

# **Notas de Aula de Algoritmos e Programação de Computadores**

FLÁVIO KEIDI MIYAZAWA

*com a colaboração de*

TOMASZ KOWALTOWSKI

Instituto de Computação - UNICAMP

Versão 2000.1

Estas notas de aula não devem ser usadas como única fonte de estudo. O aluno deve ler outros livros disponíveis na literatura.

Nenhuma parte destas notas pode ser reproduzida, qualquer que seja a forma ou o meio, sem a permissão dos autores.

Os autores concedem a permissão explícita para a utilização e reprodução deste material no contexto do ensino de disciplinas regulares dos cursos de graduação sob a responsabilidade do Instituto de Computação da UNICAMP.

© Copyright 2000

Instituto de Computação  
UNICAMP  
Caixa Postal 6176  
13083-970 Campinas-SP  
{fkm,tomasz}@ic.unicamp.br

## 4 Estruturas de Repetição

A estrutura de repetição permite que um comando (ou bloco de comandos) seja executado repetidamente até que uma determinada condição de interrupção seja satisfeita.

### 4.1 Comando For

O comando **For** permite que um comando ou bloco de comandos seja repetido um número específico de vezes. Neste comando uma variável de controle é incrementada ou decrementada de um *valor inicial* em cada interação até um *valor final*.

A sintaxe do comando **for** que incrementa a variável de controle é dada como:

**for** variável\_de\_controle := expressão\_1 **to** expressão\_2 **do** Comando\_ou\_bloco\_de\_comandos;

Para a forma que decreta a variável de controle, temos a seguinte sintaxe:

**for** variável\_de\_controle := expressão\_1 **downto** expressão\_2 **do** Comando\_ou\_bloco\_de\_comandos;

Na figura 15 apresentamos o fluxograma de uma das formas do comando **for**.

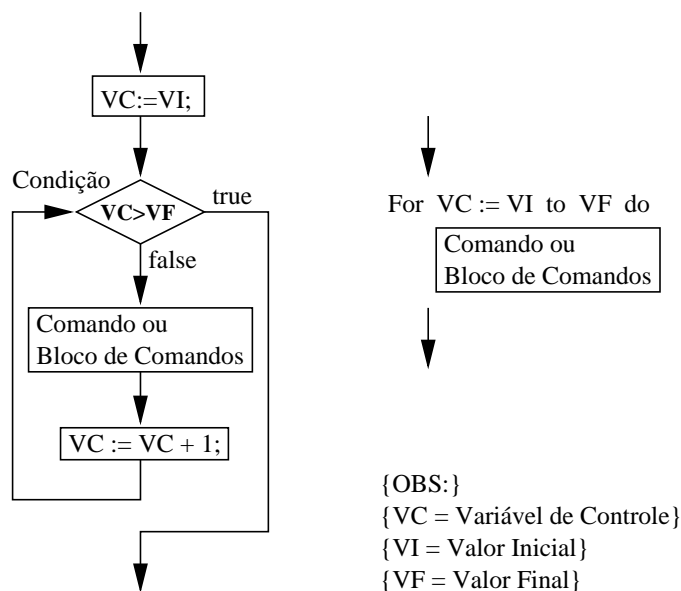


Figura 15: Fluxograma e sintaxe de uma forma do comando **for**.

**Exemplo 4.1** *Faça um programa para calcular o fatorial de um determinado número  $n$  lido.*

```
program fat;
var n,i,fatorial :integer;
begin
  write('Entre com um número: ');
  readln(n);
  fatorial := 1;
  for i:=2 to n do
    fatorial:=fatorial * i;
  writeln('O fatorial de ',n,' é igual a ',fatorial);
end.
```

**Exemplo 4.2** Faça um programa que lê um valor inteiro positivo  $n$  e em seguida lê uma seqüência de  $n$  valores reais. O programa deve imprimir o maior valor real da seqüência.

```

program Maximo;
var
  n,i      : integer;
  x        : real;
  maximo   : real;
begin
  ReadLn(n); {Supõe todos os dados não negativos.}
  maximo := 0.0
  for i:=1 to n do
    begin
      ReadLn(x);
      if x>maximo then maximo := x
    end;
  WriteLn(maximo)
end.

