

Técnicas de Programação Estruturada

Conceitos Básicos

- Lógica na Programação de Computadores
- Algoritmo
- Formas de Representação de Algoritmos
 - Gráficas
 - Fluxograma
 - Diagrama de Nassi-Schneiderman (Diagrama de Chapin)
 - Português Estruturado

Lógica na Programação de Computadores

- **A técnica mais importante no projeto da lógica de programas é chamada PROGRAMAÇÃO ESTRUTURADA, a qual consiste em uma metodologia de projetos objetivando:**
 1. Agilizar a codificação da escrita da programação;
 2. Permitir a verificação de possíveis falhas apresentadas pelos programas;
 3. Facilitar as alterações e atualizações dos programas.
- **E deve ser composta de quatro passos fundamentais:**
 1. Escrever as instruções em seqüências ligadas entre si apenas por estruturas seqüenciais, repetitivas ou de seleção;
 2. Escrever instruções em grupos pequenos e combiná-las;
 3. Distribuir módulos do programa entre os diferentes programadores que trabalharão sob a supervisão de um programador mais experiente;
 4. Revisar o trabalho executado em reuniões regulares.

Algoritmo

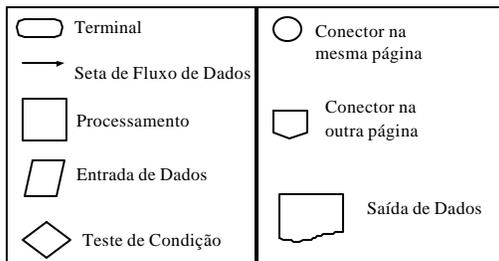
- **É um processo de cálculo matemático ou de resolução de um grupo de problemas semelhantes, em que se estipulam, com generalidade e sem restrições.**
- **São regras formais para a obtenção de um resultado ou da solução de um problema, podendo englobar fórmulas e expressões aritméticas.**

Formas de Representação de Algoritmos

- **Gráficas**
 - São ferramentas usadas pelo profissional que está envolvido diretamente com a programação, tendo como objetivo descrever o método e a seqüência dos passos de solução de um problema
 - Pode ser desenvolvido em qualquer nível de detalhe que seja necessário.
 - Usa diversos símbolos geométricos, os quais estabelecerão as seqüências de operações a serem efetuadas em um processamento computacional.

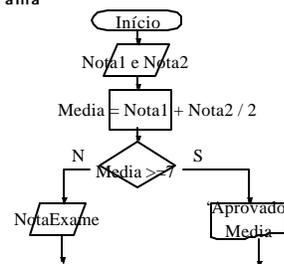
Formas de Representação de Algoritmos

• Fluxograma



Formas de Representação de Algoritmos

• Fluxograma



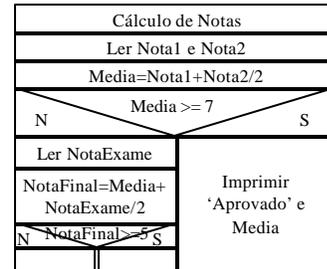
Formas de Representação de Algoritmos

- **Diagrama de Nassi-Schneiderman (Diagrama de Chapin)**
 - O diagrama N-S de Nassi-Schneiderman, também conhecido como diagrama de Chapin, é como um diagrama de fluxo em que se omitem as flechas de união e as caixas são contínuas e se escrevem em caixas sucessivas, e, como no fluxograma, pode-se escrever diferentes ações em uma caixa.



Formas de Representação de Algoritmos

- **Diagrama de Nassi-Schneiderman (Diagrama de Chapin)**



Formas de Representação de Algoritmos

- **Português Estruturado**
 - Tendo estabelecido os passos anteriores (algoritmo e diagrama gráfico), será efetuada a codificação. Esta fase obedece ao que está definido no diagrama gráfico, porém sempre deverá ser racionado com todas as variáveis que serão utilizadas dentro do programa.
 - Este relacionamento, além de definir os tipos de dados que serão utilizados, define também as variáveis e constantes que serão necessárias para manipular as informações fornecidas durante a execução de um programa.

Formas de Representação de Algoritmos

- **Português Estruturado**

```

programa SOMA_NUMEROS
var
  X : inteiro
  A : inteiro
  B : inteiro

inicio
  leia A
  leia B
  X ← A + B
  escreva X
fim
    
```

Tipos de Dados e Instruções Primitivas

Tipos de Dados, Constantes e Variáveis

- Tipos de Dados
 - Tipos Inteiros
 - Tipos Reais
 - Tipos Literais
 - Tipos Lógicos
- Constantes
- Variáveis
- Comentários

Tipos de Dados

- Os dados são representados pelas informações a serem processadas por um computador. Estas informações estão caracterizadas por quatro tipos de dados, a saber: dados numéricos inteiros e reais, dados literais e dados lógicos.
- **Tipos Inteiros**
 - São tipos inteiros, os dados numéricos positivos ou negativos, excluindo-se destes qualquer valor fracionário. Ex.: -35, 0, 56, +12.
- **Tipos Reais**
 - São tipos reais, os dados numéricos positivos, negativos e números fracionários. Ex.: 35, 0, -56, 1.2, -45.897, 6E24.
- **Tipos Literais**
 - São tipos literais as sequências contendo letras, números e símbolos especiais.
 - Uma sequência de caracteres deve ser indicada entre apóstrofes. Ex.: 'Programação', 'Rua Direita n. 3'.
- **Tipos Lógicos**
 - São tipos lógicos ou booleanos os dados com valores verdadeiro e falso ou true e false. Ex.: .true, .false.

Constantes

- As constantes podem ser numéricas, lógicas ou literais
- CONSTANTES NUMÉRICAS
 - 25; +3241; -315
 - 3.141516; -0.53
 - 7.8E3; -31521E-3; 21E+3

Variáveis

- Para podermos guardar informações no computador, necessitamos de um **tipo**, um **nome** e o sua **descrição**.
- var → tipo
- Idade → nome ou identificador
- integer → descrição
- Logo:
 - var Idade: integer;
- Significa
 - A variável **Idade** identifica-se com valores **inteiros**.

