

Algoritmos – Estruturas de Controle

Introdução à Programação para Biologia Molecular

Rosane Minghim

Apoio na confecção: Danilo Medeiros Eler
Rogério Eduardo Garcia

Baseado na Apostila: Curso Introdutório de Computação por R. Minghim e G. P. Telles

Algoritmo

- Execução sequencial:
 - Uma vez executado um comando, os demais são executados na sequência;
- Execução condicional: Deseja-se “selecionar” os comandos a serem executados:
 - Executar os comandos em determinadas condições;
- Repetição:
 - Repetir blocos de comandos pela mudança do fluxo de execução.

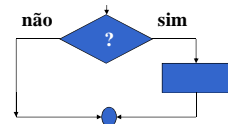
Algoritmo

- Execução seqüencial:
 - Uma vez executado um comando, os demais são executados na sequência;
- Deseja-se “selecionar” os comandos a serem executados:
 - Executar os comandos em determinadas condições;
- Repetição:
 - Repetir blocos de comandos pela mudança do fluxo de execução.

Estruturas de Controle

Escolha Simples

- Pode-se selecionar a seqüência de comandos a ser executada;
- Formato:
 - se *condição* então
 - comandos
 - fim se



Exemplo de Escolha Simples

```
constante
FATOR_CATEGORIA_1 = 1,0
TAXA_BÁSICA = 20,00

variável
categoria: inteiro
taxa: real

leia(categoria)

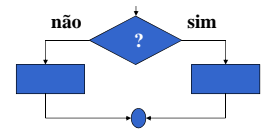
taxa ← taxa_básica
se categoria = 1 então
    taxa ← taxa - (TAXA_BÁSICA * FATOR_CATEGORIA_1)
fim se

escreva(taxa)
```

Código Python: categoria_1.py

Escolha Composta

■ Formato:
se *condição* então
comandos
senão
comandos
fim se



Exemplo

```
constante
FATOR_CATEGORIA_1 = 1,0
FATOR_GERAL = 0,2
TAXA_BÁSICA = 20,00

variável
categoria: inteiro
taxa: real

leia(categoria)

taxa ← taxa_básica
se categoria = 1 então
    taxa ← taxa - (TAXA_BÁSICA * FATOR_CATEGORIA_1)
senão
    taxa ← taxa - (TAXA_BÁSICA * FATOR_GERAL)
fim se

escreva(taxa)

Python: duas_categorias.py
```

Estruturas ‘Aninhadas’

Algoritmo maior_dretres
(algoritmo para obter o maior entre três
valores inteiros)

```
variável
maior: inteiro
numero1, numero2, numero3: inteiro

leia(numero1, numero2, numero3)
```

```
se numero1 > numero2 então
    se (numero1 > numero3) então
        maior ← numero1
    senão
        maior ← numero3
    fim se
fim se
senão
    se (numero2 > numero3) então
        maior ← numero2
    senão
        maior ← numero3
    fim se
fim se
fim
```

Python: maior_de_tres_sala.py
maior_de_tres_sala_elif.py

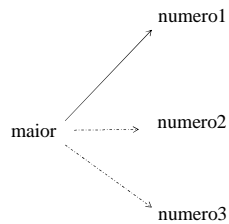
Estruturas 'Aninhadas' (simplificação)

Algoritmo maior_dos_tres_melhor
(algoritmo para obter o maior entre três valores inteiros)

variável
maior: inteiro
numero1, numero2, numero3: inteiro

```
leia(numero1, numero2, numero3)
maior ← numero1
se (numero2 > maior) então
  maior ← numero2
se (numero3 > maior) então
  maior ← numero3
fim se
fim
```

Python: maior_de_tres_melhor.py



Exercício Resolvido

Ref. Minghim, R., Telles, G.P. – Curso Introdutório de Programação

Resolver em sala de aula:

Desenvolver um algoritmo para, dados dois times de futebol (cada time identificado por um número inteiro), seus pontos ganhos e seu saldo de gols no campeonato, decidir qual dos dois está em melhor colocação (armazenando o resultado na variável ganhador). O resultado deve ser impresso. A regra diz que está na frente no campeonato o time que tiver mais pontos ganhos, com desempate pelo saldo de gols.

