

Computação I

Prof. Miguel Elias Mitre Campista

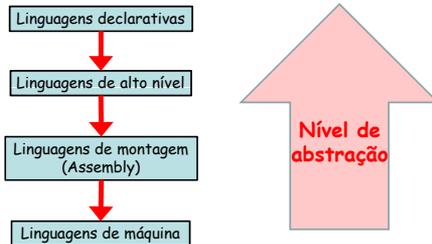
<http://www.gta.ufrj.br/~miguel>

Introdução ao Pascal

Computação I – DEL-Poli/UFRJ

Prof. Miguel Campista

Níveis de Linguagens de Programação



Computação I – DEL-Poli/UFRJ

Prof. Miguel Campista

Níveis de Linguagens de Programação

- Linguagens declarativas
 - Linguagens expressivas como a linguagem oral
 - Expressam o que fazer ao invés de como fazer
- Linguagens de alto nível
 - Linguagens típicas de programação
 - Permitem que algoritmos sejam expressos em um nível e estilo escrita fácil para leitura e compreensão
 - Possuem características de portabilidade já que podem ser transferidas de uma máquina para outra
- Linguagens de montagem e linguagem de máquina
 - Linguagens que dependem da arquitetura da máquina
 - Linguagem de montagem é uma representação simbólica da linguagem de máquina associada

Computação I – DEL-Poli/UFRJ

Prof. Miguel Campista

Como um Programa é Executado?

- Linguagens de programação
 - São projetadas em função da facilidade na construção e confiabilidade dos programas
 - Quanto mais próximo a linguagem de programação estiver da forma de raciocínio humano, mais intuitivo se torna o programa e mais simples é a programação

```
#include <stdio.h>
main() {
    ENQUANTO condição satisfeita FAÇA
        execute ação 1;
    FIM DO ENQUANTO
    imprimir "Acabou";
}
```

Como um Programa é Executado?

- Entretanto, computadores não entendem a linguagem humana...
 - Computadores entendem seqüências de 0's e 1's
 - Chamada de linguagem de máquina

```
#include <stdio.h>
main() {
    ENQUANTO condição satisfeita FAÇA
        execute ação 1;
    FIM DO ENQUANTO
    imprimir "Acabou";
}
```

1	0	1	1	0
0	0	1	1	0
...				
0	1	0	1	0
0	1	0	0	1

Computação I – DEL-Poli/UFRJ

Prof. Miguel Campista

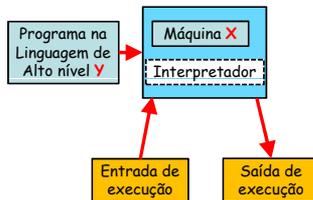
Níveis de Linguagem de Programação

- Existem duas maneiras para decodificar programas
 - Programa em linguagem de nível alto para programa em linguagem de nível baixo
 - Interpretação
 - Tradução

Programa Compilador



Programa Interpretador



Linguagem de Programação de Alto Nível

- Atualmente, há muitas linguagens de programação em alto-nível
 - C, C++, C#
 - Java
 - Perl, Python, Lua, Shell Script
 - Fortran, Cobol, Pascal

Histórico do Pascal

- Criado por Niklaus Wirth, na década de 60
 - Professor do departamento de informática da Escola Politécnica de Zurique (Suíça)
 - Objetivo era ensinar seus alunos a programar em PLI e ALGOL 60
 - Linguagem criada com objetivo de simplicidade para facilitar a compreensão

Tipos de Dados e Instruções Primitivas

- Estrutura de Dados
 - Representação da informação que ofereça facilidade de acesso e manipulação

Tipos de dados	Faixa de abrangência
inteiros	
shortint	De -128 até 127
integer	De -32.768 até 32.767
longint	De -2.147.483.648 até 2.147.483.647
byte	De 0 até 255
word	De 0 até 65.535

Tipos de Dados e Instruções Primitivas

- Estrutura de Dados
 - Representação da informação que ofereça facilidade de acesso e manipulação

Tipos de dados reais	Faixa de abrangência
real	De 2,9 E-39 até 1,7 E+38
single	De 1,5 E-45 até 3,4 E+38
double	De 5,0 E-324 até 1,7 E+308
extended	De 3,4 E-4.932 até 1,1 E+4.932
comp	De -9,2 E+18 até 9,2 E+18

Tipos de Dados e Instruções Primitivas

- Estrutura de Dados
 - Representação da informação que ofereça facilidade de acesso e manipulação

Tipos de dados caracteres (Devem vir sempre entre ' ')
string
char

Tipos de Dados e Instruções Primitivas

- Estrutura de Dados
 - Representação da informação que ofereça facilidade de acesso e manipulação

Tipos de dados lógicos
true
false

Estrutura de um Programa em Pascal

- Cabeçalho do programa
 - Área utilizada para fazer identificação de um programa
 - Uso de nome
 - Obs.: Nenhuma variável pode possuir o mesmo nome que o programa
 - Ex.: program SOMA;

```
program nome_do_programa;
```

Estrutura de um Programa em Pascal

- Área de declarações
 - Área utilizada para validar o uso de qualquer tipo de identificador que não seja pré-definido
 - var ←
 - uses
 - label
 - const
 - type
 - procedure
 - function

Estrutura de um Programa em Pascal

- Área de declarações
 - Área utilizada para validar o uso de qualquer tipo de identificador que não seja pré-definido
 - var
 - Ex.: var
 - nome: string;
 - idade: int;
 - altura, peso: real;

```
var  
nome_variavel1: tipo1;  
nome_variavel2: tipo2;  
nome_variavel3, nome_variavel4: tipo3;
```

Estrutura do Programa em Pascal

- **Corpo do programa**
 - O programa propriamente dito em Pascal está escrito na área denominada **corpo do programa**
 - Área tem início com a instrução **begin** e término com a instrução **end**, seguida do símbolo ponto (.)
 - Ex.: `begin`
`writeln(IDADE);`
`end.`

```
begin
    <instruções>
end.
```

Estrutura do Programa em Pascal

- **Comentário**
 - Parte do programa ignorada pelo compilador
 - Serve para comentar o código inserido
 - Sentença comentada deve vir entre chaves ({ })
 - » Ex.: `var`
`{ idade do usuário }`
`idade: integer;`

```
begin
    <instruções>
    {comentário}
end.
```

Estrutura do Programa em Pascal

- **Entrada e saída de dados**
 - Entrada padrão através do teclado
 - Função `readln`
 - Ex.: `readln(var);` → atribui o valor lido do teclado à variável `var`
 - Saída padrão através da tela
 - Função `writeln`
 - Ex.: `writeln(var);` → escreve o valor da variável `var` na tela

Primeiro Exemplo

- Escrever um programa que digite na tela 'Hello, world!'

