Proyecto final

Lenguajes de programación

Licenciatura en Ciencias de la Computación

Fecha de entrega: 27 de Noviembre a las 23:59.$\bar{9}$

Presentación: 28, 29 y 30 de Noviembre

Crea un directorio llamado `proyecto` en tu repositorio de Git, dentro de este directorio coloca los archivos relevantes para tu entrega.

Implementa un analizador léxico, un analizador sintáctico, un intérprete y una interfaz textual o gráfica, para el lenguaje LETREC.

Puedes utilizar el lenguaje de implementación que prefieras, así como bibliotecas estándar o de terceros. Deberás incluír en tu entrega instrucciones para correr tu intérprete y este debe poder evaluar una expresión leída del teclado o desde un archivo.

- El analizador léxico debe tomar de entrada un flujo lineal de caracteres y regresar un flujo lineal de *tokens*, estos son estructuras que permiten identificar los símbolos terminales de la sintaxis concreta, así como las categorías sintácticas *Number* y *Identifier*.
- El analizador sináctico debe tomar de entrada un flujo lineal de tokens y regresar una expresión del lenguaje de acuerdo a su sintaxis concreta. Esta expresión debe ser una estructura de acuerdo a la sintaxis abstracta.
- El intérprete debe tomar de entrada una expresión y regresar como resultado un valor expresado.
- La interfaz textual o gráfica es la fachada de tu proyecto, le debe permitir al usuario escribir expresiones del lenguaje LETREC para ser interpretadas, así como abrir un archivo con una expresión del lenguaje LETREC para también ser interpretada.

Además de la entrega por medio de tu repositorio de GIT deberás preparar una presentación y demostración de tu proyecto para los últimos tres días del curso.

Sintaxis concreta

Expression ::= *Number*
Expression ::= **-**(*Expression* , *Expression*)
Expression ::= **zero?**(*Expression*)
Expression ::= **if** *Expression* **then** *Expression* **else** *Expression*
Expression ::= *Identifier*
Expression ::= **let** *Identifier* = *Expression* **in** *Expression*
Expression ::= **proc** (*Identifier*) *Expression*
Expression ::= (*Expression Expression*)
Expression ::= **letrec** *Identifier* (*Identifier*) = *Expression* **in** *Expression*

Sintaxis abstracta

(const-exp *num*)
(diff-exp *exp1 exp2*)
(zero?-exp *exp1*)
(if-exp *exp1 exp2 exp3*)
(var-exp *var*)
(let-exp *var exp1 body*)
(proc-exp *var body*)
(call-exp *op-exp arg-exp*)
(letrec-exp *p-name b-var p-body letrec-body*)

Semántica

(value-of (const-exp n) ρ) = (num-val n)

(value-of (var-exp *var*) ρ) = ρ(*var*)

(value-of (diff-exp *exp1 exp2*) ρ)
  = (num-val (- (expval→num (value-of *exp1* ρ))
                (expval→num (value-of *exp2* ρ))))

(value-of (zero?-exp *exp1*) ρ)
  = (**let** ([*val1* (value-of *exp1* ρ)])
      (bool-val (= 0 (expval→num *val1*))))

(value-of (if-exp *exp1 exp2 exp3*) ρ)
  = (**if** (expval→bool (value-of *exp1* ρ)) (value-of *exp2* ρ) (value-of *exp3* ρ))

(value-of (let-exp *var exp1 body*) ρ)
  = (**let** ([*val1* (value-of *exp1* ρ)])
      (value-of *body* [*var* = *val1*]ρ))

(value-of (proc-exp *var body*) ρ) = (proc-val (procedure *var body* ρ))

(value-of (call-exp *op-exp arg-exp*) ρ)
  = (**let** ([*proc* (expval→proc (value-of *op-exp* ρ))]
          [*arg* (value-of *arg-exp* ρ)])
      (apply-procedure *proc arg*))

donde:
  (apply-procedure (procedure *var body* ρ) *val*)
    =(value-of *body* [*var* = *val*]ρ)

(value-of (letrec-exp *p-name b-var p-body letrec-body*) ρ)
  = (value-of *letrec-body* [*p-name* = *b-var* $\mapsto$ *p-body*]ρ)

donde:
  Si ρ_1 = [*p-name* = *b-var* $\mapsto$ *p-body*]ρ, entonces
    (apply-env ρ_1 *var*) =
      (proc-val (procedure *b-var p-body* ρ_1))
  si *var* = *p-name*, y
    (apply-env ρ_1 *var*) = (apply-env ρ *var*)
  si *var* $\neq$ *p-name*.