Fecha: 13/01/2021 17:27 (GMT-03:00) A: dcorsi@fi.uba.ar Asunto: Consultas sobre Trabajo Práctico de Clojure
Buenos días, profesor.

Disculpe que lo moleste en enero, pero quería consultar dos cosas sobre el TP.

1) Yo el intérprete de Applesoft Basic en Clojure ya lo tengo resuelto, o sea, corre los 10 programas de prueba (aunque todavía quería emprolijar un poco el código) pero en la parte de entrega dice:

El archivo para realizar las pruebas unitarias (como mínimo las correspondientes a los ejemplos del enunciado).

No entiendo a qué se refiere. O sea, tengo un archivo que tiene todos los "loads" de los programas que uso para hacer copy paste rápido a la consola y correrlo, pero nada más. No hice "tests unitarios".

2) En el campus dice:

```
] LOAD NATO.BAS

] LIST

] RUN
```

Qué vendría a ser lo de LIST? Honestamente no me acuerdo si lo hablamos en clase, pero no sabía qué hacer con eso.

Agradezco las respuestas, y nuevamente, disculpe por molestarlo en enero.

Saludos

Hola!

Qué bueno que hayas podido hacer andar el intérprete...

Sobre tus preguntas:

] LIST
muestra el listado del programa (la RI vuelta a representar en BASIC, por lo que queda más prolija que el código original). Hay una función para ello que ya está implementada...

En cuanto a los tests, los mínimos a presentar son los que están basados en los ejemplos del enunciado, más casos no se requieren.

En el campus están los videos de las dos clases donde explico estos puntos:

25/11: LIST (1:47:30 a 1:48:50)

16/12: Testing (24:05 a 35:15)

Saludos!

DCorsi

Fecha: 16/01/2021 11:49 (GMT-03:00) A: dcorsi@fi.uba.ar Asunto: Lenguajes Formales - Consulta TP
Buenos días Diego, después de resolver la función cargar-linea (donde todos los ejemplos tienen 1 sola sentencia por línea) me topé con la función contar-sentencias (más de una sentencia por línea), y me surgió la siguiente duda:
En el caso de cargar-linea, cuando debo actualizar una ya existente con una sentencia, se reemplaza la última sentencia (la que está más a la derecha), o se pisan todas las sentencias existentes con la nueva?
Gracias de antemano, y disculpa las molestias.

Buenas vacaciones,

Saludos,

Hola!
En efecto, me quedé corto con los ejemplos de cargar-linea, podría haber puesto un ejemplo de línea multisentencia...

Al ingresar (o al borrar) una línea, se la trata por completo.

Si existe:
10 print A : gosub 100 : print B

10
la borra completamente.

10 print W
la reemplaza completamente.

Te recomiendo que te inspires en el funcionamiento de alguno de los emuladores de Apple II que hay en línea para entender mejor estas cuestiones relativas a la parte interactiva (el driver-loop) del intérprete.

Saludos!
DCorsi

Fecha: 20/01/2021 18:54 (GMT-03:00) A: Diego Corsi Asunto: Re: Lenguajes Formales - Consulta TP
Buenas tardes Diego, estoy próximo a terminar el tp, pero tengo algunas dudas que no logro obtener respuestas de las referencias dadas:
1) No me queda del todo claro la función continuar-linea (la que implementa la sentencia RETURN), sacando el primer test (el cual es simplemente mirar si **[gosub-return-stack]** está vacío), en el segundo test o en otros, siempre se va a devolver como primer valor de la dupla **:omitir-restante** y el ambiente con los valores **[prog-ptrs] [gosub-return-stack]** actualizados de tal forma que **prog-ptrs** es **gosub-return-stack** con el número de sentencia -1 (llega hasta 0), y **[gosub-return-stack]** queda vacío? Hay algún caso particular? Entiendo que restarle 1 es simplemente seguir con la próxima sentencia de la línea.
2) En la función que le sigue (extraer-data), puede ser que en el segundo test, en la línea 10, haya sido omitido el (DATA 30), y en realidad debería figurar en la salida?
3) Para considerar el tp aprobado, aparte de la resolución, y los tests unitarios, deben correr sin ningún problema todos los programas .BAS proporcionados?

