A Simple Language

- This is very much like LiveOak-0.

$$
\begin{aligned}
&\mathit{Program} \rightarrow \mathit{Decls}\ \mathbf{;}\ \mathit{Expn}\\
&\mathit{Decls} \rightarrow \mathit{Decls}\ \mathbf{;}\ \mathit{Decls} \mid \mathbf{id} \mathbf{:}\ \mathit{Type}\\
&\mathit{Type} \rightarrow \mathbf{char} \mid \mathbf{int} \mid \mathit{Type}\,[\,\mathbf{num}\,] \mid \&\mathit{Type}\\
&\mathit{Expn} \rightarrow \mathbf{literal} \mid \mathbf{num} \mid \mathbf{id} \mid\\
&\qquad\quad \mathit{Expn}\ \mathtt{mod}\ \mathit{Expn} \mid \mathit{Expn}\,[\,\mathit{Expn}\,] \mid {*}\mathit{Expn}
\end{aligned}
$$

- How do we type-check programs in this language?
  - What effect do declarations have on judgments?
  - What are the typing rules for expressions?
- Initial environment
  - $F\ = \{\text{mod:}\ \mathbf{int} \times \mathbf{int} \rightarrow \mathbf{int}\}$
  - $G\ = \emptyset$

Effect of Declarations

- Variable bindings will *extend* the context component G of the environment.
- If we encounter the declaration $x\colon T$ in an existing context G, the resulting context G' will be written as
$$G' = (G, x\colon T)$$
- Typing rule for declarations
$$\frac{}{F, G \vdash x\colon T} \quad x\colon T \in G$$
- The allowable forms of T will be determined by the *Type* sublanguage.

Type Inference for Expressions

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | * *Expn*

Type Inference for Expressions

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | * *Expn*

- $\dfrac{}{F, G \vdash \textbf{literal} : \textbf{char}}$ $\qquad$ $\dfrac{}{F, G \vdash \textbf{num} : \textbf{int}}$

Type Inference for Expressions

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | * *Expn*

- $\dfrac{}{F, G \vdash \mathbf{literal} : \mathbf{char}}$ $\dfrac{}{F, G \vdash \mathbf{num} : \mathbf{int}}$
- $\dfrac{}{F, G \vdash x : T}$ $x : T \in G$

Type Inference for Expressions

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | * *Expn*

- $\dfrac{}{F, G \vdash \textbf{literal} : \textbf{char}}$ $\dfrac{}{F, G \vdash \textbf{num} : \textbf{int}}$
- $\dfrac{}{F, G \vdash x : T}\ x : T \in G$
- $\dfrac{F, G \vdash e_1 : \textbf{int} \quad F, G \vdash e_2 : \textbf{int}}{F, G \vdash e_1 \,\mathrm{mod}\, e_2 : \textbf{int}}$

Type Inference for Expressions

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | * *Expn*

- $\dfrac{}{F, G \vdash \textbf{literal} : \textbf{char}}$ $\dfrac{}{F, G \vdash \textbf{num} : \textbf{int}}$
- $\dfrac{}{F, G \vdash x : T}$ $x : T \in G$
- $\dfrac{F, G \vdash e_1 : \textbf{int} \quad F, G \vdash e_2 : \textbf{int}}{F, G \vdash e_1 \,\mathrm{mod}\, e_2 : \textbf{int}}$
- $\dfrac{F, G \vdash e : \textbf{ARRAY}(N, T)}{F, G \vdash e[x] : T}$ $0 \leq val(x) < N$

Type Inference for Expressions

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | **Expn*

- $$\frac{}{F, G \vdash \mathbf{literal}:\mathbf{char}} \qquad \frac{}{F, G \vdash \mathbf{num}:\mathbf{int}}$$
- $$\frac{}{F, G \vdash x:T} \; x:T \in G$$
- $$\frac{F, G \vdash e_1:\mathbf{int} \quad F, G \vdash e_2:\mathbf{int}}{F, G \vdash e_1 \bmod e_2:\mathbf{int}}$$
- $$\frac{F, G \vdash e:\mathbf{ARRAY}(N, T)}{F, G \vdash e[x]:T} \; 0 \le val(x) < N$$
- $$\frac{F, G \vdash e:\mathbf{PTR}(T)}{F, G \vdash *e:T}$$