Variáveis

- Identificadores permitidos:

A	Nota	Matricula	NotaFinal
X5	Sala2	LucroTotal	
- Identificadores Não Permitidos

5B	Km/h	%10*2+5	?Not a
----	------	---------	--------
- Palavras reservadas:** palavras que o Pascal usa para outras funções e que voce não pode aproveitá-la como identificador.
 - real, integer, boolean, char;

Variáveis

```
var Nota, NotaExame: real;
    Codigo: integer;
    Aprovado: boolean;
    LetraClasse: char;

type Indices = 0..63;
    Letras = 'A'..'Z';
    FaixaDeValores = -3..3;
    Sequencia = 'D'..'G';
var Let, Ch: Sequencia;

var Nome, Sobrenome: string[15];
    Endereco: string;           { 255 char }
```

Expressões Aritméticas

Prioridades:

* / div mod
+ -

Em: 11 **div** 4 dá resultado 2
11 **mod** 4 dá resultado 3

Só podem ser utilizados em operandos inteiros;

Expressões válidas em Pascal

X+Y A*B+ SqRt(C*2+G)+ Exp (X)
X-Y TOTAL/N
2*N SOMA*SOMA

sqrt(P) calcula raiz quadrada

exp (X) exponencial e^X

$$A - Bx \left(C + \frac{D}{E-1} - F \right) + G$$

$$A - B * (C + D / (E - 1) - F) + G$$

$$\sqrt{Px(P-A)x(P-B)x(P-C)}$$

$$\text{SqRt}(P*(P-A)*(P-B)*(P-C))$$

Expressões Aritméticas

- O sinal da multiplicação é indicado com asterisco (*) e nunca deve ser subentendido.
- A divisão é indicada com a barra (/).
- Não existe sinal para a potenciação e radiciação.
- Uma expressão que só tenha operandos inteiros e operadores *, mod, div, + e - terá como resultado um valor inteiro. Se pelo menos, um dos operadores for real e se os operadores forem *, /, + ou -, o resultado da expressão será real.
- O operador / sempre conduz a um resultado do tipo real, mesmo que os dois operando sejam inteiros.
- Os parênteses servem para indicar uma sequência de cálculo da expressão aritmética diferente da que seria indicada exclusivamente como uso das prioridades pré-estabelecidas.

Expressões Aritméticas

Funções predefinidas:

abs(x) : abs(A-B*B*B) : abs(-17) = 17
 trunc(x) : trunc(sqrt(B)+X) : trunc(3.3333 + 1) = 4
 round(x) : round(A-X) : round(7,5) = 8 : round(-1,3) = -1
 sqr(x) : sqr(A) : sqr(5) = 25
 sin(x) = seno de x
 cos(x) = co-seno de x
 arctan(x) = arco-tangente de x

Outro exemplo:

Sejam A e B, inteiros e X e Y reais...

(B+Y) div (X+1); (B+Y) mod (X+1) (certo ou errado?)

Expressões Lógicas

O resultado de uma relação é sempre lógico, **false** ou **true**.

Prioridade	Operadores
1ª	not
2ª	* / div mod and
3ª	+ - or
4ª	= <> < <= >= > in

Relações	Descrição
Nome = 'Joao'	Variável Nome é igual a Joao
A <> B	A é diferente de B
A < B	A é menor que B
A > B	A é maior que B
A <= B	A é menor ou igual a B
A >= B	A é maior ou igual a B
in	Contido em (conjunto)

Expressões Lógicas

VARIÁVEIS					RELAÇÕES		
X	Y	Z	COR	NOME	X*Y>Z	COR='azul'	Nome <> 'Jose'
1	2	5	'azul'	'Paulo'	false	true	true
4	3	1	'verde'	'Jose'	true	false	false
1	1	2	'branco'	'Pedro'	false	false	true
1	2	1	'azul'	'Jose'	true	true	false

```
var x, y, z : integer;
    cor, nome : string[15];
```

Expressões Lógicas

Sendo P=true, Q=false, R=false, S=true,...

- O valor lógico das conjunções:
 P and S = true Q and T = false
- O valor lógico das disjunções:
 P or S = true Q or R = false
- O valor lógico das negações:
 not P = false not R = true

Sendo X=2, Y=5, Z=9 e Nome='Maria', logo.....
 (X + Y > Z) and (Nome = 'Maria')
 (2 + 5 > 9) and ('Maria' = 'Maria')
 false and true
 false

Comandos de Atribuição

identificador := expressão ;

```
var a, b, n, soma, x, y : integer;
    k, media, total : real;
    cod, sim, teste : boolean;
    cor : string;
```

São válidas:

```
k:=1;
cor := 'verde';
teste := 'false';
a := b;
media :=soma/n;
sim := (x = 0) and (y <> 2);
```

Inválidas:

```
n := x / y;
```

Write e Writeln

Exibir todos os tipos de dados no vídeo.
 Write: escreve o parâmetro, mantendo o cursor do lado.
 Writeln passa o cursor para a próxima linha.
 Write('Digite um numero inteiro -----> ');
 Readln(i);

Read e Readln

Faz leitura de dados via teclado.
 Read: cada tecla digitada, ecoada para o vídeo permanecendo no mesmo lugar após ENTER.
 Readln: o cursor passa para a próxima linha.
 Write('Digite um numero inteiro -----> ');
 Readln(i);

ClrScr

Esta procedure tem a finalidade de limpar a tela de vídeo e colocar o cursor na primeira coluna da primeira linha. A tela de vídeo, dividida em 80 colunas e 25 linhas. O canto superior esquerdo tem coordenadas (1,1) e o inferior direito (80,25).

Writeln(lst, 'x')

Podemos enviar dados para a impressora através das procedures Write e Writeln. Para tanto, devemos colocar, antes dos parâmetros a serem enviados. O nome lógico **LST**:

```
Writeln('isto vai para o vídeo');  
Writeln(lst, 'isto vai para a impressora', ' e isto também');
```

Estrutura Seqüencial

Todos os programas em Pascal, a menos que haja indicação em contrário, devem ser executados seqüencialmente, na ordem que aparecem escritos, onde devem estar separados por \rightarrow ; \leftarrow . Como end é um delimitador, neste caso, não é necessário.

```
Program operadores_logicos ;  
Uses CRT;  
  
Var x,y : boolean;  
  
Begin  
  x:=TRUE;  
  y:=FALSE;  
  Writeln( x OR y );      { * escreve TRUE * }  
  Writeln( x AND y );     { * escreve FALSE * }  
  Writeln( x XOR y );     { * escreve TRUE * }  
End.
```

Exemplo:

```
Var i : integer;  
    r : real;  
    c : char;  
    s : string[10];  
  
Begin  
  ClrScr;  
  Write('Digite um numero inteiro -----> ');  
  Readln(i);  
  Write('Digite um numero real -----> ');  
  Readln(r);  
  Write('Digite um caractere -----> ');  
  Readln(c);  
  Write('Digite uma string -----> ');  
  Readln(s);  
  Writeln;Writeln; { * pula duas linhas * }  
  Writeln(i);  
  Writeln(r);  
  Writeln(c);  
  Writeln(s);  
End.
```

