the other node (* and the next node) and only the result of the operation remains. The node's data is number,it can no longer be func so it is important to change it is not func.

5-) main()

Scanning input, detecting error and printing if input is expression or assigning value variable if it is a expression.