# CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho Carmem Hara e Bruno Muller Jr

Departamento de Informática/UFPR

15 de agosto de 2020

# Estruturas de dados

- Os problemas abordados até o momento lidavam com pequenas quantidades de informação.
- Quando aumenta a quantidade de informação necessária para resolver um problema, são necessários mecanismos para armazenar esta informação.
- Alguns desses mecanismos já estão embutidos nas linguagens de programação: vetores, matrizes e registros.
- As próximas aulas irão abordar estas estruturas de dados, as categorias de problemas para as quais elas são apropriadas e sua sintaxe na linguagem Pascal.

# Vetores

- Aplicações
- Conceitos
- Vetores em Pascal
- Problemas Básicos
- Soma de vetores
- Produto escalar de vetores

# Aplicações

**Problema: Escreva um programa Pascal que lê até 200 valores inteiros e os imprima na ordem inversa da leitura.**

- Solução óbvia: utilizar 200 variáveis do tipo longint.
- Solução proposta: utilizar uma estrutura de dados chamada vetor capaz de armazenar os 200 elementos.
- Mas... O que são vetores, como armazenar informação neles e como recuperar informação armazenada neles?

# Aplicações

**Problema: Escreva um programa Pascal que lê até 200 valores inteiros e os imprima na ordem inversa da leitura.**

- Solução óbvia: utilizar 200 variáveis do tipo longint.
- Solução proposta: utilizar uma estrutura de dados chamada vetor capaz de armazenar os 200 elementos.
- Mas... O que são vetores, como armazenar informação neles e como recuperar informação armazenada neles?

# Aplicações

**Problema: Escreva um programa Pascal que lê até 200 valores inteiros e os imprima na ordem inversa da leitura.**

- Solução óbvia: utilizar 200 variáveis do tipo longint.
- Solução proposta: utilizar uma estrutura de dados chamada vetor capaz de armazenar os 200 elementos.
- Mas... O que são vetores, como armazenar informação neles e como recuperar informação armazenada neles?

# Conceitos

- A figura mostra conceitualmente o que é um vetor.
- Cada quadrado armazena exatamente uma informação de um mesmo tipo (inteiro, real, etc).
- Qualquer quadrado pode ser acessado individualmente através do índice.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 12 | 27 | 23 | 7 | 2 | 0 | 18 | 19 | 21 |

# Conceitos

- A figura mostra conceitualmente o que é um vetor.
- Cada quadrado armazena exatamente uma informação de um mesmo tipo (inteiro, real, etc).
- Qualquer quadrado pode ser acessado individualmente através do índice.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 12 | 27 | 23 | 7 | 2 | 0 | 18 | 19 | 21 |

# Conceitos

- A figura mostra conceitualmente o que é um vetor.
- Cada quadrado armazena exatamente uma informação de um mesmo tipo (inteiro, real, etc).
- Qualquer quadrado pode ser acessado individualmente através do índice.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 12 | 27 | 23 | 7 | 2 | 0 | 18 | 19 | 21 |

# Conceitos

- Qual o conteúdo do quadrado cujo índice é 3?
- Em quase todas as linguagens de programação, o acesso a este quadrado é dado sintaticamente por "[3]".
- Considere que o nome do vetor é "m". O que faz o comando m[3]:=m[3]+1 ?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 12 | 27 | 23 | 7 | 2 | 0 | 18 | 19 | 21 |

# Conceitos

- Qual o conteúdo do quadrado cujo índice é 3?
- Em quase todas as linguagens de programação, o acesso a este quadrado é dado sintaticamente por "[3]".
- Considere que o nome do vetor é "m". O que faz o comando m[3]:=m[3]+1 ?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 12 | 27 | 23 | 7 | 2 | 0 | 18 | 19 | 21 |

# Conceitos

- Qual o conteúdo do quadrado cujo índice é 3?
- Em quase todas as linguagens de programação, o acesso a este quadrado é dado sintaticamente por "[3]".
- Considere que o nome do vetor é "m". O que faz o comando m[3]:=m[3]+1 ?

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 12 | 27 | 23 | 7 | 2 | 0 | 18 | 19 | 21 |

# Vetores em Pascal

- O comando abaixo declara a variável v com 200 longint.
- A construção "1..200" é uma enumeração em Pascal. Significa que os índices para acessar o vetor são: 1,2,3,..., 199, 200;
- Observe os comandos para acessar os elementos deste vetor. Quais são válidos e quais não são?

```
var v: array [1..200] of longint;

(a) v[0] := 0;      (e) i:=0;            (h) v[1] := v[2]+v[3];
(b) v[1] := 0;      (f) v[i] := 0;       (i) v[i] := v[j]+v[k];
(c) v[200] := 0;    (g) v[i-1] := 0;     (j) write(v[5]);
(d) v[201] := 0;
```

