Tarea #3
Lenguajes de Programación INF-253
¡Pero esto no es teoría de compiladores!

Sven von Brand
svbrand@inf.utfsm.cl

Rodolfo Castillo
rodolfo.castillo.13@sansano.usm.cll

Marcelo Jara
marcelo.jara.13@sansano.usm.cl

Segundo semestre académico, 2015

1 Objetivos

- Que el estudiante utilice expresiones regulares
- Que el estudiante aplique el multiparadigma usando el lenguaje de scripting Python

2 Introducción

La res pública de LP ha tomado como decisión integrar la exportación de esclavos en su PIB, para ello debe realizar una prueba de calificación que consiste en la realización de un mini compilador para un subconjunto de un lenguaje antiquísimo que jamás fue implementado.

3 Tarea

El lenguaje "diseñado" para la ocasión cuenta con las siguientes reglas (separadas por funcionalidad y adjuntando ejemplo en cada caso):

- **Tipos de datos**

  El lenguaje se diseñó para poder hacer un pequeño conjunto de operaciones, estas son sobre todo matemáticas, por lo que los tipos de datos son:

  - Enteros (..., -3, -2, -1 0, 1, 2, 3, ...)
  - Booleanos (TRUE, FALSE)

- **Asignaciones**

  Las asignaciones puede escribirse en dos sentidos:

  ```
  VAR(vname) <= 5
  5 => VAR(vname)
  ```

  Ambas son equivalentes y resultan en una variable llamada **vname** conteniendo el valor **5**. Cabe notar que para declarar una variable se usa la llamada **VAR**.

  Las asignaciones pueden tomar valores enteros, booleanos, otras variables y retorno de una llamada a procedimiento.

- **Procedimientos**

  Para declarar procedimientos se debe seguir la siguiente estructura:

  ```
  $^PROC(pname)
      VAR(vname) <= 5
      #vname
  ^$
  ```

  Lo que crea un procedimiento con el nombre **pname**. Dentro de la función se puede usar el común cuerpo de instrucciones, en este ejemplo en particular se asignó el valor **5** a la variable **vname**, y luego se **retorna** la variable **vname**, (cabe notar que no se pueden definir procedimientos dentro de otros procedimientos). Para retornar un valor siempre se usa el signo # (*Se pueden retornar más tipos de valores que serán explicados más adelante.*)

  Los argumentos recibidos son siempre uno y este es una lista de nombre `PARAM`, por lo que para acceder a algun valor entregado por argumento debera llamarlo como `PARAM<I>`, siendo I un índice válido dentro de la lista. Por ejemplo: `PARAM0`, `PARAM1`, etc

- **Llamada a procedimiento**

  Para llamar a un procedimiento se usará la siguiente sintaxis:

```
(pname 0 TRUE vname)
```

En este caso se llama al procedimiento de nombre **pname**, con los argumentos **0**, **TRUE** y la variable **vname**. Cabe destacar que sólo se pueden entregar por argumento: enteros, booleanos, variables. **Por ningún motivo se puede entregar el retorno de otro procedimiento como argumento**, este debe estar almacenado en una variable para poder ser usado como argumento de otro procedimiento.

Adicionalmente, existe el procedimiento vacío, siendo la notación:

```
()
```

Lo que en Python equivale a **pass**.

- **Retorno de un procedimiento**

  Considerando todo lo anterior, el retorno de un procedimiento puede ser: entero, booleano, variable o bien una llamada a procedimiento.

- **Estructuras condicionales**

  La estructura condicional es poco convencional y restringida, no cuenta con un cuerpo de instrucciones, si no que simplemente recibe procedimientos.

  La sintaxis es la siguiente:

  ```
  IFELSE <IF-PROC> <COND> <ELSE-PROC>
  ```

  donde:

  - **IF-PROC** es el procedimiento llamado en caso de que la condición sea verdadera.
  - **ELSE-PROC** es el procedimiento llamado en caso contrario.
  - **COND** es la condición a evaluar. Esta puede ser: booleana, entera, variable o una llamada a procedimiento.

  Dado lo anterior es posible:

  ```
  IFELSE <IF-PROC> TRUE <ELSE-PROC>
  IFELSE <IF-PROC> 1 <ELSE-PROC>
  IFELSE <IF-PROC> vname <ELSE-PROC>
  IFELSE <IF-PROC> (pname 1) <ELSE-PROC>
  ```

  Siendo cada una válida.

  Adicionalmente esta estructura puede ser asignada a una variable, de la forma:

  ```
  IFELSE <IF-PROC> <COND> <ELSE-PROC> => VAR(vname)
  ```

o bien,

```
VAR(vname) <= IFELSE <IF-PROC> <COND> <ELSE-PROC>
```

Lo que asignará el valor retornado por el procedimiento que determina la condición a la variable **vname**.

