Interpreter project report

CC273 Data structure I

Alaa Kamal Ali Hassan : 6340
Abdelrahman Gaber Zayed : 6591

Interpreter project report

Table of Contents

System description

This is a simple interpreter that solves equations by converting them to postfix notation and substitutes the result of equations into each other if needed by the help of binary search tree data structure.

Flow of the code

- Checks if the user chose a file to interpret
- Parses the file into lines
- Checks that every line is in the right syntax
- Parses each line into variable/expression pair
- Makes sure that the variable naming is correct and there's one variable on the left hand side
- Evaluating each expression by converting it to postfix notation and evaluating it using a stack
- pushing a key/value pair of the result into a binary search tree that resembles the context of the current process
- In case of having a variable in the expression, it checks if it was pushed to the BST
- If the variable exists, it substitutes its value into the expression before evaluating it
- After evaluating all lines, the system shows the variables sorted by variable name using in-order traversal
- A function that converts the BST into an array is called
- The generated array is then transformed into a max-heap
- The system shows the variables sorted by value using heap sort

Files of the project

- main.c : that file contains the main function and the file parsing utilities
- evaluate.c : that file contains infix to postfix conversion and evaluation
- BST.c : that file contains the creation, handling and sorting of BST nodes
- check.c : that file contains the checks applied on user supplied input
- heap.c : that file contains the creation, handling and sorting of heap elements

Global variables

- context : that is the binary search tree that holds the key/value pairs of the current context
- contextArray : that is the array form of the 'context' variable
- peak : An integer that holds the number of variables pushed to the context BST
- base : An integer that shows how many nodes have been converted from the BST to the array

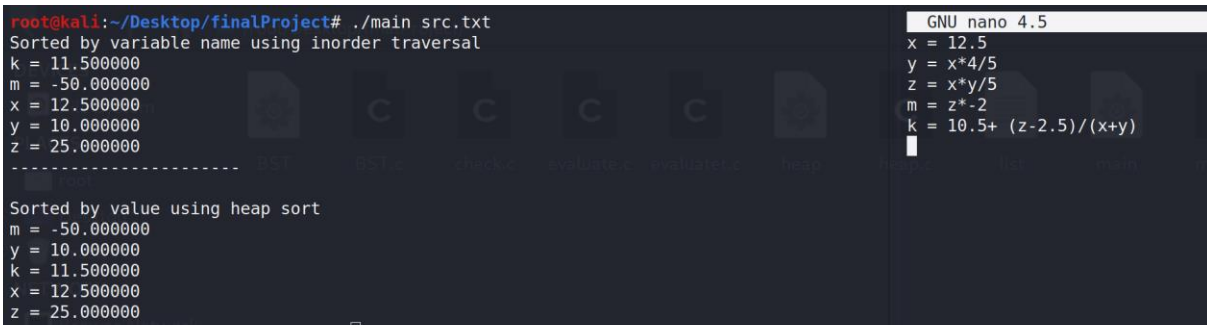
User manual

Compiling the project

This project was compiled using GCC on Debian based Linux-64bit distribution using the following command : $gcc -o main main.c -lm -w

Steps to use

1- first line of the file you want to interpret should contain a direct assignment statement since there's no variables used yet
 example : abc = 55.4
2- rest of the lines can include previously used variables in the expression they want to evaluate
 example : xyz = abc + 34 / 2.7
3- run the application followed by the path of the file you want to interpret
 example : $./main src.txt



```
root@kali:~/Desktop/finalProject# ./main src.txt
Sorted by variable name using inorder traversal
k = 11.500000
m = -50.000000
x = 12.500000
y = 10.000000
z = 25.000000
----------------------

Sorted by value using heap sort
m = -50.000000
y = 10.000000
k = 11.500000
x = 12.500000
z = 25.000000
```

```
  GNU nano 4.5
x = 12.5
y = x*4/5
z = x*y/5
m = z*-2
k = 10.5+ (z-2.5)/(x+y)
```

Struggles we faced

Following the syntax of put(k,v) in the project description

Adding nodes to the BST without providing the tree root as argument to the function led us to the idea of having one global BST as the current context as this is a simple interpreter and there's no need for sandboxing multiple contexts.

Substitution of previously used variables

As C programming language doesn't have a built in function for replacing a specific token in a string, we had to get the number of digits of the value each variable and and calculate a number of bytes that perfectly fits for each expression after substitution.