```

**Exemplo 4.3** (Tabuada) Faça um programa que imprima uma tabela, com 9 linhas e 9 colunas. Na interseção da linha  $i$  com a coluna  $j$  deve conter um valor que é a multiplicação do  $i$  com  $j$ . Isto é, o programa deve imprimir uma tabela da seguinte forma:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

```

program ProgramaTabuada;
var i,j      : integer;
begin
  for i:=1 to 9 do begin
    for j:=1 to 9 do
      write(i*j:3);
      writeln;
    end;
  end.

```

**Exercício 4.1** (Tabela de potências) Faça um programa que lê dois inteiros positivos  $n$  e  $k$  e imprime uma tabela de tamanho  $n \times k$  onde a posição  $(i, j)$  da tabela contém o número  $i^j$ .

$x$	$x^2$	$x^3$	$x^4$	$x^5$
1	1	1	1	1
2	4	8	16	32
3	9	27	81	243
4	16	64	256	1024
5	25	125	625	3125
6	36	216	1296	7776

**Exemplo 4.4** (*Triângulo de Floyd*) O seguinte triângulo formado por 6 linhas de números consecutivos, cada linha contendo um número a mais que na linha anterior, é chamado de Triângulo de Floyd.

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
```

Faça um programa que imprime o Triângulo de Floyd com  $n$  linhas (o valor de  $n$  é lido).

```
program Floyd;
var i : integer; {índice da linha}
    j : integer; {índice da coluna}
    k : integer; {próximo número}
    m : integer; {número de linhas}
begin
  ReadLn(m);
  k := 0;
  for i:=1 to m do
    begin
      for j:=1 to i do
        begin
          k := k+1;
          Write(k:3)
        end;
        WriteLn
      end
    end
end.
```

**Exercício 4.2** Faça um programa que lê um inteiro positivo  $n$  e imprime um triângulo constituído por números com o seguinte formato.

```
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

No caso a tabela foi impressa com valor de  $n$  igual a 6.

## 4.2 Comando While

O comando **while** é uma estrutura de repetição onde a condição de interrupção é testada antes de se executar os comandos a serem repetidos. Na figura 16, apresentamos o fluxograma e a sintaxe do comando While.

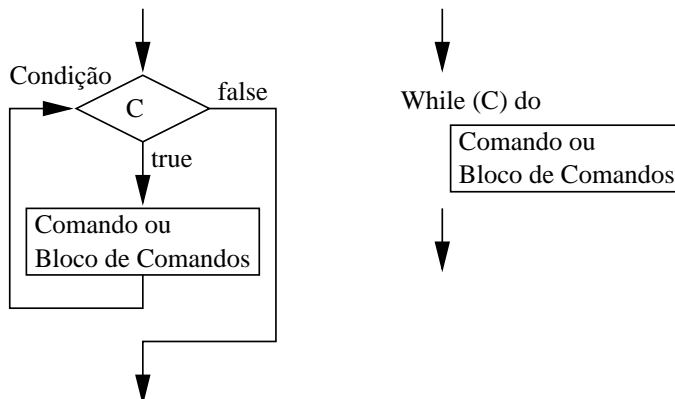


Figura 16: Fluxograma e sintaxe da rotina While.

**Exemplo 4.5** (*Validação de entrada*) Em determinado momento, um programa deve ler a partir do teclado um número que deve estar necessariamente no intervalo [10, 50]. Faça um programa que fique lendo números do teclado e pare quando o usuário entrar com o primeiro número entre [10, 50].

```
program Validacao;
var n:integer;
begin
  write('Entre com um número no intervalo [10,50]: '); readln(n);
  while ((n<10) or (n>50)) do begin
    writeln('ERRO: Número inválido. ');
    write('Entre com um número no intervalo [10,50]: '); readln(n);
  end;
  writeln('O número positivo lido foi: ',n);
end.
```