Gracias de antemano, y disculpas por el extenso email.

Saludos,

Qué bueno que estés cerca de terminarlo...
Te respondo:

1) Por lo que contás, lo entendiste bien...

2) REM mata a DATA.

3) Exacto.

Saludos!

DCorsi

Fecha: 06/02/2021 16:46 (GMT-03:00) A: dcorsi@fi.uba.ar Asunto: Trabajo practico Applesoft

Buenas tardes profesor,

Perdone que le escriba sobre una duda del trabajo práctico pero no pude encontrar nada que me ayudara y no conozco a otro compañero a quien consultarle.

Estaba queriendo definir una función la cual determina si el parámetro respeta el formato de variable y devolver true o false en torno a ello.

Mi idea era:

```
(defn es_variable? [x]
(contains? #"[A-Z][A-Z0-9]*[\%\$]?|[A-Z]" x)
)
```

Supongo que la forma de resolverlo es fácil pero ya repase todos los apuntes y vi las clases pero no pude solucionarlo.

Para poner un contexto, es para la función:

```
(defn anular-invalidos [sentencia]    ENUNCIADO
     (map anular-invalidos-aux sentencia)
     )
```

que a su vez :

```
(defn anular-invalidos-aux [x]
(if (or (palabra-reservada? x)  (integer? x) (es_variable? x) )(id x) )
)
```

Saludos,

Hola!

Tu aux está mal pensada...

Invalidaría los números de punto flotante, los operadores relacionales y los aritméticos (pues todos estos no son ni palabras reservadas, ni integers, ni variables).

El análisis léxico ignora espacios, saltos de línea, tabuladores, etc. pero reconoce los símbolos válidos e inválidos que se podían teclear en una Apple II. Los inválidos no son tantos, así que es más práctico considerarlos explicitamente en vez de hacerlo por descarte (viendo, como hacés, que no pertenezcan al conjunto de los válidos).

En cuanto a la función
es_variable?
sería conveniente implementarla como un OR entre las tres funciones que validan si el argumento es una variable entera, de punto flotante o string (estas tres las deben hacer sí o sí).

Para estas tres, una posibilidad es usar re-matches y
some?

Saludos,
DCorsi

Sent: Tuesday, February 09, 2021 at 12:53 AM
To: dcorsi@fi.uba.ar
Subject: Re: Consulta TP

Buenas noches, estoy viendo el TP y en las funciones solicitadas
palabra-reservada?
operador?
precedencia
aridad

Lo encuentro con 2 formas de hacer, una con muchisimos cond y otra definiendolos asi

```
(def operador
  [ {:name "*" }
   {:name "+" }
   {:name "/" } ] )
```

Y luego aplicar un some con la palabra ingresada

```
(some (comp (partial = "+") :name) operador)
=> true
```

Más que nada por orden y no tener que repetirlo, pero me causa dudas por si es considerado como una variable.

A espera de su respuesta.

Saludos cordiales.

Hola,

al implementar las funciones **palabra-reservada?** y **operador?**, si usás la función **contains?** te podés evitar los "muchísimos cond" que mencionás, y te quedan implementaciones muy breves.

precedencia te conviene implementarla de una forma bien "descomprimida", por una cuestión de claridad: cada operador con su precedencia (y hay muchas de estas).

aridad, en cambio, puede ser implementada de una forma más "compacta": no hay tantas de estas.

En general, tienen mucha libertad para elegir cómo implementar las funciones. No está del todo mal usar **def** para definir *constantes* (como en tu caso), ya que estas no violan la **inmutabilidad**, y permiten evitar que haya bloques iguales repetidos por todo el código. Pero si esas constantes solo se usan una única vez, en mi opinión no se justifica usar **def**.

Saludos!

DCorsi

Fecha: 11/02/2021 19:33 (GMT-03:00) A: dcorsi@fi.uba.ar Asunto: Dudas del TP Applesoft
Hola Diego! Como estas?

