Processamento de Linguagens (cod. 11567)
Departamento de Informática
Universidade da Beira Interior
2018/2019

Arith ++

Sintaxe, semântica e tipagem

João Reis
joao.reis@ubi.pt

1 Introdução

Este documento serve de referência para o resultado esperado na primeira entrega do trabalho prático desta unidade curricular. Aqui é descrita uma proposta de extensão à linguagem `Arith` chamada `Arith++`, especificando em toda a sua completude a sintaxe abstrata, semântica operacional *small-steps* e tipagem.

2 Sintaxe Abstrata

2.1 Constantes

A linguagem `Arith++` aceita constantes numéricas inteiras e de vírgula flutuante, assim como constantes *booleanas* e vetores vazios para os tipos mencionados anteriormente.

$\langle c \rangle$ ::= inteiro (`0`, `5`, `123`, ...)
| *float* (`0.01`, `.3`, `2.1`, `100.`, ...)
| *booleano* (`true`, `false`)
| $\langle t \rangle$ `[ ]` (vetor vazio de um determinado tipo)

$\langle t \rangle$ ::= `int`
| `float`
| `bool`

2.2 Expressões

Uma expressão em `Arith++` assume as seguintes formas:

- uma constante;
- uma variável

- o acesso a uma posição de um vetor;
- uma expressão dentro de parênteses;
- operações unárias aritméticas e *booleanas*;
- operações binárias aritméticas, *booleanas* e de comparação;
- atribuição local de variáveis;
- chamada de funções.

⟨*e*⟩ ::= ⟨*c*⟩
| ⟨*ident*⟩
| ⟨*ident*⟩ `.(` ⟨*e*⟩ `)`
| `(` ⟨*e*⟩ `)`
| ⟨*e*⟩ `+` ⟨*e*⟩
| ⟨*e*⟩ `-` ⟨*e*⟩
| ⟨*e*⟩ `*` ⟨*e*⟩
| ⟨*e*⟩ `/` ⟨*e*⟩
| `-` ⟨*e*⟩
| ⟨*e*⟩ `&&` ⟨*e*⟩
| ⟨*e*⟩ `||` ⟨*e*⟩
| `!` ⟨*e*⟩
| ⟨*e*⟩ `>` ⟨*e*⟩
| ⟨*e*⟩ `<` ⟨*e*⟩
| ⟨*e*⟩ `>=` ⟨*e*⟩
| ⟨*e*⟩ `<=` ⟨*e*⟩
| ⟨*e*⟩ `==` ⟨*e*⟩
| `let` ⟨*ident*⟩ `=` ⟨*e*⟩ `in` ⟨*e*⟩
| ⟨*ident*⟩ ⟨*e*⟩

2.3 Instruções

Um programa em `Arith++` é composto por uma ou mais instruções (*statements*). O conjunto de instruções desta linguagem é composto por:

- a atribuição de variáveis globais;
- impressão de uma expressão no ecrã;
- estruturas condicionais e de ciclo *if then else*, *while* e *for* (este último com duas variantes, percorrer os elementos de um vetor ou percorrer um intervalo de números inteiros);
- atribuição a uma posição de um vetor;
- declaração de funções;
- não fazer nada;
- sequência de instruções.

$\langle s \rangle$ ::= `set` $\langle ident \rangle$ `=` $\langle e \rangle$
| `print` $\langle e \rangle$
| `if` $\langle e \rangle$ `then` $\langle s \rangle$ [`else` $\langle s \rangle$] `done`
| `while` $\langle e \rangle$ `do` $\langle s \rangle$ `done`
| `for` $\langle ident \rangle$ `in` $\langle e \rangle$ `do` $\langle s \rangle$ `done`
| `for` $\langle ident \rangle$ `=` $\langle e \rangle$ `to` $\langle e \rangle$ `do` $\langle s \rangle$ `done`
| $\langle e \rangle$ `.(` $\langle e \rangle$ `)` `<-` $\langle e \rangle$
| `func` $\langle ident \rangle$ `(` $\langle params \rangle$ `)` `begin` $\langle e \rangle$ `end`
| `skip`
| $\langle s \rangle$ $\langle s \rangle$

$\langle params \rangle$ ::= $\langle t \rangle$ $\langle ident \rangle$
| $\langle t \rangle$ $\langle ident \rangle$ `,` $\langle params \rangle$

