This project translates .mat files to c files. There are 9 .c files in the project. These are

1. main.c
2. parsingFunctions.c
3. printFunctions.c
4. declarationParser.c
5. statementFunctions.c
6. forParser.c
7. expressionParser.c
8. paranthesisParser.c
9. chooseParser.c

main.c

The entrance file of the program is main.c. It basically reads input from file and sends formatted line to related function.

Opens the .mat file and reads the file line by line. Firstly, it adds white space before and after all chars that are not alphanumeric or point. Then line is ready for traversing token by token.

Ex: “n=10” → “n = 10”

For declaration lines, if variable name is valid ,this extended line is sent to the related declaration parser function. If declaration statement starts with scalar it sends to scalar_line function. Same goes for other types.

For statement lines, If line starts with a keyword (print, printsep, for)it sends to related function. Else it is considered as assignment statement and is sent to assignment_statement function.

parsingFunctions.c

This file contains helper functions for parsing strings. It has trim function for splitting. It has is_alpha_numeric functions that returns 1 if the string is alpha_numeric, otherwise returns 0. is_valid_variable_name funciton hat returns 1 if the string is valid variable name, otherwise returns 0. It has strrev funciton for reversing string.

It has is_scalar and is_matrix functions for checking expressions’ return type. It has return_type_function which returns the type of the functions. It has function first_size and second_size functions which returns sizes of matrix and vectors. For vectors second size is 1.It has also isDeclared functions. If the string is declared variable, returns type. Else returns empty.

This file also has exit_program function which terminates the program and prints the error statement to console.

printFunctions.c

In this project there are some template functions that will be printed to translated c file. These functions is written in string format and are always printed to translated c file. printFuncitons.c file stores these strings and prints to translated c file.

declarationParser.c

This file contains functions related to declarations. If the line is a declaration line main function sends line to a declarationParser function. It has three functions scalar_line,vector_line and matrix_line. scalar_line parse scalar functions and checks if it is valid declaration, if so it prints declaration to the translated c file. Also, adds variable name to scalar array. This array used in

statement functions to check if a variable is declared or not. Same goes for vector_line and matrix_line.

statementFunctions.c

This file contains functions related to statements. If the line is a statement line main function sends line to a statementFunction function. It has four functions.

First one is assignment_statement function and it is entrance function, that is it is only function that is called in another function. assignment_statement function is called in main.c. This function checks if there is an "=" sign after the variable name if there is not, statement should be variable [scalar] or variable [scalar][scalar]. For that kind of statements assignment_statement function sends statements to assign_value_specified_index function. If there is "=" sign and after that there is "{" sign then assignment_statement function sends remaining part to matrix_initializer function. If there is no such sign ssignment_statement function sends remaining part to parseParanthesis function which is a function in paranthesisParser.c. Then assign returning value to variable, if there is no type compatibility problem and there is no boundary problem for matrixes and vectors. If there is gives error

Second one is assign_value_specified_index function. It takes a string like that matrix [scalar][scalar] = <expression> and sends <expression> part to parseParanthesis function and assigns return value to variable.

Third one is matrix_initializer function. It takes a statement like that <variable_name> = { scalar scalar …}. It parses and sends scalars to parseParanthesis function and assigns return value that index.

Fourth one is print_line function. It is called only in main.c, if there is print keyword at the beginning of the statement. print_line takes statement in that form print(<expression>) and sends expression to parseParanthesis. If return value is scalar, then it prints printScalar to translated c file. If it is matrix or vector, then it prints printMatrix to translated c file.

forParser

This file contains functions related to for parsing. If the line is a for initializing line main function sends line to a forParser function. It has two functions, parseSingleFor and parseDoubleFor. ParseSinglefor parses single for situations and the other one parses double for situations.

expressionParser.c

This file contains functions related to parsing expressions. This file contains six functions.

First one is expression_parser it is entrance function of the file and it is called in parseParanthesis function. It takes line and sends to expression_divider function. Then it returns operator type and divided expressions first and second. If operator type is summation it sends to summation function, if substraction it sends to substraction function, if multiplication sends to multipliciation function. If operator is none of them, it continous in the expression_parser function. Then checks statement is a special function(tr,sqrt,choose), if so parses and return related function. For example, tr (A) returns matrixTranspose(A). If it is not a special function, it can be expression in the form A[][] and checks if it is in that form. If so, returns getValue(A,scalar,scalar). If it is not in that form again, it checks it is a variable or numeric. If so, returns that value else gives error

Second one is expression_divider. It takes line as a parameter and divides it into to part if there is "+","-","*" . It follows precedence rules and it is left recursive. It returns type of the operant.

Ex : 4 + 7 * 9 + 7 → 4 + 7 * 9 and 7

Third, fourth, and fifth ones (summation,substraction,multiplication) are almost same they takes divided line as parameters and makes the operation.

Last one is is_special_function. It takes token as parameter and returns type of special funciton.

paranthesisParser.c

This file contains functions related to parsing functions. This file contains entrance function for expression. It has one function parseParanthesis. ParseParanthesis function detects the paranthesis according to precedence rule and sends inner part to expression_parser funciton recursively. Then, emplaces the returning value.

Ex: (7+(tr(3)) first it sends 3 to expression_parser, then tr(3) and moves on.

chooseParser.c

This file contains a function for parsing choose function. It only has a function. It takes string as a parameter and parses it.

Ex : choose(<expression>,<expression>,<expression>,<expression>) then parses and sends expressions to parseParanthesis function. Then emplaces the returning value. Then it sends whole expression to parseParanthesis function.