

Proyecto No.2

Descripción

El proyecto consiste en la implementación de un generador de analizadores léxicos, tomando como base un subconjunto de las características de COCO_r. El programa aceptará como entrada un archivo en COCO_l, con la especificación del analizador léxico a generar, y dará como salida un archivo fuente en cualquier lenguaje, el cual implementará el scanner basado en la gramática ingresada. Este archivo será compilado y ejecutado para determinar su validez.

Objetivos

- **Generales**
 - Implementar un generador de analizadores léxicos basado en un subconjunto de especificaciones de COCO_r
- **Específicos**
 - Aplicar la teoría de analizadores léxicos en la construcción de un generador de analizadores léxicos.
 - Aprender las características y usos principales de COCO_r

Especificación

- **Entrada**

Un archivo que contiene la especificación del analizador léxico a generar. Esta especificación se encuentra escrita en un subconjunto de COCO_l.

- **Salida**

Programa fuente que implemente un analizador léxico basado en la especificación ingresada. El lenguaje del programa fuente queda a elección del estudiante.

Ponderación

El proyecto en total tiene un valor de **25 puntos** netos.

Vocabulario de Cocol

```
ident = letter {letter | digit}.
number = digit {digit}.
string = '"' {anyButQuote} '"'.
char = '\'' anyButApostrophe '\'
```

Sintaxis de Cocol

```
Cocol = "COMPILER" ident
      ScannerSpecification
      ParserSpecification
"END" ident '.'
```

```
ScannerSpecification =
  ["CHARACTERS" {SetDecl}]
  ["KEYWORDS" {KeywordDecl}]
  ["TOKENS" {TokenDecl}]
  {WhiteSpaceDecl}.
```

```
SetDecl      = ident '=' Set.
Set           = BasicSet { ('+' | '-') BasicSet }.
BasicSet     = string | ident | Char [".." Char].
Char         = char | "CHR" '(' number ')'.
KeywordDecl  = ident '=' string '.'.
TokenDecl    = ident ['=' TokenExpr ] ["EXCEPT KEYWORDS"] '.'.
TokenExpr    = TokenTerm { '|' TokenTerm }.
TokenTerm    = TokenFactor {TokenFactor}.
TokenFactor  = Symbol
              | '(' TokenExpr ')'
              | '[' TokenExpr ']'
              | '{' TokenExpr '}'.
Symbol       = ident | string | char
WhiteSpaceDecl = "IGNORE" Set
```

```
ParserSpecification = "PRODUCTIONS" {Production}.
Production          = ident [Attributes] [SemAction] '=' Expression '.'.
Expression          = Term { '|' Term }.
Term                = Factor {Factor}.
Factor              = Symbol [Attributes]
                    | '(' Expression ')'
                    | '[' Expression ']'
                    | '{' Expression '}'
                    | SemAction.
Attributes          = "<." {ANY} ">".
SemAction           = "(." {ANY} ".)"
```

Ejemplo de archivo de entrada

COMPILER Ejemplo

```
(.  
/*-----Scanner Specification-----*/  
.)
```

CHARACTERS

```
letter = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ".  
digit = "0123456789".  
hexdigit = digit+"ABCDEF".
```

KEYWORDS

```
if="if".  
while="while".
```

TOKENS

```
id = letter{letter} EXCEPT KEYWORDS.  
number = digit{digit}.  
hexnumber = hexdigit{hexdigit}"(H)".
```

PRODUCTIONS

END Ejemplo