3 Semântica

De seguida, definimos a semântica operacional *small-steps* para a linguagem `Arith++`. Definimos como valores desta linguagem os seguintes:

$$\begin{array}{rcl} v & ::= & i \text{ (valor inteiro)} \\ & & f \text{ (valor vírgula flutuante)} \\ & & b \text{ (valor booleano)} \\ & & x \text{ (variável vista como um endereço)} \\ & & [v_0, \cdots, v_n] \text{ (vetor de valores)} \\ & & fun\ (t_1\ x_1, \cdots, t_n\ x_n) \Rightarrow e \text{ (função)} \end{array}$$

3.1 Expressões

A avaliação das expressões desta linguagem termina sempre. Desta forma, optamos por definir a semântica das expressões com recurso à semântica *big-steps*. Os casos não-triviais serão acompanhados de uma breve descrição.

Aqui temos a semântica das constantes:

$$\frac{}{E, i \twoheadrightarrow i} \quad \frac{}{E, f \twoheadrightarrow f} \quad \frac{}{E, b \twoheadrightarrow b} \quad \frac{}{E, t\ \texttt{[]} \twoheadrightarrow []} \quad \frac{}{E, id \twoheadrightarrow E(id)}$$

De seguida, definimos a semântica do acesso a uma posição de um vetor:

$$\frac{E, id \twoheadrightarrow x \quad E, e \twoheadrightarrow v}{E, id.(e) \twoheadrightarrow E(x[v])}$$

A variável id avalia para um endereço de memória x e a expressão e avalia para um valor v. A expressão $x.(e)$ avalia desta forma para o valor localizado na memória no endereço $x + v$ ($x[v]$).

Definimos de forma trivial as operações binárias e unárias desta linguagem:

$$\frac{E, e \twoheadrightarrow i}{E, -e \twoheadrightarrow -i} \quad \frac{E, e_1 \twoheadrightarrow i_1 \quad E, e_2 \twoheadrightarrow i_2 \quad i = i_1 + i_2}{E, e_1 + e_2 \twoheadrightarrow i} \quad \frac{E, e_1 \twoheadrightarrow i_1 \quad E, e_2 \twoheadrightarrow i_2 \quad i = i_1 - i_2}{E, e_1 - e_2 \twoheadrightarrow i}$$

$$\frac{E, e_1 \twoheadrightarrow i_1 \quad E, e_2 \twoheadrightarrow i_2 \quad i = i_1 \times i_2}{E, e_1 * e_2 \twoheadrightarrow i} \quad \frac{E, e_1 \twoheadrightarrow i_1 \quad E, e_2 \twoheadrightarrow i_2 \quad i_2 \neq 0 \quad i = i_1 \div i_2}{E, e_1 / e_2 \twoheadrightarrow i}$$

$$\frac{E, e_1 \twoheadrightarrow false}{E, e_1\ \texttt{\&\&}\ e_2 \twoheadrightarrow false} \quad \frac{E, e_1 \twoheadrightarrow true \quad E, e_2 \twoheadrightarrow false}{E, e_1\ \texttt{\&\&}\ e_2 \twoheadrightarrow false} \quad \frac{E, e_1 \twoheadrightarrow true \quad E, e_2 \twoheadrightarrow true}{E, e_1\ \texttt{\&\&}\ e_2 \twoheadrightarrow true}$$

$$\frac{E, e_1 \twoheadrightarrow true}{E, e_1 \texttt{ || } e_2 \twoheadrightarrow true} \quad \frac{E, e_1 \twoheadrightarrow false \quad E, e_2 \twoheadrightarrow true}{E, e_1 \texttt{ || } e_2 \twoheadrightarrow true} \quad \frac{E, e_1 \twoheadrightarrow false \quad E, e_2 \twoheadrightarrow false}{E, e_1 \texttt{ || } e_2 \twoheadrightarrow false}$$

$$\frac{E, e \twoheadrightarrow false}{E, \texttt{!}e \twoheadrightarrow true} \quad \frac{E, e \twoheadrightarrow true}{E, \texttt{!}e \twoheadrightarrow false}$$

$$\frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 > v_2}{E, e_1 \texttt{ > } e_2 \twoheadrightarrow true} \quad \frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 <= v_2}{E, e_1 \texttt{ > } e_2 \twoheadrightarrow false}$$

$$\frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 >= v_2}{E, e_1 \texttt{ >= } e_2 \twoheadrightarrow true} \quad \frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 < v_2}{E, e_1 \texttt{ >= } e_2 \twoheadrightarrow false}$$

