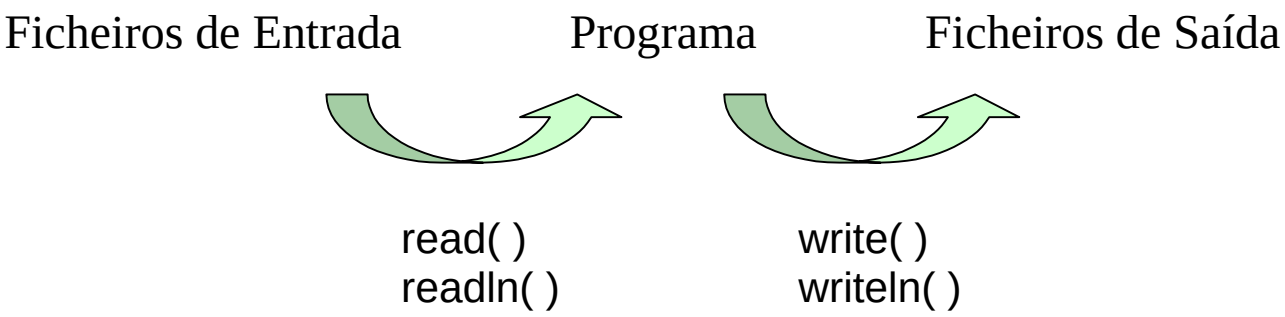
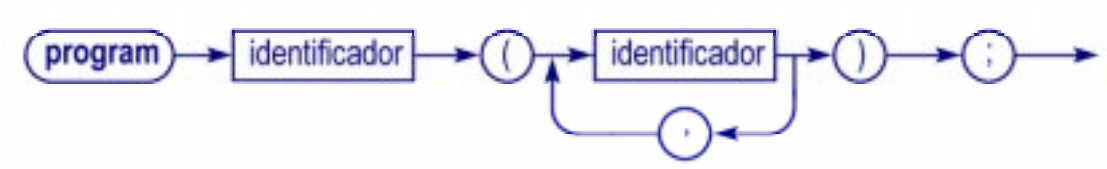
# Capítulo IV :
# A Linguagem Pascal – Ficheiros de Texto

- **Um Programa Pascal pode comunicar com um ou vários Ficheiros, tanto para a Leitura de Dados, como para a Escrita de Resultados.**

- **Os Ficheiros são exteriores ao Programa podendo existir, de forma permanente, antes e depois da sua execução.**




- **Os nomes dos Ficheiros associados, são indicados na Cabeça do Programa.**




Ficheiros de
Entrada e de Saída

- **Os Ficheiros do Tipo Texto são especialmente formatados para permitir a comunicação com o ser humano. Por isso, podem também ser directamente listados, editados, …**
- **Os Ficheiros são declarados como sendo do Tipo Texto na parte declarativa do Programa.**
- **Os Ficheiros input e output estão implicitamente declarados do Tipo Texto.**

# 1. Ficheiros de Entrada em Pascal "Standard":

**Problema:** O ficheiro numeros contem uma lista de números reais, um por cada linha, dos quais pretendemos calcular a média.

**Programa:**

```
program media (numeros, output);
var  numeros : text;
     n : integer;
     x, soma : real;

begin  reset(numeros);
       n:= 0;
       soma:= 0;

       while  not eof(numeros) do
              begin  readln(numeros, x);
                     n:= n + 1;
                     soma:= soma + x
              end;

       writeln('Média dos números = ', soma/n)
end.
```

- **O Ficheiro numeros foi indicado na Cabeça do Programa e declarado como Variável do Tipo text.**
- **A Instrução reset( ) abre o Ficheiro para Modo de Leitura.**
- **A Função eof( ), de argumento do tipo text e resultado do tipo boolean, é inicializada a false pelo reset( ) do Ficheiro e só se torna true no Fim do Ficheiro.**
- **A Instrução de Leitura indica o Ficheiro de onde os dados são lidos.**

# 1. Ficheiros de Saída em Pascal "Standard":

**Problema:** Criar um ficheiro com uma tabela de senos e cossenos dos valores de $x \in [0 , \pi/2[$ com intervalos de 0.01 radianos.