# Vetores em Pascal

- O comando abaixo declara a variável v com 200 longint.
- A construção "1..200" é uma enumeração em Pascal. Significa que os índices para acessar o vetor são: 1,2,3,..., 199, 200;
- Observe os comandos para acessar os elementos deste vetor. Quais são válidos e quais não são?

```
var v: array [1..200] of longint;

(a) v[0] := 0;      (e) i:=0;           (h) v[1] := v[2]+v[3];
(b) v[1] := 0;      (f) v[i] := 0;      (i) v[i] := v[j]+v[k];
(c) v[200] := 0;    (g) v[i-1] := 0;    (j) write(v[5]);
(d) v[201] := 0;
```

# Vetores em Pascal

- O comando abaixo declara a variável v com 200 longint.
- A construção "1..200" é uma enumeração em Pascal. Significa que os índices para acessar o vetor são: 1,2,3,..., 199, 200;
- Observe os comandos para acessar os elementos deste vetor. Quais são válidos e quais não são?

```
var v: array [1..200] of longint;

(a) v[0] := 0;      (e) i:=0;           (h) v[1] := v[2]+v[3];
(b) v[1] := 0;      (f) v[i] := 0;      (i) v[i] := v[j]+v[k];
(c) v[200] := 0;    (g) v[i-1] := 0;    (j) write(v[5]);
(d) v[201] := 0;
```

# Vetores em Pascal

- Cada comando abaixo também declara a variável v com 200 elementos, mas com outros índices;
- Observe os comandos para acessar os elementos de cada declaração de vetor. Quais são válidos e quais não são?

```
var v: array [0..199] of longint;
var v: array [201..400] of longint;
var v: array [-199..0] of longint;
var v: array [-300..-99] of longint;
var v: array [-99..100] of longint;
const min=11, max=210;
var v: array [min..max] of longint;
```

```
(a) v[0] := 0;
(b) v[1] := 0;
(c) v[200] := 0;
(d) v[201] := 0;
```

```
(e) i:=0;
(f) v[i] := 0;
(g) v[i-1] := 0;
```

```
(h) v[1] := v[2]+v[3];
(i) v[i] := v[j]+v[k];
(j) v[47] := sqrt(47);
```

```
(********************)
v[1] := -2;
v[2] := 5;
v[v[1]]   := v[1]+1;
v[v[1]-1] := 1;
```

# Vetores em Pascal

- Cada comando abaixo também declara a variável v com 200 elementos, mas com outros índices;
- Observe os comandos para acessar os elementos de cada declaração de vetor. Quais são válidos e quais não são?

```
var v: array [0..199] of longint;
var v: array [201..400] of longint;
var v: array [-199..0] of longint;
var v: array [-300..-99] of longint;
var v: array [-99..100] of longint;
const min=11, max=210;
var v: array [min..max] of longint;
```

```
(a) v[0] := 0;
(b) v[1] := 0;
(c) v[200] := 0;
(d) v[201] := 0;
```

```
(e) i:=0;
(f) v[i] := 0;
(g) v[i-1] := 0;
```

```
(h) v[1] := v[2]+v[3];
(i) v[i] := v[j]+v[k];
(j) v[47] := sqrt(47);
```

```
(********************)
v[1] := -2;
v[2] := 5;
v[v[1]]   := v[1]+1;
v[v[1]-1] := 1;
```

# Problemas Básicos

- **Problema: Escreva um programa Pascal que lê 10 valores inteiros e os imprima na ordem inversa da leitura.**

# Sub-problema: ler 10 valores inteiros

- Considere que os elementos lidos serão colocados em um vetor de 200 posições;
- Existem várias opções de lugares onde armazená-los: nos primeiros 10, nos últimos 10, nos primeiros 10 índices pares, entre outros;
- Ao escolher a opção, o programador não deve deixar "buracos" no vetor (lixo de memória);
- Sob esse ponto de vista, qualquer opção onde um elemento de índice [i] fique "colado" ao lado de um elemento de índice [i+1] é recomendável;

# Sub-problema: ler 10 valores inteiros

- Considere que os elementos lidos serão colocados em um vetor de 200 posições;
- Existem várias opções de lugares onde armazená-los: nos primeiros 10, nos últimos 10, nos primeiros 10 índices pares, entre outros;
- Ao escolher a opção, o programador não deve deixar "buracos" no vetor (lixo de memória);
- Sob esse ponto de vista, qualquer opção onde um elemento de índice [i] fique "colado" ao lado de um elemento de índice [i+1] é recomendável;

# Sub-problema: ler 10 valores inteiros

- Considere que os elementos lidos serão colocados em um vetor de 200 posições;
- Existem várias opções de lugares onde armazená-los: nos primeiros 10, nos últimos 10, nos primeiros 10 índices pares, entre outros;
- Ao escolher a opção, o programador não deve deixar "buracos" no vetor (lixo de memória);
- Sob esse ponto de vista, qualquer opção onde um elemento de índice [i] fique "colado" ao lado de um elemento de índice [i+1] é recomendável;

