

Linguagens de Programação

Nomes, Escopos e Vinculações (Bindings)

Carlos Bazilio

bazilio@ic.uff.br

<http://www.ic.uff.br/~bazilio/cursos/lp>

Nomes

- Forma mais simples de abstração
- Um nome é um caracter mnemônico que representa alguma coisa:
 - Um endereço de memória
 - Um valor, uma constante
 - Uma operação
 - Etc
- Se considerarmos o tempo, o período de existência desta associação entre nome e valor é chamado de tempo de vinculação (*binding time*)

Tipos de Vinculação de Tempo

- Tempo de projeto de linguagem
- Tempo de implementação da linguagem
- Tempo de escrita do programa
- Tempo de compilação
- Tempo de ligação
- Tempo de carga
- Tempo de execução
- Usualmente os termos estático e dinâmico se referem antes e durante o tempo de execução

Tempo de Vida de um Objeto e Armazenamento

- Eventos que sinalizam o tempo de vida de um objeto:
 - Criação de objetos
 - Criação de associações
 - Referências à variáveis, rotinas, tipos, etc
 - Remoção de associações
 - Destruição de associações
 - Destruição de objetos
- Exemplos fartos para isso são encontrados na manipulação de ponteiros

Tempo de Vida de um Objeto e Armazenamento

- O tempo de vida de um objeto geralmente está associado a 3 mecanismos distintos de armazenamento
 - Objetos estáticos
 - Objetos armazenados numa pilha
 - Objetos armazenados num heap

Objetos Estáticos

- Exemplo imediato são variáveis globais
- Ou seja, possuem a característica de existirem durante toda a execução de um programa
- Instruções de um programa também podem ser consideradas objetos estáticos
- Outros exemplos são:
 - Variáveis estáticas declaradas em rotinas
 - Valores numéricos e strings
- Compiladores também geram estruturas auxiliares para auxílio na compilação e execução

Objetos Estáticos

- Em linguagens com alocação puramente estática, o compilador previamente reserva espaço para ativação de cada rotina existente
- Dados comumente armazenados nestes espaços (registros de ativação)
 - Argumentos e valores de retorno
 - Endereço de retorno
 - Variáveis locais
 - Temporárias, etc
- Com isso, não é possível haver recursão, seja esta direta ou indireta

Objetos armazenados numa Pilha

- O empilhamento de registros de ativação permite que a recursão possa ocorrer
- Entretanto, naturalmente, devemos abdicar da reserva estática de espaço para rotinas e fazermos alocação dinâmica destas
- É de responsabilidade das rotinas chamadora/chamada manter a pilha coerente com estado atual de execução
- A decisão de quem faz o que é questão de implementação da linguagem

Objetos armazenados numa Pilha

- Os registros de ativação são também chamados de *frames*
- Da mesma maneira que temos o registrado PC (*Program Counter*) que indica a próxima instrução a ser executada, temos um outro chamado FP (*Frame Pointer*) que aponta para o endereço base do registro de ativação da rotina corrente
- Os acessos aos campos de cada frame podem ser feitos somando incrementos (*offsets*) a este ponteiro

Objetos armazenados numa Pilha

- Outro ponteiro comum neste contexto é o SP (*Stack Pointer*) que sempre aponta para o primeiro endereço livre no topo da pilha (ou o endereço do último frame alocado, dependendo da máquina)

Objetos armazenado num Heap

- Heap é uma área de armazenamento onde blocos de memória podem ser alocados/desalocados muitas vezes
- **Não há relação** com o nome heap da implementação de fila de prioridades baseada em árvore
- É necessário para alocação de estruturas de dados dinâmicas como listas, filas, objetos redimensionáveis dinamicamente, etc

Objetos armazenado num Heap

- Há várias estratégias de manipulação de espaço num heap
- Estas estratégias levam em consideração 2 fatores: velocidade e espaço
- Dois problemas a serem tratados:
 - Fragmentação Externa
 - Fragmentação Interna
- Algoritmos de alocação no heap, visando amenizar o problema de fragmentação externa:
 - First fit, Best fit, Worst fit

Coleta Automática de Lixo

- Mecanismo pelo qual áreas de memória não mais utilizadas no heap são devolvidas de forma transparente (sem o auxílio do programador para o S.O.)
- 2 algoritmos representam boa parte das implementações de coletores de lixos:
 - Contagem de referências e *Mark-Sweep* (Marca e Limpa)