**Programa:**

```
program senosecossenos (tabela);
const pi = 3.14159265;
var   tabela : text;
      x, pi2 : real;

begin rewrite(tabela);

      (* Cabeçalho da Tabela *)
      writeln(tabela, 'x' : 6, 'sin(x)' : 8, 'cos(x)' : 8);
      writeln;
      x:= 0;
      pi2:= pi/2;

      while  x < pi2 do
             begin  writeln(tabela, x:6:2, sin(x):8:4, cos(x):8:4);
                    x:= x + 0.01
             end
end.
```

- **O Ficheiro tabela foi indicado na Cabeça do Programa e declarado como Variável do Tipo text.**
- **A Instrução rewrite( ) abre o Ficheiro para Modo de Escrita.**
- **As Instruções de Escrita indicam o Ficheiro onde os resultados são escritos.**

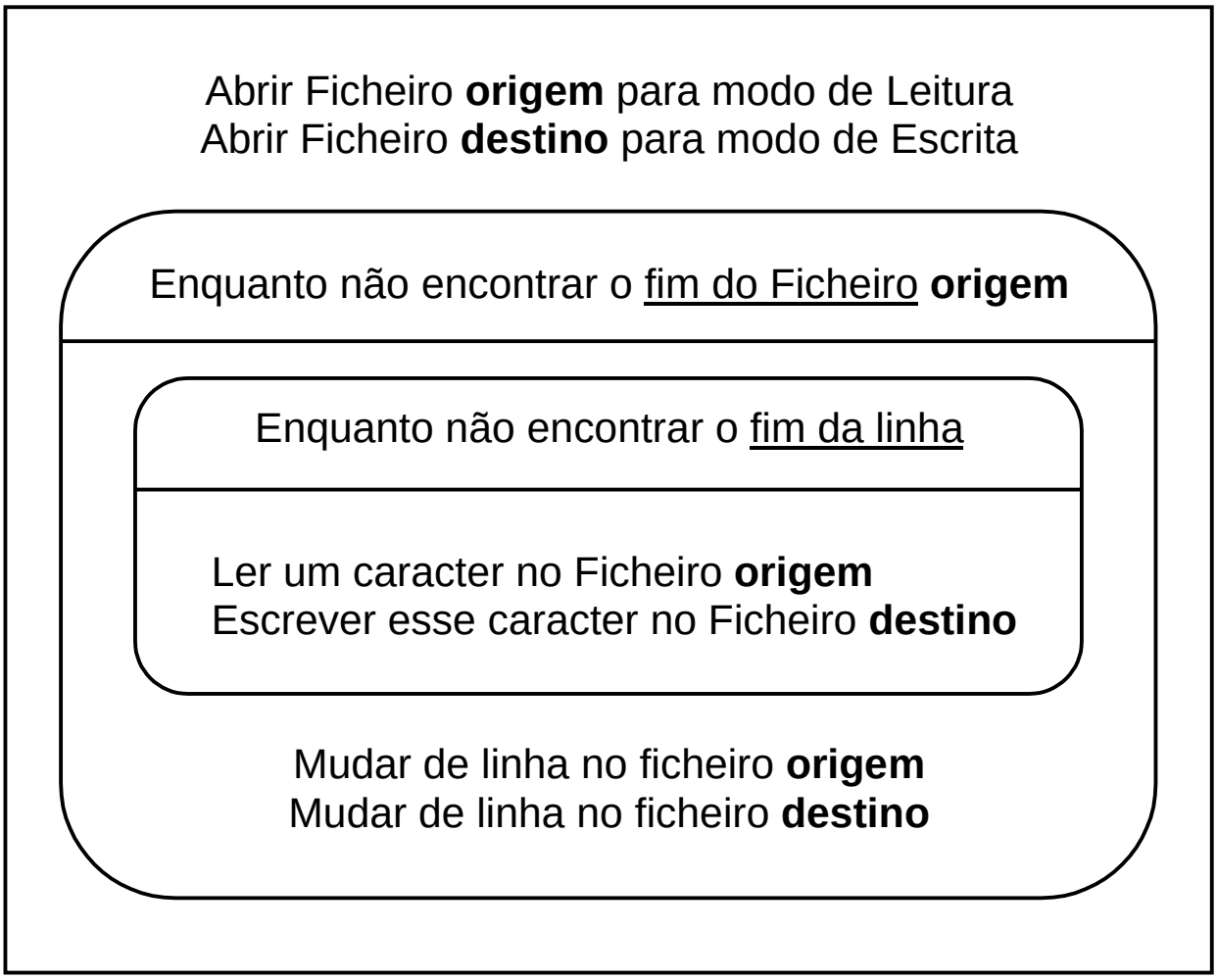
**Problema:** Registar sucessivas Pirâmides de Números num ficheiro, conforme forem sendo pedidas pelo utilizador.