Estrutura Condicional: IF

Simples, Composta e Encadeada

Tem por finalidade tomar uma decisão. Sendo a condição Verdadeira, serão executadas as instruções entre **if**, **then**, caso contrário após **else**.

```
PROGRAM Maior;  
VAR  
  A,B : INTEGER;  
BEGIN  
  WRITE('Digite os valores A e B');  
  IF A > B THEN  
    BEGIN  
      WRITE('Olhe a resposta abaixo');  
      WRITE('A é maior que B')  
    END  
  ELSE  
    IF A < B THEN  
      BEGIN  
        WRITE('Olhe a resposta abaixo');  
        WRITE('A é menor que B')  
      END  
    ELSE  
      WRITE('A é igual a B')  
  END.  
END.
```

Estrutura Condicional: IF

```
program triangulos;  
var x, y, z: real;  
  
begin  
  read(x, y, z);  
  if(x < y+z) and (y < x+z) and (z < x+y)  
  then if (x=y) and (x=z)  
       then write('Triangulo equilatero')  
       else if (x=y) or (x=z) or (y=z)  
            then write('Trinagulo isóceles')  
            else write('Triangulo escaleno')  
  else write('Não existe triangulo')  
end.
```

Estrutura Condicional: Case

Esta instrução nos permite selecionar uma opção baseada no valor de uma variável ou expressão.

A expressão ou variável no comando Case deve ser do tipo simples, normalmente Char ou Integer.

Após a avaliação da expressão, seu valor ou o valor da variável, comparado com os diversos valores discriminados.

Se houver algum que satisfaça, o comando subsequente ser executado.

Estrutura Condicional: Case

<pre>Case <expressão ou variável> of <valor 1> : Comando_1; <valor 2> : Comando_2; . . . End;</pre>	<pre>Case <expressão ou variável> of <valor 1> : Begin comando_1; comando_2; . . . End; <valor 2> : Begin comando_1; . . . End; . . . <valor n> : Begin comando_1; comando_2; . . . End; End;</pre>
---	---

Estrutura Condicional: Case

Neste caso, se o resultado da expressão ou o valor da variável não satisfizer nenhum dos valores discriminados, então o comando que estiver na frente da cláusula Else ser executado.

```
Case <expressão ou variável> of
  <valor 1> : Comando_1;
  <valor 2> : Comando_2;
  . . .
  <valor n> : Comando_n;
Else Comando;
End;
```

Estrutura de Repetição: FOR ... DO

Este comando permite que um grupo de comandos sejam repetidos um certo número de vezes. Sintaxe geral:

For <variável> := <valor inicial> to/downto <valor final> do <comando>;

A variável deve ser do tipo integer, char ou Boolean. A variação de variável entre valor inicial e valor final ser crescente e de um em um, quando utilizamos a palavra to, e decrescente de um em um, quando utilizamos a palavra downto.

```
Program Exemplo_1;
Uses CRT;
Var i : Integer;
Begin
  ClrScr;
  For i:=10 to 15 do WriteLn(i);
End.
```

Estrutura de Repetição: FOR ... DO

<pre>PROGRAM LeEscreve ; VAR Nome : STRING; Cont : INTEIRO; BEGIN FOR Cont : = 1 TO 20 DO BEGIN READ(Nome); WRITE(Nome); END; END.</pre>	<pre>Program Exemplo_3; Uses CRT; Var i : Integer; Begin ClrScr; For i:=1 to 20 do Begin Write('Valor de i --> '); Write(i:3); Write('_quadrado dei='); WriteLn(i*i:5); End; End.</pre>
--	---

Estrutura de Repetição: REPEAT...UNTIL

Repete um bloco de instruções até que uma certa condição seja satisfeita. Sua sintaxe:

```
Repeat
  Comando_1;
  Comando_2;
  . . .
Until (expressão_lógica);
```

Neste caso, todos os comandos entre as palavras reservadas Repeat e Until serão executadas, até que a expressão lógica seja verdadeira (TRUE), obviamente, devemos ter o cuidado para que ela venha a ser TRUE em determinado momento, pois caso contrário, teremos um LOOP INFINITO.

Estrutura de Repetição: REPEAT...UNTIL

<pre>PROGRAM LeEscreve ; VAR Nome : STRING; Total: INTEGER; BEGIN Total := 0; REPEAT READ(Nome); WRITE('Nome=',Nome); Total := Total + 1; UNTIL Total >=20; END.</pre>	<pre>Program Exemplo; Uses CRT; Var i : Integer; Begin ClrScr; i:=1; Repeat WriteLn(i); i:=i+1; Until i=10; End.</pre>
---	--

Estrutura de Repetição: **WHILE ... DO**

A estrutura While..Do permite controlar o número de vezes que uma instrução ou bloco de instruções ser executado. Ela difere da Repeat.Until porque esta só avalia a expressão lógica no final do primeiro Loop, enquanto que a instrução While ..Do avalia a expressão lógica antes da primeira interação, isto significa que, eventualmente, pode não ocorrer sequer a primeira interação.

While <expressão_lógica> Do <comando>;
ou

```
While <expressão_lógica> Do
Begin
  comando_1;
  comando_2;
  ...
End;
```

Estrutura de Repetição: **WHILE ... DO**

```
PROGRAM LeEscreve ;
VAR Nome : STRING;
    Total: INTEGER;
BEGIN
  Total:=0;
  WHILE Total<20 DO
  BEGIN
    READ(Nome);
    WRITE('Nome=',Nome);
    Total:=Total + 1;
  END;
END.
```

```
Program Exemplo_1;
Uses CRT;
Var i : Integer;
Begin
  ClrScr;
  i:=0;
  While (i<11) Do
  Begin
    Writeln(i);
    i:=i+1;
  End;
End.
```

Exemplos

soma dos pares de 100 a 200

```
PROGRAM SomaPares ;
VAR Soma, Par: INTEGER;
BEGIN
  Soma:=0;
  Par:= 100;
  WHILE Par<=200 DO
  BEGIN
    Soma:=Soma+Par;
    Par:=Par + 2
  END;
  WRITE (Soma);
END.
```

```
Program SomaPares ;
Var Soma, Par : INTEGER;
BEGIN
  Soma:=0;
  Par:= 100;
  REPEAT
    Soma:=Soma+Par;
    Par:=Par + 2;
  UNTIL Par > 200;
  WRITE (Soma)
End.
```

Exemplos

soma dos pares de 100 a 200

```
PROGRAM SomaPares ;
VAR Soma, Par: INTEGER;
BEGIN
  Soma:=0;
  FOR Par:=0 to 50 DO
  Soma:=Soma+Par*2+100;
  WRITE (Soma);
END.
```

```
Program SomaPares ;
Var Soma, Par : INTEGER;
BEGIN
  Soma:=0;
  FOR Par:=100 downto 50
  DO
    Soma:=Soma+Par*2;
    WRITE (Soma);
  End.
```

Estrutura de Dados

Varáveis Compostas Homogêneas :

Conhecidas no Pascal como arrays, correspondem a posições de memória identificadas por um único nome, individualizadas por índice e cujo conteúdo é de um mesmo tipo.