$$\frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 < v_2}{E, e_1 \texttt{ < } e_2 \twoheadrightarrow true} \quad \frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 >= v_2}{E, e_1 \texttt{ < } e_2 \twoheadrightarrow false}$$

$$\frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 <= v_2}{E, e_1 \texttt{ <= } e_2 \twoheadrightarrow true} \quad \frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 > v_2}{E, e_1 \texttt{ <= } e_2 \twoheadrightarrow false}$$

$$\frac{E, e_1 \twoheadrightarrow v \quad E, e_2 \twoheadrightarrow v}{E, e_1 == e_2 \twoheadrightarrow true} \quad \frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \; v_1 \neq v_2}{E, e_1 == e_2 \twoheadrightarrow false}$$

A semântica da expressão `let in` é definida da seguinte forma:

$$\frac{E, e_1 \twoheadrightarrow v_1 \quad E\{x \mapsto v_1\}, e_2 \twoheadrightarrow v_2}{E, \texttt{ let } x = e_1 \texttt{ in } e_2 \twoheadrightarrow v_2}$$

A expressão e_1 é avaliada no ambiente E para o valor v_1. De seguida, avalia-se a expressão e_2 no ambiente E, acrescido da variável x que tem o valor v_1, para v_2.

$$\frac{E, e \twoheadrightarrow fun\ (x_1, \cdots, x_n) \Rightarrow e_f \quad E, e_1 \twoheadrightarrow v_1 \ \cdots \quad E, e_n \twoheadrightarrow v_n \quad E\{x_1 \mapsto v_1, \cdots, x_n \mapsto v_n\}, e_f \twoheadrightarrow v}{E, e(e_1, \cdots, e_n) \twoheadrightarrow v}$$

3.2 Instruções

Definimos a seguir a semântica das instruções da linguagem `Arith++` usando a semântica *small-steps*, mas mantendo sempre a semântica *big-steps* sempre que nos referimos às expressões.

Para a sequência de instruções, definimos dois casos:

$$\frac{}{E, \texttt{ skip } s \to E, s} \quad \frac{E, s_1 \to E_1, s_1'}{E, s_1 \; s_2 \to E_1, s_1' \; s_2}$$

A atribuição de variáveis globais resulta num novo ambiente em que a variável em causa irá ter associado o valor resultante da avaliação da expressão de atribuição:

$$\frac{E, e \twoheadrightarrow v}{E, \texttt{ set } x = e \to E\{x \mapsto v\}, \texttt{ skip}} \quad \frac{E, e \twoheadrightarrow v}{E, \texttt{ print } e \to E, \texttt{ skip}}$$

A instrução condicional `if` resulta no passo em que se avalia a instrução s caso a condição e se avalie para `true`, ou na avaliação da instrução `skip` caso a condição se avalie para `false`.

$$\frac{E, e \twoheadrightarrow false}{E,\ \texttt{if}\ e\ \texttt{then}\ s\ \texttt{done}\ \rightarrow E,\ \texttt{skip}} \qquad \frac{E, e \twoheadrightarrow true}{E,\ \texttt{if}\ e\ \texttt{then}\ s\ \texttt{done}\ \rightarrow E, s}$$

De forma semelhante, definimos a instrução condicional `if else`, mas desta vez avaliamos s_2 caso a condição se avalie para `false`.

$$\frac{E, e \twoheadrightarrow false}{E,\ \texttt{if}\ e\ \texttt{then}\ s\ \texttt{else}\ s_2\ \texttt{done}\ \rightarrow E,\ s_2} \qquad \frac{E, e \twoheadrightarrow true}{E,\ \texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{done}\ \rightarrow E, s_1}$$

Do ciclo `while` resulta no passo de avaliação da sequência de instruções composta pelo seu corpo seguido do mesmo ciclo `while` sempre que a condição se avalie para `true`. Caso a condição se avalie para `false`, resulta no passo de avaliação da instrução `skip`.

$$\frac{E, e \twoheadrightarrow true}{E,\ \texttt{while}\ e\ \texttt{do}\ s\ \texttt{done}\ \rightarrow E, s\ \texttt{while}\ e\ \texttt{do}\ s\ \texttt{done}} \qquad \frac{E, e \twoheadrightarrow false}{E,\ \texttt{while}\ e\ \texttt{do}\ s\ \texttt{done}\ \rightarrow E,\ \texttt{skip}}$$