**Exemplo 4.6** (*Seqüência de números positivos*) Faça um programa que lê uma seqüência de números positivos (pode ser vazia) e seguida pela leitura de um número negativo. O programa deve parar de ler números quando o usuário entrar com o número negativo. O programa deve imprimir a soma, média e quantidade dos números não negativos.

```
program SequenciaPositivos;
var x,soma : real;
    nelementos : integer;
begin
  write('Entre com um número: '); readln(x);
  soma := 0; nelementos := 0;
  while (x>=0) do begin
    soma := soma + x; nelementos := nelementos + 1;
    write('Entre com um número: '); readln(x);
  end;
  if (nelementos>0) then begin
    writeln('A soma dos elementos é: ',soma);
    writeln('A media dos elementos é: ',soma/nelementos);
    writeln('A quantidade de elementos é: ',nelementos);
  end
  else writeln('Não foi lido nenhum elemento positivo. ');
end.
```

**Exemplo 4.7** Faça um programa que leia uma quantidade  $n$ , (vamos supor que  $n \geq 0$ ) e em seguida o programa deve ler  $n$  idades inteiras e então deve imprimir a média das idades lidas.

```

program MediaIdades;
var x,soma,lidos,n : integer;
begin
  write('Entre com a quantidade de idades a ler: ');
  readln(n);
  lidos:=0;
  soma := 0;
  while (lidos<n) do begin
    write('Entre com uma idade: ');
    readln(x);
    soma := soma + x;
    lidos := lidos + 1;
  end;
  if (lidos >0) then writeln('A média das idades é ',soma/lidos)
  else writeln('Não foi lido nenhuma idade. ');
end.

```

**Exemplo 4.8** O cálculo da raiz quadrada de um número positivo  $n$  pode ser aproximado usando se a seguinte série:

$$n^2 = 1 + 3 + 5 + \dots + (2 \cdot n - 1) = \sum_{k=1}^n (2 \cdot k - 1)$$

Se  $n$  é um quadrado perfeito, então podemos calcular a raiz usando-se o seguinte código:

```

...
readln(n);
soma := 0; i := 1; raiz := 0;
while soma<>n do
  begin
    soma := soma+i;
    i := i+2;
    raiz := raiz+1
  end;
writeln(raiz);
...

```

Caso  $n$  não seja um quadrado perfeito podemos obter uma aproximação considerando a parte inteira da raiz. Seja  $r = \lfloor \sqrt{n} \rfloor$ . Então  $r^2 \leq n < (r + 1)^2$  Fazendo duas modificações no trecho acima:

```

...
readln(n);
soma := 0; i := 1; raiz := 0;
while soma<=n do { <===== }
  begin
    soma := soma+i;
    i := i+2;
    raiz := raiz+1;
  end;
  raiz := raiz-1; { <===== }
writeln(raiz);
...

```

Note que a atribuição `raiz:=raiz-1;` foi feita para voltar o valor da raiz de uma unidade, uma vez que a condição de parada do comando `while` é `soma<=n;`. Uma versão que atualiza a variável soma com atraso é apresentada a

seguir:

```
...  
readln(n);  
soma := 0; i := 1; raiz := -1; {atraso na soma}  
while soma <= n do  
  begin  
    soma := soma + i;  
    i := i + 2;  
    raiz := raiz + 1;  
  end;  
writeln(raiz);  
...
```

**Exercício 4.3** Faça um programa para calcular a raiz aproximada, como no exemplo 4.8, mas usando o comando **repeat** em vez do comando **while**.

**Exercício 4.4** Para calcular a raiz quadrada de  $n$  com uma casa decimal basta calcular a raiz de  $100 \cdot n$  como no exemplo 4.8, e dividir o resultado por 10. Faça um programa que calcula a raiz de um valor  $n$  ( $n \geq 0$ ) pelo método do exemplo 4.8, com  $d$  casas decimais, ( $n$  e  $d$  são lidos).

### 4.3 Comando Repeat

O comando **repeat** é uma estrutura de repetição onde a condição de interrupção da estrutura de repetição é testada no fim do bloco de repetição. Note que neste comando não é preciso usar **begin** e **end** para especificar os vários comandos a serem repetidos. Na figura 17, apresentamos o fluxograma e a sintaxe do comando Repeat.

