Tipagem de comandos

- Os comandos TINY por si só não têm tipos, mas as regras de tipagem garantem que toda expressão usada dentro de um comando está consistente

$$\frac{T \vdash e : bool \quad T \vdash c_1 \quad T \vdash c_2}{T \vdash \text{if } e \text{ then } c_1 \text{ else } c_2}$$

$$\frac{T[\overline{v \to t}] \vdash c}{T \vdash \text{var } \overline{v:t}\ \overline{c}}$$

$$\frac{T(i) = t \quad T \vdash e : t}{T \vdash i := e}$$

$$\frac{T(i) = ref\ t \quad T \vdash e : t}{T \vdash i := e}$$

Tipagem de procedimentos

- Os procedimentos que colocamos em TINY não possuem parâmetros, então não faz sentido falar de verificação de tipos nas chamadas de procedimentos

- Mas e se colocamos parâmetros e retorno?

```
PROC  -> procedure id ( [DECLS] ) [: TIPO] CMDS end
CMD   -> id ( [EXPS] )
       | …
EXPS  -> EXPS , EXP
       | EXP
EXP   -> id ( [EXPS] )
       | ...
```

- Seguindo a linhagem Pascal, definimos o valor de retorno de um procedimento com uma atribuição para uma variável com o nome do procedimento

Tipagem de procedimentos – linhas gerais

- Como os procedimentos estão em um espaço de nomes separado das variáveis, seu contexto de tipagem também é diferente

- A ideia é representar o tipo de um procedimento como combinação dos tipos de seus parâmetros e do seu tipo de retorno

- A verificação da chamada checa o número de parâmetros, e o tipo de cada um versus o tipo dos argumentos

- A verificação do *corpo* do procedimento põe o tipo de cada parâmetro e da variável de retorno no ambiente de tipos de variáveis

Tipagem de procedimentos - regras

$$\frac{p(i2p) = (\bar{t_p}) \to t_r \quad \overline{r+e : t_a} \quad |\bar{F_p}| = |\bar{t_a}| \quad \overline{t_p \simeq t_a}}{T_1\ p \vdash i2p(\bar{e}) : b_r}$$

$$t_1 \simeq t_2 = \begin{cases} t_1 = t_2 \\ t_1 = real \wedge t_2 = int \end{cases}$$