# Sub-problema: ler 10 valores inteiros

- Considere que os elementos lidos serão colocados em um vetor de 200 posições;
- Existem várias opções de lugares onde armazená-los: nos primeiros 10, nos últimos 10, nos primeiros 10 índices pares, entre outros;
- Ao escolher a opção, o programador não deve deixar "buracos"no vetor (lixo de memória);
- Sob esse ponto de vista, qualquer opção onde um elemento de índice [i] fique "colado"ao lado de um elemento de índice [i+1] é recomendável;

# Imprime Invertido - v1

```
program imprime_invertido_v1;
var v: array [1..200] of longint;
    i : integer ;
begin
    i:= 1;
    while i <= 10 do
    begin
        read (v[i] ) ;
        i:= i + 1;
    end;

    i:= 10;
    while i >= 1 do
    begin
        write (v[i] ) ;
        i:= i - 1;
    end;
end.
```

# Imprime Invertido - v2

- O comando for é bem útil quando estamos lidando com vetores.
- Porém, ele não substitui todas as funcionalidades do comando while.
- Assista a aula gravada do Professor Castilho para utilizá-lo corretamente.

```
program imprime_invertido_v2;
var v: array [1..200] of longint;
   i : integer ;
begin
   for i:=1 to 10 do
      read (v[i]);

   for i:=10 downto 1 do
      write (v[i]);

end.
```

# Imprime Invertido - v3

```
program imprime_invertido_v3;
const inicio_vet =  1;
const fim_vet    =  10;
const tam_vet    =  200;
var v: array [inicio_vet..tam_vet] of longint;
   i : integer ;
begin
   for i:=inicio_vet to fim_vet do
      read (v[i]);

   for i:=fim_vet downto inicio_vet do
      write (v[i]);
end.
```

# Imprime Invertido - v4

```
program ler_e_imprimir_ao_contrario ;
const min=0; max=200;
type vetor= array [min..max] of real ;
var v: vetor ;
   n: integer ;

   procedure ler (var v: 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 (v[i] ) ;
   end;

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

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

# Imprime Pares

- **Problema: Escreva um programa Pascal que imprime os elementos pares de um vetor.**

```
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;
```

# Imprime Pares

- **Problema: Escreva um programa Pascal que imprime os elementos pares de um vetor.**

- 
```
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;
```

# Encontrar o menor de uma sequência de números

- **Problema: Escreva um programa Pascal que encontra o menor elemento de uma sequência de números; Compare com uma função que retorna o menor elemento de um vetor.**

```
program menordoslidos1;
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.
```

```
function menordoslidos2 (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] ;
    menordoslidos2:= menor;
end;
```

# Encontrar o menor de uma sequência de números

- **Problema: Escreva um programa Pascal que encontra o menor elemento de uma sequência de números; Compare com uma função que retorna o menor elemento de um vetor.**

```
program menordoslidos1;
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.
```

```
function menordoslidos2 (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] ;
   menordoslidos2:= menor;
end;
```

# Variação: Encontrar o índice do menor elemento de um vetor

**Problema: Altere a função abaixo para retornar o índice do menor elemento de um vetor**

```
function menordoslidos2 (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] ;
   menordoslidos2:= menor;
end;
```

# Somar dois Vetores

- **Problema: Escreva um programa Pascal que soma dois vetores com o mesmo número de elementos.**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $v$ | 5 | 2 | 7 | 3 | 7 | 2 | 0 | 8 | 9 | 1 |
| $w$ | 5 | 3 | 2 | 4 | 1 | 2 | 6 | 1 | 4 | 2 |
| $v + w$ | 10 | 5 | 9 | 7 | 8 | 4 | 6 | 9 | 13 | 2 |

# Somar dois Vetores

```
procedure somar_vetores (var v, w, soma_v_w: vetor_i ; tam: integer) ;
(* este procedimento considera que os dois vetores tem o mesmo tamanho *)
var i : integer ;
begin
   for i:= 1 to tam do
      soma_v_w[i]:= v[i] + w[i];
end;
```

# Multiplicar dois Vetores

- **Problema: Escreva um programa Pascal que calcula o produto escalar de dois vetores (eles tem o mesmo número de elementos).**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $v$ | 5 | 2 | 7 | 3 | 7 | 2 | 0 | 8 | 9 | 1 |
| $w$ | 5 | 3 | 2 | 4 | 1 | 2 | 6 | 1 | 4 | 2 |
| $v \times w$ | 25 | 6 | 14 | 12 | 7 | 4 | 0 | 8 | 36 | 2 |

$$produto\_escalar = 25 + 6 + 14 + 12 + 7 + 4 + 0 + 8 + 36 + 2 = 114$$

# Multiplicar dois Vetores

```
function prod_escalar (var v, w: vetor_r ; tam: integer) : real ;
var i : integer ;
   soma: real ;
begin
   soma:= 0;
   for i:= 1 to tam do
      soma:= soma + v[i] * w[i];
   prod escalar:= soma;
end;
```

# Fim

- este material está no livro no capítulo 9, seções de 9.1 até 9.4