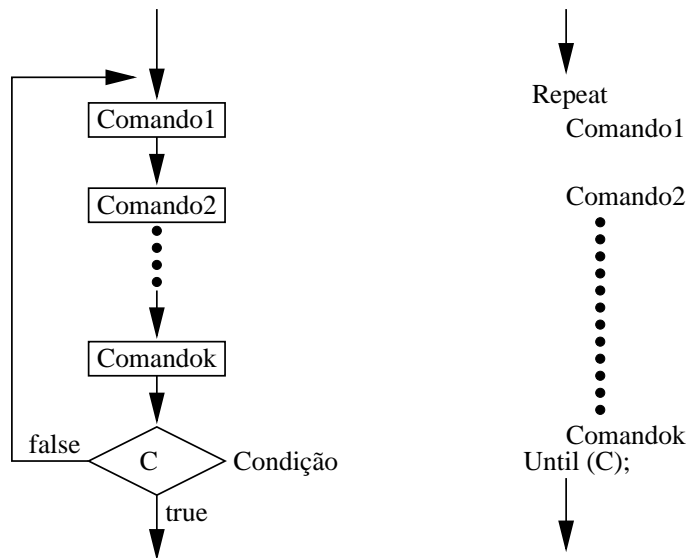


Figura 17: Fluxograma e sintaxe do comando Repeat.

**Exemplo 4.9** (Validação de entrada) Em determinado momento, um programa deve ler a partir do teclado um número que deve estar necessariamente no intervalo [10, 50]. Faça um programa que fique lendo números do teclado e pare quando o usuário entrar com o primeiro número entre [10, 50].

```
program Validacao;
var n:integer;
begin
  repeat
    write('Entre com um número no intervalo [10,50]: '); readln(n);
    if ((n<10) or (n>50)) then writeln('ERRO: Número inválido. ');
  until (n>=10) and (n<=50);
  writeln('O número positivo lido foi: ',n);
end.
```

**Exemplo 4.10** A seguir apresentamos a implementação do Algoritmo de Euclides usando o comando **repeat**.

```
program Euclides;
var x,y,r,m,n : integer;
begin
  Readln(m,n);
  x := m; y := n;
  repeat
    r := x mod y;
    x := y; y := r
  until r=0;
  Writeln(x)
end.
```

**Exemplo 4.11** (Seqüência de números positivos) Faça um programa para ler uma seqüência de números positivos (pode ser vazia) e seguido pela leitura de um número negativo. O programa deve parar de ler números quando o usuário entrar com o número negativo. O programa deve imprimir a soma, média e quantidade dos números não negativos.

```
program SequenciaPositivos2;
var x,soma : real;
    nelementos : integer;
begin
  soma := 0; nelementos := 0;
  repeat
    write('Entre com um número: '); readln(x);
    if (x>=0) then begin
      soma := soma + x;
      nelementos := nelementos + 1;
    end;
  until (x<0);
  if (nelementos>0) then begin
    writeln('A soma dos elementos é: ',soma);
    writeln('A media dos elementos é: ',soma/nelementos);
    writeln('A quantidade de elementos é: ',nelementos);
  end
  else writeln('Não foi lido nenhum elemento positivo. ');
end.
```

**Exemplo 4.12** A raiz quadrada de um número positivo  $N = N_0$  pode ser calculada pelo método de aproximações sucessivas de Newton. As aproximações  $N_1, N_2, \dots$  são tais que  $\lim_{i \rightarrow \infty} N_i = \sqrt{N_0}$ , onde

$$N_i = \begin{cases} \frac{N_0}{2}, & \text{se } i = 1, \\ \frac{(N_{i-1})^2 + N_0}{2N_{i-1}}, & \text{caso contrário.} \end{cases}$$

Faça um programa que calcula a raiz de um número calculando o valor da raiz pelo método de aproximações sucessivas de Newton parando as interações quando a diferença entre o valor calculado em uma interação com o valor calculado na interação anterior seja menor que 0,00001.

```

program raizquadrada;
var i      : integer;
    n,raiz,raizanterior : real;

begin
  write('Entre com um número: ');
  readln(n);
  raiz := n/2;
  repeat
    raizanterior := raiz;
    raiz := (raiz*raiz + n)/(2*raiz);
  until (abs(raiz-raizanterior) < 0.000001);
  writeln('A raiz de ',n:20:10,' é ',raiz:20:10);
end.

