Comandos e estruturas de controle

Prof. Hugo de Paula

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Departamento de Ciência da Computação

Sumário

Colateralidade

- Comandos sequenciais: C1; C2;
- Comandos colaterais:
  - C1, C2 são executados, sem uma ordem.
  - Disponíveis nas linguagens concorrentes
  - `n := 7, n := n + 1;` suponha n = 0
    - C1 seguido de C2 -> **n = 8**
    - C2 seguido de C1 -> **n = 7**
    - C1 avaliado entre **n+1** e sua atribuição a **n** -> **n = 1**

Estilos de computação

- Computação determinista.
  - determina-se *a priori* a sequencia de comandos que serão executados
- Computação não-determinista.
  - Não determina *a priori* a ordem de execução dos comandos.
  - Computação efetivamente determinista.
    - não determinista, tem efeito previsível
- Comandos colaterais
  - não deterministas
  - efetivamente determinista
    - nenhum comando inspeciona variáveis atualizadas por outro
  - m := 10, **write**(m);
  - m := m + 1, n := n + 2;

Estruturas de controle

Estrutura de controle

Uma estrutura de controle é uma instrução de controle e os comandos que ela controla.

Questões de projeto:

- Uma estrutura de controle pode ter múltiplas entradas?

Estruturas de seleção

Instrução de seleção

Provê os meios para se escolher entre dois ou mais caminhos de execução.

- Estruturas podem ser de **dois caminhos** (*two-way*) ou de **múltiplos caminhos** (*multiple-way*).

Seleção de dois caminhos

- Escolha de sub-comandos (condicionais): SE (**if**).

Forma geral da estrutura condicional de dois caminhos

```
SE <expressão_de_controle>
ENTÃO <cláusula>
SENÃO <cláusula>
```

- Comando condicional produz computação não determinista.

Exemplo em PASCAL

```
if (x >= y)
then max := x
else max := y
```

Seleção de dois caminhos: projeto

- Qual o tipo e formato da expressão de controle?
- Como as instruções “ENTÃO” E “SENÃO” são especificadas?
- Qual deve ser o significado de seletores aninhados?

Seleção de dois caminhos: Expressão de controle

- Se linguagem não possui palavra reservada para a cláusula “ENTÃO”, expressão de controle vem entre parêntesis.
- Linguagens como o C99, Python e C++, expressão de controle pode ser aritmética.
- Na maioria das linguagens, expressão de controle deve ser **boolean**.
- Cláusulas podem ser simples ou blocos de comandos.

Seleção de dois caminhos: Expressão de controle

- Python utiliza indentação para definir cláusulas:

Exemplo em Python

```
if x > y:
    print("X é maior que Y")
    x = y
else:
    print("X não é maior que Y")
```

Seleção de dois caminhos aninhada

Exemplo em Java

```
if (soma == 0)
    if (contador == 0)
        resultado = 0;
else resultado = 1;
```

- A qual instrução do **if** o comando **else** está relacionado?
- Java utiliza regra semântica estática: **else** está sempre casado com o **if** mais próximo.
- Usar chaves para alternar a semântica: C, C++ e C#.

Seleção de dois caminhos aninhada

Exemplo em Java

```
if (soma == 0) {
    if (contador == 0)
        resultado = 0;
}
else resultado = 1;
```

Exemplo em Python

```
if soma == 0:
    if contador == 0:
        resultado = 0
else:
    resultado = 1
```

Expressões seletoras

- Em linguagens funcionais, como o Lisp e F#, seletor é uma expressão.

Exemplo em Lisp

```
let y = if x > 0 then x else 2 * x
```

- Se expressão **if** produz valor, então deve haver uma cláusula **else**.
- Tipos dos valores retornados deve ser o mesmo.

Operador ternário condicional em C

```
y = (x > 0) ? x : 2 * x;
```

Seleção de múltiplos caminhos

- Permite a seleção de um dentre várias cláusulas ou blocos de comando.
- Questões de projeto:
  1. Qual o tipo e formato da expressão de controle?
  2. Como os segmentos selecionáveis são especificados?
  3. O fluxo está condicionado à execução de apenas um segmento?
  4. Como os valores de cada caso são especificados?
  5. O que é feito para valores não representados?

Seleção de múltiplos caminhos: exemplos

Exemplo em Java, C, C++, JavaScript

```
switch (expressao_de_controle) {
    case expressao_constante_1: instruções_1;
    ...
    case expressao_constante_n: instruções_n;
    [default: instrucoes_n+1]
}
```

Seleção de múltiplos caminhos: projeto

- Decisões de projeto da linguagem C:
  1. Expressões de controle só podem ser de tipos inteiros.
  2. Segmentos selecionáveis podem ser sequências de comandos, blocos de comandos ou instruções compostas.
  3. Qualquer número de segmentos pode ser executado em um determinado momento (não há fim de segmento implícito).
  4. Cláusula **default** usada para valores não representados.

Seleção de múltiplos caminhos: projeto

- Decisões de projeto da linguagem C#:
  1. Regra semântica estática não permite a execução implícita de mais de um segmento.
  2. Cada segmento selecionável deve terminar com um salto não condicional (**goto** ou **break**).
  3. Em C# a expressão de controle e as constantes das cláusulas **case** podem ser strings.

