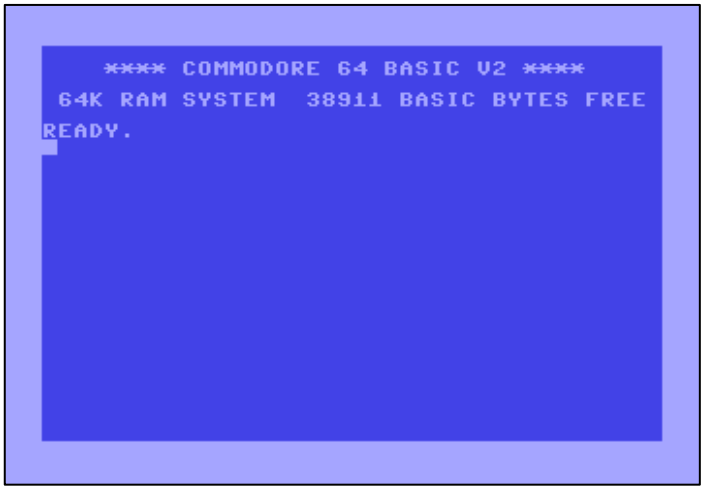
Trabajo Práctico (*individual y obligatorio*)

Intérprete de COMMODORE 64 BASIC V2 en Clojure

La Commodore 64 es una computadora doméstica (*home computer*) de 8 bits comercializada por Commodore International entre agosto de 1982 y abril de 1994. El precio minorista original fue de 595 dólares. Tomó su nombre de sus 64 kilobytes (65.536 bytes) de RAM. Con soporte para *sprites* multicolores y un chip de sonido dedicado, podía crear imágenes y audio superiores en comparación con los sistemas que carecían de dicho hardware. La Commodore 64 traía en ROM un intérprete del lenguaje BASIC (denominado COMMODORE 64 BASIC V2), que también servía como *shell* de la interfaz de usuario y, al encender la máquina, estaba disponible inmediatamente en el indicador READY.



Se ha comparado la Commodore 64 con el automóvil *Ford Modelo T* por su papel en llevar una nueva tecnología a los hogares de clase media a través de la producción en masa. Ha sido incluida en el Libro Guinness de los Records como el modelo de computadora individual más vendido de todos los tiempos, con estimaciones que sitúan el número vendido entre 12,5 y 17 millones de unidades.

Si bien existen actualmente muchos emuladores de la Commodore 64 (incluso hay algunos que corren *online* como, por ejemplo, https://floooh.github.io/tiny8bit/c64.html), su manejo es bastante complicado (deben ingresarse manualmente los programas o, menos frecuentemente, a algunos emuladores como el citado se les pueden arrastrar los archivos con los programas). Además, por emularse exactamente el hardware original, si un programa genera una salida demasiado extensa, no es posible verla después de que ocurrió el *scroll* (la Commodore 64 original no disponía de barras de desplazamiento vertical). Es por ello que, en este trabajo práctico, se pide desarrollar, en el lenguaje **Clojure**, no un emulador sino un **intérprete de COMMODORE 64 BASIC V2**, a fin de poder correr, en una ***Java Virtual Machine*** contemporánea, los siguientes programas escritos para la Commodore 64:

Programa N° 1

STARS.BAS

```
10 INPUT "WHAT IS YOUR NAME"; U$
20 PRINT "HELLO "; U$
30 INPUT "HOW MANY STARS DO YOU WANT"; N
40 IF N < 1 THEN GOTO 90
50 S$ = "" : FOR I = 1 TO N
60 S$ = S$ + "*"
70 NEXT I
80 PRINT S$
90 INPUT "DO YOU WANT MORE STARS [Y/N]"; A$
100 IF A$ = "N" THEN 130
110 IF A$ = "Y" GOTO 30
120 GOTO 90
130 PRINT "GOODBYE "; U$
140 END
```

Programa N° 2

NAME.BAS

```
10 INPUT "WHAT IS YOUR NAME"; N$
20 L = LEN(N$): IF L = 0 THEN GOTO 10
30 PRINT "HELLO " N$: PRINT
40 FOR I = 1 TO L
50 PRINT MID$(N$, I)
60 NEXT I
70 FOR I = L TO 1 STEP -1
80 PRINT MID$(N$, I)
90 NEXT I
100 PRINT : PRINT "GOODBYE "; N$
```

Programa N° 3

SUM.BAS

```
30 LET N=1
40 LET S = S + N
50 PRINT N,S
60 LET N = N + 1
70 IF N <= 100 GOTO 40
80 PRINT : REM PRINT EMPTY LINE
90 PRINT "FINAL SUM: ";S
100 END
```

Programa N° 4

FIBO.BAS

```
05 PRINT "FIBONACCI NUMBERS"
10 LET M% = 5000 : LET C% = 0
20 LET X% = 0 : LET Y% = 1
30 IF X% > M% GOTO 100
40 PRINT "F(" C% ") = " X% : C% = C% + 1
50 X% = X% + Y%
60 IF Y% > M% GOTO 100
70 PRINT "F(" C% ") = " Y% : C% = C% + 1
80 Y% = X% + Y%
90 GOTO 30
100 END
```