```

**Exemplo 4.13** Faça um programa que escreve individualmente os dígitos de um número inteiro positivo da direita para a esquerda.

```

program Digitos1;
var n,d    : Integer;
begin
  write('Entre com um número inteiro positivo: ');
  Readln(n);
  repeat
    d := n mod 10;
    n := n div 10;
    Write(d:2)
  until n=0;
  writeln;
end.

```

**Exercício 4.5** Faça um programa que escreve individualmente os dígitos de um número inteiro positivo da esquerda para a direita.

**Exemplo 4.14** O valor  $\pi$  pode ser calculado através da série  $\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \dots$ . Cada termo desta série tem um formato  $(-1)^{k+1} \cdot \frac{4}{2k-1}$ ,  $k = 1, 2, \dots$ . A medida que  $k$  cresce, o termo vai se tornando cada vez menor, e sua contribuição para o valor de  $\pi$  se torna menor. Faça um programa que calcula o valor de  $\pi$  através da série acima, somando termo a termo, parando quando a diferença absoluta entre o valor de  $\pi$  calculado em uma interação e o valor de  $\pi$  calculado na interação anterior for menor que 0,0001.

```

program pi;
var pi,piant,termo,sinal : real ;
    i           : integer;
begin
    pi := 0;
    i := 1;
    sinal := -1;
    termo := 4;
    repeat
        piant := pi;
        pi := pi + termo;
        i := i+2;
        termo := sinal*4/i;
        sinal := sinal*(-1);
    until abs(pi-piant) < 0.00001;
    writeln('pi = ',pi);
end.

```

**Exercício 4.6** *Um programa deve ler um inteiro positivo  $n$  e em seguida ler mais  $n$  valores reais sendo que o programa deve imprimir a soma, a média, o menor valor e o maior valor dos  $n$  valores reais lidos. Faça três versões deste programa, usando os comandos **while**, **repeat** e **for**.*

#### 4.4 Exercícios

1. Faça um programa que descubra um número entre 0 e 1000 imaginado pelo usuário. O programa deve fazer interações com o usuário. A cada interação, o programa deve tomar um número e perguntar para o usuário se este número é igual, menor ou maior do que o valor imaginado. O usuário deve responder de forma correta. A execução do programa deve terminar assim que o programa "adivinhar" o valor imaginado pelo usuário. O programa deve imprimir o número imaginado e o número de perguntas feitas pelo programa. Seu programa não pode fazer mais que 10 perguntas.
2. Faça um programa que leia uma seqüência de números inteiros positivos e termine com um número negativo (este último não deve ser considerado, serve apenas para finalizar a seqüência). O programa deve verificar se os números positivos:
  - (a) Estão em ordem crescente.
  - (b) Estão em ordem decrescente.
  - (c) Se a seqüência é uma progressão aritmética, neste caso dizer a razão.
  - (d) Se a seqüência é uma progressão geométrica, neste caso dizer a razão.
3. O desvio padrão  $dp$  e a variância  $var$  dos números  $x_1, \dots, x_n$  podem ser calculados usando as seguintes fórmulas

$$dp(x_1, \dots, x_n) = \sqrt{\frac{1}{n-1} \left( \sum_{i=1}^n x_i^2 - \frac{1}{n} \left( \sum_{i=1}^n x_i \right)^2 \right)}$$

$$var(x_1, \dots, x_n) = (dp(x_1, \dots, x_n))^2.$$

Faça um programa que lê o valor  $n$  e a seqüência dos  $n$  números reais e depois imprime a média, o desvio padrão e a variância dos  $n$  números lidos.

4. Um banco faz empréstimos com uma taxa de juros mensal igual a  $t$ ,  $0 < t \leq 1$ . Faça um programa que imprime quanto uma pessoa que toma emprestado  $q$  reais ficará devendo após  $m$  meses. Os valores de  $q$ ,  $j$  e  $m$  são lidos.