?

Primeiro Exemplo

```
program HELLO;
begin
    writeln('Hello, world!');
end.
```

```
Compilação
fpc -o<executavel> <codigo_fonte>
```

```
shell$>fpc -oex1 exemplo1.pas
```

Segundo Exemplo

- Escrever um programa que digite na tela 'Hello, ' seguido do seu nome

?

Segundo Exemplo

```
program HELLO;
var
  NOME: string;
begin
  { Solicita ao usuário a entrada do nome }
  writeln('Digite o seu nome, por favor!');
  readln(NOME);
  writeln('Hello, ', NOME, '!');
end.
```

Terceiro Exemplo

- Escrever um programa que some três inteiros passados pelo teclado e imprima na tela o resultado da soma



Terceiro Exemplo

```
program SOMA;
var
  RESULTADO, P1, P2, P3: integer;
begin
  { Solicita ao usuário os três inteiros }
  writeln('Entre com as tres parcelas');
  { Lê os três inteiros do teclado }
  readln(P1);
  readln(P2);
  readln(P3);
  { Calcula a soma }
  RESULTADO := P1 + P2 + P3;
  { Imprime o resultado }
  writeln('O resultado é: ', RESULTADO);
end.
```

Quarto Exemplo

- Escrever um programa que calcule o salário líquido de um trabalhador. Para isso, deve ser calculado o salário bruto como sendo o produto entre o valor da hora e o número de horas trabalhadas e, em seguida, calculado o INSS sobre o salário bruto para encontrar o salário líquido



Quarto Exemplo

```
program SALARIO;
var
  HT, VH, PD, TD, SB, SL: real;
begin
  write('Quantas horas de trabalho? ');
  readln(HT);
  write('Qual o valor da hora? ');
  readln(VH);
  write('Qual o percentual de desconto? ');
  readln(PD);
  SB := HT * VH;
  TD := (PD/100) * SB;
  SL := SB - TD;
  { Imprime os resultados }
  writeln('Salario bruto: ', SB);
  writeln('Total de desconto: ', TD);
  writeln('Salario liquido: ', SL);
end.
```

Quarto Exemplo

```
program SALARIO;
var
  HT, VH, PD, TD, SB, SL: real;
begin
  write('Quantas horas de trabalho? ');
  readln(HT);
  write('Qual o valor da hora? ');
  readln(VH);
  write('Qual o percentual de desconto? ');
  readln(PD);
  SB := HT * VH;
  TD := (PD/100) * SB;
  SL := SB - TD;
  { Imprime os resultados }
  writeln('Salario bruto: ', SB:2);
  writeln('Total de desconto: ', TD:2);
  writeln('Salario liquido: ', SL:2);
end.
```

Tomada de Decisão

- Desvio condicional simples
 - Tomada de decisão que pode gerar um desvio na execução do programa
 - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO
 - Ex.: `if (A > 0) then`
`writeln('A > 0');`

```
if (<condição>) then
  <instrução se verdadeiro>
```

Tomada de Decisão

- Desvio condicional simples
 - Tomada de decisão que pode gerar um desvio na execução do programa
 - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO

```
if (<condição>) then
  begin
    <instrução1 se verdadeiro>
    <instrução2 se verdadeiro>
  end;
```

Tomada de Decisão

- Desvio condicional simples
 - Tomada de decisão que pode gerar um desvio na execução do programa
 - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO
 - Ex.: `if (A > 0) then`
`begin`
`writeln('A > 0');`
`A := B + C;`
`end;`

Tomada de Decisão

- Desvio condicional composto
 - Tomada de decisão que gera um desvio na execução do programa
 - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO

```
if (<condição>) then
  begin
    <instrução1 se verdadeiro>
    <instrução2 se verdadeiro>
  end
else
  begin
    <instrução1 se verdadeiro>
    <instrução2 se verdadeiro>
  end
end;
```

Tomada de Decisão

- Desvio condicional composto
 - Tomada de decisão que gera um desvio na execução do programa
 - Desvio depende da avaliação de uma sentença lógica em VERDADEIRO ou FALSO
 - Ex.: `if (A > 0) then`
`begin`
`writeln('A > 0');`
`A := B + C;`
`end`
`else`
`begin`
`writeln('A < 0');`
`A := B - C;`
`end;`

Não se pode colocar
";" antes do else!

Quinto Exemplo

- Escrever um programa que ordene duas variáveis inteiras



Quinto Exemplo

```
program ORDENA;
var
  TEMP, A, B: integer;
begin
  writeln('Entre com o valor de A:');
  readln(A);
  writeln('Entre com o valor de B:');
  readln(B);
  writeln('Sequencia: ', A, ' e ', B);
  if (A > B) then
    begin
      TEMP := A;
      A := B;
      B := TEMP;
      writeln('Sequencia ordenada: ', A, ' e ', B);
    end;
end.
```

Sexto Exemplo

- Escrever um programa que ordene duas variáveis inteiras, se elas já estiverem ordenadas, o programa avisa que não há nada para fazer



Sexto Exemplo

```
program ORDENA_V2;
var
  TEMP, A, B: integer;
begin
  writeln('Entre com o valor de A:');
  readln(A);
  writeln('Entre com o valor de B:');
  readln(B);
  writeln('Sequencia: ', A, ' e ', B);
  if (A > B) then
    begin
      TEMP := A;
      A := B;
      B := TEMP;
      writeln('Sequencia ordenada: ', A, ' e ', B);
    end
  else
    writeln('Sequencia jáh ordenada: ', A, ' e ', B);
  end;
end.
```

Operadores Lógicos

- Ou
– OR
• Ex.: se (<condição1> or <condição2>) then
sentença1;
- E
– AND
• Ex.: se (<condição1> and <condição2>) then
sentença1;
- Negação
– NOT
• Ex.: se not <condição1> then
sentença1;