## Programa:

```
program CriarPiramides (input, piramides, output);
var  piramides : text;
     n, linha, k : integer;
     resposta : char;

begin  rewrite(piramides);
       writeln('As Pirâmides que vai solicitar, serão registadas
                                          num ficheiro com esse nome');
       (*Cabeçalho do Ficheiro*)
       writeln(piramides, 'Pirâmides Diversas');

       repeat  (*Linhas em branco no Ficheiro*)
               for k:=1 to 5 do writeln(piramides);

               writeln('Qual o tamanho desta piramide?');
               readln(n);

               for  linha:=1 to n do
                    begin  write(piramides, ' ':40-linha);
                           for  k:=1 to linha do
                                write(piramides, k:2);
                           for  k:=linha-1 downto 1 do
                                write(piramides, k:2);
                           writeln(piramides)
                    end;

               writeln('Deseja construir mais pirâmides?');
               writeln('Por favor, responda S ou N');
               readln(resposta)

       until resposta = 'N'
end.
```

**Problema:** Copiar um texto dum ficheiro origem para um ficheiro destino, mantendo a mesma estrutura de linhas.

## Diagrama de Estrutura:




- **A Função eoln( ), de argumento do tipo text e resultado do tipo boolean, só toma o valor true em cada Fim de Linha do Ficheiro de Entrada.**

- **Note-se que, um mesmo Ficheiro, não pode estar simultaneamente em Modo de Leitura e Modo de Escrita.**

## Programa:

```
program copiar (origem, destino);
var  origem, destino : text;
     c : char;

begin  (*Abrir Ficheiro origem para Leitura*)
       reset(origem);
       (*Abrir Ficheiro destino para Escrita*)
       rewrite(destino);

       while  not eof(origem) do
              begin  (*Copiar uma Linha*)

                     while  not eoln(origem) do
                            begin  (*Copiar um caracter*)
                                   read(origem, c);
                                   write(destino, c)
                            end;

                     (*Mudar de Linha em ambos os Ficheiros*)
                     readln(origem);
                     writeln(destino)
              end
end.
```

## Exercícios:

- **Listar no écran um Ficheiro de Texto já existente, isto é, fazer o mesmo que o comando cat do unix.**

- **Guardar num Ficheiro, um Texto escrito no teclado.**

❖ **Em alguns sistemas, existem diferenças na utilização de Ficheiros, em relação Pascal "Standard".**

# Utilização de Ficheiros no Sistema Operativo unix:

- **A diferença essencial, em relação ao Pascal "Standard", consiste no facto de que, cada Ficheiro possui dois nomes: o nome externo (nome do Ficheiro no sistema unix) e o nome interno (nome do Ficheiro dentro do programa).**

- **As instruções reset( ) e rewrite( ) abrem o Ficheiro e associam o seu nome interno ao seu nome externo.**

- **É também utilizada a instrução close( ) para fechar cada Ficheiro.**

**Para o exemplo anterior:**

```
program copiar (f, g);
var   f, g : text;
      c : char;

begin (*Abrir Ficheiro origem para Leitura*)
      reset(f, 'origem');
      (*Abrir Ficheiro destino para Escrita*)
      rewrite(g, 'destino');

      while  not eof(f) do
            begin while  not eoln(f) do
                         begin read(f, c);
                               write(g, c)
                         end;
                  readln(f);
                  writeln(g)
            end;

      (* Fechar ambos os Ficheiros *)
      close(f);
      close(g)
end.
```

# Utilização de Ficheiros no TurboPascal:

- **O sistema é análogo ao dos Compiladores Pascal do Sistema Operativo unix.**
- **As instruções reset( ) ou rewrite( ) abrem um Ficheiro, mas não associam os seus dois nomes.**
- **É necessária a instrução assign( ) para esse efeito.**
- **O nome externo deve possuir o sufixo pretendido.**

