

Linguagens de Programação

Fluxo de Controle

Carlos Bazilio

bazilio@ic.uff.br

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

Fluxo de Controle

- Mecanismos utilizados para especificar ordem de execução de instruções recaem em 8 principais categorias:
 - Sequenciamento: usualmente, a ordem no código
 - Seleção: instruções condicionais
 - Iteração: repetição de um conjunto de instruções
 - Abstração procedural: procedimentos e funções
 - Recursão: rotinas que se auto-referenciam
 - Concorrência: 2 ou mais fragmentos de um programa que executam ao mesmo tempo

Fluxo de Controle

- ... (mais categorias):
 - Tratamento de exceção: recurso pelo qual um trecho de código pode ser controlado para, no caso de uma falha, um tratador adequado possa ser acionado
 - Não-determinismo: a ordem de execução das instruções é indeterminada
- Cada linguagem disponibiliza construções que implementam 1 ou + destas categorias
- A importância de cada categoria pode variar em função do paradigma da linguagem adotada

Goto

- Instrução mais básica de desvio
- Debatida profundamente no passado por não conter uma construção estruturante
- Disponível em linguagens como o C e Pascal
- Tem como “desvantagem” principal o fato de permitir desvios entre instruções de escopos/aninhamentos distintos, pois:
 - Dificulta legibilidade
 - Aumenta esforço do compilador para tornar a execução consistente

Goto

Exemplo em Pascal

```
function busca(chave: string) : string;
var s: string; ...
  procedure busca_arquivo(fnome: string);
  begin ...
    for ... (* itera sobre o conjunto de linhas *)
      ...
      if achou(chave, linha) then
        begin
          s := linha;
          goto 100;
        end;
      ...
    end;
  begin (* busca *) ...
    for ... (* itera sobre o conjunto de arquivos *)
      ...
      busca_arquivo(fnome);
      ...
100: return s;
end;
```

Seleção

- A construção base para instruções deste tipo é o if-then-else
- Algumas variações:
 - *elseif* <condição> : permite que os ramos else sejam conjugados com outra condição
 - *case/switch* : alternativa ao aninhamento de várias instruções de seleção

