Mehmet Süzer – 2019400213
Harun Reşid Ergen - 2019400141

CMPE230 Project 1

Libraries:

<stdio.h>, <string.h>, <stdlib.h>

File Reading:

- Space is allocated for lines of the file.
- File is read line by line.
- Every token is seperated by a single whitespace.
- Comments are removed.
- Space is allocated for lines of output file.

Line Processing:

- First word is taken for every line and statement type is determined for this line. (print, for, matrix, etc.)

Declaration:

The type of the variable name is determined by the first word which can be ‘Scalar’, ‘Vector’, ‘Matrix’. Variables are stored in type arrays with their names. Row and column numbers are stored if the variable is matrix or vector. Every vector is interpreted as matrix with column number 1. Redeclaration is checked for every variable and error is given if the variable is redeclared.

For:

For statement is checked whether the loop is nested or not. Considering the type of the for loop, parameters and expressions are taken. If necessary tokens (in, :, }, {, etc.) are missing, error is given.

Assignment:

If a statement is not a for statement, declaration statement, or print operation, it must be an assignment statement. Since the first token must be a variable name in an assignment statement, its decleration and type are checked. The right-hand side of the statement is interpreted as an expression.

- At the end, unnecessary characters are detected. In such cases, error is given.

Output File:

- If there is no error until the end of the line processing, output file is ready to be generated.
- Necessary libraries, functions, and definitions which are used for arithmetic operations are present in a text file called content.txt.
- Content of the content.txt is copied to file.c in order to do necessary operations.
- Generated output lines during the line processing are printed into file.c.
- Allocated space for line processing is freed.

Important Functions in matlang2c.c:

separate:

Every token is separated by a single whitespace in a line.

valid:

Checks whether a variable name is valid or not.

adjust_space:

Reduces multiple whitespaces into a single whitespace.

strstrip:

Removes unnecessary characters from the beginning and the end of the string.

Expressions:

BNF notations:

<expr> := <term> **|** <expr> - <term> **|** <expr> + <term>

<term> := <term> * <factor> **|** <factor>

<factor> := <matrix> **|** <scalar> **|** <matrix>[<expr>] **|** <matrix>[<expr>, <expr>] **|** choose(<expr>, <expr>, <expr>, <expr>) **|** tr(<expr>) **|** sqrt(<expr>) **|** (<expr>) **|** double

- Expressions written in infix notation are converted to prefix notation.
- Prefix notation is converted to functional notation by using a stack.
- Matrix indexing is considered as a function called "value()" such that it returns the value at a given position.

Infix notation:

j = sqrt(tr(A[3,2] + i * (count – 1))) ------------- expr(), term(), factor() ----------------->

Prefix notation:

j = sqrt tr + [] A var(3) var(2) * i – count var(1) -------------convert() -------------------->

Functional notation:

j = square_root(tr(add(value(A,var(3),var(2)),multp(i,sub(count,var(1))))));

- Compatible function checks whether an expression written in prefix notation is valid or not by using stacks. For example, type of operands, row and column compatibility in operations are checked.
- In order to deal with operation overloading, all scalars, vectors, and matrices in file.mat are declared in file.c as structure Variable.

```
typedef struct {
        int type; // if type == 0, then it is a scalar, if type is 1, then it is a matrix
        union {
                double scalar;
                Matrix matrix;
        } value;
} Variable;
```

- "var" function converts a double to a structure Variable.

!!! Notes !!!

The code gives “Segmentation fault (core dumped)” in Ubuntu Virtual Machine, even though it runs well as it runs on Windows 10 where the project is developed. “file.c” file is still created.