Linguagem com construções imperativas (L2)

Sintaxe abstrata:

$$
\begin{array}{lcl}
e & \in & \mathsf{Expr} \\
e & ::= & n \mid b \mid e_1\ op\ e_2 \mid \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \\
 & \mid & x \mid \mathsf{let}\ x : T = e_1\ \mathsf{in}\ e_2 \\
 & \mid & e_1 := e_2 \mid\ !\ e \mid \mathsf{new}\ e \mid () \mid \mathsf{while}\ e_1\ \mathsf{do}\ e_2 \mid e_1; e_2 \mid \boxed{\mathrm{l}} \\
 & \mid & \mathsf{read}\ () \mid \mathsf{print}\ \ e \\
 & & \\
v & \in & \mathsf{Values} \\
v & ::= & n \mid b \mid () \mid \boxed{\mathrm{l}} \\
 & & \\
T & \in & \mathsf{Types} \\
T & ::= & \mathsf{int} \mid \mathsf{bool} \mid \mathsf{ref}\ T \mid \mathsf{unit} \\
 & & \\
\mathsf{in}, \mathsf{out} & \in & \mathbb{Z}^* \qquad \text{(listas finitas de inteiros)} \\
\mathsf{in} & ::= & [] \mid n :: \mathsf{in} \\
\mathsf{out} & ::= & [] \mid n :: \mathsf{out} \\
n.l & & \text{significa prefixar a lista } l \text{ com o elemento } n \\
l.n & & \text{significa posfixar a lista } l \text{ com o elemento } n \\
 & & \\
n & \in & \mathbb{Z} \\
b & \in & \{\mathsf{true}, \mathsf{false}\} \\
op & \in & \{+, <, \ldots\} \qquad \text{(operações binárias, no mínimo adição e menor-que)} \\
\boxed{\mathrm{l}} & \in & \mathsf{Locations} \qquad \text{(localizações/endereços de memória)}
\end{array}
$$

Semântica operacional *small-step*:

$$\frac{[\![n]\!] = [\![n_1]\!] + [\![n_2]\!]}{n_1{+}n_2, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow n, \sigma, \mathtt{in}, \mathtt{out}} \quad (\textsc{op}+)$$

$$\frac{[\![n_1]\!] < [\![n_2]\!]}{n_1{<}n_2, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow \mathsf{true}, \sigma, \mathtt{in}, \mathtt{out}} \quad (\textsc{op}{<}\textsc{true})$$

$$\frac{[\![n_1]\!] \geq [\![n_2]\!]}{n_1{<}n_2, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow \mathsf{false}, \sigma, \mathtt{in}, \mathtt{out}} \quad (\textsc{op}{<}\textsc{false})$$

$$\frac{e_1, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow e_1', \sigma', \mathtt{in}', \mathtt{out}'}{e_1\ op\ e_2, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow e_1'\ op\ e_2, \sigma', \mathtt{in}', \mathtt{out}'} \quad (\textsc{op}1)$$

$$\frac{e_2, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow e_2', \sigma', \mathtt{in}', \mathtt{out}'}{v\ op\ e_2, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow v\ op\ e_2', \sigma', \mathtt{in}', \mathtt{out}'} \quad (\textsc{op}2)$$

$$\mathsf{if\ true\ then}\ e_2\ \mathsf{else}\ e_3, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow e_2, \sigma, \mathtt{in}, \mathtt{out} \quad (\textsc{if}1)$$

$$\mathsf{if\ false\ then}\ e_2\ \mathsf{else}\ e_3, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow e_3, \sigma, \mathtt{in}, \mathtt{out} \quad (\textsc{if}2)$$

$$\frac{e_1, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow e_1', \sigma', \mathtt{in}', \mathtt{out}'}{\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow \mathsf{if}\ e_1'\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3, \sigma', \mathtt{in}', \mathtt{out}'} \quad (\textsc{if}3)$$