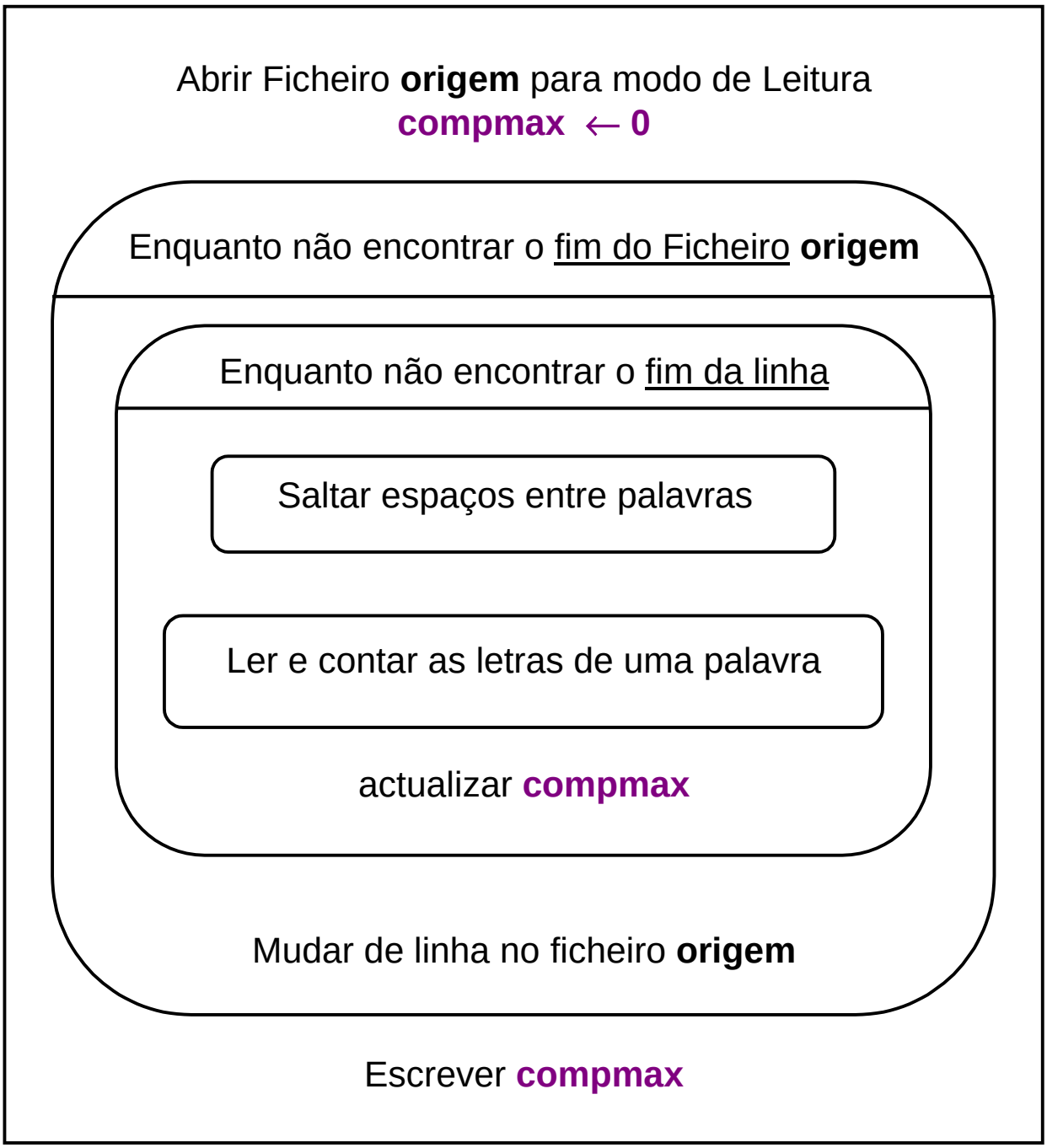
**Para o mesmo exemplo:**

```
program copiar (f, g);
var   f, g : text;
      c : char;

begin  (*Abrir Ficheiro origem para Leitura*)
       assign(f, 'origem.txt');
       reset(f);
       (*Abrir Ficheiro destino para Escrita*)
       assign(g, 'destino.txt');
       rewrite(g);

       while  not eof(f) do
              begin  while  not eoln(f) do
                            begin  read(f, c);
                                   write(g, c)
                            end;
                     readln(f);
                     writeln(g)
              end;

       (* Fechar ambos os Ficheiros *)
       close(f);
       close(g)
end.
```

**(*No que se segue, será utilizada a versão Pascal "Standard"*)**

**Problema:** O ficheiro origem contem um texto cujas palavras estão separadas por, um ou mais, espaços. Pretende-se determinar quantas letras tem a **maior palavra** que nele ocorre.

## Diagrama de Estrutura:



## Programa:

```
program MaiorPalavra (origem, output);
var   origem : text;
      c : char;
      comp, compmax : integer;

begin reset(origem);
      compmax:= 0;

      (* Ler o primeiro caracter *)
      read(origem, c);
      while  not eof(origem) do
            begin while  not eoln(origem) do
                        begin  (* Início de linha *)
                               (* Saltar espaços *)
                               while  c = ' '  do
                                      read(origem, c);
                               (* Aqui caracter ≠ espaço *)

                               (* Início de palavra *)
                               comp:= 0;
                               while  c <> ' '  do
                                      begin  comp:= comp + 1;
                                             read(origem, c)
                                      end;
                               (* Fim de palavra *)

                               (* Actualização de compmax *)
                               if comp > compmax
                               then compmax:= comp

                        end; (* Fim de linha *)
                  readln(origem)
            end; (* Fim do texto *)
      writeln('A maior palavra do texto tem', compmax, ' letras.')
end.
```

## Notas:

- **Normalmente as palavras de um texto estão separadas, não apenas por espaços, mas também por . , : ; ! ? ( ) …, ou seja, todo e qualquer caracter que não seja uma letra.**

