

This project translates .mat files to c files. There are 9 c files one header file 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
10. main.h

### **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 function that returns 1 if the string is valid variable name, otherwise returns 0. It has strev function 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 assignment\_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.c**

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 subtraction it sends to subtraction function, if multiplication sends to multiplication function. If operator is none of them, it continues 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 two parts if there is “+”, “-”, “\*” . It follows precedence rules and it is left recursive. It returns type of the operand.

Ex :  $4 + 7 * 9 + 7 \rightarrow 4 + 7 * 9$  and 7

Third, fourth, and fifth ones (summation, subtraction, 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 function.

### **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` function recursively. Then, replaces 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 replaces the returning value. Then it sends whole expression to `parseParanthesis` function.

### **main.h**

There are declarations of functions, global variables in this file.

Ex : `char * scalarArray[256]`  
`char * matrixArray[256]`  
`char * vectorArray[256]`  
`extern char* matrixFirstSize(char *mat);`  
`extern char* matrixFirstSize(char *mat);`