O nome de uma variável composta é um identificador que obedece as mesmas regras de formação de identificadores de variáveis simples. Para referenciar um elemento, é necessário colocar o nome da variável, seguindo de um ou mais índices, entre colchetes.

Alfabeto

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

O conteúdo armazenado na posição 3 de Nota, ou seja Nota[3] é C.

I:= 6;

Então Nota[i] é F.

Estrutura de Dados

Programa: Leia um conjunto de 10 notas, digitadas uma em cada linha, armazene-as na variável composta. Nota e calcular a sua média.

```
program Project2;
.....
begin
  ...
  soma := 0;
  for i:= 1 to 10 do
  begin
    readln (nota[i]);
    soma:=soma+nota[i];
  end;
  media:=soma/10;
  ...
end.
```

Pl.1: Codificar em Pascal a estrutura que compare o conteúdo da sétima posição da var Nota com valor 60.

```
program ProjectA;
.....
begin
  ...
  if nota[7] > 60
  then.....
  else.....
  ...
end.
```

Estrutura de Dados

P1.2: Com INDICE: Codificar em Pascal a estrutura que compare o conteúdo da sétima posição da var Nota com valor 60.

```

program ProjectA;
.....
begin
...
indice:=7;
if nota[indice] > 60
then.....
else .....
...
end.
    
```

P1.3: Agora que calcule e escreva o número de alunos com nota superior a média.

```

program ProjectA;
.....
begin
...
quantidade:= 0;
for i:= 1 to 10 do
if nota[i] > media then
quantidade := quantidade + 1;
writeln ('Número de alunos com nota superior a media : ', quantidade);
...
end.
    
```

Estrutura de Dados

A variável matriz é constituída por nove elementos dispostos em três linhas de três colunas.

	1	2	3
1	12	19	32
2	69	14	37
3	11	112	42

Matriz [2,2] referencia o elemento da segunda linha e segunda coluna, cujo conteúdo é 14. Matriz[2,3] referencia o elemento da segunda linha e terceira coluna, cujo conteúdo é 37

Uma outra notação para referenciar o elemento da segunda linha e segunda coluna é matriz[2][2]; já a matriz [2][3] faz referência ao elemento da segunda linha e terceira coluna.

Estrutura de Dados

Variáveis Compostas Unidimensionais : Vars onde seu conteúdo é referenciado por um único índice.

Nota:

72	43	94	67	96	84	84	15	78	42
----	----	----	----	----	----	----	----	----	----

Peso:

46.5	66.1	99.5	33.7	44.8
------	------	------	------	------

Idade:

27	28	15	12	64
----	----	----	----	----

Nome:

J	O	S	E
---	---	---	---

Declaração: identificadores: array [li..ls] of t;
 Identificadores: nomes associados as vars que se deseja declarar;
 li..ls: limite inferior..limite superior do intervalo de variação do índice;
 t: tipo de componente da variável.
 Exemplo:
 nota: array [1..10] of real;

Estrutura de Dados

Preencher a variável composta M com o valor 2 e depois escrever o seu conteúdo.

```

program Preenche;
var m: array[1..6] of integer;
i: integer;
begin
for i := 1 to 6 do
m[i] := 2;
for i := 1 to 6 do
write(m[i], ' ');
end.
    
```

Fazer um programa que leia uma variável de 100 elementos numéricos e verificar se existem elementos iguais a 30. Se existirem, escrever as posições em que estão armazenados.

```

program Iguais;
var numeros: array[1..100] of integer;
i: integer;
begin
for i := 1 to 100 do
readln(numeros[i]);
for i := 1 to 100 do
if numeros[i] = 30 then
writeln(i)
end.
    
```

Estrutura de Dados

Variáveis Compostas Multidimensionais : Vars onde seu conteúdo é referenciado por mais de um índice.

Escaninho:

	1	2	3
1	211	412	630
2	936	118	1120
3	437	591	667
4	331	212	430

 => 4lin X 3col = 12 elementos

Declaração: identificadores: array [li1..ls1, li2..ls2, li3..ls3] of t;
 Identificadores: nomes associados as vars que se deseja declarar;
 li..ls: limite inferior..limite superior do intervalo de variação do índice;
 t: tipo de componente da variável.
 Exemplo:
 var escaninho: array[1..4,1..3] of real;

Estrutura de Dados

Prog: Leia uma matriz 4x4, multiplique os elementos da diagonal principal por um constante K, também lida e escreva a matriz resultante. Seja A a matriz descrita e K = 3.

$$A = \begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}$$

$$A \times K = \begin{bmatrix} 3 & 5 & 9 & 13 \\ 2 & 18 & 10 & 14 \\ 3 & 7 & 33 & 15 \\ 4 & 8 & 12 & 48 \end{bmatrix}$$

```

program Diagonal;
var a: array[1..4,1..4] of integer;
i, j, k: integer;
begin
for i:=1 to 4 do
for j:=1 to 4 do
read(a[i,j]);
readln(k);
writeln('K = ', k);
{calculo da diagonal}
for i:=1 to 4 do
a[i,i] := a[i,i] * k;
for i:=1 to 4 do
begin
for j:= 1 to 4 do
write(a[i,j]:3, ' ');
writeln
end;
end.
    
```

Exercício

Ordenação Numérica

Faça um programa que:

- Pergunte quantos números devem ser ordenados;
- Ordene-os;
- Mostre a listagem dos números ordenados na tela.

Utilize-se do fragmento de código ao lado.

```

program ordenacao;
var  n,i,inic: 1..100;
    x: array [1..100] of real;
    temp: real;

writeln;
writeln('Dados Ordenados: ');
writeln;
for i:=1 to n do
    writeln('x [',i:3,']= ',x [i]:4:1);
end.

