

### Exercício:

Considerando a seguinte gramática e as respectivas ações semânticas para criação de uma tabela de símbolos hierárquica, construa a tabela de símbolos gerada para o trecho de código apresentado abaixo.

As ações semânticas contam com os seguintes dados:

- `entrada`: endereço inicial para a tabela de símbolos.
- `tabPtr`: pilha para armazenamento da referência ao contexto de símbolos em uso.
- `desloc`: pilha para contabilizar o tamanho da área de dados de um procedimento.

As ações semânticas exploram os seguintes serviços de biblioteca:

- `geraTab`: inicializa a tabela de símbolos de um contexto.
  - Parâmetro: endereço da tabela do contexto envolvente.
- `push`: coloca na pilha uma informação.
  - Parâmetros: o dado a ser empilhado e a pilha a ser manipulada.
- `pop`: retira o elemento no topo da pilha.
  - Parâmetros: a pilha a ser manipulada.
- `top`: retorna o elemento no topo da pilha.
  - Parâmetros: a pilha a ser lida.
- `adSimb`: adiciona o identificador de uma variável na tabela de símbolos.
  - Parâmetros: a tabela correspondente ao contexto onde o símbolo deve ser inserido, o identificador, seu tipo e posição de memória onde se encontra relativo ao início do contexto.
- `adProc`: adiciona o identificador de um procedimento na tabela de símbolos.
  - Parâmetros: a tabela correspondente ao contexto onde o símbolo deve ser inserido, o identificador e o endereço.

### Gramática:

```
P → M D          {      defTam( top(tabPtr), top(desloc) );
                    pop(tabPtr);
                    pop(desloc);      }
M → ε            {      t = geraTab(entrada);
                    push(t, tabPtr);
                    push(0, desloc);  }
D → D ; D
D → id : T       {      adSimb( top(tabPtr), id.nome, T.tipo, top(desloc) );
                    top(desloc) = top(desloc) + T.tam;      }
D → proc id ; N D ; S {      t = top( tabPtr );
                    defTam( t, top(desloc) );
                    pop( tabPtr );
                    pop( desloc );
                    adProc( top(tabPtr), id.nome, t );      }
N → ε            {      t = geraTab( top(tabPtr) );
                    push( t, tabPtr );
                    push( 0, desloc);  }
T → int          {      T.tipo = int; T.tam = 4;      }
T → real         {      T.tipo = real; T.tam = 8;      }
T → array [num] of T1 {      T.tipo = arranjo(num.val, T1.tipo);
                    T.tam = num.val* T1.tam;      }
T → ^T1        {      T.tipo = ponteiro(T1.tipo);
                    T.tam = 4;      }
```

### Programa fonte:

```
a : real;
b : int;
proc p1;
  c : real;
  ...
end p1;
proc p2;
  d: array[5] of int;
  proc p3;
    e, f : real;
    ...
  end p3;
  ...
end p2;
...
```