# CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho, Bruno Müller Jr, Carmem Hara

Departamento de Informática/UFPR

18 de agosto de 2020

## Resumo

Resolvendo um problema básico

# Objetivos da aula

- Resolver um problema básico que motiva o uso de vetores
- Entender a diferença de problemas básicos com e sem o uso de vetores
- Revisar este problema à luz das funções e procedimentos

# Um problema básico

**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 vetores

# 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;

# Exemplos

- Vejamos alguns exemplos na prática
  - alocar da primeira posição em direção à última
  - alocar da última posição em direção à primeira
  - alocar nas posições pares (não faz muito sentido, mas...)
  - alocar da posição `ini` em direção à posição `fim`

# Uma boa política para o momento

Até ordem contrária, ou até uma necessidade surgir, vamos sempre alocar da primeira posição em direção à última.

Vamos também usar sempre esta definição aqui (enquanto pudermos!):

```
1 const MAX=200;
2 v: array [1..MAX] of longint;
```

# Imprime invertido - v1

```
1  program imprime_invertido_v1;
2  var v: array [1..200] of longint;
3      i: longint;
4  begin
5      i:= 1;
6      while i <= 10 do
7      begin
8          read (v[i]);
9          i:= i + 1;
10     end;
11
12     i:= 10;
13     while i >= 1 do
14     begin
15         write (v[i]);
16         i:= i - 1;
17     end;
18     writeln;
19 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

```
1  program imprime_invertido_v2;
2  var v: array [1..200] of longint;
3      i: longint;
4  begin
5     for i:=1 to 10 do
6        read (v[i]);
7
8     for i:=10 downto 1 do
9        write (v[i]);
10     writeln;
11 end.
```

# Imprime invertido - v3

```
1  program imprime_invertido_v3;
2  const inicio_vet =  1;
3        fim_vet    =  10;
4        tam_vet    =  200;
5
6  var v: array [inicio_vet..tam_vet] of longint;
7      i: longint;
8
9  begin
10    for i:=inicio_vet to fim_vet do
11       read (v[i]);
12
13    for i:=fim_vet downto inicio_vet do
14       write (v[i]);
15    writeln;
16 end.
```

# Imprime invertido - v4

```
1  program ler_e_imprimir_ao_contrario;
2  const min=0; max=200;
3  type vetor= array [min..max] of longint;
4  var v: vetor;
5      n: longint;
6
7     procedure ler (var v: vetor; var tam: longint); (* leitura *)
8     var i: longint;
9     begin
10       read (tam); (* 1<= tam <= 200, define o tamanho util do vetor *)
11       for i:= 1 to tam do
12          read (v[i]);
13    end;
14
15    procedure imprimir_ao_contrario (var v: vetor; tam: longint); (* impressao *)
16    var i: longint;
17    begin
18       for i:= tam downto 1 do
19          write (v[i]);
20      writeln;
21    end;
22
23 begin (* programa principal *)
24    ler (v, n);
25    imprimir_ao_contrario (v, n);
26 end.
```

# Fim do tópico

- este material está no livro no capítulo 9, seções 9.1 a 9.3.2

# Licença

- Slides feitos em LaTeX usando beamer
- Licença

  *Creative Commons* Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`