**Programming Languages Project**: TextJedi Lexical Analyzer and Interpreter

TextJedi Language is a programming language that was designed for editing text files mainly by cutting and pasting operations. (it is imaginary)

**Data Types:** The only three data types in TextJedi are Strings (called as text), Chars and Integers. String constants are written “like this” and integer constants are natural (zero or positive whole) numbers.

All variables should be declared before they are used. Declarations are done with the “new” keyword.

**Ex:** new text myText; new int myInt; new char myChar; /\* This is a comment\*/

/\* And two lines of code on one line is OK \*/

**Assignment, Operations and Functions:**

“:=” is the assignment operator. On the left hand side there has to be a variable name and on the right hand side there may be either a value, an arithmetic expression or a function call. A value is either a constant or a variable. An arithmetic expression is composed of two operands and an operator written in infix form. Valid operators are “+” and “-”.

**Ex:**

MyText := “Hello world”;

myInt:=0;

myChar := ‘X’;

myInt := 2 + yourInt;

MyText:= FirstText+SecondText;

* Subtracting a larger value from a smaller value generates an error message. Other than that Addition and subtraction on integers works just as expected.
* Addition over text values results in concatenation of the two texts

Ex: s := “Hello”+ “ world”; /\*Assigns “Hello world” to s\*/

* Subtraction over text values results in the removal of the first occurrence of the Subtrahend from the Minuend (finds the substring and cuts and removes it)

**Ex:** myString := “Hello world”-“wo”; /\* Assigns “Hello rld” to myString\*/

If the substring cannot be found, it is not an error condition. However, if the second string is bigger than the first one, an error message is generated.

* A function call is a call made to one of the predefined (built in) functions of the language. The predefined functions are as follows:

size(myText) : returns the size of myText as an int

subs(myText, begin, end) : returns substring of myText between characters marked by the two integers begin and end

locate(bigText, smallText, start): Finds (the first occurrence of) smallText in bigtext and returns its position as an integer. It returns 0 if it fails to find it. Starts its search from the location marked by the integer “start”.

insert(myText, location, insertText) : inserts insertText into the position marked by the integer “location” of myText and returns the resulting text.

override(myText,location, ovrText): writes ovrText onto myText by overriding whatever that was previously there, starting from the location (given as an integer) If the writing operation exceeds the size of myText, it terminates at the end of myText without creating any error condition.

asString(myInt): returns myInt as a string (type conversion)

asText(myString): returns myString as an int

replaceChar(myText,oldChar,newChar): replaces all occurrences of oldChar in myText with newChar (character) and returns the result as a string value (so that it is assigned to newString)

**Commands:**

read myString from myTextFile; /\* Reads a string from a text file called myTextFile.txt Please note that there is no size limit\*/

write myText to yourTextFile; /\* Write the string called myText onto a text file named as yourTextFile \*/

input myText prompt promptText; /\* Receive input from keyboard into myText. Use promptText as a prompt. Max 100 characters buffer is enough\*/

output myText; /\* Write myText on the screen\*/

***Lexical rules for the programming language TextJedi are as follows:***

1- Identifiers:

* Maximum identifier size is 30 characters. If you use an identifier larger than that, the lexical analyzer issues an error message.
* TextJedi language is case sensitive, and all keywords are lower case.
* Identifiers start with an alphabetic character (any letter) and are composed of one or more letters/digits/\_ (underscore)

2- Integer constants

* Maximum integer size is the same as a C unsigned int.
* Negative values are not supported. All integers are zero or positive.

3- Operators

* Valid operators of the language are +,-,:=

4- String constants

* String constants of TextJedi are delimited by double quotes (ASCII code 34)as in “this is a string”
* String constants have unlimited size
* String constants cannot contain the double quote character. when you reach one, the string terminates.
* If a string constant cannot terminate before the file end, there should be a lexical error issued.

6- Keywords:

* Keywords are: new, int, text, size, subs, locate, insert, override, read, write, from, to, input, output, asText, asString

7- End of line mark: ;

8- Comments: Anything between /\* and \*/ is a comment.

* If a comment fails to terminate before the file end, there should be a lexical error issued.

**------------------------------------------------------------------------**

**Project Definition:** This project is an interpreter project for theTextJedi language. Interpreter executable (TextJedi.exe) runs a TextJedi program when called with an argument that contains the filename of the code to be run. (from the operating system command prompt)

Example: TextJedi myProg.tj (.tj is just a .txt file)