A instrução `for` é definida em duas variações, como descrito anteriormente. No caso da iteração sobre os elementos de um vetor, a avaliação do `for` resulta no passo de avaliação no mesmo ambiente, acrescido da variável de ciclo mapeada para o valor da cabeça do vetor, da sequência composta pelo seu corpo e novo ciclo `for` sobre a cauda do vetor a iterar. Quando o vetor a iterar está vazio, resulta no passo de avaliação da instrução `skip`.

$$\frac{E, e \twoheadrightarrow [v_0, \cdots, v_n]}{E,\ \texttt{for}\ x\ \texttt{in}\ e\ \texttt{do}\ s\ \texttt{done} \rightarrow E\{x \mapsto v_0\}, s\ \texttt{for}\ x\ \texttt{in}\ [v_1, \cdots v_n]\ \texttt{do}\ s\ \texttt{done}}$$

$$\frac{E, e \twoheadrightarrow [\ ]}{E,\ \texttt{for}\ x\ \texttt{in}\ e\ \texttt{do}\ s\ \texttt{done} \rightarrow E,\ \texttt{skip}}$$

Temos de seguida o caso do `for` sobre um intervalo de valores. Quando a expressão que define o limite da iteração se avalia para um valor superior ao valor para o qual se avaliação a expressão que define o valor inicial, a avaliação deste caso resulta no passo de avaliação da sequência de instruções composta pelo corpo do `for` seguido de novo `for` em que a expressão inicial é o valor inicial anterior somado com 1. Quando a expressão que define o limite da iteração se avalia para um valor inferior ao valor para o qual se avaliação a expressão que define o valor inicial, a avaliação deste caso resulta num passo igual ao anterior com a exceção de que se subtrai 1 ao valor inicial. Quando a expressão inicial e a expressão de limite avaliam para o mesmo valor, resulta no passo de avaliação da instrução `skip`.

$$\frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 < v_2}{E,\ \texttt{for}\ x = e_1\ \texttt{to}\ e_2\ \texttt{do}\ s\ \texttt{done} \rightarrow E\{x \mapsto v_1\}, s\ \texttt{for}\ x = (v_1 + 1)\ \texttt{to}\ e_2\ \texttt{do}\ s\ \texttt{done}}$$

A atribuição de uma posição de um vetor e definição de uma função comportam-se de forma semelhante à atribuição de variáveis globais.

$$\frac{E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2 \quad v_1 > v_2}{E,\ \texttt{for}\ x = e_1\ \texttt{to}\ e_2\ \texttt{do}\ s\ \texttt{done} \rightarrow E\{x \mapsto v_1\}, s\ \texttt{for}\ x = (v_1 - 1)\ \texttt{to}\ e_2\ \texttt{do}\ s\ \texttt{done}}$$

$$\frac{E, e_1 \twoheadrightarrow v \quad E, e_2 \twoheadrightarrow v}{E,\ \texttt{for}\ x = e_1\ \texttt{to}\ e_2\ \texttt{do}\ s\ \texttt{done} \rightarrow E,\ \texttt{skip}}$$

$$\frac{E, id \twoheadrightarrow x \quad E, e_1 \twoheadrightarrow v_1 \quad E, e_2 \twoheadrightarrow v_2}{E, id.(e_1) <- e_2 \rightarrow E\{x[v_1] \mapsto v_2\},\ \texttt{skip}}$$

$$\frac{}{E,\ \texttt{func}\ x(t_1\ a_1, \cdots, t_n\ a_n)\ \texttt{begin}\ e\ \texttt{end} \rightarrow E\{x \mapsto fun\ x(a_1, \cdots, a_n) \Rightarrow e\},\ \texttt{skip}}$$

4 Tipagem

$$\begin{array}{rcl} \tau & ::= & \texttt{int} \\ & | & \texttt{bool} \\ & | & \texttt{float} \\ & | & [\tau] \\ & | & \tau \times \tau \times \cdots \times \tau \rightarrow \tau \end{array}$$

$$\frac{}{\Gamma \vdash i : \texttt{int}} \quad \frac{}{\Gamma \vdash f : \texttt{float}} \quad \frac{}{\Gamma \vdash b : \texttt{bool}} \quad \frac{}{\Gamma \vdash x : \Gamma(x)}$$

$$\frac{}{\Gamma \vdash \texttt{int}\ \texttt{[]} : [\texttt{int}]} \quad \frac{}{\Gamma \vdash \texttt{float}\ \texttt{[]} : [\texttt{float}]} \quad \frac{}{\Gamma \vdash \texttt{bool}\ \texttt{[]} : [\texttt{bool}]} \quad \frac{\Gamma \vdash id : [\tau] \quad \Gamma \vdash e : \texttt{int}}{\Gamma \vdash id.(e) : \tau}$$