Escolhas Múltiplas

- Permite escolher uma entre várias alternativas expressas por valor inteiro ou caracter;

■ Formato:

```
selecione expressão entre
constante:
  comandos
constante:
  comandos
...
constante:
  comandos
senão
  comandos
fim seleção
```

Escolhas Múltiplas

- Na seleção múltipla, a **expressão** é calculada e os comandos relacionados abaixo da constante com o mesmo valor da expressão são executados.
- Se não houver valor igual ao da expressão, os comandos subordinados à palavra **senão** são executados.
- A cláusula **senão** é opcional.
 - Se ela não existir e o valor da expressão não for igual a nenhuma constante, nenhum comando da estrutura é executado.
- A **seleção** é exclusiva
 - No máximo uma das opções é executada.

Exemplo

Algoritmo desconto_taxa

constante

FATOR_CATEGORIA_1 = 1,0

FATOR_CATEGORIA_2 = 0,8

FATOR_CATEGORIA_3 = 0,5

FATOR_GERAL = 0,2

TAXA_BÁSICA = 20,00

variável

categoria: inteiro

taxa: real

leia(categoria)

Python:multipla.py

seleccione categoria entre

1:

taxa ← taxa - (taxa_básica *
FATOR_CATEGORIA_1)

2:

taxa ← taxa - (taxa_básica *
FATOR_CATEGORIA_2)

3:

taxa ← taxa - (taxa_básica *
FATOR_CATEGORIA_3)

senão

taxa ← taxa - (taxa_básica *
FATOR_GERAL)

fim seleção

escreva(taxa)

fim

Exercício Resolvido

Ref. Minghim, R., Telles, G.P. – Curso Introdutório de Programação

Resolver em sala de aula:

Desenvolver um algoritmo para calcular o valor de uma cartela de passes de ônibus para um passageiro. Uma cartela pode ter 50 ou 100 passes. Determinados tipos de usuários possuem desconto na compra de passes, de acordo com a tabela abaixo:

idosos 50%

estudantes 50%

trabalhadores faixa I 50%

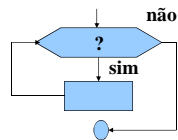
trabalhadores faixa II 25%

trabalhadores faixa I e estudante 75%

Repetição por Condição

- Uma das formas de repetir um conjunto de comandos de um algoritmo é subordiná-lo a um comando de repetição usando uma estrutura da forma:

enquanto *condição* faça
comandos
fim enquanto



Repetição por Condição

- Os comandos serão repetidos zero ou mais vezes, enquanto o valor da condição for verdadeiro.
- Essa estrutura normalmente é denominada **laço** ou **loop**.

Repetição por Condição – Funcionamento

- A condição da cláusula **enquanto** é testada.
- Se ela for verdadeira os comandos seguintes são executados em seqüência como em qualquer algoritmo, até a cláusula **fim enquanto**.
- O fluxo nesse ponto é desviado de volta para a cláusula **enquanto**.
- Se a condição agora for falsa (ou quando finalmente for), o fluxo do algoritmo é desviado para o primeiro comando após a cláusula **fim enquanto**.
- Se a condição ainda for verdadeira, o processo se repete.

Repetição por Condição

- A condição pode ser qualquer expressão que resulte em um valor do tipo lógico e pode envolver operadores aritméticos, lógicos, relacionais e resultados de funções.