Coleta de Lixo

- O algoritmo de contagem de referências é uma técnica básica para implementação da coleta de lixo
- Neste, um contador de referências é mantido para cada área de memória alocada no heap
- A cada alteração do número de referências a uma dada área, o contador é atualizado
- Coletores com estas características costumam ser classificados como gulosos

Coleta de Lixo

- O algoritmo *mark-sweep* pode ser dividido em 3 partes:
 - O coletor marca todas as células como lixo
 - Em seguida, todo ponteiro para o heap é percorrido e as células para os quais estes apontam são marcados como áreas úteis
 - Finalmente, todas as áreas ainda marcadas como lixo são devolvidas para a lista de áreas livres
- Este processo só se inicia após o esgotamento do heap
- Algoritmos deste tipo são chamados de *lazy*

Escopo

- 2 divisões básicas:
 - Estático (Léxico): refere-se à estrutura do programa; ou seja, a visibilidade dos objetos pode ser definida em tempo de compilação
 - Dinâmico: a visibilidade depende da ordem de execução das instruções num programa

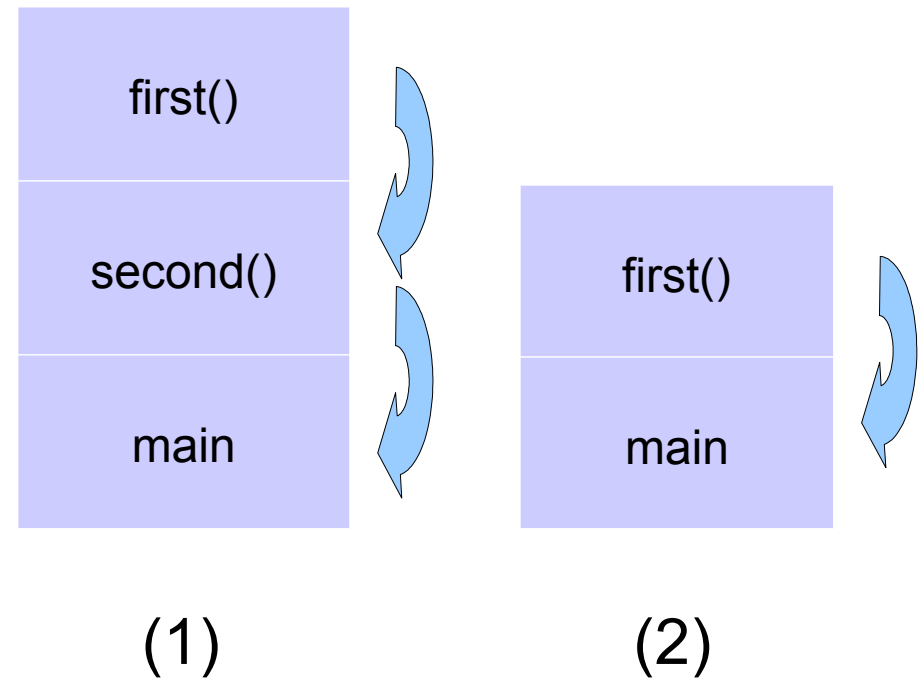
Escopo

```
a: integer

procedure first
  a := 1

procedure second
  a: integer
  first()

a := 2
if read_integer() > 0
  second()      // (1)
else
  first()       // (2)
write_integer(a)
```



Escopo Perl

- Perl permite tanto tratamento estático quanto dinâmico no acesso à variáveis
- A diferenciação é feita pelos modificadores *my* (estático) e *local* (dinâmico)

```
$x = 0;  
sub f { return $x; }  
sub g { my $x = 1; return f(); }  
print g()."\n";
```

```
$x = 0;  
sub f { return $x; }  
sub g { local $x = 1; return f(); }  
print g()."\n";
```

Escopo

```
type tpessoa = record
  idade : integer
limite : integer
pessoas : collection
```

```
function mais_antigo (p : tpessoa) : boolean
  return p.idade >= limite
```

```
procedure imprime_pessoa (p : tpessoa)
  // Chamar rotinas apropriadas de I/O
  // Fazer uso da variável não local tamanho_linha
```

```
procedure imprime_registros_selecionados
  (bd : collection, predicado, rotina_impressao : procedure)
  tamanho_linha : integer
  if device_type(stdout) = terminal
    tamanho_linha = 80
  else
    tamanho_linha = 132
  foreach record r in bd
    if predicado(r)
      rotina_impressao(r)
```

```
limite := 35
```

```
imprime_registros_selecionados(pessoas, mais_antigo, imprime_pessoa)
```