Sétimo Exemplo

```
program TEMPERATURA;
var
  TEMP: real;
begin
  writeln('Qual a temperatura de hoje?');
  readln(TEMP);
  if (TEMP > 30) and (TEMP <= 25) then
    writeln('Temperatura agradável!');
  else
    if not (TEMP < 25) then
      writeln('Muito calor!');
    else
      writeln('Muito frio!');
  end;
end.
```

Repetição

- Utiliza o conceito de loop de programação
– Repetição é realizada até que uma condição falhe
• Modos: utilizando
repeat-until
while-do

```
repeat
  <instrução1 se verdadeiro>
  <instrução2 se verdadeiro>
until (<condição>);
```

Repetição

- Utiliza o conceito de loop de programação
 - Repetição é realizada até que uma condição falhe

- Modos: utilizando
 - repeat-until
 - while-do

```
while (<condição>) do
  begin
    <instrução1 se verdadeiro>
    <instrução2 se verdadeiro>
  end;
```

Oitavo Exemplo

```
program FATORIAL;
var
  NUMBER, TEMP, RESULTADO: integer;
begin
  writeln('Entre com o numero inteiro:');
  readln(NUMBER);
  RESULTADO := NUMBER;
  TEMP := NUMBER;
  repeat
    TEMP := TEMP - 1;
    RESULTADO := RESULTADO * TEMP;
  until TEMP = 1;
  writeln('Fatorial de ', NUMBER, ' eh: ', RESULTADO);
end.
```

Nono Exemplo

```
program FATORIAL;
var
  NUMBER, TEMP, RESULTADO: integer;
begin
  writeln('Entre com o numero inteiro:');
  readln(NUMBER);
  RESULTADO := NUMBER;
  TEMP := NUMBER;
  while (TEMP > 1) do
    begin
      TEMP := TEMP - 1;
      RESULTADO := RESULTADO * TEMP;
    end;
  writeln('Fatorial de ', NUMBER, ' eh: ', RESULTADO);
end.
```

Repetição

- Utiliza o conceito de loop de programação
 - Repetição pode ser realizada com variável de controle
 - Modos: utilizando
 - for-to //Loop com variável crescente
 - for-downto //Loop com variável decrescente

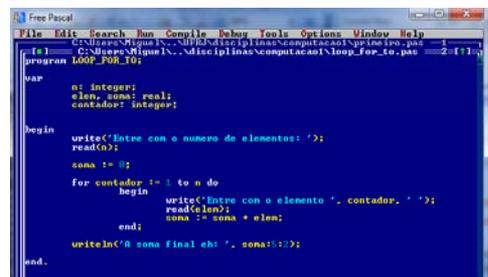
```
for <variável> := <inicio> to <fim> do
  begin
    <instrução1 se verdadeiro>
    <instrução2 se verdadeiro>
  end;
```

Repetição

- Utiliza o conceito de loop de programação
 - Repetição pode ser realizada com variável de controle
 - Modos: utilizando
 - for-to //Loop com variável crescente
 - for-downto //Loop com variável decrescente

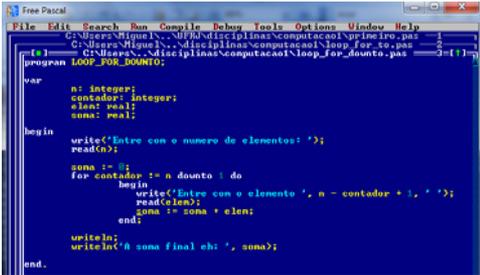
```
for <variável> := <inicio> downto <fim> do
  begin
    <instrução1 se verdadeiro>
    <instrução2 se verdadeiro>
  end;
```

Décimo Exemplo



```
Free Pascal
File Edit Search Run Compile Debug Tools Options Window Help
G:\Users\Miguel\Documents\Disciplinas\computacao\loop_for_to.pas 1
program LOOP_FOR_TO;
var
  n: integer;
  soma: real;
  contador: integer;
begin
  write('Entre com o numero de elementos: ');
  readln(n);
  soma := 0;
  for contador := 1 to n do
    begin
      write('Entre com o elemento ', contador, ' ');
      readln(soma);
      soma := soma + soma;
    end;
  writeln('A soma final eh: ', soma);
end.
```

Décimo Primeiro Exemplo



```
Free Pascal
File Edit Search Run Compile Debug Tools Options Window Help
C:\Users\Miguel\...disciplinas\computacao\primeiro.pas
C:\Users\Miguel\...disciplinas\computacao\loop_for.pas
C:\Users\Miguel\...disciplinas\computacao\loop_for_daunto.pas
program LOOP_FOR_DAUNTO;

var
  n: Integer;
  contador: Integer;
  soma: real;
begin
  write('Entre com o numero de elementos: ');
  read(n);

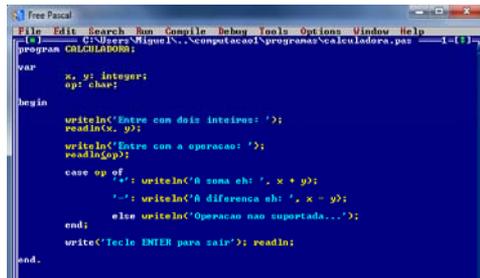
  soma := 0;
  for contador := n downto 1 do
    begin
      write('Entre com o elemento ', n - contador + 1, ' ');
      read(soma);
      soma := soma + soma;
    end;
  writeln;
  writeln('A soma final eh: ', soma);
end.
```

Seleção

- Utiliza o conceito de seleção de sentença
 - Seleciona caso verdadeiro
 - Ex.: case var of
 - 1: <sentenças>;
 - 2: <sentenças>;
 - else: <sentenças>;
 - end;

```
case <variável> of
  valor1: <instrução1 se verdadeiro>
  valor2: <instrução2 se verdadeiro>
  else <instrução2 se verdadeiro>
end;
```

Décimo Segundo Exemplo



```
Free Pascal
File Edit Search Run Compile Debug Tools Options Window Help
C:\Users\Miguel\...computacao\preparacao\calculadora.pas
program CALCULADORA;

var
  x, y: Integer;
  op: char;
begin
  writeln('Entre com dois inteiros: ');
  readln(x, y);
  writeln('Entre com a operacao: ');
  readln(op);

  case op of
    '+': writeln('A soma eh: ', x + y);
    '-': writeln('A diferenca eh: ', x - y);
    else writeln('Operacao nao suportada...');
  end;
  write('Tecle ENTER para sair: '); readln;
end.
```

