Remoção de Recursão com Pilhas

Daniel Oliveira

Departamento de Informática - UFPR

Fevereiro 2021

Remoção de Recursão - Ideia Geral

- A ideia de pilha de chamadas é bem simples:
  - Ao chamar uma função: empilha as informações necessárias para executá-la
  - Ao retornar de uma função: desempilha as informações, retoma função que está no topo atual da pilha

- Um padrão que muitas vezes funciona:
  - Chamadas recursivas são substituídas por 'empilha'
  - Retorno de funções são substituídos por 'desempilha'
  - A computação principal é feita dentro de um laço de repetição até a pilha ficar vazia

Exemplo - Quicksort

```
Quicksort(v, a, b)
Se a ≥ b
    Devolva v
Senão
    m ← Particiona(v, a, b, v[b])
    Quicksort(v, a, m − 1)
    Quicksort(v, m + 1, b)
```

```
Quicksort_i(v, a, b)
iniciaPilha(p)
empilha(p, a, b)
Enquanto p não está vazia
    a, b ← desempilha(p)
    Se a < b
        m ← Particiona(v, a, b, v[b])
        empilha(p, a, m − 1)
        empilha(p, m + 1, b)
```

Pegadinhas - Ordem de execução

- Múltiplas chamadas recursivas serão executadas de forma reversa na versão iterativa
  - caso a ordem de execução seja importante, devemos empilhar na ordem inversa que acontece as chamadas recursivas

Funcao(a, b)

```
...
Funcao(a + 1, b)
Funcao(a, b + 1)
```

Funcao_i1(a, b)

```
...
Enquanto p não está vazia
    a, b ← desempilha(p)
    ...
    empilha(p, a + 1, b)
    empilha(p, a, b + 1)
```

Funcao_i2(a, b)

```
...
Enquanto p não está vazia
    a, b ← desempilha(p)
    ...
    empilha(p, a, b + 1)
    empilha(p, a + 1, b)
```

Pegadinhas - Continuidade

- Ao empilhar os dados de uma nova chamada recursiva a execução atual deve parar
  - Operações após as chamadas recursivas devem ser feitas apenas ao final dessa chamada
  - Emular esse efeito torna o código difícil de ler
    - necessário armazenar qual linha deve ser executada

Contador(n)

```
Se n = 1
    Imprime n
Senão
    Contador(n − 1)
    Imprime n
```

Contador(n)

```
iniciaPilha(p)
empilha(p, n)
Enquanto p não está vazia
    n ← desempilha(p)
    Se n = 1
        Imprime n
    Senão
        empilha(p, n − 1)
        Imprime n
```

Pegadinhas - Continuidade

- Podemos usar a estrutura *Switch* ou *GOTO* para pular a linha correta
  - emulando mais precisamente a pilha de chamadas do sistema

Contador(n)

```
iniciaPilha(p)
empilha(p, n)
Enquanto p não está vazia
    n ← desempilha(p)
    Se n = 1
        Imprime n
    Senão
        empilha(p, n − 1)
        Imprime n
```

Contador(n)

```
iniciaPilha(p)
empilha(p, n, L1)
Enquanto p não está vazia
    n, linha ← desempilha(p)
    GOTO linha
    L1:
    Se n = 1
        Imprime n
    Senão
        empilha(p, n, L2)
        empilha(p, n − 1, L1)
        continua
        L2:
        Imprime n
```

Remoção de Recursão - Resumo

- Remover recursão pode resultar em ganho de desempenho
- Algumas linguagens/dispositivos não permitem recursão
- Para remover recursão fazemos:
  1. Se recursão cauda, tradução direta para iterativo
  2. Caso contrário, tentamos transformar para recursão de cauda e voltamos ao passo 1
  3. Caso não for possível transformar em recursão de cauda, usamos pilha para emular a pilha do sistema
- Algumas recursões simples se tornam extremamente complicadas na forma iterativa
  - 'Debugar' e manter o código se torna um trabalho árduo