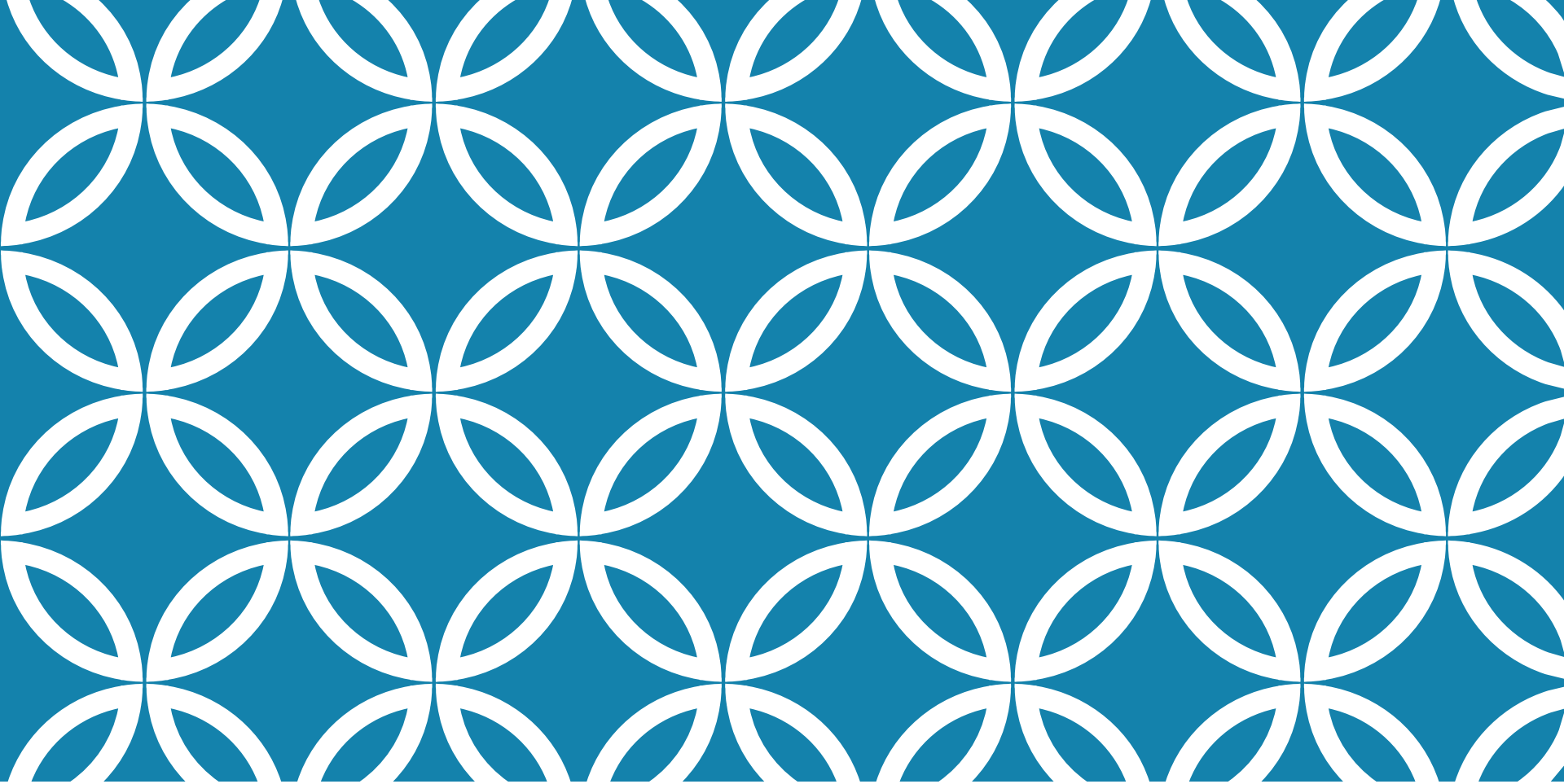


ESTRUTURAS DE DADOS

Prof. André Vignatti

ESTRUTURA DE DADOS

Modo de armazenamento e organização de dados no computador para serem usados eficientemente.

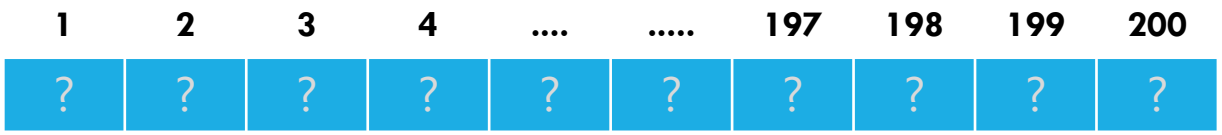
- **Vetores/Matrizes (ALG 1)**
- Tipos de Dados Compostos: registros e estruturas
- Lista, Fila, Pilha
- Árvores
- Heaps
- Tabelas Hash
- Estruturas para Grafos
- .....