- **Regla de nombres**:

  Tanto los nombres de variables como los de funciones deben ser únicamente letras ya sean minúsculas o mayúsculas. Ejemplos válidos:

  ```
  HELLOlp
  WHYDOYOUHATEUSSOMUCH
  OHmyGOD
  instaGG
  imdone
  ```

- **Librería estándar**:

  Usted deberá crear una librería estándar para su implementación del lenguaje. Esto implica que deberá crear manualmente un archivo llamado `stdlib.py` que deberá implementar las funciones:

  1. `EQ(A, B)`: Retorna `True` si A y B son iguales, `False` en caso contrario
  2. `NEQ(A, B)`: Opuesto a `EQ`
  3. `LT(A, B)`: Retorna `True` si A es menor que B, `False` en caso contrario
  4. `GT`: Retorna `True` si A es mayor que B, `False` en caso contrario
  5. `LEQ(A, B)`: Retorna `True` si A es menor o igual que B, `False` en caso contrario
  6. `GEQ(A, B)`: Retorna `True` si A es mayor o igual que B, `False` en caso contrario
  7. `ADD(A, B)`: Retorna la suma entre A y B
  8. `SUB(A, B)`: Retorna la resta entre A y B
  9. `INPUT()`: Obtiene input y lo entrega como `int`
  10. `OUTPUT(OUT)`: Escribe `OUT` a la salida estándar

  Pista: Al transformar el input a Python considere siempre insertar la línea:

  ```
  from stdlib import *
  ```

  al inicio del archivo de output.

4 Resumen resumidor

Se le pide a usted y a su grupo que a partir de un archivo de input escrito en el lenguaje creado, se genere un código equivalente en Python. A continuación se presenta un ejemplo de juguete:

```
$^PROC(Dummy)
    #1024
^$

$^PROC(CaseA)
    #42
^$

$^PROC(CaseB)
    #(Dummy)
^$

$^PROC(Default)
    #0
^$

$^PROC(IfCaseB)
    IFELSE (CaseB) (EQ PARAM0 2) (Default) => VAR(Res)
    #Res
^$

$^PROC(Switch)
    VAR(Res) <= IFELSE (CaseA) (EQ PARAM0 1) (IfCaseB PARAM0)
    #Res
^$

VAR(In) <= (INPUT)
(Switch In) => VAR(Res)
VAR(Res) <= (ADD Res 5)
(OUTPUT Res)
```

Lo que se vería traducido como:

```
from stdlib import *

def Dummy(*params):
    return 1024

def CaseA(*params):
    return 42

def CaseB(*params):
    return Dummy()

def Default(*params):
    return 0

def IfCaseB(*params):
    Res = CaseB() if EQ(params[0], 2) else Default()
    return Res

def Switch(*params):
    Res = CaseA() if EQ(params[0], 1) else IfCaseB(params[0])
    return Res

In = INPUT()
Res = Switch(In)
Res = ADD(Res, 5)
OUTPUT(Res)
```

5 Consideraciones adicionales

- Para ayudarlo en esta tarea se levantará una pequeña plataforma web donde podrá escribir código en este lenguaje y se le entregará el resultado traducido a Python. Esté atento a los medios de comunicación dispuestos en el ramo.

  DISCLAIMER: El resultado que de la plataforma es **referencial**, esto es, el presente documento sigue siendo el que manda ante cualquier eventualidad.

- Su programa debe poder ser llamado desde una terminal de la siguiente forma:

  ```
  python tarea4.py <archivo-a-traducir>
  ```

  Y generar como output un archivo de la forma

  ```
  <archivo-a-traducir>.py
  ```

- Usted puede realizar supuestos respecto a su tarea en el archivo README, serán considerados. Recuerde que los supuestos son para simplificar su tarea, **no para evitar hacer alguna parte de esta**.

- La tarea se podrá realizar en grupos de una o dos personas. No necesariamente los mismos grupos formados para la tarea 1 y 2. Por lo que es **mandatorio** que se incluya en el archivo README la identificación de los integrantes (Nombre, rol y usuario del DI)

- No es necesario el uso de IDE para esta tarea, pero si le interesa, se recomienda **PyCharm** (misma línea de productos de IntelliJ IDEA).

6 Evaluación del código

- Uso correcto del lenguaje.

- Al igual que en la tarea de Java, ahora se exige usar la convención PEP8, sin embargo, no es necesario que esté de acuerdo con cada una de las clausulas de ésta, por lo que si alguna no le parece, podrá no usarla siempre y cuando lo explicite en el README y comente, brevemente, por qué decidió no usarla (debe indicar cual fue la cláusula con el nombre o algún ejemplo). (*No se asuste con la longitud de esta guía de estilo, por lo general usará un subconjunto pequeño de esta.*)

7 Sobre la entrega

- La tarea debe entregarse antes de las 23:55 hrs el día **15 de enero del 2015**. Sin embargo habrá una modificación a la regla de plazos que hemos llevado acabo durante el semestre: Su grupo recibirá una **bonificación de 10 puntos** si entrega a más tardar en la fecha oficial de entrega. Por otro lado se aceptarán tareas en los días no hábiles inmediatamente seguidos de la fecha oficial de entrega, siendo estos **plazo fatal**, es decir usted no obtendrá la bonificación de 10 puntos y será calificado con nota 0 si es que entrega fuera de él.

- Para consultas sobre el reglamento de tareas y/o el desarrollo de este enunciado, por favor dirigirse a la sección correspondiente en la plataforma Moodle.

- La entrega será en el siguiente formato:

  `tarea4-user1[-user2].tar.gz`

  donde `user<i>` es su usuario del `DI`.

- Se pide que se explique la estrategia usada en el archivo README, dada la libertad de la tarea, se será especialmente estricto con que explique claramente su algoritmo, sin ser demasiado extenso. Esto requerirá que entienda bien lo que hizo.

- El archivo `.tar.gz` debe contener todos los archivos necesarios para correr su tarea, junto con el archivo README.