

# Compiladores

## Analizador Semântico

José Luis Seixas Junior

# Índice

- Escopo;
- Expressões;
- Casting;
- Outros;



# Análise Semântica

- Erros não detectados sintaticamente:
  - Necessitam de contexto;
  - Quais variáveis estão no escopo?
  - Tipos?
- Todos os nomes usados foram declarados?
- Nomes não são declarados mais de uma vez?
- Tipos das operações são consistentes?

# Escopo

- Escopo é o trecho de código onde a variável está visível.
- Consequentemente, onde a variável poderá ser usada;
- Quem determina se poderá ser usada é o analisador semântico;

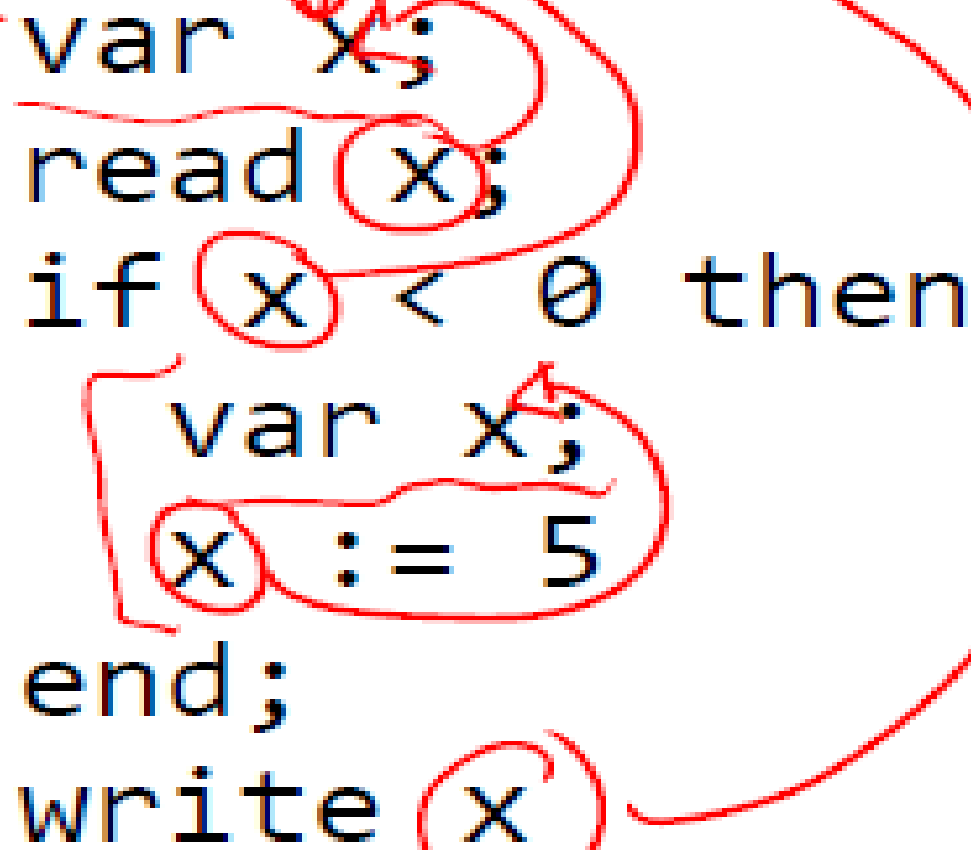


# Escopo

- Uso de um nome na declaração vigente:

```
procedure funcao1(a: integer);  
begin  
    write(a);  
end;  
  
procedure funcao2(a: varchar);  
begin  
    write(a);  
end;
```

# Escopo



```
var x;  
read x;  
if x < 0 then  
  var x;  
  x := 5  
end;  
write x
```

The diagram illustrates variable scope using red annotations on the code. A large red bracket on the left groups the entire code block. A red circle highlights the first 'x' in 'var x;'. A red arrow points from this circle to the 'x' in 'read x;'. Another red circle highlights the 'x' in the condition 'x < 0'. A red arrow points from this circle to the 'x' in the inner 'var x;'. A red circle highlights the 'x' in 'x := 5'. A red arrow points from this circle to the 'x' in 'write x'. A final red circle highlights the 'x' in 'write x'. A large red arrow originates from the first 'x' and points to the 'x' in 'write x', indicating the scope of the first variable declaration.

# Escopo

- Tabela de Símbolos Encadeada;
- Tabela que mapeia um nome a algum atributo:
  - Tipo e armazenamento em tempo de execução;
- Cada tabela corresponde a um escopo e ligadas com a tabela responsável;
- Insere as declarações nas tabelas e usa as tabelas como análise de escopo.



# Escopo

- Mutuamente Recursivo:

```
procedure par()  
  if 0 < n then  
    n := n - 1;  
    impar()  
  else  
    res := 1  
  end  
end;
```

```
procedure impar()  
  if 0 < n then  
    n := n - 1;  
    par()  
  else  
    res := 0  
  end  
end;
```



# Escopo

- Declaração duplicada:
  - Se a declaração aconteceu mais de uma vez;
  - Se a variável é usada em nível inferior, sem um novo bloco;
- Declaração efetuada:
  - Se a declaração foi feita alguma vez;
  - Se existe um bloco interno com declaração querendo sobrescrever uma variável sem um novo bloco;

# Escopos

- Variáveis;
- Procedimentos;
- Funções;
- Tipo:
  - Se existe;
- Campos:
  - Se determinada variável é um array;



# Variáveis e Campos

- Respeita a hierarquia de funções;
  - Uma função pode usar uma variável de declaração em nível superior;
- Em caso de classes:
  - Visibilidade com subclasses e pacotes;
  - *Extends, implements*;
  - Métodos necessitam sobrescrever os das classes superiores;

# Métodos

- Não podem ser declarados no mesmo nível de implementação;
- Mas aceitam declarações sobre níveis mais baixos dentro de uma única função;
- A assinatura do método é seu tipo, nome, parâmetros, na ordem que aparecem.



# Expressões

- Atribuições:
  - Dependem da resultante de tipo;
  - Variável que será atribuída é do mesmo tipo da que terá a atribuição;
- Funções matemáticas:
  - Adição;
  - Subtração;
  - Divisão;
  - Multiplicação;

# Expressões

- Também deve servir para inequações:
  - Maior;
  - Menor;
  - Maior igual;
  - Menor igual;
  - Diferente;
- Resultado:
  - O retorno da função é o mesmo de atribuição;



# Expressões

- Entre “if” e “then”:
  - O retorno da expressão é booleana?
- while;
- not;
- and/or;
- array → Deve ter o mesmo tipo durante toda a sua inserção;

# Casting

- Automático:
  - `char c = 'a';`
  - `int a = c;`
- Não automático:
  - `char c = 'a';`
  - `int a = (int) c;`



# Casting

- *char* e *short* são convertidos para *int*, *float* para *double*;
- Se um dos operandos é *double*, o outro é convertido para *double* e o resultado é *double*;
- Se um dos operandos é *long*, o outro é convertido para *long* e o resultado é *long*;
- Se um dos operandos é *unsigned*, o outro é convertido para *unsigned* e o resultado é *unsigned*;
- Se não, todos os operandos são *int* e o resultado é *int*.

# Outros

- *goto* depende de *label* ou espaço de linha;
- *case* duplicado;
- tipo deve conter atributo;





Obrigado.

Dúvidas?