DECLARANDO VETORES

Exemplo: reservar 200 posições de inteiros na memória

mais usual:

```
var V: array[1..200] of integer;
```



alternativas:

```
var v: array [0..199] of integer;

var v: array [201..400] of integer;

var v: array [−199..0] of integer;

var v: array [−300..−99] of integer;

var v: array [−99..100] of integer;

const min=11, max=210;
var v: array [min..max] of integer;
```

Armazenando valores: (por ex, na posição 98 do vetor)

```
v[98]:= 12345;

read(v[98]);   (* e se digita 12345 no teclado *)
```

Alguns exemplos:

```
read (v[1]); (* le do teclado e armazena na primeira posicao de v *)

i:= 10;
v[i+3]:= i * i; (* armazena o quadrado de i (100) na posicao 13 de v *)

write (i, v[i]); (* imprime o par (10, 100) na tela *)

write (v[v[13]]); (* imprime o valor de v[100] na tela *)

v[201]:= 5; (* gera erro, pois a posicao 201 nao existe em v *)

v[47]:= sqrt (4); (* gera erro, pois sqrt retorna um real, mas v eh de inteiros *)

var x: real;
v[x]:= 10; (* gera erro, pois x eh do tipo real, deveria ser ordinal *)
```

LENDO VÁRIOS VALORES

```
program lendo_vetores;
var v: array [1..200] of real;   (* define um vetor de reais *)
    i: integer;

begin
      i:= 1;
      while i <= 10 do
      begin
            read (v[i]);
            i:= i + 1;
      end;
end.
```

Se digitarmos 15, 12, 27, 23, 7, 2, 0, 18, 19, 21, teremos a seguinte "foto" do vetor:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...	197	198	199	200
15	12	27	23	7	2	0	18	19	21	?	?	?	?	?	...	?	?	?	?

COMANDOS DE REPETIÇÃO: FOR E REPEAT...UNTIL

Os comandos a seguir são equivalentes ao **while:**

```
begin
        for i:= 1 to 10 do
                read (v[i]);
end.
```

```
begin
        i:= 1;
        repeat
                read (v[i]);
                i:= i + 1;
        until i > 10;
end.
```

IMPRIMINDO VETORES

```
program lendo_e_imprimindo_vetores;
var v: array [1..200] of real;
    i: integer;

begin
     for i:= 1 to 10 do
     begin
          read (v[i]);
          writeln (v[i]);
     end;
end.
```

Poderíamos resolver sem usar vetores

```
program lendo_e_imprimindo;
var x: real; i: integer;

begin
     for i:= 1 to 10 do
     begin
          read (x);
          writeln (x);
     end;
end.
```

Organizando de outra forma:

```
program lendo_e_imprimindo_vetores;
var v: array [1..200] of real;
    i: integer;

begin
     (* este pedaco de codigo trata apenas da leitura dos dados *)
     for i:= 1 to 10 do
          read (v[i]);

     (* este outro pedaco de codigo trata apenas da impressao   *)
     for i:= 1 to 10 do
          writeln (v[i]);
end.
```

Se usar funções, o programa principal ficaria:

```
begin (* programa principal *)
      ler_vetor (v);
      imprimir_vetor (v);
end.
```

Como gostaríamos de fazer:

```
procedure ler (var v: array [1..200] of real);
```

Infelizmente, é impossível de fazer isso em Pascal!! ☹

Como deve ser feito:

```
const min=1; max=200;
type vetor= array [min..max] of real;
var v: vetor;

procedure ler_vetor (var w: vetor);
```

PROGRAMA COMPLETO

```
program ler_e_imprimir_com_procedures;
const min=1; max=200;
type vetor= array [min..max] of real;
var v: vetor;

procedure ler_vetor (var w: vetor);
var i: integer;
begin
      for i:= 1 to 10 do
            read (w[i]);
end;

procedure imprimir_vetor (var w: vetor); (* impressao dos dados *)
var i: integer;
begin
      for i:= 1 to 10 do
            write (w[i]);
end;

begin (* programa principal *)
      ler_vetor (v);
      imprimir_vetor (v);
end.
```

IMPRIMINDO AO CONTRÁRIO

PROBLEMA: ler diversos valores, imprimir ao contrário.

```
procedure imprimir_ao_contrario (var w: vetor);
var i: integer;
begin
      for i:= 10 downto 1 do
            write (w[i]);
end;
```

```
program ler_e_imprimir_ao_contrario;
const min=0; max=50;
type vetor= array [min..max] of real;
var v: vetor;
    n: integer;

procedure ler (var w: vetor; var tam: integer); (* leitura *)
var i: integer;
begin
     read (tam);  (* 1<= tam <= 200, define o tamanho util do vetor *)
     for i:= 1 to tam do
          read (w[i]);
end;

procedure imprimir_ao_contrario (var w: vetor; tam: integer); (* impressao *)
var i: integer;
begin
     for i:= tam downto 1 do
          write (w[i]);
end;

begin (* programa principal *)
     ler (v, n);
     imprimir_ao_contrario (v, n);
end.
```

IMPRIMINDO PARES

```
const min_r=0; max_r=50;
      min_i=1; max_i=10;

type vetor_r= array [min_r..max_r] of real;
     vetor_i= array [min_i..max_i] of integer;
```

```
procedure imprimir_pares (var v: vetor_i; tam: integer);
var i: integer;
begin
      for i:= 1 to tam do
            if eh_par (v[i]) then
                  write (v[i],' ');
      writeln;
end;
```

O que acontece se eu trocar para essa linha?

```
if eh_par (i) then  // no lugar de: if eh_par (v[i]) then
```

MENOR NÚMERO LIDO

Resolvendo SEM usar vetor:

```
program menor_dos_lidos;
var N, i: integer; x, menor: real;
begin
      read (N);
      read (x);
      menor:= x;
      for i:= 2 to N do
      begin
            read (x);
            if x < menor then
                  menor:= x;
      end;
      writeln (menor);
end.
```

Solução usando vetores:

```
function menor_dos_lidos (var v: vetor_r; N: integer): real;
var i: integer; menor: real;
begin
      menor:= v[1];
      for i:= 2 to N do
            if v[i] < menor then
                  menor:= v[i];
      menor_dos_lidos:= menor;
end;
```