```

Exercício

Ordenação Numérica

```

Borland Pascal Version 7.0
Quantos sao os numeros: 4

x [ 1]= 21
x [ 2]= 23
x [ 3]= 25
x [ 4]= 20

Dados Ordenados:

x [ 1]= 0.0
x [ 2]= 1.0
x [ 3]= 3.0
x [ 4]= 5.0

```

Exercício

Ordenação Numérica - Código Completo

```

program ordenacao;
var  n,i,inic: 1..100;
    x: array [1..100] of real;
    temp: real;
begin
write('Quantos sao os numeros: ');
readln(n);
writeln;
for i:=1 to n do
begin
write('x [',i:3,']= ?');
readln(x[i]);
end;

```

```

for inic:=1 to n-1 do
for i:=inic+1 to n do
if x[i] < x[inic] then
begin
temp:=x[inic];
x[inic]:=x[i];
x[i]:=temp;
end;
writeln;
writeln('Dados Ordenados: ');
writeln;
for i:=1 to n do
writeln('x [',i:3,']= ',x [i]:4:1);
end.

```

Exercício

MediaNotas

Faça um programa que:

- Leia 10 notas;
- Mostre na tela a quantidade de alunos que tiveram nota superior a media.

Utilize-se do fragmento de código ao lado.

```

program medianotas;
var  nota: array[1..10] of integer;
begin
soma:=0;

readln(nota[i]);

media:= soma /10;
writeln('Numero de alunos com nota superior a media = ',qtde);
end.

```

Exercício

MediaNotas

```

Borland Pascal Version 7.0 Copyright (c) 1983
Digite aqui a nota[1]: 34
Digite aqui a nota[2]: 42
Digite aqui a nota[3]: 95
Digite aqui a nota[4]: 60
Digite aqui a nota[5]: 40
Digite aqui a nota[6]: 70
Digite aqui a nota[7]: 70
Digite aqui a nota[8]: 46
Digite aqui a nota[9]: 51
Digite aqui a nota[10]: 80

Numero de alunos com nota superior a media = 5

```

Exercício

MediaNotas - Código Completo

```

program medianotas;
var  nota: array[1..10] of integer;
    soma, qtde, i: integer;
    media: real;
begin
soma:=0;
qtde:=0;
for i:=1 to 10 do
begin
write('Digite aqui a nota[',i,']: ');
readln(nota[i]);
soma:=soma + nota[i];
end;

```

```

media:= soma /10;
for i:= 1 to 10 do
if nota[i] > media then
qtde:=qtde +1;
writeln;
writeln('Numero de alunos com nota superior a media = ',qtde);
end.

```

Exercício

MaxMin

Faça um programa que:

- Monta uma lista com valores min e max;
- Como resultado, mostre o maior e o menor da lista;
- Também mostre a lista.

Utilize-se do fragmento de código:

```

Program MinMax;
const maxsize = 20;
type listsize = 1..maxsize;
while item < maxsize do
begin
  first:=a[item];
  second:=a[item+1];
  if first > second then
  begin
    if first > max then max:= first;
  
```

```

    end
  else
  begin
    if second > max then
      max:=second;
    end;
    item:=item+2
  end;
end.
  
```

Exercício

MaxMin

```

Digite os numeros para selecionar: 1
Digite os numeros para selecionar: 2
Digite os numeros para selecionar: 3
Digite os numeros para selecionar: 4
Digite os numeros para selecionar: 5
Digite os numeros para selecionar: 6
Digite os numeros para selecionar: 7
Digite os numeros para selecionar: 8
Digite os numeros para selecionar: 9
Digite os numeros para selecionar: 11
Digite os numeros para selecionar: 22
Digite os numeros para selecionar: 33
Digite os numeros para selecionar: 44
Digite os numeros para selecionar: 55
Digite os numeros para selecionar: 66
Digite os numeros para selecionar: 77
Digite os numeros para selecionar: 88
Digite os numeros para selecionar: 99
Digite os numeros para selecionar: 12
Digite os numeros para selecionar: 13

Numero maior : 99
Numero menor : 1
  
```

Exercício

MaxMin - Código Completo

```

Program MinMax;
{este programa monta uma lista com
valores min e max e depois mostra o
maior e o menor da lista}
const maxsize = 20;
type listsize = 1..maxsize;
var item: listsize;
    min, max, first, second: integer;
    a: array[listsize] of integer;
begin
  for item:=1 to maxsize do
    readln(a[item]);
  min:=a[1];
  max:=min;
  item:=2;
  while item < maxsize do
  begin
    first:=a[item];
    second:=a[item+1];
    if first > second then
  
```

```

    if first > max then max:= first;
    if second < min then
      min:=second
    end
  else
  begin
    if second > max then
      max:=second;
    if first < min then min:= first
    end;
    item:=item+2
  end;
  if item = maxsize then
    if a[maxsize] > max then
      max:=a[maxsize]
    else
      if a[maxsize] < min then
        min:=a[maxsize];
        writeln(max, min);
      end.
  
```

Estrutura de Dados

Variáveis Compostas Heterogêneas :: Registros :: São conjuntos de dados logicamente relacionados, mas de tipos diferentes (num, literal, lógico).

Inscrição	Nome	09214	José da Silva
Rua	Num	CEP	Sagres 210 30250-160
CPF	274165106-44		
Sexo	Data Nascimento	M	03 02 55
Tem Dependentes	Horas Trab.	falso	22,5

Assim, constituem as informações cadastrais do mesmo indivíduo.

O conceito de registro visa facilitar o agrupamento de variáveis que não são do mesmo tipo, mas que guardam estreita relação lógica.

A referência ao conteúdo de um componente do registro é indicada pela notação:

identificador do registro . identificador do componente

Estrutura de Dados

DECLARAÇÃO: a criação de registros é feita assim:

```

ListaDeIdent: record componentes
end;
  
```

Nome			
Rua	Num	CEP	
CPF			
Sexo	HorTr ₁	HorTr ₂	HorTr ₃
Nascimento			
Tem Dependentes			

```

var...;
...
cadastro: record nome,
  rua: string[25];
  num: integer;
  cep: real;
  cpf: real;
  sexo: char;
  ht: array[1..3] of real;
  nascimento: real;
  temdep: boolean
end;
  
```

CEP e CPF foram declaradas como real porque os valores excedem 32767.

É possível declarar algumas vars como literal.

Estrutura de Dados

Outro ex1: Declarar o registro CAD

Nome	
Endereço	
CPF	
Sexo	HorTr
Nascimento	

```

var...;
CAD: record nome: string[30];
  end: record rua: string[30];
  num: integer;
  cep: real;
  end;
  cpf: real;
  sexo: char;
  ht: array[1..3] of real;
  nascimento: real;
  temdep: boolean
end;
  
```

Outro ex2: Declarar o registro CAD

```

type end = record rua: string[30];
  num: integer;
  cep: real;
end;
var...;
.....
CAD: record nome: string[30];
  end: end;
  cpf: real;
  sexo: char;
  ht: array[1..3] of real;
  nascimento: real;
  temdep: boolean
end;
  
```

Estrutura de Dados

Varíáveis Compostas Heterogêneas :: Conj. de Registros : É referenciado por um mesmo nome e individualizado por índices.