Programa N° 5

GCD.BAS

```
10 REM FIND GREATEST COMMON DIVISOR (WITH USER SUPPLIED VALUES)
20 PRINT "ENTER A" : INPUT A : PRINT "ENTER B" : INPUT B
25 IF A<=0 OR B<=0 OR INT(A)<>A OR INT(B)<>B THEN GOTO 20 : REM RE-ENTER
30 IF A < B THEN C = A : A=B : B=C
40 PRINT A,B
50 LET C = A - INT(A/B)*B
60 LET A = B : B = C
70 IF B > 0 GOTO 40
80 PRINT "GCD IS ";A
90 END
```

Programa N° 6

PRIMES.BAS

```
10 PRINT "SEARCH PRIME NUMBERS UP TO: " : INPUT MAX
20 X = 1
30 LET P = .
40 IF X < 2 OR X <> INT(X) GOTO160
50 IF X=2 OR X = 3 OR X = 5THEN P=1 : GOTO 160
60 IF X/2 = INT(X/2) GOTO 160
70 IF X/3 = INT(X/3) GOTO 160
80 D = 5
90 Q = X/D : IF Q = INT(Q) GOTO 160
100 D = D + 2
110 IF D*D> X GOTO 150
120 Q = X/D : IF Q = INT(Q) THEN GOTO 160
130 D=D+4
140 IF D*D <= X GOTO 90
150 P = 1
160IFP=1THENPRINTX;" ";
170 X = X + 1
180 IF X < MAX THEN 30
190 PRINT
200 END
```

Programa N° 7

NATO.BAS

```
100 PRINT "ENTER A WORD: " : INPUT W$
110 T = LEN(W$): IF T = 0 THEN 100
120 PRINT "IT'S SPELLED AS FOLLOWS: ";
130 FOR I = 1 TO T
140 L = ASC(MID$(W$, I, 1)) - 64
150 IF L < 1 OR L > 26 THEN PRINT "??? "; : GOTO 190
160 FOR J = 1 TO L : READ S$ : NEXT J
170 PRINT S$; " ";
180 RESTORE
190 NEXT I
200 END
1000 DATA ALFA, BRAVO, CHARLIE, DELTA, ECHO, FOXTROT, GOLF, HOTEL
1010 DATA INDIA, JULIETT, KILO, LIMA, MIKE, NOVEMBER, OSCAR, PAPA
1020 DATA QUEBEC, ROMEO, SIERRA, TANGO, UNIFORM, VICTOR, WHISKEY, X-RAY
1030 DATA YANKEE: DATAZULU :REM EL ULTIMO VALOR ES ZULU
1040 DATA HOLA MUNDO
```

Programa N° 8

SINE.BAS

```
100 PRINT "X","SIN(X)"
110 PRINT "---","------"
120 FOR I = 1 TO 19 : PRINT " "; : NEXT I
130 FOR A = 0 TO 8 * ATN(1) STEP 0.1
140 PRINT "*"
150 PRINT INT (A * 100) / 100, "  "; INT (SIN(A) * 100000) / 100000
160 FOR I = 1 TO 20 + SIN(A) * 18 : PRINT " "; : NEXT I,A
170 PRINT "*" : A = 8 * ATN(1)
180 PRINT INT (A * 100) / 100, "  "; INT (SIN(A) * 100000) / 100000
190 FOR I = 1 TO 19 : PRINT " "; : NEXT I : PRINT "*"
```

Programa N° 9

HEX.BAS

```
10 REM THIS PROGRAM PRINTS DECIMAL AND CORRESPONDING HEXADECIMAL NUMBERS.
15 REM IT SHOWS 16 NUMBERS PER PAGE. TO SHOW THE NEXT PAGE PRESS ANY KEY.
30 LET S = 0
40 REM NEXT SCREEN
50 PRINT "DECIMAL","HEXA"
60 PRINT "-------","----"
70 FOR I = S TO S + 15
80 LET A = I
90 GOSUB 200
100 PRINT I, HEX$
110 NEXT I
120 PRINT "PRESS RETURN OR TYPE 'QUIT' AND RETURN" : INPUT V$ : REM WAIT
130 IF V$ = "QUIT" GOTO 180
140 IF V$ <> "" THEN 120
150 S = S + 16
160 PRINT
170 GOTO 50
180 END
200 HEX$ = ""
210 LET B = A - INT (A/16) * 16 : REM B = A MOD 16
220 IF B < 10 THEN H$ = MID$(STR$(B), 2, 1)
230 IF B >= 10 AND B <= 15 THEN H$ = CHR$(65 + B - 10)
240 LET HEX$ = H$ + HEX$
250 LET A = (A - B) / 16
260 IF A > 0 THEN GOTO 210
270 RETURN
```