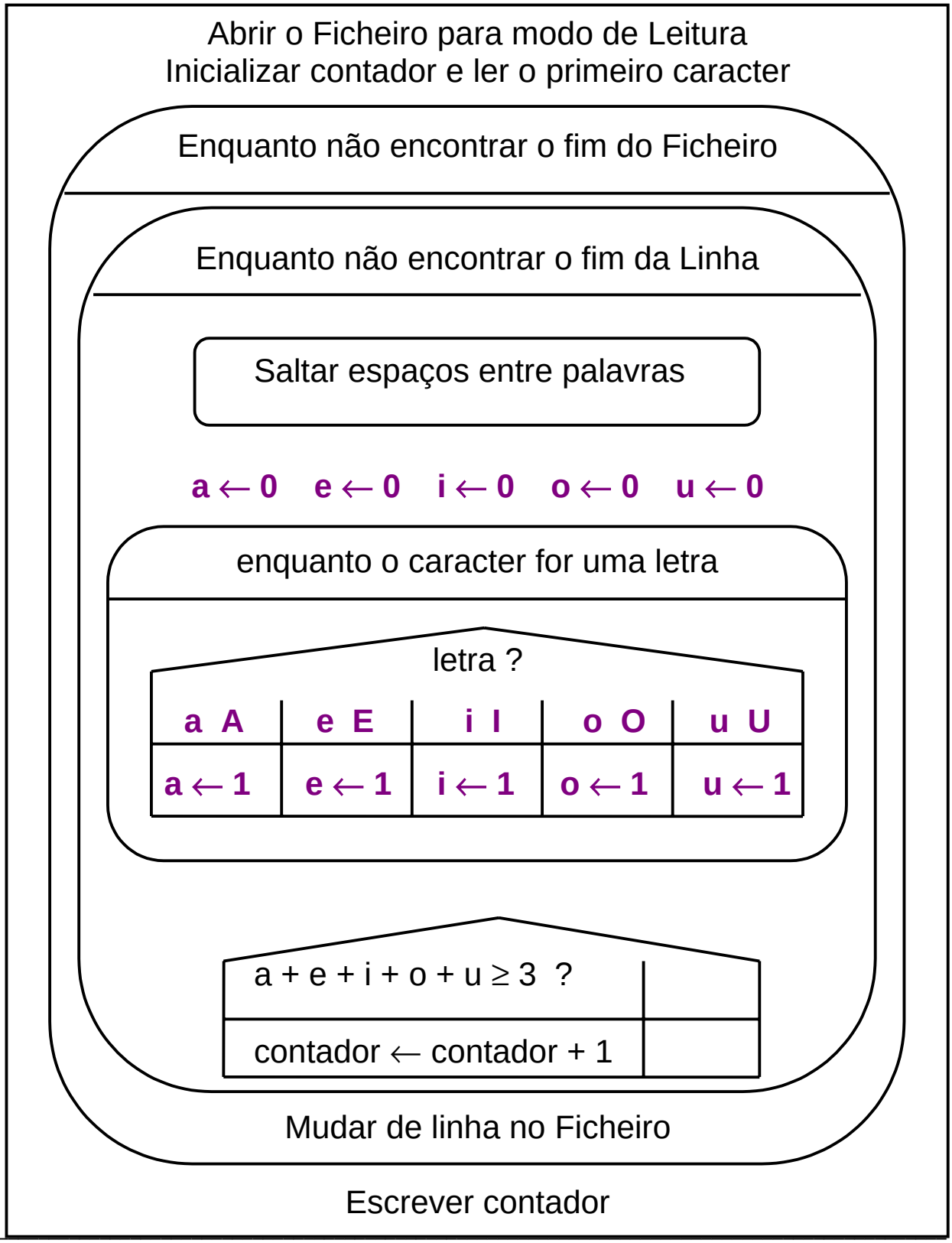
```
…

(* Saltar caracteres que não sejam letras *)

while  ((c < 'a') or (c > 'z')) and ((c < 'A') or (c > 'Z')) do
       read(origem, c);

(* Aqui o caracter é uma letra *)

(* Início de palavra *)
comp:= 0;
while ((c >= 'a') and (c <= 'z')) or ((c >= 'A') and (c <= 'Z')) do
      begin  comp:= comp + 1;
             read(origem, c)
      end;
(* Fim de palavra *)
…
```

- **Não foi resolvido o problema das palavras divididas, por hifen, em duas linhas.**

- **Não foi guardada a maior (ou maiores) palavra.**

**Problema:** É dado um texto cujas palavras estão separadas por um ou mais espaços. **Contar** as palavras com, pelo menos, **três vogais distintas**.