Exemplo

Algoritmo calcula_senos

variável
n, i: inteiro

leia(n)

i ← 0

enquanto i ≤ n faça

 escreva(i, seno(i))

 i ← i + 1

fim enquanto

 escreva ('fim do cálculo')

fim

i ≤ n = falso

Exemplo

Algoritmo calcula_senos

variável

n, i: inteiro

leia(n)

i ← 0

enquanto (i < 45) e (i < n + 1) faça

 escreva(i, seno(i))

 i ← i + 1

fim enquanto

 escreva ('fim do cálculo')

fim

Loop Infinito

```
enquanto verdadeiro faça  
  comandos  
fim enquanto
```

- Um loop infinito pode acontecer também quando cometemos algum erro ao especificar a condição lógica que controla a repetição ou ao esquecer de algum comando dentro da iteração.

Exemplo

Algoritmo calcula_senos

```
variável  
n, i: inteiro
```

```
leia(n)  
i ← 0  
enquanto (i < 45) e (i < n + 1) faça  
  escreva(seno(i))  
fim enquanto  
fim
```

A variável i não é incrementada

Exemplo – soma consecutiva de dados sem estrutura de repetição

Algoritmo somasimples

```
variável  
valor1, valor2, valor3, valor4: real  
soma: real  
  
leia(valor1, valor2, valor3, valor4)  
soma ← (valor1 + valor2 + valor3 + valor4)/4  
escreva(soma)  
fim
```

Exemplo – soma consecutiva de dados

Algoritmo somasimples

```
variável  
valor, soma: real  
  
soma ← 0  
leia(número)  
enquanto número > -100 faça  
  soma ← soma + número  
  leia(número)  
fim enquanto  
escreva(soma)  
fim
```

Exercício: Além da soma, calcular a média aritmética

Outra forma de Repetição

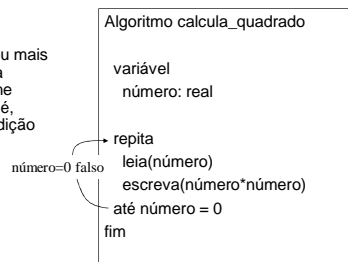
- Com teste no final do bloco de comandos.
- Uma diferença dessa forma para a anterior é que os comandos dentro da estrutura são executados uma vez antes que a condição seja testada pela primeira vez, e serve para os processos iterativos onde existe garantia de execução correta do bloco pelo menos uma vez.

Outra forma de Repetição

- Outra diferença com relação à forma anterior de repetição é que enquanto aquela estabelecia uma condição de continuidade, esta estabelece a condição de parada para a repetição.
- O seu formato é dado por:
repita
comandos
até *condição*

Exemplo

- Nesse caso, os comandos são repetidos uma ou mais vezes, até que a condição se torne verdadeira (isto é, enquanto a condição for falsa).



Exercício

- Fazer a soma consecutiva de valores inteiros lidos até a soma atingir 25500 ou até que tenham sido lidos 150 termos.
- Imprimir o número de termos da soma. Se a soma excedeu o limite, imprimir a diferença, senão imprimir a soma.
- Casos de teste

Repetição por Contagem

- Na iteração baseada em contagem, sabe-se **antecipadamente** quantas vezes um conjunto de comandos vai ser repetido.
- Formato:**
para variável de valor_inicial até valor_final passo valor_do_passo
faça
comandos
fim para

Repetição por Contagem

- Inicialmente a **variável**, que chamamos de variável controladora, é inicializada com o **valor_inicial**. Devem ser do tipo inteiro.
- Em seguida, os comandos são repetidos zero ou mais vezes, enquanto o valor da **variável** estiver entre o **valor_inicial** e o **valor_final**, inclusive.
- No final de cada repetição do conjunto de comandos, a variável controladora é automaticamente acrescida do **valor_do_passo** e o teste do limite é repetido

Exemplo

Algoritmo conta_com_para

```
variável
n, i: inteiro

leia(n)
para i de 1 até n passo 1 faça
  escreva(i)
fim para
fim
```

Repetição por Contagem

- A estrutura de controle **para** admite uma variação em que a repetição se dá em ordem decrescente. Basta que o valor do passo seja negativo.