Seleção

```
case <variável> of
  valor1:
    begin
      <instrução1 se verdadeiro>
    end;
  valor2:
    begin
      <instrução2 se verdadeiro>
    end;
  else
    begin
      <instrução2 se verdadeiro>
    end;
end;
```

Estrutura de Dados Homôgenea

- Matrizes de uma dimensão ou vetores
 - Estruturas de dados que armazenam variáveis do mesmo tipo
 - Estrutura deve ser dimensionada antes do uso por constantes inteiras e positivas
 - Nomes dados às matrizes seguem as mesmas regras de nomenclatura de variáveis simples

Matrizes de uma Dimensão ou Vetores

- Uma matriz de uma dimensão ou vetor é representada por:
 - Nome
 - Tamanho (dimensão)
 - Tipo

```
<matriz>: array[<dimensão>] of <tipo de dados>;
```

Décimo Terceiro Exemplo

```
Free Pascal IDE
[0] C:\Users\Miguel\Documents\computacao\programas\media.pas end(1)
program MEDIA;

var
  notas: array [1..5] of real;
  soma: real;
  idx: integer;

begin
  soma := 0;
  writeln('Entre com as notas dos alunos');
  for idx := 1 to 5 do
    begin
      writeln('Entre com a nota do aluno ', idx, ': ');
      readln(notas[idx]);
      soma := soma + notas[idx];
    end;

  writeln('A media foi: ', soma/5);
  write('Tecle <ENTER> para continuar'); readln;
end.
```

Décimo Quarto Exemplo

```
Free Pascal IDE
[0] C:\Users\Miguel\Documents\computacao\programas\media.pas end(1)
program MEDIA;

var
  h: array [1..5] of real;
  idx: integer;

begin
  writeln('Entre com as notas dos alunos');
  for idx := 1 to 5 do
    begin
      writeln('Informe ', idx, 'o. valor: ');
      readln(h[idx]);
    end;

  for idx := 1 to 5 do
    begin
      if h[idx] >= 0 then
        begin
          h[idx] := 0 + h[idx];
        end
      else
        begin
          h[idx] := 0 + h[idx];
        end;
    end;

  for idx := 1 to 5 do
    writeln(h[idx]);
  write('Tecle <ENTER> para continuar'); readln;
end.
```

Décimo Quinto Exemplo

```
Free Pascal IDE
[0] C:\Users\Miguel\Documents\computacao\programas\vetring.pas end(1)
program VETRING;

var
  nomes: array [1..6] of string;
  idx: integer;

begin
  writeln('Entre com as nomes dos alunos');
  for idx := 1 to 6 do
    begin
      writeln('Informe o ', idx, 'o. nome: ');
      readln(nomes[idx]);
    end;

  for idx := 1 to 6 do
    writeln('Nome [', idx, ']: ', nomes[idx]);
  write('Tecle <ENTER> para continuar'); readln;
end.
```

Décimo Sexto Exemplo

```
Free Pascal IDE
[0] C:\Users\Miguel\Documents\computacao\programas\vetring_ordenado.pas end(2)
program ORDENADO;

var
  nomes: array [1..6] of string;
  temp: string;
  i, j: integer;

begin
  writeln('Entre com 6 nomes');
  writeln;
  for i := 1 to 6 do
    begin
      readln(nomes[i]);
    end;

  for i := 1 to 6 do
    begin
      writeln('Nome ', i, ': ', nomes[i]);
    end;

  (Ordenação do vetor de strings)
  for i := 1 to 6 do
    for j := i + 1 to 6 do
      if (nomes[i] < nomes[j]) then
        begin
          temp := nomes[i];
          nomes[i] := nomes[j];
          nomes[j] := temp;
        end;
    end;

  writeln;
  for i := 1 to 6 do
    begin
      writeln('Nome Ordenado ', i, ': ', nomes[i]);
    end;

  writeln;
  writeln('Tecle <ENTER> para continuar'); readln;
end.
```

Décimo Sétimo Exemplo

```
Free Pascal IDE
[0] C:\Users\Miguel\Documents\computacao\programas\vetring_busca.pas end(1)
program BUSCA;

var
  nomes: array [1..6] of string;
  temp: string;
  i, j: integer;
  busca: integer;
  achou: boolean;

begin
  writeln('Entre com 6 nomes');
  writeln;
  for i := 1 to 6 do
    begin
      readln(nomes[i]);
    end;

  for i := 1 to 6 do
    begin
      writeln('Nome ', i, ': ', nomes[i]);
    end;

  (Ordenação do vetor de strings)
  for i := 1 to 6 do
    for j := i + 1 to 6 do
      if (nomes[i] < nomes[j]) then
        begin
          temp := nomes[i];
          nomes[i] := nomes[j];
          nomes[j] := temp;
        end;
    end;

  writeln;
  for i := 1 to 6 do
    begin
      writeln('Nome Ordenado ', i, ': ', nomes[i]);
    end;

  busca := False;
  i := 1;

  writeln('Entre com o nome a ser buscado');
  readln(busca);
```

```
while (busca = False) and (i <= 6) do
  if (nomes[i] = busca) then
    begin
      achou := true;
    end
  else
    begin
      i := i + 1;
    end;

  if (achou = true) then
    begin
      writeln('String encontrada na posição: ', i);
    end
  else
    begin
      writeln('String não encontrada...');
    end;

  writeln;
  writeln('Tecle <ENTER> para terminar'); readln;
end.
```

Décimo Sétimo Exemplo

Décimo Sétimo Exemplo

Estrutura de Dados Homogênea

- Matrizes de mais de uma dimensão
 - Estruturas de dados que armazenam variáveis do mesmo tipo
 - Estrutura deve ser dimensionada antes do uso por constantes inteiras e positivas
 - Nomes dados às matrizes seguem as mesmas regras de nomenclatura de variáveis simples

Matrizes de Mais de uma Dimensão

- Uma matriz de mais de uma dimensão é representada por:
 - Nome
 - Tamanho de cada uma das suas dimensões
 - Caso possua duas: dimensão de linhas e colunas
 - Tipo

```
<matriz>:  
array[<dimensão linha>, <dimensão coluna>]  
of <tipo de dados>;
```