Ternary Choice Operator

Type → **char** | **int** | **bool** | *Type* [**num**] | & *Type*

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | * *Expn*
  | *Expn* **?** *Expn* **:** *Expn*

Ternary Choice Operator

Type → **char** | **int** | **bool** | *Type* [**num**] | & *Type*

Expn → **literal** | **num** | **id** | *Expn* mod *Expn* | *Expn* [*Expn*] | * *Expn*
| *Expn* ? *Expn* : *Expn*

- $$\frac{F, G \vdash e_1 : \mathbf{bool} \quad F, G \vdash e_2 : T \quad F, G \vdash e_3 : T'}{F, G \vdash e_1 ? e_2 : e_3 : T} \; T \equiv T'$$

Ternary Choice Operator

Type → **char** | **int** | **bool** | *Type* [**num**] | & *Type*

Expn → **literal** | **num** | **id** | *Expn* mod *Expn* | *Expn* [*Expn*] | * *Expn*
| *Expn* ? *Expn* : *Expn*

- $$\frac{F, G \vdash e_1 : \mathbf{bool} \quad F, G \vdash e_2 : T \quad F, G \vdash e_3 : T'}{F, G \vdash e_1 ? e_2 : e_3 : T} \; T \equiv T'$$

- $$\frac{F, G \vdash e_1 : \mathbf{bool} \quad F, G \vdash e_2 : T \quad F, G \vdash e_3 : T'}{F, G \vdash e_1 ? e_2 : e_3 : \mathbf{error}} \; T \not\equiv T'$$

Adding and Checking Statements

Program → *Decls* **;** *Stmt*

Decls → *Decls* **;** *Decls* | **id :** *Type*

Type → **char** | **int** | *Type* [**num**] | & *Type*

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | * *Expn*

Stmt → **id** = *Expn* | *Stmt* **;** *Stmt* | **if** *Expn* **then** *Stmt* | **while** *Expn* **do** *Stmt*

Adding and Checking Statements

Program → *Decls* **;** *Stmt*

Decls → *Decls* **;** *Decls* | **id :** *Type*

Type → **char** | **int** | *Type* [**num**] | & *Type*

Expn → **literal** | **num** | **id** | *Expn* `mod` *Expn* | *Expn* [*Expn*] | * *Expn*

Stmt → **id** = *Expn* | *Stmt* **;** *Stmt* | **if** *Expn* **then** *Stmt* | **while** *Expn* **do** *Stmt*

- $$\frac{F, G \vdash e : T'}{F, G \vdash x = e \text{ is valid}} \; x : T \in G, T \equiv T'$$

Adding and Checking Statements

Program → *Decls* **;** *Stmt*

Decls → *Decls* **;** *Decls* | **id :** *Type*

Type → **char** | **int** | *Type* [**num**] | & *Type*

Expn → **literal** | **num** | **id** | *Expn* mod *Expn* | *Expn* [*Expn*] | * *Expn*

Stmt → **id** = *Expn* | *Stmt* **;** *Stmt* | **if** *Expn* **then** *Stmt* | **while** *Expn* **do** *Stmt*

- $\dfrac{F, G \vdash e : T'}{F, G \vdash x = e \text{ is valid}} \; x : T \in G, T \equiv T'$
- $\dfrac{F, G \vdash e : T'}{F, G \vdash x = e \text{ is invalid}} \; x : T \in G, T \not\equiv T'$

Adding and Checking Statements

Program → *Decls* **;** *Stmt*

Decls → *Decls* **;** *Decls* | **id :** *Type*

Type → **char** | **int** | *Type* [**num**] | & *Type*

Expn → **literal** | **num** | **id** | *Expn* mod *Expn* | *Expn* [*Expn*] | * *Expn*

Stmt → **id** = *Expn* | *Stmt* **;** *Stmt* | **if** *Expn* **then** *Stmt* | **while** *Expn* **do** *Stmt*

- $\dfrac{F, G \vdash e : T'}{F, G \vdash x = e \text{ is valid}} \quad x : T \in G, T \equiv T'$
- $\dfrac{F, G \vdash e : T'}{F, G \vdash x = e \text{ is invalid}} \quad x : T \in G, T \not\equiv T'$

- [Effect of declaration] $\dfrac{F, (G, x : T) \vdash s \text{ is valid}}{F, G \vdash x : T ; s \text{ is valid}} \quad x \notin G$