Algoritmo conta_decrescente

```
variável
i,n: inteiro

leia (n)
para i de n até 1 passo -1 faça
  escreva(i)
fim para
fim
```


Exemplo

```
Algoritmo expoente
variável
base, expoente, resultado: inteiro
i: inteiro

leia(base, expoente)
resultado ← 1
para i de 1 até expoente passo 1 faça
    resultado ← resultado*base
fim para
escreva (resultado)
fim
```

Exemplos – dos and don'ts

Algoritmo laço_ruim

```
constante
LIMITANTE = 25500

variável
n, soma, i: inteiro

leia (n)
soma ← 0
para i de 1 até n passo 1 faça
    leia(termo)
    soma ← soma + termo
    se soma > LIMITANTE então
        i ← n + 1
    fim se
fim para
escreva (soma)
fim
```

Algoritmo laço_bom

```
constante
LIMITANTE = 25500

variável
n, soma, i: inteiro

leia (n)
soma ← 0
i ← 1
Enquanto (i ≤ n) e (soma ≤ LIMITANTE) faça
    leia(termo)
    soma ← soma + termo
    i ← i + 1
fim enquanto
escreva (soma)
fim
```

Equivalência entre as três formas de repetição

- É possível perceber que apenas uma forma de iteração (por exemplo, aquela da cláusula **enquanto**), seria suficiente para desenvolver qualquer algoritmo baseado em repetição.
- As demais formas existem para facilitar a estruturação dos algoritmos e aumentar a clareza do código.

Exemplo

- Qualquer laço de repetição baseado em contagem é equivalente a uma estrutura **enquanto** no seguinte formato:

```
variável ← valor_inicial
enquanto variável ≤ valor_final faça
    comandos
    variável ← variável + valor_do_passo
fim enquanto
```

Exemplo

Algoritmo expoente

variável
base, expoente, resultado: inteiro;
i: inteiro;

leia(base, expoente)

resultado \leftarrow 1

$i \leftarrow 1$

Enquanto $i \leq$ expoente faça

 resultado \leftarrow resultado * base

$i \leftarrow i + 1$

fim para

escreva (resultado)

fim

Percorrendo um Algoritmo: Casos de Teste

- Um algoritmo deve ser revisado buscando melhorias. Além disso, é preciso verificar se sua execução está correta.
- Um recurso para iniciar esse processo é percorrer o algoritmo.
 - Simular manualmente a execução de cada passo do algoritmo até chegar ao fim, assumindo valores para aqueles dados que são lidos do usuário.

Exemplo

```

1 Algoritmo maiorDetres
2
3 variável
4 maior: inteiro
5 número1, número2, número3: inteiro
6 i: inteiro
7
8
9 Para i de 1 até 3 passo 1 faça
10 leia(número1, número2, número3)
11 se número1 > número2 então
12 se (número2 > número3) ou (número2 = número3) então
13 maior ← número2
14 senão
15 se número1 > número3 então
16 maior ← número1
17 senão
18 maior ← número3
19 fim se
20 fim se
21 se (número2 > número3) ou (número2 = número3) então
22 maior ← número2
23 senão
24 maior ← número3
25 fim se
26 fim se
27 escreva(maior)
28 fim para
29
30 fim
  
```

Passo	Linha	i	número1	número2	número3	maior	condição
1	9	1	?	?	?	?	$1 \leq 3$
2	10	1	3	1	2	?	
3	11	1	3	1	2	?	verdadeiro
4	12	1	3	1	2	?	falso
5	13	1	3	1	2	3	
6	28	1	3	1	2	3	
7	29	2	3	1	2	3	
8	9	2	3	1	2	3	$2 \leq 3$
9	10	2	1	4	4	3	
10	11	2	1	4	4	3	falso
11	22	2	1	4	4	3	verdadeiro
12	23	2	1	4	4	4	
13	28	2	1	4	4	4	
14	29	3	1	4	4	4	
15	9	3	1	4	4	4	$3 \leq 3$
16	10	3	5	4	1	4	
17	11	