Décimo Oitavo Exemplo

```
program MEIRI;  
var  
  nomes : array[1..5, 1..3] of string;  
  temp_pri : string;  
  temp_ult : string;  
  i, j : integer;  
  busca : boolean;  
  achou : boolean;  
begin  
  writeln('Entre com primeiro e ultimo nomes de seis pessoas');  
  writeln;  
  for i := 1 to 6 do  
    begin  
      writeln('Primeiro nome de candidato ', i, ' ');  
      readln(nomes[i, 1]);  
      writeln('Ultimo nome de candidato ', i, ' ');  
      readln(nomes[i, 3]);  
    end;  
  for i := 1 to 6 do  
    begin  
      writeln('Nome: ');  
      for j := 1 to 3 do  
        write(nomes[i, j], ' ');  
      writeln;  
    end;  
  (Ordenação do vetor de strings)  
  for i := 1 to 6 do  
    for j := 1 to 3 do  
      if nomes[i, j] < nomes[i, j+1] then  
        begin  
          temp_pri := nomes[i, j];  
          temp_ult := nomes[i, j+1];  
          nomes[i, j] := nomes[i, j+1];  
          nomes[i, j+1] := temp_pri;  
          nomes[i, j] := temp_ult;  
          nomes[i, j+1] := temp_pri;  
        end;  
    end;  
end;
```

Décimo Oitavo Exemplo

```
writeln;  
for i := 1 to 6 do  
  begin  
    write('Nome Ordenado ');  
    for j := 1 to 3 do  
      begin  
        write(nomes[i, j], ' ');  
      end;  
    writeln;  
  end;  
achou := false;  
i := 1;  
writeln('Digite com o nome a ser buscado:');  
readln(busca);  
(Busca do elemento no vetor)  
while (busca = false) and (i <= 6) do  
  begin  
    if (nomes[i, 1] = busca) then  
      begin  
        writeln('Busca encontrada!');  
        achou := true;  
        i := i + 1;  
      end;  
    else  
      begin  
        writeln('Busca não encontrada...');  
        i := i + 1;  
      end;  
    end;  
  if (achou = true) then  
    begin  
      writeln('String encontrada na posição ', i);  
    end;  
  else  
    begin  
      writeln('String não encontrada...');  
    end;  
  writeln('Tecla (ENTER) para terminar'); readln;  
end;  
92177
```

Registros

- Estrutura de dados composta por dados de tipos diferentes
 - Matriz heterogênea
 - Declarada dentro do bloco type
 - Bloco type deve ser declarado antes de var porque o registro define tipo de dados

```
type  
<nome_registro> = record  
  var1: <tipo var1>;  
  var2: <tipo var2>;  
  ...  
  varn: <tipo var n>;  
end;
```

Registros

- Os registro podem ser usados como tipos

```
type  
<nome_registro> = record  
  var1: <tipo var1>;  
  var2: <tipo var2>;  
  ...  
  varn: <tipo var n>;  
end;  
var  
var_registro: <nome_registro>
```

Arrays de Registros

- Registros podem ser usados como tipos de arrays

```
type
<nome_registro> = record
    var1: <tipo var1>;
    var2: <tipo var2>;
    ...
    varn: <tipo var n>;
end;

var
var_array: array[1..N] of <nome_registro>
```

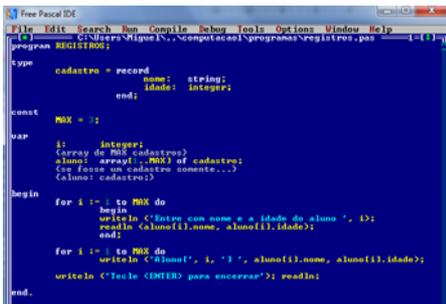
Arrays de Registros

- Registros podem conter como um de seus elementos um array

```
type
<nome_reg> = record
    var1: <tipo var1>;
    var2: array[1..4] of <tipo var2>;
    ...
    varn: <tipo var n>;
end;

var
var_registro: array[1..N] of <nome_reg>
```

Décimo Nono Exemplo



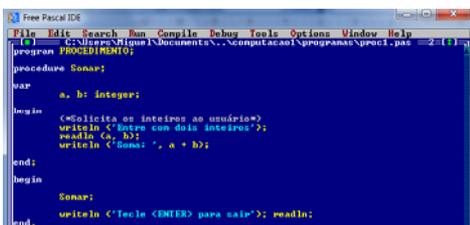
```
Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
program NOME19.pas
type
    cadastro = record
        nome: string;
        idade: integer;
    end;
const
    MAX = 10;
var
    i: integer;
    aluno: array[1..MAX] of cadastro;
begin
    for i := 1 to MAX do
        begin
            writeln('Entre com nome e a idade do aluno ', i);
            readln(aluno[i].nome, aluno[i].idade);
        end;
    for i := 1 to MAX do
        writeln('Aluno[', i, ']', aluno[i].nome, aluno[i].idade);
    writeln('Tecla <ENTER> para encerrar'); readln;
end.
```

Procedimento

- Utilizado para modularizar o programa
 - Reuso reduz o tempo de codificação do programa
- Deve sempre ser declarado antes da função principal

```
procedure NOME;
var
    var1: <tipo1>
    var2: <tipo2>
begin
    código...
end;
```

Vigésimo Exemplo



```
Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
program PROCEDIMENTO;
procedure Somar;
var
    a, b: integer;
begin
    (<Solicita os inteiros ao usuário>)
    writeln('Entre com dois inteiros');
    readln(a, b);
    writeln('Soma: ', a + b);
end;
begin
    Somar;
    writeln('Tecla <ENTER> para sair'); readln;
end.
```

Passagem de Parâmetro para Procedimento

- Um procedimento pode receber uma lista de variáveis como entrada da função

```
procedure NOME (<var1>: <tipo1>; <var2>: <tipo2> );
var
    var_local1: <tipo1>
    var_local2: <tipo2>
begin
    código...
end;
```

Variáveis Globais X Variáveis Locais

- Variáveis globais
 - Existem durante toda a execução do programa
 - Podem ser manipuladas em qualquer ponto do programa
- Variáveis locais
 - Existem durante a execução do procedimento/função
 - Podem ser manipuladas apenas dentro da função na qual foi declarada

