Construção de Compiladores
Aula 5 - Variáveis

Bruno Müller Junior

Departamento de Informática
UFPR

26 de Agosto de 2014

Declaração de Variáveis

- O escopo das variáveis da linguagem Pascal seguem um modelo bem definido, porém nem sempre intuitivo.
- Por esta razão, antes de explicar como implementar a declaração e uso de variáveis, iremos primeiro explicar questões específicas da própria linguagem.

Escopo das Variáveis

- O programa abaixo tem duas variáveis globais.

```
program declaraVars (input, output);
    var a, b:  integer
    begin
      a:=a+b;
    end.
```

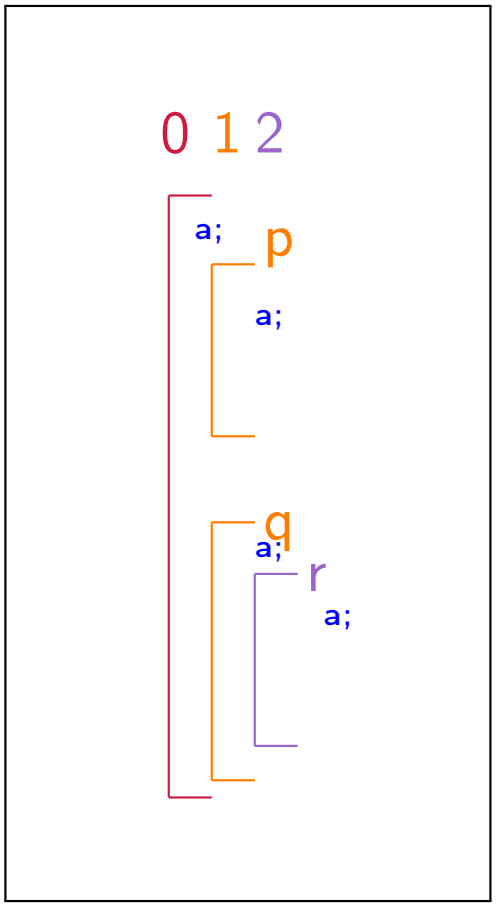
- Uma forma de acessá-las seria utilizar instruções que referenciam diretamente o nome de cada variável.
- Porém, isto inviabilizaria o acesso a variáveis de procedimentos e funções: em uma chamada recursiva, como fazer?

Exemplo de Programa

```
program variaveis (input, output);
var a, b:  integer
    procedure p(x,y:integer);
    var a:integer
    begin
        ...
    end
    procedure q(a:integer);
        procedure r(a:integer);
        begin
        ...
        end
    begin
        ...
    end
begin
    a:=a+b;
end.
```

- Neste programa, as chaves indicam o escopo das variáveis.
- Por exemplo, as variáveis “a” e “b” são visíveis em todos os lugares, porém nem sempre o mesmo “a”.

Esquema



instâncias da variável “a”.

- Cada instância está em um nível léxico diferente.
- A idéia é usar isto em nosso favor: colocar o nível léxico como parte do endereço e cada variável.
- Observe que quando um procedimento é chamado, o espaço para as variáveis tem de ser criado. Quando sai do procedimento, este

Instruções Novas

- O acesso às variável na MEPA utiliza dois parâmetros: nível léxico e deslocamento.
- O espaço para as variáveis é criado na pilha: basta acrescentar espaço usando o registrador s da MEPA.

Instrução	Ação	Significado
AMEM m	`s:=s+m;`	Aloca Memória
DMEM m	`s:=s-m;`	Desaloca Memória
CRVL k,n	`s:=s+1;` `M[s]:=M[D[k]+n];`	Carrega Valor
ARMZ k,n	`M[D[k]+n]:=M[s];` `s:=s-1`	Armazena Valor

Implementação na MEPA

- A idéia é usar o Vetor de Registradores de Base (D) como apontador do registro de ativação corrente daquele nível.
- Assim, `D[0]` aponta para o R.A. ativo de nível 0, `D[1]` para o R.A. de nível 1, ... , `D[k]` para o R.A. de nível k.

Exemplo de Tradução

- Traduza o programa Pascal abaixo:

```
program varsGlobais (input,
output);
    var a, b:  integer
    begin
        a:=0;
        b:=a+10;
    end.
```

```
    INPP
    AMEM 2
    CRCT 0
    ARMZ 0,0
    CRVL 0,0
    CRCT 10
    SOMA
    ARMZ 0,1
    DMEM 2
    PARA
```

Simulação da Execução na MEPA

- Na MEPA, utiliza-se o vetor de registradores para indicar o R.A. "ativo" de cada nível léxico.
- `D[k]` aponta para a primeira variável local do R.A.
- Isto explica melhor as operações do tipo `M[D[k]+n]`.
- No programa do próximo slide, considere
  - onde cada variável é "viva" (código estático).
  - o uso do vetor `D[]` para apontar as variáveis na sequência de chamadas

$$Principal \rightarrow q \rightarrow r \rightarrow r \rightarrow p$$

Simulação da Execução na MEPA

```
program variaveis (input, output);
var a, b:  integer
    procedure p(x,y:integer);
    var a:integer
    begin
        ...
    end
    procedure q(a:integer);
        procedure r(a:integer);
        begin
        ... r; ... p; ...
        end
    begin
        ... r ...
    end
begin
    ... q ...
end.
```

Implementação do Compilador

- Todos os tokens reconhecidos como identificadores são chamados *símbolos*.
- Símbolos válidos são aqueles que já foram declarados (por exemplo na declaração “var”;
- Quando entra em um procedimento, os símbolos lá declarados são “validados”, e quando sai devem ser “invalidados”.
- Confira novamente no programa do slide anterior.
- Observe que a “validação” e a “invalidação” de símbolos segue a estrutura de uma pilha.

Tabela de Símbolos

- A tabela de símbolos é uma estrutura de dados de um compilador que armazena os símbolos "válidos" para uso naquele momento.
- Se um identificador não está presente na tabela de símbolos, então isto significa que não foi declarado.
- As operações básicas são:

  Insere (ident, atributos) Insere o identificador indicado na TS, assim como seus atributos;

  Busca(ident) Retorna a entrada (os atributos) da TS associados ao ident procurado.

  Retira(n) Retira as últimas *n* entradas da TS.
- Considere o programa anterior para exemplificar.

Categorias

- Os atributos que a T.S. deve armazenar dependem do símbolo, ou melhor, da categoria a que o símbolo pertence.
- O livro do Tomasz (cap.10) descreve as categorias de símbolos que serão utilizadas.
- Os atributos são aqueles que são utilizados, por exemplo na geração de código.

  Variável Simples {nível léxico, tipo, deslocamento}
  Procedimento {...}
  Função {...}
  Parâmetro Formal {...}
  Rótulo {...}

Projeto

- Consulte o capítulo 10 do livro do Tomasz, em especial no que ele referencia a TS. Atenção às sugestões de implementação.
- Implemente um Tipo Abstrato de Dados "Tabela de Símbolos" (TS).
- Ela deve funcionar como uma pilha na inserção e na remoção.
- A busca deve procurar do **último** para o `primeiro`.
- A remoção simplesmente altera o topo da pilha.