Identificador da variável[i].identificador do registro.identificador do componente

Exemplo: Mercadorias[5].Camisas.Nome

Contas

1
2
3
4
5

Cliente

Nome		
Rua	Numero	CPF
Saldo		

Type registro= record nome,

 rua: string[20];

 numero, cep,

 saldo: real

 end;

Var contas: array [1..5] of registro;

Estrutura de Dados

Programa: Escreva um programa que dados 500 códigos de profissão, emita o nome das profissões correspondentes.

Estrutura de Dados

1
2
3
...
100

numérico literal

Código	Nome
--------	------

```

program profissoes;
var tabela: array [1..100] of record codigo:
integer;
                  nome: string[12];
                  end;
cod_desejado, i, k: integer;
begin
for i:=1 to 100 do
  readln(tabela[i].codigo, tabela[i].nome);
for k:=1 to 500 do
  begin
    readln(cod_desejado);
    i:=1;
    while (tabela[i].codigo <> cod_desejado) and
      (i < 100) do
      i:=i+1;
    if tabela[i].codigo = cod_desejado then
      writeln (cod_desejado, ' ', tabela[i].nome)
    else
      writeln ('Codigo invalido: ', cod_desejado);
    end;
  end.

```

Estrutura de Dados

Arquivos:
São estruturas de dados armazenados em meio magnético ou outro, mas fora da memória principal, por dois motivos: a RAM é pequena e volátil.

Arquivos :: Binário: Utiliza-se da numeração binária para armazenar tanto as informações numéricas como as literais.

Exemplo: Para armazenar o numeral 65, ele se utiliza da seguinte forma:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	1

65 =

Sendo formado por uma seqüência de registros, todos do mesmo tipo, que sua vez são formados por campos. Um registro é a menor qtd de informação que pode ser lida/escr num arquivo binário. Um campo é a informação representada pelo valor de uma variável simples.

Estrutura de Dados

A organização dos registros é interna e seqüencial.

DECLARAÇÃO:
type nome-do-tipo= file of tipo-do-registro;
var lista-de-identificadores: nome-do-tipo;
- ou - var lista-de-identificadores: file of tipo-do-registro;

Onde:
type e var : são palavras chaves que permitem declarar tipos e variáveis;
nome do tipo : é um identificador associado ao tipo do arquivo;
file of: são palavras-chaves que caracterizam os arquivos do tipo binário;
tipo-de-registro: indica o tipo dos registros que formar os arquivos binários;
lista-de-identificadores: são os nomes associados a arquivos binários.

Por exemplo:
type ArqInt = file of integer; ou var Notas: file of integer;
var Notas: ArqInt;

Dá o nome ArqInt para o tipo dos arquivos em que cada registro é um inteiro.

Estrutura de Dados

```

type endereco = record nome, rua, cidade : string[25];
                  numero: integer;
                  end;
var Agenda: file of endereco;

```

Campos: nome rua num cidade

Registro_ Endereco

Arquivo_ Agenda

Estrutura de Dados

Tratando um arquivo binário :
Todo arq tem dois nomes: um *interno*, declarado dentro do programa e outro *externo*, usado pelo sistema operacional. O Pascal associa os dois nomes a um arquivo com o seguinte comando:
assign (nome-interno-do- arquivo, nome -do- arquivo-externo);
ex: assign (Agenda, '2002-INE.BIN');

Utilizando-se do comando string, também é possível dar nome a um arquivo....:

```

...
var externo: string;
begin
  readln(externo);
  assign(Agenda, externo);
...

```

mas cuidado: Cada arquivo binário possui um apontador, que é uma variável que indica qual é a posição do registro que está pronto para ser usado. Quando iniciar o arquivo, é necessário abri-lo.

Estrutura de Dados

reset (nome-interno-do-arquivo) : o qual é usado para se abrir um arquivo já existente e faz o seu apontador indicar o primeiro registro.

```
.....
begin
  assign(binario, 'dados.bin');
  reset(binario);
.....
```

rewrite (nome-interno-do-arquivo) : o qual é usado para criar um arquivo novo, ou sobrepor este arquivo no já existente. Também faz o seu apontador indicar o primeiro registro.

```
.....
begin
  write('Quantidade de pessoas a ler: ');
  readln(n);
  assign(binario, 'dados.bin');
  rewrite(binario);
.....
```

Estrutura de Dados

close (nome-interno-do-arquivo) : o qual é usado para fechar um arquivo criado ou alterado. Não se usa quando o arquivo só foi lido.

```
.....
begin
  writeln:
  writeln('Digite o nome da pessoa e
  sua idade. ');
  readln(umapessoa.nome,
  umapessoa.idade);
  write(binario, umapessoa)
end;
close(binario)
end.
```

read (nome-interno-do-arquivo, lista-de-registros) : (nome interno do arquivo no qual os registros serão lidos , nomes das variáveis que receberão os valores lidos de acordo com sua declaração para os registros do arquivo binário)

```
....
begin
  read(binario, umapessoa);
  writeln(umapessoa.nome:15,
  umapessoa.idade:4)
end;
.....
```

Estrutura de Dados

write (nome-interno-do-arquivo, lista-de-registros) : (nome interno do arquivo no qual os registros serão escritos , nomes das variáveis que receberão os valores escritos de acordo com sua declaração para os registros do arquivo binário)

```
.....
begin
  writeln;
  writeln('Digite o nome da pessoa e
  sua idade. ');
  readln(umapessoa.nome,
  umapessoa.idade);
  write(binario, umapessoa)
end;
.....
```

EOF (nome-interno-do-arquivo) : retorna valor verdadeiro se tiver sido atingido o fim do arquivo ou se ele estiver vazio. caso contrário retorna false . End of file.

```
.....
while not EOF (binario) do
  begin
    read(binario, umapessoa);
    writeln(umapessoa.nome:15,
    umapessoa.idade:4)
  end;
.....
```

Estrutura de Dados

O programa completo que grava informações de registro em um arquivo binário.

```
program gravabinario;
type pessoa = record nome: string[15];
  idade: integer
end;
var binario: file of pessoa;
  umapessoa: pessoa;
  i, n: integer;
begin
  write('Quantidade de pessoas a ler: ');
  readln(n);
  assign(binario, 'dados.bin');
  rewrite(binario);
  for i:=1 to n do
    begin
      writeln;
      writeln('Digite o nome da pessoa e sua idade. ');
      readln(umapessoa.nome, umapessoa.idade);
      write(binario, umapessoa)
    end;
  close(binario)
end.
```