Computação 1 – DEL-Poli/UFRJ

Prof. Miguel Campista

Vigésimo Primeiro Exemplo

```

Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
G:\Miguel\Programas\UFRJ\Computacao1\programas\proci1\proci1.pas 2
program GLOBAISLOCAL;
var
  <variável global>
  soma: Integer;
procedure Somar (a, b: Integer);
begin
  <=soma é uma variável global, enquanto a e b são variáveis locais= >
  soma := a + b;
end;
var
  a, b: Integer;
begin
  writeln ('Entre com as variáveis:');
  readln (a, b);
  Somar (a, b);
  writeln ('Soma: ', soma);
  writeln ('Tecla <ENTER> para sair:'); readln;
end.
    
```

Computação 1 – DEL-Poli/UFRJ

Prof. Miguel Campista

```

Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
G:\Miguel\Programas\UFRJ\Computacao1\programas\proci1\proci1.pas 2
program CALCULADORA;
procedure Somar (a, b: Integer);
begin
  resultado := a + b;
end;
procedure Subtrair (a, b: Integer);
begin
  resultado := a - b;
end;
procedure Multiplicar (a, b: Integer);
begin
  resultado := a * b;
end;
procedure Dividir (a, b: Integer);
begin
  resultado := a / b;
end;
<= Variáveis locais ao programa principal = >
var
  v1, v2: Integer;
  op: char;
begin
  writeln ('Entre com dois inteiros:');
  readln (v1, v2);
  writeln ('Entre com a operação desejada:');
  readln (op);
  case op of
    '+': Somar (v1, v2);
    '-': Subtrair (v1, v2);
    '*': Multiplicar (v1, v2);
    '/': Dividir (v1, v2);
  else writeln ('Operação desconhecida...');
  end;
  writeln ('Resultado: ', resultado:1:1);
  writeln; writeln ('Tecla <ENTER> para sair:'); readln;
end.
    
```

Passagem Parâmetro por Valor X por Referência

- Passagem de parâmetro por valor
 - Valor da variável é passada para função e é usada para inicializar uma variável local definida como um argumento da função
 - Após o término da execução da função, a variável é desalocada e o valor é perdido
- Passagem de parâmetro por referência
 - Endereço da variável é passada para função e é usada como referência para a posição da variável em memória
 - Após o término da execução da função, a variável é alterada

Computação 1 – DEL-Poli/UFRJ

Prof. Miguel Campista

```

Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
G:\Miguel\Programas\UFRJ\Computacao1\programas\proci1\proci1.pas 2
program CALCULADORA;
procedure Somar (a, b: Integer; var resultado: real);
begin
  resultado := a + b;
end;
procedure Subtrair (a, b: Integer; var resultado: real);
begin
  resultado := a - b;
end;
procedure Multiplicar (a, b: Integer; var resultado: real);
begin
  resultado := a * b;
end;
procedure Dividir (a, b: Integer; var resultado: real);
begin
  resultado := a / b;
end;
<= Variáveis locais ao programa principal = >
var
  v1, v2: Integer;
  op: char;
  resultado: real;
begin
  writeln ('Entre com dois inteiros:');
  readln (v1, v2);
  writeln ('Entre com a operação desejada:');
  readln (op);
  case op of
    '+': Somar (v1, v2, resultado);
    '-': Subtrair (v1, v2, resultado);
    '*': Multiplicar (v1, v2, resultado);
    '/': Dividir (v1, v2, resultado);
  else writeln ('Operação desconhecida...');
  end;
  writeln ('Resultado: ', resultado:1:1);
  writeln; writeln ('Tecla <ENTER> para sair:'); readln;
end.
    
```

Function

- Cumpre papel semelhante aos das procedures
 - Entretanto, retorna sempre um valor de um tipo pré-determinado
 - Valor de retorno é retornado no próprio nome da function

```

function NOME (<variáveis>) : <tipo_var_retorno>;
var
  var_local1: <tipo1>
  var_local2: <tipo2>
begin
  código...
end;
    
```

Computação 1 – DEL-Poli/UFRJ

Prof. Miguel Campista

```

Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
G:\Users\Miguel\Documents\computacao\programas\calc.pas
program CALCFUNC;

function Somar (a, b: Integer) : real;
begin
  Somar := a + b;
end;

function Subtrair (a, b: Integer) : real;
begin
  Subtrair := a - b;
end;

function Multiplicar (a, b: Integer) : real;
begin
  Multiplicar := a * b;
end;

function Dividir (a, b: Integer) : real;
begin
  Dividir := a / b;
end;

(* Variáveis locais ao programa principal *)
var
  a, b: Integer;
  op: char;
  resultado: real;
begin
  writeln ('Entre com dois inteiros');
  readln (a, b);
  writeln ('Entre com a operação desejada');
  readln (op);
  case op of
    '+': resultado := Somar (a, b);
    '-': resultado := Subtrair (a, b);
    '*': resultado := Multiplicar (a, b);
    '/': resultado := Dividir (a, b);
  else writeln ('Operação desconhecida...');
  end;
  writeln ('Resultado: ', resultado:1:1);
  writeln ('Tecla <ENTER> para sair'); readln;
end.

```

Recursividade

- Uma função pode chamar a mesma função para um problema reduzido
 - As chamadas são realizadas até que o problema seja mínimo
 - Caso base

Vigésimo Quinto Exemplo

```

Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
G:\Users\Miguel\Documents\computacao\programas\fatrec.pas
program FATREC;

function Fatorial (n: Integer) : Integer;
begin
  if n = 1 then
    Fatorial := 1;
  else
    Fatorial := n * Fatorial (n - 1);
  end;
end;

(* Variáveis locais ao programa principal *)
var
  v: Integer;
begin
  writeln ('Entre com o inteiro');
  readln (v);
  writeln ('Fatorial de ', v, ' é: ', Fatorial (v));
  writeln ('Tecla <ENTER> para sair'); readln;
end.

```

Units

- Arquivo em Pascal (*.pas) utilizado para pré-programar procedimentos e funções
 - Biblioteca
 - Muitas já são padrão do Pascal (Ex.: crt)
 - Outras podem ser criadas pelo programador
 - Sintax
 - **unit**
 - Define o nome da unit e deve ser o mesmo nome do arquivo
 - **Interface**
 - Define a interface dos procedimentos e funções
 - **Implementation**
 - Define a implementação de cada um dos procedimentos e funções da interface

```

exemplo.pas

unit exemplo;

interface
  function NOMEFUNC (<variáveis>):<tipo_var_retorno>;
  procedure NOMEPROC (<variáveis>);
end;

implementation
  function NOMEFUNC (<variáveis>):<tipo_var_retorno>;
  var
    ...
  begin
    ...
  end;
  procedure NOMEPROC (<variáveis>);
  var
    ...
  begin
    ...
  end;
end.

