Functions

1. **Main()**
2. **Description**

This function is the main driver. Opens the script file and executes it.

1. **Flow Chart**

Mark all variables as empty

File successfully opened? opened

No

Exit with an error

Yes

Get the first character in the file. Get the next token and begin processing. Close the file

1. **addChar()**
2. **Description**

A function to add the next character to the current lexeme

1. **Flow Chart**

Maximum Lexeme length exceeded by adding next character?

Yes

Exit with an error

No

Add character to the lexeme

1. **lookup()**
2. **Description**

A function to lookup operators or special characters (newline, carriage return, and space) and return the token for these characters.

1. **Flow Chart**

Yes

Add character to the lexeme. Set and return the next token.

Character is ‘:’?

No

Yes

Add character to the lexeme. Set and return the next token.

Character is ‘+’?

No

Yes

Add character to the lexeme. Set and return the next token.

Character is ‘-‘?

No

Character is ‘\*’?

Yes

Add character to the lexeme. Set and return the next token.

No

Yes

Add character to the lexeme. Set and return the next token.

Character is ‘/’?

No

Character is ‘\n’?

Yes

Start a new line. Add character to the lexeme. Set and return the next token.

Yes

Add character to the lexeme. Set and return the next token.

Character is ‘\r’?

No

Add character to the lexeme. Set and return the next token.

Character is ‘ ‘?

Yes

No

Exit with an error

1. **getChar()**
2. **Description**

A function to get the next character of input and determine its character class and marks it as part of the current line if it is not the newline or carriage return.

1. **Flow Chart**

Get the next character

yes

Set the character class to EOF

Next character is EOF?

No

Mark character as part of the current line as long as it is not the carriage return or newline character

Yes

Next character is A, B, C, D or E?

Set the character class to LETTER

No

Yes

Set the character class to DISPLAY

Next character is ‘d’?

No

Next character is a digit?

Set the character class to DIGIT

Yes

No

Set character class to UNKNOWN

1. **lex()**
2. **Description**

A simple lexical analyzer. It moves to the next lexeme. It then adds the current character to the current lexeme.

1. **Flow Chart**

Move to the next lexeme

Yes

Parse the lexeme as an identifier

Character class is LETTER?

No

Character class is

DISPLAY?

Yes

Parse the lexeme as the display keyword

No

Yes

Parse lexeme as either an operator or a special character (an operator or the newline or the carriage return)

Character class is UNKNOWN?

Nop

Yes

Parse the lexeme as a digit

Character class is DIGIT?

No

Set character class to EOF

1. **Script()**
2. **Description**

Parses and executes the strings in the language generated by the rule:

<stmts> -> <stmt> | <stmt><eos> | <stmt><eos><stmts>

1. **Flow Chart**

Parse and execute the script statements

1. **eos()**
2. **Description**

Parses the strings in the language generated by the rule:

<eos> -> <lf> | <cr><lf>

1. **Flow Chart**

Yes

Get the next token

First character is the newline character?

No

Yes

Get the next token

First character is the carriage return

No

Get the next token

Yes

Get the next token

Next character is the newline?

No

Exit with error.

1. **stmt()**
2. **Description**

Parses the strings in the language generated by the rule:

<stmt> -> <var> : <expr> | display<msp><var>

And then executes the statement

1. **Flow Chart**

Get the identifier’s value.

Assignment Statement?

yes

Next character is assignment operator?

Parse and execute the expression. Store the result in the variable identified by the identifier.

No

Yes

No

Exit with error

No

Exit with error

Display statement?

Yes

Get the next token. Parse the multiple spaces between the display keyword and the identifier.

No

Next character is an identifier?

Exit with error

Yes

Get the identifier

No

Corresponding variable has a value?

Exit with error

Yes

Print the value of the variable

1. **expr()**
2. **Description**

Parses the strings in the language generated by the rule:

<expr> -> <term> | <term> + <term> | <term> - <term> | <term> \* <term> | <term> / <term>

Returns the value of the expression represented the string.

1. **Flow Chart**

Parse the first term and retrieve its value

Parse the second term and retrieve its value. Execute the appropriate operation on the first and second term. Return the result.

Expression includes an operator?

Yes

No

Return the first term.

1. **term()**
2. **Description**

Parses the strings in the language generated by the rule:

<term> -> <var> | <const>

Returns the value of the term represented by the string.

1. **Flow Chart**

Return the value of the identifier

Term is an identifier?

Return the integer represented by the term

Term is an integer?

Exit with an error

1. **var()**
2. **Description**

Parses the strings in the language generated by the rule:

<var> -> A | B | C | D | E

Returns the identifier referenced by the string

1. **Flow Chart**

Return the identifier

Current character is an identifier?

Yes

No

Exit with error

1. **constant()**
2. **Description**

Parses the strings in the language generated by the rule:

<const> -> <digit> | <digit><const>

Return the integer represented by this string.

1. **Flow Chart**

First character is an integer?

No

Exit with error

Yes

Get all the following digit-representing characters if any. Return the integer represented by all these sequence of digits.

Yes

Next character is an integer?

No

Return the integer represented by the only digit-representing character.

1. **msp()**
2. **Description**

Parses the strings in the language generated by the rule:

<msp> -> <sp> | <sp><msp>

1. **Flow Chart**

No

Exit with error

First character is a space

Yes

No

Next character is space?

Return

Yes

Get the next token

1. **digit()**
2. **Description**

Parses the strings in the language generated by the rule:

<digit> -> 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

1. **Flow Chart**

Yes

Return the digit represented.

Character is a digit?

No

Exit with an error

1. **error()**
2. **Description**

Exits the program with an error message specified by the parameter.

1. **Flow Chart**

At a new line?

Yes

Print error message as well as the previous line.

Close the file. Exit program.

No

Get the remaining part of the current line.

Print the error message as well as the current line. Exit program.

1. **hash()**
2. **Description**

Returns the value of an identifier which is used to index the value of the variable it references.

1. **Flow Chart**

Returns the value of a identifier which is used to index the value of the variable it references.

1. **addToLine()**
2. **Description**

Adds a character to the current line. Exits with an error if the current line will be exceed its maximum length (maximum length is 99. The last cell is reserved for the null character).

1. **Flow Chart**

Adding new character makes line length exceed its maximum?

Exit with error

Yes

No

Add character to the line.

1. **startNewLine()**
2. **Description**

Marks the current line as the previous line and then moves to the next line.

1. **Flow Chart**

Move to the next line

Mark the current line as the previous line