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 tipo 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

$$\frac{\Gamma \vdash e : t_e \qquad t_e = procs[id].t_p \qquad procs[id].t_r = t_r}{\Gamma \vdash id(\overline{e}) : t_r}$$

$$\frac{\Gamma[id : t_r, \overline{p : t_p}] \vdash b}{\Gamma \vdash procedure\ id(\overline{p : t_p}) : t_r\ b}$$

Subtipagem

- O conjunto de valores de um tipo pode ser um subconjunto do conjunto de valores de outro tipo
- Podemos querer expressar isso no sistema de tipos através de uma *relação de subtipagem* $\leq$
- Em uma linguagem OO essa relação é declarada pelo programador; em Java ela é dada pelas cláusulas *extends* e *implements*
- A relação de subtipagem é *simétrica* ($t \leq t$) e *transitiva* ($r \leq s$ e $s \leq t$ implica $r \leq t$)
- Podemos usar a relação de subtipagem explicitamente nas regras, ou podemos introduzir uma *regra de subsunção*

Subtipagem explícita vs. subsunção

- Subtipagem explícita:

$$\frac{\Gamma \vdash e : t \quad t \leq T(id)}{\Gamma \vdash id := e}$$

- Subsunção:

$$\frac{\Gamma \vdash e : t_1 \quad t_1 \leq t_2}{\Gamma \vdash e : t_2}$$

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

- Usar subsunção deixa o sistema mais sintético, usar a subtipagem explícita deixa ele mais fácil de implementar

Classes

- Para mostrar como funciona a subtipagem, podemos adicionar classes com herança simples a Tiny*:*

```
s : classes ';' procs ';' cmds
  | classes ';' cmds
  | procs ';' cmds
  | cmds
  ;

classes : classes ';' classe
        | classe
        ;

classe : CLASS ID VAR decls END
       | CLASS ID VAR decls ';' procs END
       | CLASS ID procs END
       | CLASS ID END
       | CLASS ID ':' ID VAR decls END
       | CLASS ID ':' ID VAR decls ';' procs END
       | CLASS ID ':' ID procs END
       | CLASS ID ':' ID END
       ;
```

```
cmd : <outros>
    | rexp '.' ID '(' ')'
    | rexp '.' ID '(' exps ')'
    ;

exp : <outras>
    | rexp '.' ID '(' ')'
    | rexp '.' ID '(' exps ')'
    | NEW ID '(' exps ')'
    ;

rexp : <outras>
     | rexp '.' ID '(' ')'
     | rexp '.' ID '(' exps ')'
     ;
```