Emilio Hernández López A01336418
Samantha Barco Mejía A01196844
Diseño de Compiladores
Edgar Manoatl

Definición de Proyecto

- Propuesta general
  Para el proyecto final correspondiente a la clase de Diseño de Compiladores nuestro equipo desarrollará un lenguaje de programación funcional usando como referencia el lenguaje existente Racket.

- Análisis Léxico
  - Sentencias de control
    Para nuestro proyecto implementaremos funciones básicas como:
    1. If
    2. Do .. While
    3. While
    4. For
    5. Asignaciones
  - Símbolos
    6. +
    7. -
    8. *
    9. /
    10. ^
    11. =
    12. !=
    13. %
    14. ||
    15. &&
    16. ==
  - Utilidades:
    17. Imprimir
    18. Comentarios
    19. Input
    20. Stringify
    21. Floatify
    22. Intify
    23. Boolfy

- Análisis sintáctico
  En nuestro lenguaje, todas las funciones y sentencias de control deberán ubicarse dentro de paréntesis en conjunto con los parámetros. Ejemplo, asumiendo que existen variables a y b:
  - (if (a==1)

(
(display a)
)
(
(display b)
)
)

- (integer x (2 + 2))
- (float y (2 + (3.1 - 5.7)))

Las sentencias de control se declararán de la siguiente manera:

a. If:
   - (**if** (condition)
     (true-statements)
     (false-statements)
)
b. Do .. While
   - (**do-while** (condition)
     (statements)
)
c. While:
   - (**while** (condition)
     (statements)
)
d. For:
   - (**for** (iterator **in** array)
     (statements)
)
e. Asignaciones: (**integer** name value) (**char** name value) (**array** name value) (**string** name value) (bool name value) INT CHAR FLOAT ARRAY STRING BOOL

Las operaciones aritméticas seguirán el siguiente formato (donde x puede ser una variable o valor):

f. +: (x **+** x)
g. -: (x **-** x)
h. *: (x ***** x)
i. /: (x **/** x)
j. ^: (x **^** x)
k. =: (x **=** x) Sirve para volver a asignar valores a variables
l. !=: (x **!=** x)
m. %: (x **%** x)
n. ||: (x **||** x)
o. &&: (x **&&** x)
p. ==: (x **==** x)

Las utilidades tendrán el siguiente formato

q. Imprimir: (**display** variable)
r. Comentarios: **[**comments inside square brackets**]**

s. Input: (**input** variable)
t. Stringify: (**stringify** variable)
u. Floatify: (**floatify** variable)
v. Intify: (**intify** variable)
w. Boolfy: (**boolfy** variable)

El usuario también podrá crear sus propias funciones y llamarlas de manera recursiva:

(**function** (name received-values)
(statements)
(function-name received-values)
)

- Validación de tipado
  - Tipos:
    - Entero: números 0-9
    - Flotante: números con punto decimal
    - Char: ‘x’ siendo x un caracter cualquiera
    - String: “xyz” siendo xyz un conjunto de caracteres cualquiera
    - Booleano: true/false t/f
  - Cómo proceder al hacer operaciones con diferentes tipos de valores
  - eg: entero + flotante
  - Suma:
    - entero + entero: entero
    - flotante + flotante: flotante
    - entero + flotante: flotante
    - string + * = string (concatenación)
    - cualquier otra combinación: error
  - Resta:
    - entero - entero: entero
    - flotante - flotante: flotante
    - entero - flotante: flotante
    - cualquier otra combinación: error
  - Multiplicación:
    - entero * entero: entero
    - flotante * flotante: flotante
    - entero * flotante: flotante
    - string * entero = string (concatenación del mismo string n veces)
    - cualquier otra combinación: error
  - División
    - entero / entero: flotante
    - flotante / flotante: flotante
    - string / entero: string (substring desde 0 hasta sin incluir el entero)
    - cualquier otra combinación: error
  - Potencia
    - entero ^ entero: entero

    - flotante ^ entero: flotante
    - flotante ^ flotante: flotante
    - cualquier otra combinación: error
- Módulo
    - entero % entero: entero
    - entero % float: float
    - float % float: float
    - float % entero: float
    - string % entero: char (devuelve el carácter n en el string)
    - cualquier otra combinación: error
- Stringify
    - flotante: string
    - entero: string
    - char: string
    - Bool: string
    - cualquier otra combinación: error
- Floatify
    - flotante: flotante
    - entero: flotante
    - string: flotante (si es válido)
    - cualquier otra combinación: error
- Intify
    - flotante: entero
    - entero: entero
    - char: entero
    - string: entero (si es válido)
    - bool: entero
    - cualquier otra combinación: error
- Boolfy:
    - entero: bool (>0 true)
    - string: bool
    - cualquier otra combinación: error
- Comparación
    - entero == entero :bool
    - string == string: bool
    - float == float: bool
    - bool == bool : bool
    - Cualquier otra combinación: error
- AND
    - bool & bool: bool

- OR
    - bool | bool: bool
- Imprimir:
    - string
    - bool

        - char
        - entero
        - flotante
- Estructuras de control
    - if recibe bool
    - for recibe un arreglo
    - do-while recibe bool
    - while recibe bool