Estaba terminando el interprete de Applsoft y me surgieron un par de dudas con respecto a algunas funciones a completar.

ejecutar-asignacion

Con respecto a esta funcion, no me trae mayores problemas en si la asignacion, pero no veo como hacer para evaluar y llevar a un literal el lado de la derecha del igual:

En los ejemplos parciera que no puede ser cualquier cosa, sino literales u operaciones con literales. Sea cualquiera de los casos, podría utilizar el evaluar pero con un ambiente solo con las variables? O directamente una copia del amb (de manera que no me afecte al ambiente el evaluar)

precedencia

En cuanto a la precedencia, busque numerosos articulos en internet sobre la precedencia de Applesoft pero no pude encontrar nada, hay alguna regla en general sobre la misma que haya que saber? O simplemente tengo que usar el emulador y a prueba y error deducir el orden de cada una.

Un saludo,

Hola!
La evaluación se hace en dos pasadas:
1) una función usa el algoritmo del patio de maniobras de Dijkstra para convertir la expresión original a la notación polaca inversa (RPN).
2) una función evalúa la RPN y devuelve el resultado.

Ambas funciones ya son provistas completas.

En uno de los enlaces que aparecen comentados arriba de todo en el código fuente están las precedencias. Igual allí hay un error que no impide que anden los programas del TP: el menos unario y la potencia están al revés...

Saludos!
DCorsi

Sent: Friday, February 12, 2021 at 3:28 PM
To: "Diego Corsi" <dcorsi@fi.uba.ar>
Subject: Consultas del TP de Clojure

Buenas tardes profesor,
Quería hacerle unas consultas por el TP de Clojure.
- En primer lugar, estoy medio perdido a cómo encararlo. Sé que tenemos que completar las funciones evaluar y aplicar, y además implementar las funciones que les siguen, pero no estoy seguro si lo estoy haciendo en el orden correcto. Yo comencé con las que hay que implementar porque parecen más simples, similares a los ejercicios de la guía, y luego eso ya me aclararía como completar las funciones evaluar y aplicar, pero tal vez me estoy confundiendo por no saber bien cómo encararlo y por ahora no parece aclararse. Tal vez es porque no estoy muy avanzado pero siento que no sé cómo resolverlo.
- En segundo lugar, no entiendo la parte final del PDF del TP:

Características de Applesoft BASIC y Apple DOS 3.3 a implementar en el intérprete

Comandos de Apple DOS 3.3

■ **Para la transferencia de archivos**
LOAD: carga un archivo.
SAVE: graba un archivo.

Sentencias de Applesoft BASIC

■ **Para la Entrada/Salida de datos**
INPUT: espera, con *prompt*, valor(es) por teclado.
PRINT: muestra valor(es) por pantalla.
?: lo mismo que PRINT.

■ **Para el uso de datos embebidos**
DATA: define datos embebidos en el código.
READ: lee variable(s) desde los datos embebidos.
REM: embebe un comentario en el código.
RESTORE: reinicia los datos embebidos.

■ **Para manipular el ambiente**
CLEAR: borra todas las variables.
LET/=: hace la asignación de un valor a una variable.
LIST: muestra el listado de sentencias del programa.
NEW: borra el programa y las variables.
RUN: ejecuta el programa.

■ **Para el control de flujo**
END: detiene la ejecución manteniendo el ambiente.
FOR/TO/NEXT/STEP: ciclo controlado por contador.
GOSUB/RETURN: va... (a una subrutina) y vuelve.
GOTO: va... (a determinada línea del programa).
IF/GOTO: si la condición es verdadera, va...
IF/THEN: si la condición es verdadera, ejecuta o va...
ON/GOSUB/RETURN: según un valor, va... y vuelve.
ON/GOTO: según un valor, va...

Sentencias del intérprete (ni en BASIC ni en DOS)

ENV: muestra el ambiente (*environment*).
EXIT: sale del intérprete y vuelve al REPL de Clojure.

Funciones de Applesoft BASIC

■ **Numéricas**
ATN: retorna la arcotangente de un número.
INT: retorna la parte entera de un número.
SIN: retorna el seno de un ángulo dado en radianes.

■ **Para cadenas de caracteres**
LEN: retorna la longitud de una cadena.
MID$: retorna una subcadena.