## Diagrama de Estrutura:



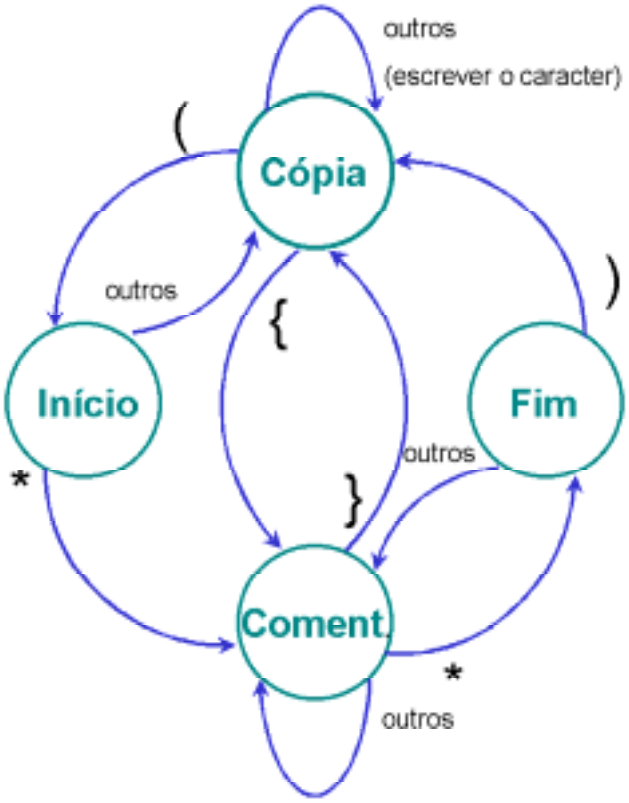
## Programa:

```
...
var  texto : text;
     car : char;
     contador : integer;    (* Contador geral *)
     a, e, i, o, u : 0 ..1;    (* Um Interruptor para cada Vogal *)
...

                ...
                (* Saltar espaços *)
                while  car = ' '  do
                       read(texto, car);

                (* Início de palavra *)
                (* Inicializar os 5 Interruptores *)
                a:= 0; e:= 0; i:= 0; o:= 0; u:= 0;

                while  car <> ' '  do
                       begin  case  car of
                                    'a', 'A'   : a:=1;
                                    'e', 'E'   : e:=1;
                                    'i',   'I'   : i:=1;
                                    'o', 'O'   : o:=1;
                                    'u', 'U'   : u:=1;
                              otherwise ;
                              end;
                              read(texto, car)
                       end; (* Fim de palavra *)

                (* Actualização do contador geral *)
                if a + e + i + o + u >= 3
                then contador:= contador + 1;
                ...

...
        writeln('O texto tem', contador, ' palavras com, pelo menos,
                                                três vogais distintas.')
```

**Problema:** O Ficheiro origem contem um programa Pascal.
Pretendemos **remover todos os comentários** e registar a nova versão do programa no Ficheiro destino.
Os comentários podem estar nas duas formas possíveis, **(* … *)** ou **{ … }**.

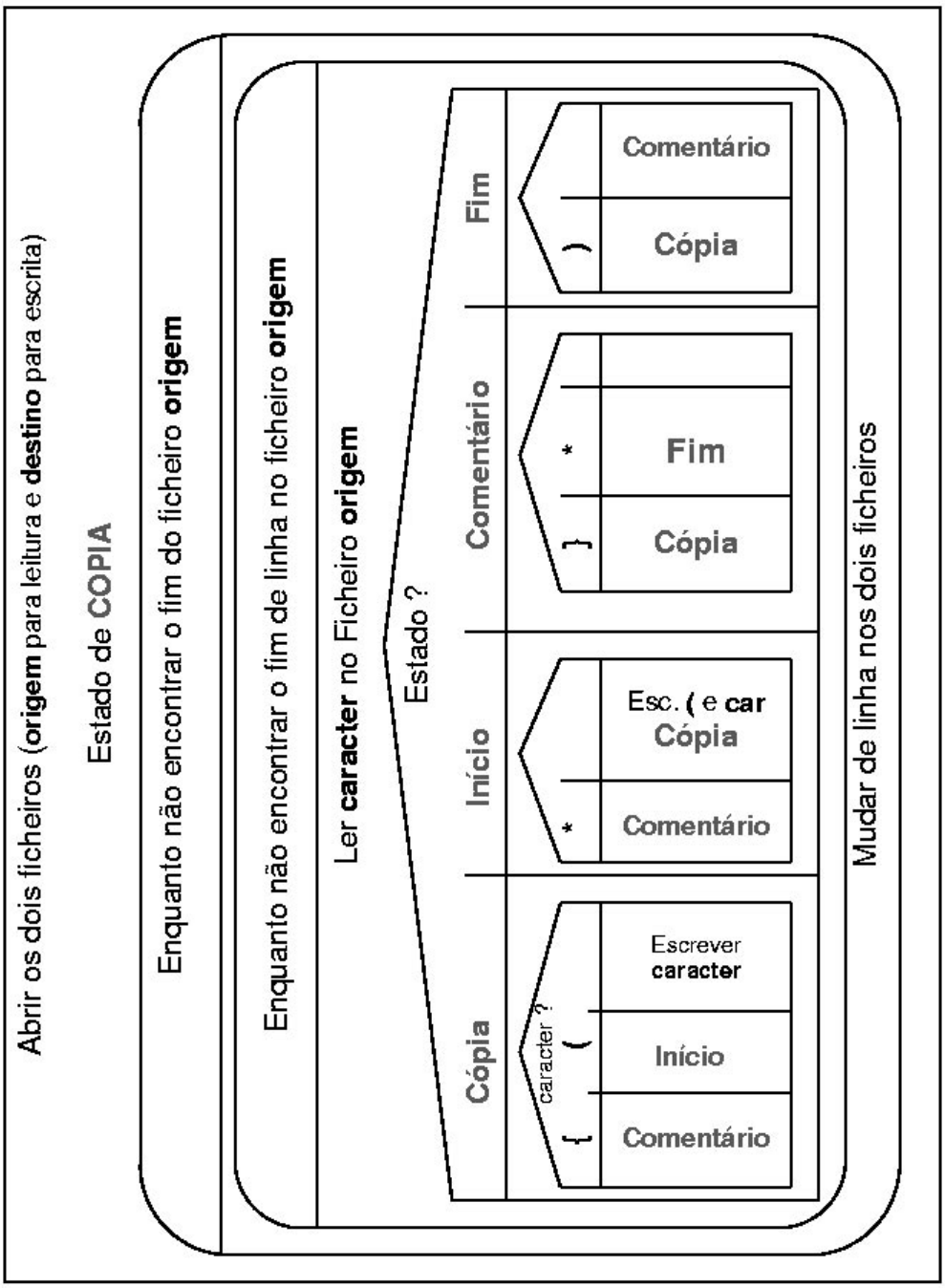
## Resolução por um Autómato Determinista:

