Doc

There is a single “Scanner” function that should be used

This function receives a filename for the file to be scanned, a regex pattern with the separators and operators of the language and a dictionary with the reserved words.

It returns the ST and PIF if the program is correct, or nothing if the program has an error

Also, it prints all errors found (as possible, may not do that well since it keeps constructing the program from an erroneous form).

Updated specs:

Alphabet:

1. Upper case and lower case letters of the english alphabet
2. Underline
3. Decimal digits
4. other symbols: + - / * = < > ( ) { } [ ] ; . : , " ' ! | &

Lexic:

a. Special symbols:

1. operators: + - * / = <= < == != > >= // && || ++ -- %
2. separators: [ ] { } ( ) : ; ,
3. reserved words: main bool number char string if else for while read write const

b. identifiers

1. identifier ::= letter{letter|digit};
2. letter ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
3. digit ::= "0" | "1" | ... | "9"

c. constants

1. number rule :

    number ::= ["+"|"-"]non_zero_number | "0"

    non_zero_number ::= non_zero_digit{digit}

    non_zero_digit ::= "1" | "2" | "3" | ... | "9"

2. char rule:

    char ::= '<any valid character>'

    character ::= any valid character (UTF8)

3. string rule:

    string ::= "charseq"

    charseq ::= {character}

4. bool rule:

    bool ::= "true" | "false"

IDENTIFIER ::= identifier

CONSTANT ::= number | char | string | bool

positive_number ::= [+]non_zero_number

Syntax:

The words - predefined tokens are specified between " and ":

function ::= type IDENTIFIER "(" decllist ")" cmpdstmt

program ::= int "main" "(" decllist ")" cmpdstmt

decllist ::= single_declaration {"," single_declaration}

single_declaration ::= type IDENTIFIER

chained_declaration ::= type identifier_list

identifier_list ::= IDENTIFIER {"," IDENTIFIER}

simple_type ::= "bool" | "char" | "number" | "string"

array_type ::= simple_type "[" [positive_number] "]"

type ::= simple_type | array_type

cmpdstmt ::= "{" stmtlist "}"

stmtlist : stmt {stmt}

stmt ::= simplstmt ";" | structstmt

simplstmt ::= assignstmt | iostmt | chained_declaration | func_call

func_call ::= IDENTIFIER "(" identifier_list ")"

assignstmt ::= IDENTIFIER "=" expression

expression ::= expression ("+" | "-") term | term

term ::= term ("*" | "/" | "//" | "%") factor | factor

factor ::= "(" expression ")" | IDENTIFIER["[" positive_number "]"] | CONSTANT | func_call

iostmt ::= ("read" "(" IDENTIFIER ")" | "write" "(" (IDENTIFIER | CONSTANT | expression) ")")

structstmt ::= cmpdstmt | ifstmt | whilestmt | forstmt

ifstmt ::= "if" condition (cmpdstmt | stmt) ["else" (cmpdstmt | stmt)]

whilestmt ::= "while" condition (cmpdstmt | stmt)

forstmt ::= "for" "(" assign_list ";" condition ";" assign_list ")" (cmpdstmt | stmt)

assign_list ::= assignstmt | assignstmt "," assign_list

condition ::= expression RELATION expression

RELATION ::= "<" | "<=" | "==" | "!=" | ">=" | ">" | "&&" | "||"