■ **Para la conversión de tipos**
ASC: retorna el cód. ASCII de la inicial de una cadena.
CHR$: retorna el carácter de un código ASCII dado.
STR$: retorna un número convertido en cadena.

Operadores de Applesoft BASIC

■ **Aritméticos**
+: calcula la suma de dos números.
-: calcula la resta de dos números.
***:** calcula la multiplicación de dos números.
/: calcula la división de dos números.
^: calcula la potencia de dos números.

■ **Para cadenas de caracteres**
+: realiza la concatenación de dos cadenas.

■ **Relacionales**
=: determina si dos valores son iguales.
<>: determina si dos valores son distintos.
<: determina si un valor es menor que otro.
<=: determina si un valor es menor o igual que otro.
>: determina si un valor es mayor que otro.
>=: determina si un valor es mayor o igual que otro.

■ **Lógicos**
AND: determina si ambos valores son verdaderos.
OR: determina si alguno de los valores es verdadero.

¿Quiere decir que además de hacer lo que figura en el código fuente hay que implementar estas características? ¿O qué cuando se vaya desarrollando van a ir quedando naturalmente funciones similares (o operadores) y deben llamarse de esa manera?
- En tercer lugar, en las funciones que piden ver si algo es una palabra reservada o un operador por ejemplo, lo hice guardándolo en un set y luego buscándolo ahí, pero no sé si los datos que puse son correctos. Los saqué de la documentación en el código fuente pero creo que son demasiados y me generan problemas al buscarlos.
Por ejemplo, en palabra-reservada? tengo lo siguiente:

```
(defn palabra-reservada? [x]
  (let [palabras-reservadas (set '(ABS, AND, ASC, AT, ATN, CALL, CHR$, CLEAR, COLOR=, CONT, COS,
          DATA, DEF, DEL, DIM, DRAW, END, EXP, FLASH, FN, FOR, FRE, GET,
          GOSUB, GOTO, GR, HCOLOR=, HGR, HGR2, HIMEM:, HLIN, HOME, HPLOT,
```

Lo cual no corre porque HIMEM: y LOMEM: contienen : y creo que Clojure piensa que es el comienzo de una keyword y se rompe. Lo mismo sucede con operador donde ^ no puede ser un elemento del set, pero al escaparlo con \ no funciona cuando se busca como símbolo:

Perdón por hacer tantas preguntas juntas pero la verdad que me siento bastante perdido para encarar el TP.
Quedo atento a cualquier respuesta.
Desde ya, muchas gracias
Saludos,

Hola,

si primero resolvés correctamente las funciones que les siguen a evaluar y a aplicar, ya habrás avanzado bastante. Así el intérprete ya debería tener alguna funcionalidad parcial. Si luego completás evaluar y aplicar, tendrás el intérprete terminado y andando.

La Apple II tenía un sistema operativo (Apple DOS 3.3) y un lenguaje principal (Applesoft BASIC). Implementarlos completamente en Clojure sería posible, pero extremadamente complicado como TP de una materia cuatrimestral como la nuestra (habría que manejar varios modos gráficos, E/S para diversos dispositivos, etc.). Por eso, solamente implementamos en el intérprete una selección de características, suficiente para poder cargar, visualizar (listar) y correr los 10 programas dados (los cuales correrían sin cambios en una Apple II original).

Así que no hay un "además"... Las características de Applesoft BASIC y Apple DOS 3.3 seleccionadas son lo mínimo que hay que tener implementado para que el intérprete funcione correctamente y los 10 programas dados se puedan cargar, visualizar (listar) y correr. Si algo falta, el manejo del intérprete se complicará (por ejemplo, sin EXIT no se podrá salir del intérprete) o alguno de los programas fallará.

Tu implementación de la función **palabra_reservada?** es demasiado ambiciosa (serviría para una implementación completa de Applesoft BASIC y Apple DOS 3.3). Deberías limitarte a reconocer las palabras reservadas usadas en las características que fueron seleccionadas para nuestra implementación.

Por último, a veces en Clojure es necesario escribir un símbolo en un string y convertirlo luego en símbolo, por ejemplo:
(symbol "^")

No se usa la barra invertida como escape en estos últimos casos.

Saludos!

DCorsi