

Scope

①

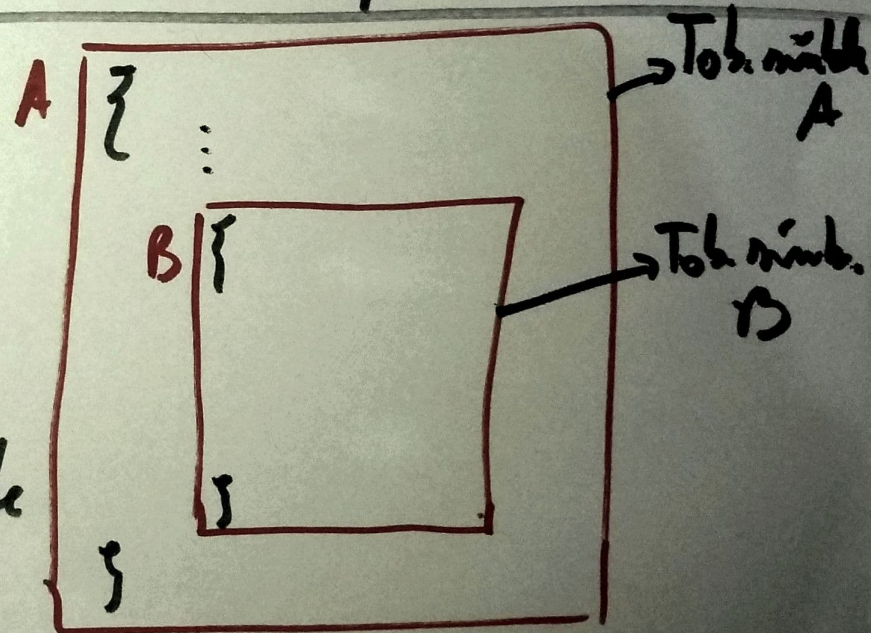
- * Sempre existirá um scope global
- * Exceto pelo scope global, um scope não é definido pelos símbolos {...}
- * Uma variável deverá sempre ser declarada antes de ser usada.

```
{  
  x = x + 1;  
  int x;  
}
```

=> erro remanêcia deve ser reportado

- * Escopos podem ser aninhados

* Sugestão de implementação: uma tabela de símbolos por scope definindo ponteiros aninhados de tabelas.



* Não deve ser permitida a declaração de dois identificadores ② com o mesmo nome em um mesmo bloco.

```
{ int x;  
  :  
  int x;  
}
```

=> erro semântico deve ser reportado.

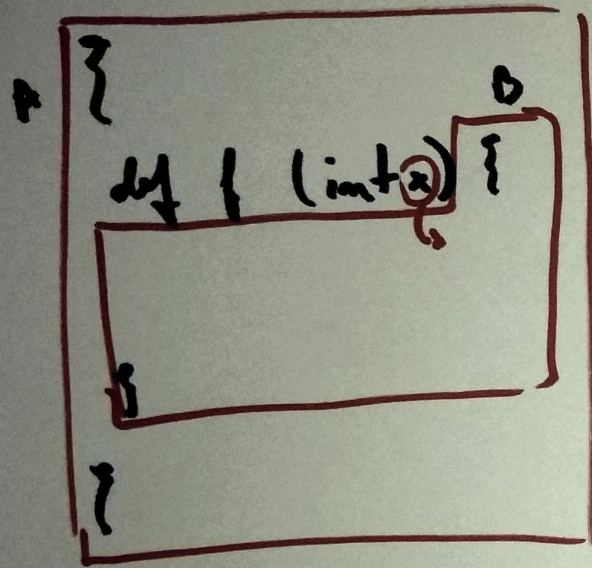
* Na tabela de símbolos, cada identificador deverá armazenar a linha de sua declaração, o tipo do identificador e uma variável booleana indicando se o identificador é nome de função.

```
linha 57 { int x;  
          {  
            x = x + 1;  
          }  
linha 63  
linha 97 }
```

=> OK!

Na m. x é declarado
na linha 57, um erro
semântico deve ser reportado.

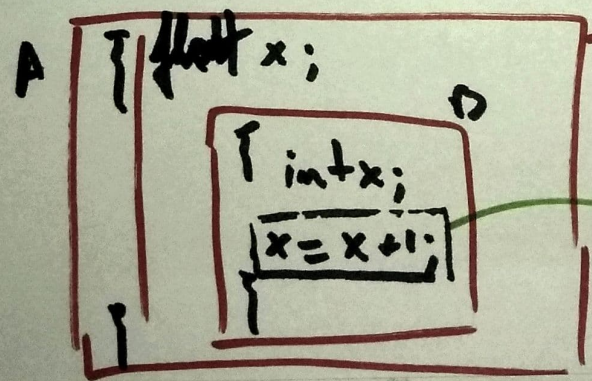
* Parâmetros de função devem pertencer ao scope da função.



Tab. símbolos B

x	

* Poderá ocorrer identificadores com mesmo nome em tabelas símbolos. A tabela mais "próxima" de um identificador deverá sempre ser considerada para este identificador.



→ A tab. de simb. A contém também o símbolo x.

→ Consideramos a tabela de símbolos mais próxima para as ocorrências de x (Tab. simb. B).

* Cada tabela de símbolos deverá conter ~~os~~ a uma
tabela de símbolos externa imediata (se existir) (4)

* Cada funç. terá o seu próprio scope e os parâmetros sempre
são considerados parâmetros por referência
↳ não nomes implementados!