```

Units

```

usaUnit.pas

program USAUNITS;
uses exemplo;
var
  ...
begin
  ...
end.

```

Exemplo de Uso de Unit

```
unit FatUnit;  
interface  
function Fatorial (n: integer) : integer;  
implementation  
function Fatorial (n: integer) : integer;  
begin  
    if n = 1 then  
        <begin>  
            Fatorial := 1  
        <end>  
    else  
        <begin>  
            Fatorial := n * Fatorial (n - 1);  
        <end>  
    end;  
end.  
end.
```

Exemplo de Uso de Unit

```
program FATREC;  
uses FatUnit;  
<= Variáveis locais ao programa principal =>  
var  
    v: integer;  
begin  
    writeln ('Entre com o inteiro');  
    readln (v);  
    writeln ('Fatorial de ', v, ' eh: ', Fatorial (v));  
    writeln; writeln ('Tecla <ENTER> para sair'); readln;  
end.
```

Arquivos

- Permite escrever e ler dados da memória secundária
 - Operações principais
 - Assign (<variável>, <arquivo>)
 - Associa o nome lógico de um arquivo ao arquivo físico, o parâmetro <variável> é a indicação da variável do tipo arquivo e <arquivo> é o nome do arquivo a ser manipulado
 - Rewrite (<variável>)
 - Cria um arquivo para uso, utilizando o nome associado ao parâmetro <variável>. Caso o arquivo já exista, esta instrução o apaga para criá-lo novamente
 - Reset (<variável>)
 - Abre um arquivo existente, colocando-o disponível para leitura e escrita, utilizando o nome associado ao parâmetro <variável>.

Arquivos

- Permite escrever e ler dados da memória secundária
 - Operações principais
 - Write (<variável>, <dado>)
 - Escreve a informação <dado> no arquivo indicado
 - Read (<variável>, <dado>)
 - Lê a informação <dado> no arquivo indicado pela <variável>
 - Close (<variável>)
 - Fecha um arquivo em uso dentro de um programa. Nenhum programa deve ser encerrado sem antes fechar os arquivos abertos

Arquivos de Texto

- Cria-se variável do tipo text

```
program ARQUIVO;  
var  
    arquivo: text
```

- A variável é, então, associada a um nome de arquivo

```
program ARQUIVO;  
var  
    arquivo: text  
begin  
    assign (arquivo, 'arquivo.txt');  
    <sentenças>...  
    close (arquivo);  
end;
```

- Depois o programa é escrito manipulando a variável...

Arquivos de Texto

Exemplo de Escrita

```
program ESCREVERQUIVO;
var
  arquivo : text;
  nome: string;
  continua: char;
begin
  assign(arquivo, 'pessoas.txt');
  rewrite(arquivo);
  repeat
    write('Digite o Nome: ');
    readln(nome);
    writeln(arquivo, nome);
    write('Deseja continuar (s/n)? ');
    readln(continua);
  until (continua = 'N') or (continua = 'n');
close(arquivo);
end.
```

Exemplo de Leitura

```
program LERQUIVO;
var
  arquivo: text;
  nome: string;
begin
  assign(arquivo, 'pessoas.dat');
  reset(arquivo);
  while not eof(arquivo) do
    begin
      readln(arquivo, nome);
      writeln(nome);
    end;
close(arquivo);
writeln('Tecla <ENTER> para sair'); readln;
end.
```

Exemplo de Leitura com Registro

```
type
  registro = record
    nome: string;
    idade: integer;
  end;
var
  arquivo : text;
  reg: registro;
  continua: char;
begin
  assign(arquivo, 'pessoas.txt');
  reset(arquivo);
  repeat
    write('Digite o Nome: ');
    readln(reg.nome);
    write('Digite a idade: ');
    readln(reg.idade);
    write(arquivo, reg.nome);
    write(arquivo, ' ');
    writeln(arquivo, reg.idade);
    write('Deseja continuar (s/n)? ');
    readln(continua);
  until (continua = 'N') or (continua = 'n');
close(arquivo);
end.
```

Arquivos com Tipo Definido

- Arquivos denominados em Pascal como arquivos tipados
 - Arquivos do tipo binário, diferentes dos arquivos de texto
 - Operações de leitura e escrita são mais rápidas

```
program ARQUIVOTIPADO;
var
  arquivo: file of integer;
begin
  assign(arquivo, 'arquivo.bin');
  <sentenças>...
  close(arquivo);
end;
```

Exemplo de Escrita com Tipo Definido

```
program ARQUIVOTIPADO;
var
  arquivo: file of integer;
  regi: integer;
  continua: char;
begin
  assign(arquivo, 'inteiros.dat');
  rewrite(arquivo);
  repeat
    write('Digite o inteiro: ');
    readln(reg);
    write(arquivo, reg);
    write('Deseja continuar (s/n)? ');
    readln(continua);
  until (continua = 'N') or (continua = 'n');
close(arquivo);
end.
```

Exemplo de Leitura com Tipo Definido

```
program LERARQUIVOTIPADO;
var
  arquivo: file of integer;
  numero: integer;
begin
  assign(arquivo, 'inteiros.dat');
  reset(arquivo);
  while not eof(arquivo) do
    begin
      read(arquivo, numero);
      writeln(numero);
    end;
close(arquivo);
writeln('Tecla <ENTER> para sair'); readln;
end.
```

Exemplo de Escrita e Leitura com Tipo Definido

```
Free Pascal IDE
File Edit Search Run Compile Behave Tools Options Window Help
G:\Users\MiguelN...compucao1\programas\arquivo4.pas 6-13
program ESCREVERLEITURADO;
type
  registro = record
    nome: string;
    idade: integer;
  end;
procedure ler;
var
  arquivo_proc: file of registro;
  reg: registro;
begin
  assign(arquivo_proc, 'registro.dat');
  reset(arquivo_proc);
  while not eof(arquivo_proc) do
    begin
      read(arquivo_proc, reg);
      writeln('reg.nome: ', reg.nome);
    end;
  close(arquivo_proc);
end;
```