Seleção de múltiplos caminhos com **if**

Exemplo em Python

```
if   contador < 10:
     resultado = 0
elif contador < 100:
     resultado = 1
elif contador < 1000:
     resultado = 2
```

Exemplo em Ruby

```
case
     when contador < 10 then   resultado = 0
     when contador < 100 then   resultado = 1
     when contador < 1000 then   resultado = 2
```

Comandos iterativos

Estruturas de repetição

Repetição pode ser alcançada com iteração ou com recursividade.

- Comandos iterativos: Composto por um corpo e uma construção que determina o ponto de parada (ponto de controle).

Comandos iterativos

- Indefinidos (ponto de controle baseado em lógica):

```
while not eof(f) do
begin
        read( f, ch );     write( ch );
end
```

- Definidos (ponto de controle baseado em contadores):

```
for i := 1 to 10 do
        write( i );
```

Loops controlados por contadores

Ponto de controle definido por uma variável, em que se especifica: valor inicial, valor final e tamanho do passo.

- **Perguntas**
  1. Qual o tipo e escopo da variável de controle do loop?
  2. Qual o valor da variável de controle após o loop?
  3. É permitido mudar o valor da variável dentro do loop?
  4. Qual o valor da mesma se forçarmos a saída?
  5. Os parâmetros do loop devem ser avaliados apenas uma vez ou a cada iteração?

Loops controlados por contadores: exemplos

Exemplo em C

```
for ([expr_1] ; [expr_2] ; [expr_3]) comandos|
```

- Linguagens baseadas no C:
  - As expressões podem ser instruções completas, ou mesmo sequências de instruções (separados por vírgula).
  - O valor de uma expressão com múltiplas instruções é o valor da última instrução.
  - Não há definição explícita de variável de loop.
  - Tudo pode ser alterado dentro do loop.
  - Primeira expressão é avaliada apenas uma vez, mas demais são avaliadas a cada iteração.

Loops controlados por contadores: exemplos

- C++ difere do C em:
  - Expressão de controle também pode ser **boolean**.
  - Expressão inicial também pode incluir definição de variáveis (escopo é o bloco do corpo do loop).
- Java e C#:
  - Expressão de controle deve ser **boolean**.

Loops controlados por contadores: exemplos

Exemplo em Python

```
for variavel_do_loop in object:
    corpo do loop
[else:
    clausula do else]
```

- Variável é normalmente um objeto do tipo **range**, que representa uma lista de valores ou uma chamada à função **range**. Ex.:

  range(5) retorna 0, 1, 2, 3, 4.
- A variável do loop recebe os valores de **range** a cada iteração.
- Cláusula **else** é opcional é executada se o loop termina normalmente

Exemplo em Pascal

```
for i := 1 to 10 do
    write ( i );
write ( i );
```

- saída final: 10

```
for i := 1 to 10 do
begin
    i := i * 2;     { * }
    write ( i );    { ** }
end;
```

- *) i = 1 **) i = 2
- *) i = 3 **) i = 6
- *) i = 7 **) i = 14 **loop infinito**

Loops controlados por lógica

- Ponto de controle baseado em uma expressão **boolean**
- Perguntas:
  - Pré-teste ou pós teste?
- Linguagens baseadas em C possuem ambas as formas:

Pré-teste

```
while (expressao_de_controle)
    corpo do loop
```

Pós-teste

```
do
    corpo do loop
while (expressao_de_controle);
```

Mecanismos de controle de loops

- Conveniente permitir ao programador controlar o loop dentro do corpo do loop.
- Saídas: **break** e **continue**
  - Problema: loops aninhados.
- C, C++, Python, Ruby e C# possuem saídas não rotuladas.
- Java e Perl possuem versões rotuladas das saídas.

Iteração baseada em estruturas de dados

- O numero de elementos na estrutura de dados controla a iteração no loop.
- Mecanismos de controle é uma chamada a uma função *iterator* que retorna o próximo elemento em uma ordem predeterminada.

Exemplo em Java

```
Iterator<?> it = list.iterator();
while(it.hasNext())
    if (!cond(it.next()))
        it.remove();

for (String meuElemento : minhaLista) { ... }
```

Iteração baseada em estruturas de dados

- C# e F# possuem classes genéricas que podem ser iteradas com o comando **foreach**
- *Collections* que implementam a interface `IEnumerator` pode usar **foreach.**

Exemplo em C#

```
List<String> nomes = new List<String>();
nomes.Add("Joao");
nomes.Add("Ana");
nomes.Add("Maria");
foreach (Strings nome in nomes)
    Console.WriteLine("Nome: {0}", nome);
```

Chamadas de procedimento

- Chamadas a procedimentos: *P (A1, A2, A3 ... An)*
  - aplicar uma abstração dado os argumentos
- Parâmetro real
  - se **expressão** implica num argumento **valor**
  - se **acesso a variável** implica uma **referência a variável**
- Altera o fluxo do programa para o código da função.
- Estrutura de controle determinista (sempre chama o procedimento).
- Saída pode ser não determinista (múltiplos **return**).

Comandos de entrada/saída

- Associação (*binding*): *open*, *close*.
- Acesso: sequencial ou aleatório.
- *Stream* (fluxo contínuo) $\times$ registros de tamanho fixo.
- Codificação: textual $\times$ binário.

Arquivos

- Arquivos padrões:
  - Unix: stdin, stdout, stderr
  - C: stdin, stdout, stderr
  - C++: cin, cout, cerr
  - Java: System.in, System.out, System.err
- *Input/Output streams (em Java)*
  - file, pipe, memory, url
  - filter
  - reader, writer