$$\frac{e_1, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow e_1', \sigma', \mathtt{in}', \mathtt{out}'}{\mathsf{let}\ x : T = e_1\ \mathsf{in}\ e_2, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow \mathsf{let}\ x : T = e_1'\ \mathsf{in}\ e_2, \sigma', \mathtt{in}', \mathtt{out}'} \quad (\textsc{e-let}1)$$

$$\frac{}{\mathsf{let}\ x : T = v\ \mathsf{in}\ e_2, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow \{v/x\}\ e_2, \sigma, \mathtt{in}, \mathtt{out}} \quad (\textsc{e-let}2)$$

$$\frac{l \in Dom(\sigma)}{l := v, \sigma, \mathtt{in}, \mathtt{out} \longrightarrow (), \sigma[l \mapsto v], \mathtt{in}, \mathtt{out}} \quad (\textsc{atr}1)$$

$$\frac{e, \sigma, \texttt{in}, \texttt{out} \longrightarrow e', \sigma', \texttt{in}', \texttt{out}'}{l := e, \sigma, \texttt{in}, \texttt{out} \longrightarrow l := e', \sigma', \texttt{in}', \texttt{out}'} \quad (\textsc{atr2})$$

$$\frac{e_1, \sigma, \texttt{in}, \texttt{out} \longrightarrow e_1', \sigma', \texttt{in}', \texttt{out}'}{e_1 := e_2, \sigma, \texttt{in}, \texttt{out} \longrightarrow e_1' := e_2, \sigma', \texttt{in}', \texttt{out}'} \quad (\textsc{atr})$$

$$\frac{l \in Dom(\sigma) \quad \sigma(l) = v}{! \; l, \sigma, \texttt{in}, \texttt{out} \longrightarrow v, \sigma, \texttt{in}, \texttt{out}} \quad (\textsc{deref1})$$

$$\frac{e, \sigma, \texttt{in}, \texttt{out} \longrightarrow e', \sigma', \texttt{in}', \texttt{out}'}{! \; e, \sigma, \texttt{in}, \texttt{out} \longrightarrow ! \; e', \sigma', \texttt{in}', \texttt{out}'} \quad (\textsc{deref})$$

$$\frac{l \notin Dom(\sigma)}{\mathsf{new}\ v, \sigma, \texttt{in}, \texttt{out} \longrightarrow l, \sigma[l \mapsto v], \texttt{in}, \texttt{out}} \quad (\textsc{new1})$$

$$\frac{e, \sigma, \texttt{in}, \texttt{out} \longrightarrow e', \sigma', \texttt{in}', \texttt{out}'}{\mathsf{new}\ e, \sigma, \texttt{in}, \texttt{out} \longrightarrow \mathsf{new}\ e', \sigma', \texttt{in}', \texttt{out}'} \quad (\textsc{new})$$

$$\frac{}{(); \; e_2, \sigma, \texttt{in}, \texttt{out} \longrightarrow e_2, \sigma, \texttt{in}, \texttt{out}} \quad (\textsc{seq1})$$

$$\frac{e_1, \sigma, \texttt{in}, \texttt{out} \longrightarrow e_1', \sigma', \texttt{in}', \texttt{out}'}{e_1; e_2, \sigma, \texttt{in}, \texttt{out} \longrightarrow e_1'; e_2, \sigma', \texttt{in}', \texttt{out}'} \quad (\textsc{seq})$$

$$\mathsf{while}\ e_1\ \mathsf{do}\ e_2, \sigma, \texttt{in}, \texttt{out} \longrightarrow \mathsf{if}\ e_1\ \mathsf{then}\ (e_2; \mathsf{while}\ e_1\ \mathsf{do}\ e_2)\ \mathsf{else}\ (), \sigma, \texttt{in}, \texttt{out} \quad (\textsc{e-while})$$

$$\mathsf{print}\ n, \sigma, \texttt{in}, \texttt{out} \longrightarrow (), \sigma, \texttt{in}, \texttt{out}.n \quad (\textsc{print-n})$$

$$\frac{e, \sigma, \texttt{in}, \texttt{out} \longrightarrow e', \sigma', \texttt{in}', \texttt{out}'}{\mathsf{print}\ e, \sigma, \texttt{in}, \texttt{out} \longrightarrow \mathsf{print}\ e', \sigma', \texttt{in}', \texttt{out}'} \quad (\textsc{print})$$

$$\mathsf{read}\ (), \sigma, n.\texttt{in}, \texttt{out} \longrightarrow n, \sigma, \texttt{in}, \texttt{out} \quad (\textsc{read})$$