Exemplo de Escrita e Leitura com Tipo Definido

```
VAR
  arquivo : file of registro;
  reg: registro;
  continua: char;
begin
  assign(arquivo, 'registro.dat');
  reset(arquivo);
  if (IORESULT = 0) then
    begin
      writeln(arquivo);
      write(arquivo, reg);
      seek(arquivo, FilePos(arquivo) - 1);
    end;
  else
    seek(arquivo, FilePos(arquivo));
  repeat
    write('Digite o Nome: ');
    readln(reg.nome);
    write('Digite a Idade: ');
    readln(reg.idade);
    write(arquivo, reg);
    writeln('Deseja continuar (s/n)? ');
    readln(continua);
  until (continua = 'N') or (continua = 'n');
  close(arquivo);
  ler;
end;
```

Ponteiros

Exemplo de Ponteiro

```
program PONTIERS;
procedure enviainteiro (var a: integer);
var
  x: integer;
begin
  new(x);
  x := a;
  writeln('x: ', x);
  dispose(x);
end;
var
  ptrNome: ^string;
  ptrInt: ^integer;
begin
  writeln('Escreva um nome: ');
  new(ptrNome);
  ptrNome := 'Miguel';
  writeln(ptrNome);
  dispose(ptrNome);
  writeln('Escreva um inteiro: ');
  GetMem(ptrInt, 4);
  readln(ptrInt);
  writeln(ptrInt);
  enviainteiro(ptrInt);
  freeMem(ptrInt);
  write('Inicio EMER para terminar:'); readln;
end;
```

Exemplo de Ponteiro

```
program PONTIERS;
uses
  Crt;
type
  ptrMat = ^matriz;
  matriz = array [1..10] of integer;
var
  a: ptrMat;
  i, j, n, x: integer;
begin
  clrscr;
  writeln('Informe o numero de elementos:');
  readln(n);
  writeln;
  GetMem(a, n * SizeOf(integer));
  for i := 1 to n do
    begin
      write('Entre o ', i, 'o. elemento: ');
      readln(a[i]);
    end;
  for i := n - 1 downto 1 do
    for j := 1 to i do
      if a[i] > a[j] + 1 then
        begin
          x := a[i];
          a[i] := a[j] + 1;
          a[j] := x;
        end;
    end;
  clrscr;
  writeln('Classificacao de ', n, ' elementos:');
  for i := 1 to n do
    write(' ');
  writeln(a[i]);
  writeln;
  writeln('Escreva Ca, n = SizeOf (Integer);');
  writeln('Inicio EMER para terminar:'); readkey;
end;
```

Exemplo de Ponteiro

```
program LISTPONTIERS;
uses
  Crt;
type
  lista = ^tabela;
  tabela = record
    elemento: real;
    proc: lista;
  end;
var
  a: primeiro_elen, segundo_elen, lista_saida: lista;
  i: integer;
  nome, media: real;
  m: string;
  codigos: integer;
begin
  soma := 0;
  i := 1;
  clrscr;
  writeln('Matriz Dinamica');
  writeln;
  primeiro_elen := nil;
  repeat
    if primeiro_elen = nil then
      begin
        new(a);
        write('Informe o ', i, 'o. valor: ');
        readln(a);
        if n < 0 then
          begin
            writeln('Matriz Dinamica');
            primeiro_elen := a;
            a := nil;
            i := i + 1;
          end;
        else
          begin
            writeln('Matriz Dinamica');
            primeiro_elen := a;
            a := nil;
            i := i + 1;
          end;
        end;
    end;
  until (primeiro_elen = nil);
end;
```

Exemplo de Ponteiro

```
else
  begin
    sequencia_elen := a;
    new (a);
    write ('Informe o ', i, 'o. valor: ');
    readln (a);
    if a <> '' then
      begin
        wal (a, a^elemento, codigo);
        sequencia_elen^prox := a;
        a^prox := nil;
        i := i + 1;
      end;
    until a = '' do
      lista_saida := primeira_elen;
      writeln;
      write ('A lista eh: ');
      while lista_saida <> nil do
        begin
          writeln (lista_saida^elemento);
          soma := soma + lista_saida^elemento;
          lista_saida := lista_saida^prox;
        end;
        media := soma / i;
        writeln;
        write ('Media = ', media);
        writeln;
        write ('Tecla qualquer tecla para encerrar. ');
        readkey;
      end;
end;
```

Exemplo de Ponteiro

```
program LISTAPOINTER;
uses
  Crt;
type
  lista = ^dados;
  dados = record
    nome: string;
    salario: real;
    tempo: integer;
    prox: lista;
  end;
var
  cadfunc, atual, list_cf: lista;
  ent_nome: string;
  ent_salario: real;
  ent_tempo: integer;
  i, linha: integer;
  resp: char;
begin
  i := 1;
  cadfunc := nil;
  repeat
    clrscr;
    writeln ('cadfunc');
    writeln;
    new (cadfunc);
    writeln ('Entre o ', i, 'o. registro');
    writeln;
    write ('Nome.....: '); readln (ent_nome);
    write ('Salario.....: '); readln (ent_salario);
    write ('Tempo de servico..: '); readln (ent_tempo);
    cadfunc^nome := ent_nome;
    cadfunc^salario := ent_salario;
    cadfunc^tempo := ent_tempo;
    if (list_cf = nil) or (ent_nome < list_cf^.nome) then
      begin
        cadfunc^prox := list_cf;
        list_cf := cadfunc;
      end;
  until i = 0;
```

Exemplo de Ponteiro

```
else
  begin
    atual := list_cf;
    while (atual^prox <> nil) and
      (ent_nome > atual^prox^.nome) do
      atual := atual^prox;
      cadfunc^prox := atual^prox;
      atual^prox := cadfunc;
    end;
    i := i + 1;
    writeln;
    writeln ('Basta continuar (S)in ou (N)ao: ');
    readln (resp);
    until upcase(resp) = 'N' do
      i := i;
      linha := i;
      gotoxy (i, 1); write ('RegM');
      gotoxy (i, 2); write ('RegN');
      gotoxy (i, 3); write ('Salario');
      gotoxy (i, 4); write ('Tempo de Servico');
      while list_cf <> nil do
        begin
          gotoxy (i, linha); write (i);
          gotoxy (i, linha); write (list_cf^.nome);
          gotoxy (i, linha); write (list_cf^.salario);
          gotoxy (i, linha); write (list_cf^.tempo);
          list_cf := list_cf^prox;
          linha := linha + 1;
        end;
        i := i + 1;
      end;
      writeln;
      writeln ('Tecla qualquer coisa para encerrar. '); readkey;
    end;
```