$$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 + e_2 : \texttt{int}} \quad \frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 - e_2 : \texttt{int}} \quad \frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 * e_2 : \texttt{int}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 / e_2 : \texttt{int}} \quad \frac{\Gamma \vdash e : \texttt{int}}{\Gamma \vdash -e : \texttt{int}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{float} \quad \Gamma \vdash e_2 : \texttt{float}}{\Gamma \vdash e_1 + e_2 : \texttt{float}} \quad \frac{\Gamma \vdash e_1 : \texttt{float} \quad \Gamma \vdash e_2 : \texttt{float}}{\Gamma \vdash e_1 - e_2 : \texttt{float}} \quad \frac{\Gamma \vdash e_1 : \texttt{float} \quad \Gamma \vdash e_2 : \texttt{float}}{\Gamma \vdash e_1 * e_2 : \texttt{float}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{float} \quad \Gamma \vdash e_2 : \texttt{float}}{\Gamma \vdash e_1 / e_2 : \texttt{float}} \quad \frac{\Gamma \vdash e : \texttt{float}}{\Gamma \vdash -e : \texttt{float}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1\ \texttt{\&\&}\ e_2 : \texttt{bool}} \quad \frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1\ \texttt{||}\ e_2 : \texttt{bool}} \quad \frac{\Gamma \vdash e : \texttt{bool}}{\Gamma \vdash \texttt{!}e : \texttt{bool}}$$

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\ \texttt{>}\ e_2 : \texttt{bool}} \quad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\ \texttt{<}\ e_2 : \texttt{bool}} \quad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\ \texttt{>=}\ e_2 : \texttt{bool}} \quad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\ \texttt{<=}\ e_2 : \texttt{bool}}$$

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\ \texttt{==}\ e_2 : \texttt{bool}} \quad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma + x : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash\ \texttt{let}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 : \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma + x : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{let}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 : \tau_2}$$

$$\frac{\Gamma \vdash x : \tau_1 \times \cdots \times \tau_n \rightarrow \tau \quad \Gamma \vdash e_1 : \tau_1 \quad \cdots \quad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash x(e_1, \cdots, e_n) : \tau}$$

$$\frac{}{\Gamma \vdash \texttt{skip}} \qquad \frac{\Gamma \vdash s_1 \quad \Gamma \vdash s_2}{\Gamma \vdash s_1\ s_2} \qquad \frac{\Gamma \vdash e : \tau \quad \Gamma + id : \tau}{\Gamma \vdash \texttt{set}\ id\ \texttt{=}\ e} \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \texttt{print}\ e}$$

$$\frac{\Gamma \vdash e : \texttt{bool} \quad \Gamma \vdash s}{\Gamma \vdash \texttt{if}\ e\ \texttt{then}\ s\ \texttt{done}} \qquad \frac{\Gamma \vdash e : \texttt{bool} \quad \Gamma \vdash s_1 \quad \Gamma \vdash s_2}{\Gamma \vdash \texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{done}} \qquad \frac{\Gamma \vdash e : [\tau] \quad \Gamma + id : \tau \quad \Gamma \vdash s}{\Gamma \vdash \texttt{for}\ id\ \texttt{in}\ e\ \texttt{do}\ s\ \texttt{done}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int} \quad \Gamma + id : \texttt{int}}{\Gamma \vdash \texttt{for}\ id\ \texttt{=}\ e_1\ \texttt{to}\ e_2\ \texttt{do}\ s\ \texttt{done}} \qquad \frac{\Gamma \vdash e : \texttt{bool} \quad \Gamma \vdash s}{\Gamma \vdash \texttt{while}\ e\ \texttt{do}\ s\ \texttt{done}}$$

$$\frac{\Gamma \vdash id : [\tau] \quad \Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash id\texttt{.(}e_1\texttt{)}\ \texttt{<-}\ e_2} \qquad \frac{\Gamma \vdash e : \tau \quad \Gamma + id : t_1 \times \cdots \times t_n \to \tau}{\Gamma \vdash \texttt{func}\ id\ \texttt{(}t_1\ id_1, \cdots, t_n\ id_n\texttt{)}\ \texttt{begin}\ e\ \texttt{end}}$$