Consideremos um Autómato Determinista com quatro Estados: **Cópia**, **Comentário**, **Início** e **Fim**.
A leitura de cada caracter provoca uma **Transição de Estado** e também uma operação associada.




Os comentários estão **Sintáticamente Correctos** se e só se, partindo do Estado de **Cópia**, for antingido o mesmo Estado após a leitura do último caracter.

# Diagrama de Estrutura:



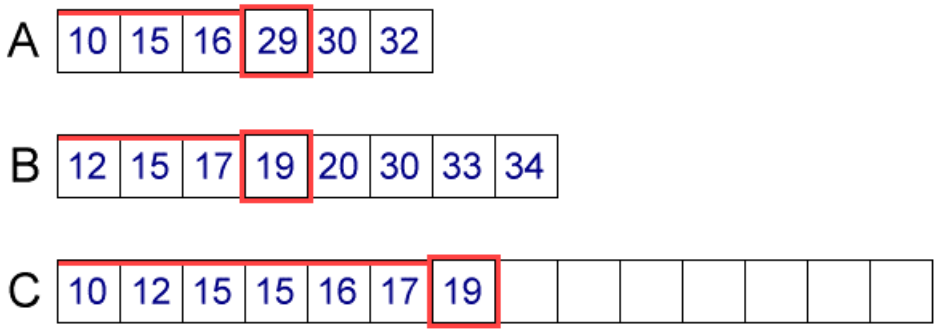
## Programa:

```
program DesComentar (origem, destino);
var  origem, destino : text;
     c : char;
     estado : (copia, inicio, comentario, fim);

begin reset(origem);
      rewrite(destino);

      estado:= copia;

      while  not eof(origem) do
            begin while  not eoln(origem) do

                        begin read(origem, c);
                              case estado of
                                    copia : case  c of
                                                '{' : estado:= comentario;
                                                '(' : estado:= inicio
                                          otherwise write(destino, c)
                                          end;
                                    inicio : if c = '*'
                                          then estado:= comentario
                                          else  begin write(destino, '(', c);
                                                      estado:= copia
                                                end;
                                    comentario : case  c of
                                                      '}' : estado:= copia;
                                                      '*' : estado:= fim
                                                otherwise (* nada *);
                                                end;
                                    fim :  if c = ')'
                                          then  estado:= copia
                                          else  estado:= comentario
                              end; (* Fim do case *)
                        end; (* Fim de linha *)

                  readln(origem);
                  writeln(destino)
            end (* Fim do texto *)
end. (* Fim do programa *)
```