Estrutura de Dados

O programa completo que lê informações do arquivo binário.

```
program lebinario;
type pessoa = record nome: string[15];
  idade: integer;
end;
var binario: file of pessoa;
  umapessoa: pessoa;
begin
  assign(binario, 'dados.bin');
  reset(binario);
  while not EOF (binario) do
    begin
      read(binario, umapessoa);
      writeln(umapessoa.nome:15, umapessoa.idade:4)
    end;
end.
```

Estrutura de Dados

seek (nome-interno-do-arquivo, num-do-registro) : (nome interno do arquivo no qual os registros serão escritos , num do registro ao qual se quer acessar , **num** come ça em 0.

```
...
For i:=1 to 100 do Write(Arq,i);
Seek(Arq,0);
Read(Arq,i);
Read(Arq,i);
....
```

filesize (nome-interno-do-arquivo) : retorna quantos registros possui o arquivo.

```
N := filesize(turma);
seek(arquivo, filesize(arquivo));
_ posiciona o apontador de registros no final do arquivo.
```

Estrutura de Dados

Programa que grava num arquivo em disco, o quadrado dos números de 0 a 100 e depois permite consulta através da instrução seek.

```
program Exemplo_2;
uses CRT;
var Arq : File of Real;
    i : Integer;
    s : real;
begin
assign(Arq,'Arquivo.dta');
rewrite(Arq);
for i:=0 to 100 do begin
s:=i*i;
write(Arq,s);
end;
```

```
close(Arq);
reset(Arq);
clrscr;
while i>=0 do begin
write(Numero -> ');
readln(i);
if (i>=0) And (i<=100) then begin
seek(Arq,i);
read(Arq,s);
writeln;
writeln(s:10:0);
writeln;
end;
end;
close(Arq);
end.
```

Estrutura de Dados

Programa que cadastra uma base de dados pelo seus registros.

```
Nome: Reinaldo
Rua: Eufrasio
Numero: 123
CEP: 8804002
CPF: 1234567
Sexo(M/F): M
Data de Nascimento(dd/mm/aaaa): 12/12/1975
RG: 654321
```

```
program sequenc;
type
endereco = record
rua : string[40];
numero : integer;
CEP : real;
end;
cadastro = record
nome : string[30];
ender : endereco;
CPF: real;
Sexo : char;
nascimento : real;
rg : real;
end;
arqcad = file of cadastro;
var
arq_in,arq_out : arqcad;
reg: cadastro;
```

Estrutura de Dados

```
begin
assign(arq_out,'sample.dat'); {associando nome interno com nome externo}
rewrite(arq_out); {abrindo arquivo para gravação}
write('Nome: ');
readln(reg.nome);
while (reg.nome <> 'fim') do
begin
write('Rua: '); readln(reg.ender.rua);
write('Numero: '); readln(reg.ender.numero);
write('CEP: '); readln(reg.ender.CEP);
write('CPF: '); readln(reg.CPF);
write('Sexo(M/F): '); readln(reg.Sexo);
write('Data de Nascimento(dd/mm/aaaa): '); readln(reg.nascimento);
write('RG: '); readln(reg.rg);
write(arq_outreg); {grava registro após o último registro do arquivo}
writeln;
write('Nome: ');
readln(reg.nome);
end;
close(arq_out);
assign(arq_in,'sample.dat'); {associando nome interno com nome externo}
```

Estrutura de Dados

```
reset(arq_in); {abrindo arquivo para leitura}
read(arq_in,reg);
while (not EOF(arq_in)) do
begin
write(reg.ender.rua, ' ');
write(reg.ender.numero, ' ');
write(reg.ender.CEP, ' ');
write(reg.CPF, ' ');
write(reg.Sexo, ' ');
write(reg.nascimento, ' ');
writeln(reg.rg, ' ');
end;
writeln('O numero de registros do arquivo eh: ',filesize(arq_in));
end.
```

Estrutura de Dados

write (variável:comprimento-do-campo:casadec);

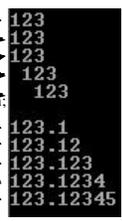
Usado para formatar esteticamente na tela campos ou valores decimais.

variável: comumente tipo real declarada no programa;

comprimento-do-campo: área delimitada em bytes que o numero a esquerda do campo/virgula que irá ocupar;

casadec: área delimitada em bytes que o número a direita da virgula irá ocupar.

```
program formata;
var r:real;
begin
r:=123.12345;
writeln(r:1:0);
writeln(r:2:0);
writeln(r:3:0);
writeln(r:4:0);
writeln(r:5:0);writeln;
writeln(r:1:1);
writeln(r:2:2);
writeln(r:3:3);
writeln(r:4:4);
writeln(r:5:5);
end.
```



Estrutura de Dados

Arquivos :: Texto: Armazem todas as informações numéricas ou literais através de códigos de seus caracteres (ASCII), ex: quando pedir para armazenar a palavra aluno, ele armazenará desta forma → aluno.

Os caracteres de um arquivo texto são agrupados em linhas e depois em páginas. Possui a vantagem de ser lido e entendido mais facilmente em até computadores muito diferentes.

Enquanto que os arquivos binários armazenam informações no mesmo formato com que elas são utilizadas na memória, os arquivos do tipo texto exigem conversões, portanto exigem mais tempo de processamento.

Sua organização se dá através da sua forma sequencial. Não é permitido acesso direto a arquivos do tipo texto.

Estrutura de Dados

<p>var lista-de-identificadores <i>text</i>;</p> <p>lista-de-identificadores = são os nomes associados aos arquivos.</p> <p>var documento: text;</p> <p>Declara o arquivo, denominado Documento, como sendo do tipo texto.</p>	<p>assign (<i>nome-interno-do-arquivo</i>, <i>nome-do-arquivo-externo</i>);</p> <p>Todo arq tem dois nomes: um <i>interno</i>, declarado dentro do programa e outro <i>externo</i>, usado pelo sistema operacional. O Pascal associa os dois nomes a um arquivo com o seguinte comando:</p> <pre>assign (documento, 'DOC.TXT')</pre> <p>Utilizando-se do comando string, também é possível dar nome a um arquivo.....:</p> <pre>... var externo: string; begin readln(externo); assign(documento, externo);</pre>
---	--

Estrutura de Dados

<p>reset (<i>nome-interno-do-arquivo</i>) : o qual é usado para se abrir um arquivo já existente e faz o seu apontador indicar o primeiro registro.</p> <pre>..... begin assign(documento, 'doc.txt'); reset(documento);</pre>	<p>rewrite (<i>nome-interno-do-arquivo</i>) : o qual é usado para criar um arquivo novo, ou sobrepor este arquivo no já existente. Também faz o seu apontador indicar o primeiro registro.</p> <pre>..... begin assign(documento, 'doc.txt'); rewrite(documento);</pre>
---	---

