Registros

- Como aquecimento para geração de código para classes e métodos, vamos adicionar um tipo *record* a TINY similar a structs de C:

```
s    : regs ';' procs ';' cmds
     | regs ';' cmds
     | procs ';' cmds
     | cmds
     ;
regs : regs ';' reg
     | reg
     ;
reg : RECORD ID BEGIN decls END
    ;
```

```
tipo : REAL
     | INT
     | BOOL
     | ID
     ;
exp : <outras>
    | rexp '.' ID
    | NEW ID
    | NIL
    ;
rexp : rexp '.' ID
     | '(' exp ')'
     | ID
     ;
```

```
cmd : <outros>
    | rexp '.' ID ATTRIB exp
    ;
```

escrever campo
instancia (alocar)
do campo

- Note que o tipo do campo de um registro pode ser outro registro (ou mesmo o próprio, para um registro recursivo)

Subtipagem

- Os nomes de tipos registro estão em seu próprio espaço de nomes
- A ordem em que os campos estão definidos importa
- Um tipo registro reg1 é subtipo de outro tipo registro reg2 se:
  - O número de campos do primeiro de reg1 é maior ou igual ao de reg2,
  - Os tipos do i-ésimo campo de reg1 e do i-ésimo campo de reg2 são subtipo um do outro, para qualquer i
- Para evitar loops infinitos, mantemos em cada registro um conjunto de registros que já sabemos que são supertipos dele, ou que estamos assumindo que são

Análise de tipos

- Fora a subtipagem, os outros termos envolvendo registros têm uma análise bem direta:
  - O construtor de registros tem o tipo do registro que ele constrói
  - Uma indexação tem o tipo do campo correspondente, se ele existir
  - Uma escrita precisa ter o lado direito como subtipo do tipo do campo correspondente, se existir
  - O tipo de nil é nil, e nil é subtipo de todos os tipos registros (tipo *bottom*)

$\bot \leq t$

Geração de código

- Em tempo de execução, uma variável com um tipo registro na verdade é um *ponteiro* para o registro, que será alocado no *heap*
- O tamanho de um registro então é |campos|*4 bytes
- Vamos criar uma instrução **newrec *n*** na máquina de pilha, que recebe o número de campos do registro, aloca memória para ele (usando *malloc*, para simplificar) e empilha esse endereço
- A instrução **ldfld *i*** desempilha o endereço de um registro e empilha o valor do seu i-ésimo campo
- A instrução **stfld *i*** desempilha o valor a ser escrito e o endereço de um registro, e escreve esse valor em seu i-ésimo campo