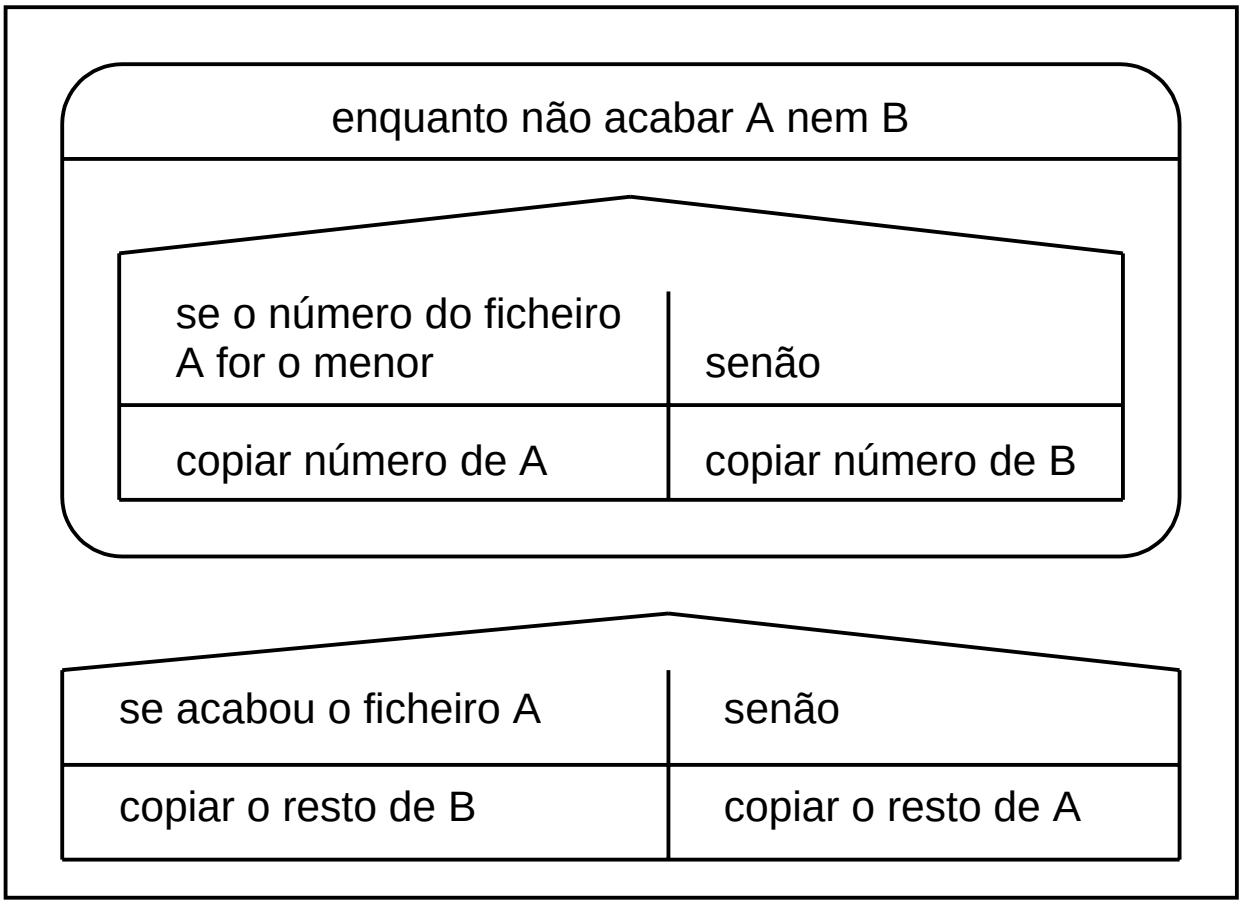
## Problema: Fusão de Ficheiros

São dados **dois** Ficheiros origem, A e B, cada um contendo uma **lista de números reais**, um por linha, dispostos por **ordem crescente**.
Pretende-se construir **um** Ficheiro C, com a **totalidade** dos números contidos em A e em B, por ordem crescente.

## Estratégia:




## Diagrama de Estrutura:



## Programa:

```
program Fusao (A, B, C);
var  A, B, C : text;
     numA, numB : real;

begin  (* Abrir Ficheiros *)
       reset(A); reset(B); rewrite(C);

       (* Ler o primeiro numero de cada Ficheiro *)
       readln(A, numA); readln(B, numB);

       while  not ( eof(A) or eof(B) ) do
              (* Escrever o menor em C *)
              if numA < numB
              then  begin  writeln(C, numA);
                           (* Ler o seguinte em A *)
                           readln(A, numA)
                    end
              else  begin  writeln(C, numB);
                           (* Ler o seguinte em B *)
                           readln(B, numB)
                    end;
       (* Um dos Ficheiros A ou B chegou ao fim *)

       (* Acabar de copiar o resto de A *)
       while  not eof(A) do
              begin  writeln(C, numA);
                     readln(A, numA)
              end;
       (* Acabar de copiar o resto de B *)
       while  not eof(B) do
              begin  writeln(C, numB);
                     readln(B, numB)
              end;

end.
```