Estrutura de Dados

<p>append (<i>nome-interno-do-arquivo</i>) : o qual é usado para se abrir um arquivo texto, já existente, exclusivamente para a escrita, prolongando o arquivo a partir do último caracter gravado.</p> <pre>..... begin assign(documento, 'doc.txt'); append(documento);</pre>	<p>close (<i>nome-interno-do-arquivo</i>) : o qual é usado para fechar um arquivo criado ou alterado. Não se usa quando o arquivo só foi lido.</p> <pre>..... begin readln(texto, nome, idade); writeln(nome:15, idade:4) end; close(texto) end.</pre>
--	--

Estrutura de Dados

<p>readln ou read (<i>nome-interno-do-arquivo</i>, <i>lista-de-variáveis</i>) : (nome interno do arquivo no qual os registros serão lidos , nomes das variáveis que receberão os valores lidos)</p> <pre>..... begin readln(nome, idade);</pre>	<p>write ou writeln (<i>nome-interno-do-arquivo</i>, <i>lista-de-variáveis-ou-const</i>) : (nome interno do arquivo no qual os registros serão lidos , são constantes ou nomes das variáveis que terão seus valores escritos; o nome ou valor pode aparecer na lista seguido da indicação do formato de escrita)</p> <pre>..... begin writeln(texto, nome:15, idade:4);</pre>
--	--

Estrutura de Dados

<p>EOLn (<i>nome-interno-do-arquivo</i>): retorna o valor booleano verdadeiro se o apontador do arquivo tipo texto estiver sobre a marca de fim de linha ou se EOF for verdadeiro. Caso contrário, retorna o valor falso. EOLn significa End Of Line.</p>
--

Estrutura de Dados

<p>O programa completo que grava informações de registro em um arquivo texto.</p>	<pre>program gravatxt; var texto: text; nome: string[15]; i, n, idade: integer; begin write('Digite a qtde de pessoas a cadastrar: '); readln(n); assign(texto, 'dados.txt'); rewrite(texto); for i:= 1 to n do begin writeln; write('Digite o nome e a idade'); readln(nome, idade); writeln(texto, nome:15, idade:4) end; close(texto) end.</pre>
---	---

Estrutura de Dados

O programa completo que lê informações do arquivo texto.

```

program letexto;
var texto: text;
    nome: string[15];
    idade: integer;
begin
    assign(texto, 'dados.txt');
    reset(texto);
    while not EOF(texto) do
        begin
            readln(texto, nome, idade);
            writeln(nome:15, idade:4);
        end;
    close(texto);
end.
    
```

Estrutura de Dados

Este programa que lê (do teclado) um cadastro de 10 alunos, indicando o nome, nota1, nota2. Calcule a média simples dos 10 alunos e depois escreva um arquivo .TXT contendo os dados de cada aluno: nome, nota1, nota2, média.

```

CADASTRO DE ALUNOS
-----
Nome: Beatriz
Nota 1: 10
Nota 2: 10
-----
#Fim do Cadastro-----
Pressione qualquer tecla para continuar...
    
```

```

program Cadastro_de_Alunos;
uses crt;
var CadArq:Text;
    Cont:Integer;
    Nome:String;
    Nota1,nota2,media:Real;
begin
    Cont:=1;
    assign(CadArq, 'Cadastro.txt');
    rewrite(CadArq);
    while (Cont <= 10) do
        begin
            clrscr;
            { Cadastro dos Alunos }
            writeln('-----');
            writeln('CADASTRO DE ALUNOS');
            writeln('-----');
            writeln;
            write (Nome: ');
            readln(nome);
            write (Nota 1: ');
            
```

Estrutura de Dados

```

readln(nota1);
write('Nota 2: ');
readln(nota2);
media:=(nota1+nota2*2)/3;
{ Armazenando as notas no arquivo }
write (CadArq, nome, #9);
write (CadArq, nota1:2:2, #9);
write (CadArq, nota2:2:2, #9);
writeln(CadArq, media:2:2);
{ Apos o Cadastro e o Armazenamento , incrementado o Contador }
inc(Cont);

{ Mensagem ao usu rio }
writeln;
writeln('---Aluno Cadastrado---');
writeln;
write ('Pressione qualquer tecla para continuar...');
readln;
end;
close(CadArq);
end.
    
```

Estrutura de Dados

Este programa que lê os dados gerados pelo programa anterior

```

DADOS DO ALUNO
-----
Nome: Carla
Nota 1: 8.00
Nota 2: 9.00
Media: 8.00
-----
Pressione qualquer tecla para ver mais....
    
```

```

program Leitura_de_Dados;
uses crt;

var Arq:Text;
    Nome,nota1,nota2,media,temp:String;
    Tam,Cont:Integer;

begin
    assign(Arq, 'Cadastro.txt');
    reset(Arq);
    Cont:=1;
    while (Cont<=10) do
        begin
            clrscr;
            writeln('-----');
            writeln('DADOS DO ALUNO');
            writeln('-----');
            writeln;

            { Leitura dos Dados no Arquivo }
            readln(Arq, Temp);
        end;
    end;
end.
    
```

Estrutura de Dados

```

{ Sequencia de Passos da Separacao:
- Copia a primeira parte da linha para o nome ate a tabulacao
- Deleta essa parte
- Copia a parte da linha restante para o nota1 ate a tabulacao
- Deleta essa parte
- Copia a parte da linha restante para o nota2 ate a tabulacao
- Deleta essa parte
- O que sobrou copia para a media }

nome:= Copy(temp,1,Pos(#9,temp) - 1);
Delete (temp,1,Pos(#9,temp));
nota1:= Copy(temp,1,Pos(#9,temp) - 1);
Delete (temp,1,Pos(#9,temp));
nota2:= Copy(temp,1,Pos(#9,temp) - 1);
Delete (temp,1,Pos(#9,temp));
Tam:= length(temp);
media:= Copy(temp,1,tam);
    
```

Estrutura de Dados

```

{ Mostrando as informacoes na tela }
writeln('Nome: ',nome);
writeln('Nota 1: ',nota1);
writeln('Nota 2: ',nota2);
writeln('Media: ',media);

{ Apos Exibicao, Incrementa o contador }
inc(Cont);

{ Mensagem para a exibicao de novos dados }
writeln;
write ('Pressione qualquer tecla para ver mais...');
readln;
end;
close(Arq);
end.
    
```