Programa N° 10

BIN-DEC.BAS

```
 80 PRINT "————————————————————————————————————————"
 90 PRINT " BIN -> DEC  /  DEC -> BIN  CONVERTER "
100 PRINT "————————————————————————————————————————"
110 PRINT
120 PRINT "1) BINARY -> DECIMAL"
130 PRINT "2) DECIMAL -> BINARY"
140 PRINT "3) QUIT"
150 INPUT "YOUR CHOICE"; CH
160 IF CH < 0 THEN 180
170 ON CH GOTO 200, 340, 420
180 PRINT "WRONG CHOICE! RETRY..."
190 GOTO 110
200 LET P = 0 : LET S = 0
210 PRINT "ENTER BINARY NUMBER" : INPUT N$
220 L = LEN (N$) : IF L=0 GOTO 310
230 FOR I=1 TO L
240 LET B$ = MID$(N$, L-I+1, 1)
250 IF B$ <> "0" AND B$ <> "1" GOTO 310
260 K = 0 : IF B$ = "1" THEN K = 1
270 IF K > 0 THEN S = S + INT(0.1 + EXP(P * LOG(2)))
280 LET P = P + 1
290 NEXT
300 GOTO 320
310 PRINT "ERROR, INVALID BINARY ENTERED" : GOTO 200
320 PRINT "AS DECIMAL: ";S
330 GOTO 110
340 X$ = "" : PRINT "ENTER AN INTEGER"
345 PRINT "(GREATER OR EQUAL THAN ZERO)" : INPUT A
350 IF A < 0 OR A<>INT(A) GOTO 340
360 LET B = A - INT (A/2) * 2
370 LET X$ = MID$(STR$(B), 2, 1) + X$
380 LET A = (A - B) / 2
390 IF A > 0 GOTO 360
400 PRINT "AS BINARY: ";X$
410 GOTO 110
420 END
```

El intérprete a desarrollar en **Clojure** ofrecerá los dos modos de ejecución de **COMMODORE 64 BASIC V2**: ejecución inmediata y ejecución diferida. Deberá estar basado en un REPL (*read-eval-print-loop*) que acepte, además de sentencias de **COMMODORE 64 BASIC V2**, los comandos del intérprete. No será necesario utilizar espacios para separar los distintos símbolos del lenguaje. Soportará tres tipos de datos: números enteros, números de punto flotante y cadenas de caracteres.

Características de COMMODORE 64 BASIC V2 a implementar en el intérprete

Comandos del intérprete

ENV: muestra el ambiente (*environment*).
LOAD: carga un archivo.
SAVE: graba un archivo.
RUN: ejecuta el programa.
EXIT: sale del intérprete y vuelve al REPL de Clojure.

Sentencias de COMMODORE 64 BASIC V2

■ **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.

■ **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...

Funciones de COMMODORE 64 BASIC V2

■ **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.
EXP: retorna la constante E elevada al argumento.
LOG: retorna el logaritmo natural del argumento.

■ **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 COMMODORE 64 BASIC V2

■ **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.

■ **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.

A los efectos de desarrollar el intérprete solicitado, se deberá partir de los siguientes materiales proporcionados por la cátedra en el aula virtual de la materia en el campus FIUBA:

- Apunte de la cátedra: *Interpretación de programas de computadora*, donde se explican la estructura y el funcionamiento de los distintos tipos de intérpretes posibles, en particular la *interpretación iterativa*, que es la estrategia a utilizar en este trabajo práctico.
- Apunte de la cátedra: *Clojure*, donde se resume el lenguaje a utilizar para el desarrollo.
- Tutorial: *Pasos para crear un proyecto en Clojure usando Leiningen*, donde se indica cómo desarrollar un proyecto dividido en código fuente y pruebas, y su despliegue como *archivo jar*.
- Código fuente de un intérprete de **Commodore 64 BASIC V2** sin terminar, para completarlo.
- 10 archivos para interpretar.

Con este trabajo práctico, se espera que las/los estudiantes adquieran conocimientos profundos sobre el proceso de interpretación de programas y el funcionamiento de los intérpretes de lenguajes de programación y que, a la vez, pongan en práctica los conceptos del paradigma de *Programación Funcional* vistos durante el cuatrimestre.

Para aprobar la cursada, se deberá entregar **hasta el 21/06/2023** (por e-mail a **dcorsi@fi.uba.ar** o a **corsi@mail.com** un **proyecto** compuesto por el código fuente del intérprete de **Commodore 64 BASIC V2** en Clojure y las pruebas de las funciones desarrolladas (como *mínimo*, las pruebas correspondientes a los ejemplos que acompañan el código fuente del intérprete sin terminar proporcionado por la cátedra). El archivo enviado deberá estar comprimido y no deberá contener archivos ejecutables dentro (para evitar que el servidor de correo electrónico lo rechace). El nombre del archivo deberá estar formado por el apellido y el número de padrón de su autor/a, separados por un guion medio (por ejemplo: **Corsi-123456.zip**).

Al momento de rendir la evaluación final de la materia, se deberá modificar el intérprete presentado en este trabajo práctico, incorporándole alguna funcionalidad adicional.