

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

3 Estrutura Condicional

A estrutura condicional permite a execução de instruções quando uma condição representada por uma expressão lógica é satisfeita.

3.1 Estrutura Condicional Simples

if (*condição*) **then** Comando;

A seguir, exemplificamos o uso da estrutura condicional simples.

```
program maximo1;  
var a,b: integer;  
begin  
  write('Entre com o primeiro número: ');  
  readln(a);  
  write('Entre com o segundo número: ');  
  readln(b);  
  if (a > b)  
  then  
    writeln('O maior valor é ',a);  
  if (a <= b)  
  then  
    writeln('O maior valor é ',b)  
end.
```

3.2 Estrutura Condicional Composta

Note que no exemplo do algoritmo *maximo1*, exatamente um dos dois comandos **if**'s é executado, i.e., ou ($a > b$) ou ($a \leq b$). Podemos considerar ambos os casos através de uma única condição:

```
if (condição)  
  then Comando1_ou_Bloco_de_Comandos1  
  else Comando2_ou_Bloco_de_Comandos2;
```

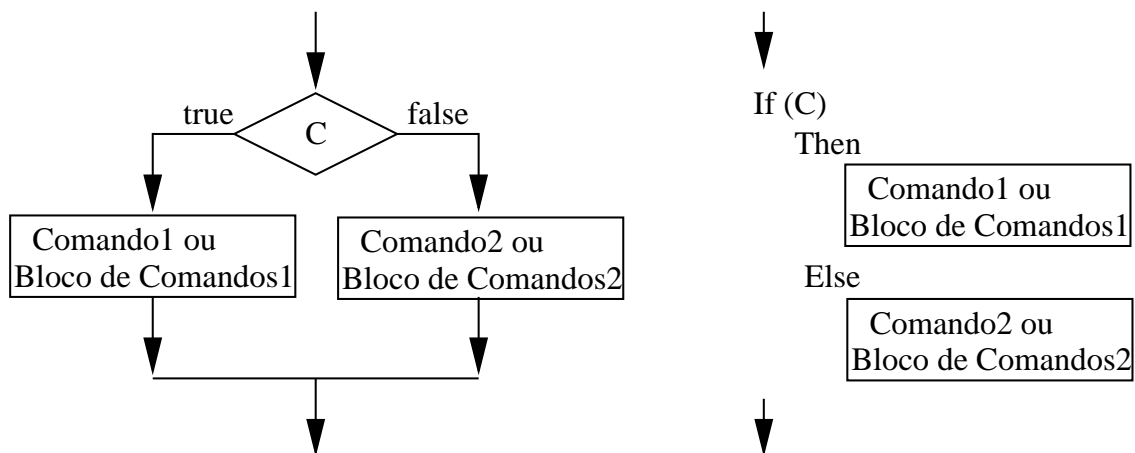


Figura 14: Fluxograma do comando If-Then-Else

Note que depois do *Comando1_ou_Bloco_de_Comandos1* não há ; (ponto e vírgula). Caso tivesse um ponto e vírgula, este estaria separando o comando **if** de outro comando e portanto a palavra **else** não seria continuação deste

comando **if**. Na figura 14 apresentamos o fluxograma do comando **if**, e no programa seguinte exemplificamos o uso da estrutura condicional composta.

```
program maximo2;
var a,b: integer;
begin
  write('Entre com o primeiro número: '); readln(a);
  write('Entre com o segundo número: '); readln(b);
  if (a > b)
  then
    writeln('O maior é ',a)
  else
    writeln('O maior é ',b)
end.
```

3.3 Bloco de Comandos

Um bloco de comandos é um conjunto de instruções que vai ser considerado como sendo um comando único. Desta forma, é possível compor estruturas, como no caso do comando **if**, envolvendo mais comandos. Um bloco de comandos é definido pelas palavras **begin** e **end**.

```
begin
  Comando1;
  Comando2;
  Comando3;
  :
end;
```

Aqui parece ser o momento para apresentar mais uma prática de programação estruturada. Sempre que um comando atuar sobre um bloco de comandos, dispomos este bloco de comandos deslocados mais a direita. Os comandos dentro de um bloco, entre **begin** e **end** estarão alinhados e também estarão deslocados mais a direita. Isto facilita muito a visualização da atuação de cada comando.

Exemplo 3.1 *No programa a seguir, exemplificamos o uso dos blocos de comandos.*

```
program maximo3;
var a,b: integer;
begin
  write('Entre com o primeiro número: '); readln(a);
  write('Entre com o segundo número: '); readln(b);
  if (a > b)
  then
    begin
      writeln('O maior está na variável a');
      writeln('O maior é ',a)
    end
  else
    begin
      writeln('O maior está na variável b');
      writeln('O maior é ',b)
    end
  end
end.
```

Exemplo 3.2 No exemplo seguinte apresentamos um programa que lê os coeficientes de uma equação de segundo grau, a , b e c (equação $ax^2 + bx + c = 0$), e imprime as raízes desta equação (se existir).

```
program equacaosegundograu;
var a, b, c: real;
    var x1, x2, delta: real;
begin
    writeln('Programa para resolver uma equação de segundo grau. ');
    writeln('Equação na forma: a*x*x + b*x+c = 0');
    write('Entre com o valor de a (diferente de zero): '); readln(a);
    write('Entre com o valor de b: '); readln(b);
    write('Entre com o valor de c: '); readln(c);
    delta := b * b - 4 * a * c;
    if (delta >= 0)
    then
        begin
            x1 := (-b + sqrt(delta))/(2 * a);
            x2 := (-b - sqrt(delta))/(2 * a);
            writeln('O valor de x1 = ',x1,' e o valor de x2 = ',x2);
        end
    else
        writeln('não é possível calcular raízes reais para esta equação');
    end.
end.
```

3.4 Comando Case

O comando **case** é um comando que permite selecionar um conjunto de operações conforme uma expressão com resultado escalar. Neste comando o valor escalar é comparado com vários outros valores que dividem as opções de execução em casos (vamos chamar estes de valores de caso). No máximo um caso pode ser verdadeiro. Os valores de caso podem ser tanto valores escalares como faixas de valores de escalares. Além disso, todos os valores de caso devem ser disjuntos (sem interseções). Se o valor escalar é igual ao valor (ou está dentro de uma faixa) de um caso então os comandos associados a este caso são executados.