Sistema de Tipos:

$$\frac{}{\Gamma \vdash n : \mathsf{int}} \quad (\textsc{t-int})$$

$$\frac{}{\Gamma \vdash b : \mathsf{bool}} \quad (\textsc{t-bool})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{int} \qquad \Gamma \vdash e_2 : \mathsf{int}}{\Gamma \vdash e_1 + e_2 : \mathsf{int}} \quad (\textsc{t-op}+)$$

$$\frac{\Gamma \vdash e_1 : \mathsf{int} \qquad \Gamma \vdash e_2 : \mathsf{int}}{\Gamma \vdash e_1 < e_2 : \mathsf{bool}} \quad (\textsc{t-op}<)$$

$$\frac{\Gamma \vdash e_1 : \mathsf{bool} \qquad \Gamma \vdash e_2 : T \qquad \Gamma \vdash e_3 : T}{\Gamma \vdash \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 : T} \quad (\textsc{t-if})$$

$$\frac{\Gamma(x) = T}{\Gamma \vdash x : T} \quad (\textsc{t-var})$$

$$\frac{\Gamma \vdash e_1 : T \qquad \Gamma, x \mapsto T \vdash e_2 : T'}{\Gamma \vdash \mathsf{let}\ x : T = e_1\ \mathsf{in}\ e_2 : T'} \quad (\textsc{t-let})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{ref}\ T \qquad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 := e_2 : \mathsf{unit}} \quad (\textsc{t-atr})$$

$$\frac{\Gamma \vdash e : \mathsf{ref}\ T}{\Gamma \vdash ! \; e : T} \quad (\textsc{t-deref})$$

$$\frac{\Gamma \vdash e : T}{\Gamma \vdash \mathsf{new}\ e : \mathsf{ref}\ T} \quad (\textsc{t-new})$$

$$\frac{}{\Gamma \vdash () : \mathsf{unit}} \quad (\textsc{t-unit})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{bool} \qquad \Gamma \vdash e_2 : \mathsf{unit}}{\Gamma \vdash \mathsf{while}\ e_1\ \mathsf{do}\ e_2 : \mathsf{unit}} \quad (\textsc{t-while})$$

$$\frac{\Gamma \vdash e_1 : \mathsf{unit} \qquad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1; \; e_2 : T} \quad (\textsc{t-seq})$$

$$\frac{}{\Gamma \vdash \mathsf{read}\ () : \mathsf{int}} \quad (\textsc{t-read})$$

$$\frac{\Gamma \vdash e : \mathsf{int}}{\Gamma \vdash \mathsf{print}\ e : \mathsf{unit}} \quad (\textsc{t-print})$$

Trabalho

O trabalho consiste em implementar em OCaml um interpretador para a linguagem L2 da especificação acima e com variações definidas abaixo que serão deixadas propositalmente subespecificadas.

O trabalho será avaliado da seguinte forma:

- nota máxima 9,0 para os trabalhos que implementarem somente L2 conforme a especificação dada acima
- nota máxima 10,0 para os trabalhos que implementarem também uma dentre as seguintes opções:
  - arrays
  - mecanismo de exceções
  - expressão *for* para repetições

Arquivo com as definições dos datatypes necessários e com alguns casos de teste referentes a L2 da especificação dada será disponibilizado no Moodle da disciplina.

O trabalho deve ser realizado em grupo (2 a 3 componentes), e ser entregue via Moodle no prazo especificado. Após a entrega, o trabalho será apresentado em laboratório pelos componentes do grupo, conforme cronograma de apresentações disponível no Moodle.