Iteração

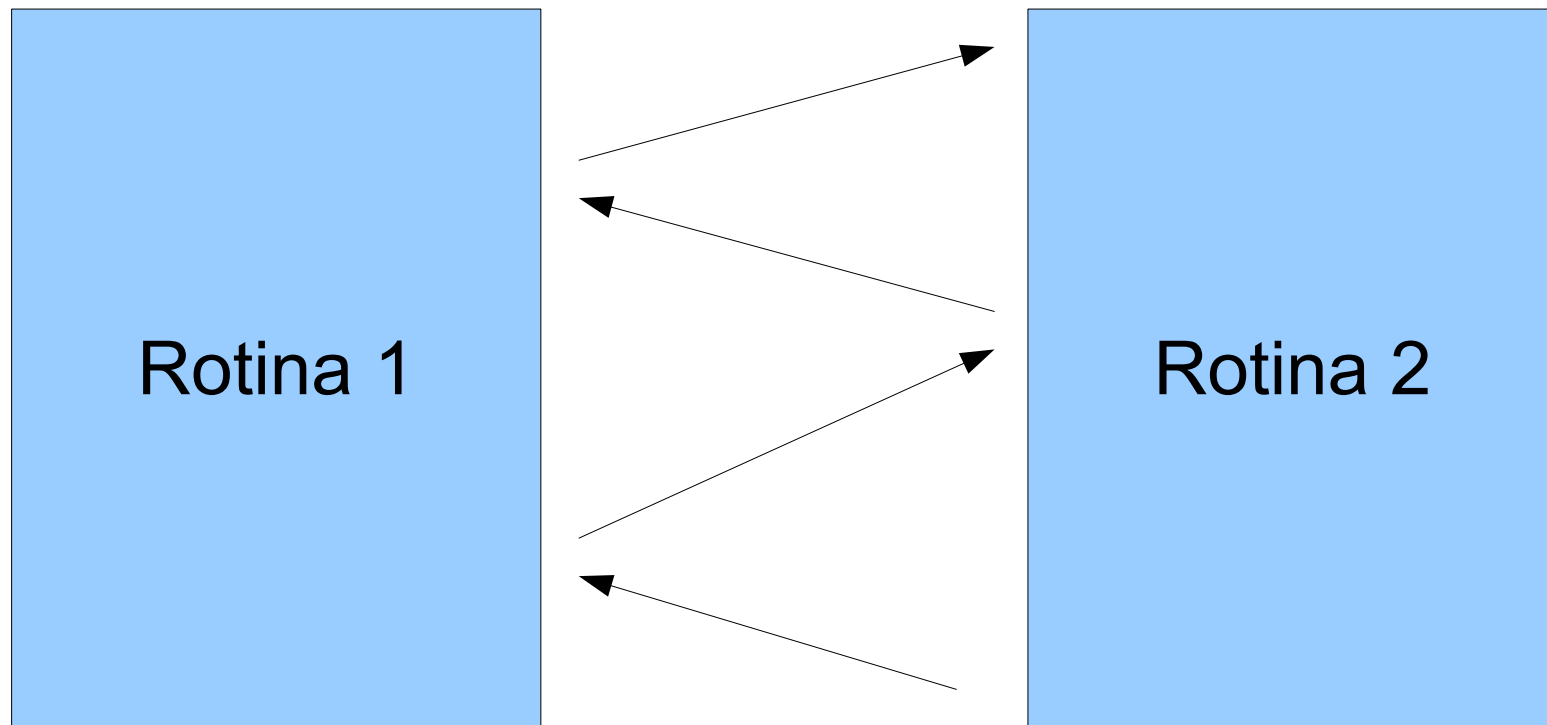
- Recurso para se repetir a execução de um conjunto de instruções
- As construções básicas são o *for* (repetição definida), *while* (repetição indefinida) e a chamada recursiva de funções
- Existem inúmeras variações destas construções; o *for* de C é um bom exemplo:
 - `for (i=0; i<=10; i++) {}`
 - `for (i=0,j=0; (i+j)<=100; i+=2, j++) {}`
 - `for (;;) {}`

Iteração

- Iterações também são utilizadas em enumerações (faixas de valores)
 - for i:=0 to 10 by 2 do ...
 - for c:='a' to 'z' do ...
- Iteradores são construções oferecidas por diversas linguagens que permitem uma iteração mais genérica
 - cidades = [“petropolis”, “riodasostras”, “macae”, “niteroi”]
 - for c in cidades do ...

Co-rotina

- Construção que permite a cooperação entre 2 rotinas para a execução de uma tarefa



Co-rotina

- A Rotina 1 poderia representar o for, enquanto que a Rotina 2 representa o iterador
- O iterador retorna os elementos, os quais podem estar armazenados de forma distinta:
 - Um vetor de strings
 - Um arquivo
 - Uma lista encadeada
- Ou seja, o iterador faz com que o programa abstraia da forma como um conjunto de valores é armazenado e percorrido

Recursão

- Funções recursivas se caracterizam por funções que se auto-referenciam

```
(define fatorial  
  (lambda (n)  
    (if (= n 0)  
        1  
        (* n (fatorial (- n 1))))))
```

- Usualmente, as implementações recursivas tendem a ser mais custosas que as iterativas
- Este custo pode ser amenizado com a implementação de recursões “em cauda”

Recursão em Cauda

- Este tipo de recursão se caracteriza pela computação dos valores ao longo da chamada da função, e não do retorno

```
(define fatorial
  (lambda (n)
    (fatorial_aux n 1)))
(define fatorial_aux
  (lambda (n acum)
    (if (= n 0)
        acum
        (fatorial_aux (- n 1) (* acum n)))))
```

- Desta maneira, ao final das chamadas recursivas, o valor a ser computado já está produzido, diferente da versão anterior