As sintaxes do Extended Pascal e Borland Pascal diferem levemente. Assim, daremos a sintaxe do Borland Pascal e posteriormente explicitaremos a diferença com a sintaxe do Extended Pascal.

Há duas formas para a sintaxe do comando **case**:

A seguir apresentamos a sintaxe do caso onde não definimos comandos para o caso de não haver correspondência com os valores escalares de cada caso.

```
case (Expressão_Escalar) of
    Lista_Constantes_Escalares1: Comando_ou_Bloco_de_Comandos1;
    Lista_Constantes_Escalares2: Comando_ou_Bloco_de_Comandos2;
    :
    Lista_Constantes_EscalaresK: Comando_ou_Bloco_de_ComandosK;
end
```

A sintaxe para a situação onde definimos comandos para o caso onde não há correspondência com os valores escalares de cada caso é apresentada a seguir:

```

case (Expressão_Escalar) of
  Lista_Constantes_Escalares1: Comando_ou_Bloco_de_Comandos1;
  Lista_Constantes_Escalares2: Comando_ou_Bloco_de_Comandos2;
  ⋮
  Lista_Constantes_EscalaresK: Comando_ou_Bloco_de_ComandosK;
  else
    ComandoE1;
    ComandoE2;
    ⋮
    ComandoEE;
end

```

Cada Lista_Constantes_Escalares pode ser uma seqüência de escalares ou faixas de escalares separados por vírgula.

A diferença com a sintaxe do padrão Extended Pascal é que em vez da palavra **else** na sintaxe anterior, é usada a palavra **otherwise**. No decorrer deste texto, usaremos a sintaxe do padrão Borland Pascal.

Exemplo 3.3 *O seguinte programa mostra um exemplo do uso do comando **case**.*

```

program exemplo;
var c: char;
begin
  write('Entre com um caracter: ');
  readln(c);
  case c of
    'A'..'Z','a'..'z':
      writeln('Caracter lido é letra.');
    '0'..'9': begin
      writeln('Caracter lido é dígito.');
      writeln('i.e., um caracter em [0,9].');
    end;
    '+','-', '*', '/':
      writeln('Caracter é um operador matemático.');
    '$': writeln('Caracter é o símbolo $.');
    else
      writeln('O caracter lido não é letra, nem dígito.');
      writeln('nem operador matemático e nem é o símbolo $.');
  end;
end.

```

Exemplo 3.4 O seguinte programa mostra um exemplo de menu implementado com o comando **case**.

```
Program ExemploMenu;
var Opcao : char;
begin
  repeat
    { Escolha de uma opção do Menu }
    writeln('Entre com uma opção do menu abaixo: ');
    writeln(' [1] - Inserir dados de aluno novo no cadastro. ');
    writeln(' [2] - Remover aluno do cadastro. ');
    writeln(' [3] - Alterar os dados de um aluno. ');
    writeln(' [4] - Sair do sistema de cadastro. ');
    write('Opcao: '); readln(Opcao); writeln;
  case Opcao of
    '1' : begin
      writeln('Inserir Funcionário. ');
      writeln('Opção não implementada. ');
    end;
    '2' : begin
      writeln('Remover Funcionário. ');
      writeln('Opção não implementada. ');
    end;
    '3' : begin
      writeln('Alterar Funcionário. ');
      writeln('Opção não implementada. ');
    end;
    '4' : writeln('Fim da execução do sistema. ');
  else
    writeln('Opção inválida. ');
  end; { case }
  writeln;
until (Opcao = '4 ');
end.
```

3.5 Exercícios

1. Escreva um programa que determina a data cronologicamente maior de duas datas fornecidas pelo usuário. Cada data deve ser fornecida por três valores inteiros onde o primeiro representa um dia, o segundo um mês e o terceiro um ano.
2. Faça um programa que lê uma medida em *metros* e escreve esta medida em *polegadas*, *pés*, *jardas* e *milhas*.
Obs.:

1 polegada = 25.3995 milímetros
1 pé = 12 polegadas
1 jarda = 3 pés
1 milha = 1760 jardas

3. Sabendo que o valor numérico de uma letra na codificação ASCII é dada pela função **ord**,
 - (a) faça um programa que leia uma letra e escreva a codificação em binário desta letra.
Exemplo: Suponha que a letra a ser lida é a letra 'a'. A função **ord('a')** retorna o valor 97 e o programa deve imprimir: **01100001**.
Note que $0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 97$.

(b) faça um programa que leia uma letra e escreva a codificação em hexadecimal desta letra.

Exemplo: Suponha que a letra a ser lida é a letra 'm'. A função **ord**('m') retorna o valor 109 e o programa deve imprimir: **6D**. Note que $6 \cdot 16^1 + D \cdot 16^0 = 109$, onde D em hexadecimal é igual a 13 em decimal.

Obs.: a função **ord** sobre um caracter devolve um número entre 0 e 255 (que pode ser representado em 8 dígitos binários ou 2 dígitos hexadecimais).