Operational Semantics

Initial configuration of executing statement S: $\langle S, \emptyset, \emptyset \rangle$.

Arithmetic expressions (E):

$$\begin{aligned}
\langle n\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \boldsymbol{c}, n\boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle v\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \boldsymbol{c}, m(v), \boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle (E_1 \ iop \ E_2)\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle E_1 \ E_2 \ iop \ \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle iop \ \boldsymbol{c}, n_2 n_1 \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \boldsymbol{c}, n\boldsymbol{s}, \boldsymbol{m} \rangle \text{ where } n = n_1 \ \underline{iop} \ n_2
\end{aligned}$$

Execution will be guarded against division by zero.

Boolean conditions (C):

$$\begin{aligned}
\langle b\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle b, n\boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle (E_1 \ bop \ E_2)\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle E_1 \ E_2 \ bop \ \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle bop \ \boldsymbol{c}, n_2 n_1 \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \boldsymbol{c}, b\boldsymbol{s}, \boldsymbol{m} \rangle \text{ where } b = n_1 \ \underline{bop} \ n_2
\end{aligned}$$

Statements (S):

$$\begin{aligned}
\langle ()\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle (S_1; S_2)\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle S_1 S_2 \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle v = E\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle E \ save \ \boldsymbol{c}, v\boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle save \ \boldsymbol{c}, nv\boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m}[v = n] \rangle \\
\langle continue \ \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \boldsymbol{c'}, \boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle break \ \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \boldsymbol{c''}, \boldsymbol{s}, \boldsymbol{m} \rangle \\
\langle exit \ \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m} \rangle &\to \langle \emptyset, \emptyset, \emptyset \rangle
\end{aligned}$$

where $\boldsymbol{c'}$:= the first $artificial\ while$ in $\boldsymbol{c}$ and the commands after it; and $\boldsymbol{c''}$:= the commands after the first $artificial\ while$ (excluding it).

Branching (if):

$$\langle (if\ C\ then\ S_t\ else\ S_f)\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m}\rangle \to \langle C\ branch\ \boldsymbol{c}, S_t S_f \boldsymbol{s}, \boldsymbol{m}\rangle$$
$$\langle branch\ \boldsymbol{c}, true\ S_t S_f \boldsymbol{s}, \boldsymbol{m}\rangle \to \langle S_t \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m}\rangle$$
$$\langle branch\ \boldsymbol{c}, false\ S_t S_f \boldsymbol{s}, \boldsymbol{m}\rangle \to \langle S_f \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m}\rangle$$

Looping ($while$):

$$\langle (while\ C\ do\ S)\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m}\rangle \to \langle C\ loop\ \boldsymbol{c}, CS\boldsymbol{s}, \boldsymbol{m}\rangle$$
$$\langle loop\ \boldsymbol{c}, false\ CS\boldsymbol{s}, \boldsymbol{m}\rangle \to \langle \boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m}\rangle$$
$$\langle loop\ \boldsymbol{c}, true\ CS\boldsymbol{s}, \boldsymbol{m}\rangle \to \langle S(while\ C\ do\ S)\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{m}\rangle$$

Execution will be guarded against infinite cycles.

After evaluating the condition, we flip the $artificial$ flag on. That is because we should only *break/continue* inside loops. For a given loop, having evaluated its condition guarantees that we are inside of it. A loop we have not entered (not inserted in the command stack $artificially$), should be invisible to the *break/continue*.