CI1055: Algoritmos e Estruturas de Dados I

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

Departamento de Informática/UFPR

3 de agosto de 2020

Resumo

Aplicações das técnicas elementares.

Aplicações de Técnicas Elementares

- Inverter um número de três dígitos
- Convertendo para binário
- Cálculo do máximo divisor comum (MDC)
- Tabuada

Inverter um número de três dígitos

- Objetivos
  - Utilização de acumuladores
  - Melhoramentos sucessivos da solução
  - Generalização da solução

Primeira Solução

```
program inverte3_v0;
var num, unidade, dezena, centena, inverso: integer;
begin
      write( 'Entre com um numero de tres digitos: ');
      readln( num );
      centena:= numero div 100;
      dezena:= (numero mod 100) div 10;
      unidade:= numero mod 10;
      inverso:= unidade*100 + dezena*10 + centena;
      writeln( inverso );
end.
```

Qual o problema desta versão?

- A solução pode ser generalizada?
- Como é uma solução para números com 4 dígitos?

Solução para Quatro dígitos

```
program inverte4;
var num, unidade, dezena, centena, milhar, inverso: integer;
begin
      write( 'Entre com um numero de quatro digitos: ');
      readln( num );
      milhar:= numero div 1000;
      centena:= numero mod 1000 div 100;
      dezena:= (numero mod 100) div 10;
      unidade:= numero mod 10;
      inverso:= unidade*1000 + dezena*100 + centena*10 + milhar;
      writeln( inverso );
end.
```

Generalização

- E se a entrada tiver 10 dígitos?
- Temos que achar uma solução genérica
- Encontrar um padrão repetitivo para **separar os dígitos** do número

Generalização

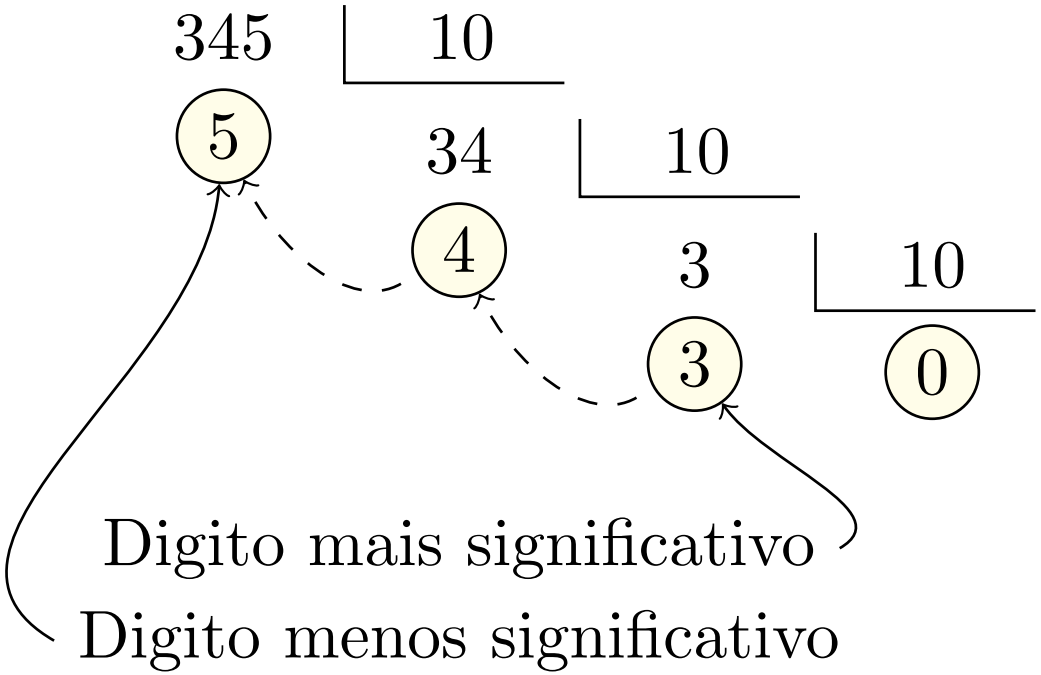
- E se a entrada tiver 10 dígitos?
- Temos que achar uma solução genérica
- Encontrar um padrão repetitivo para **separar os dígitos** do número

Generalização

- E se a entrada tiver 10 dígitos?
- Temos que achar uma solução genérica
- Encontrar um padrão repetitivo para **separar os dígitos** do número

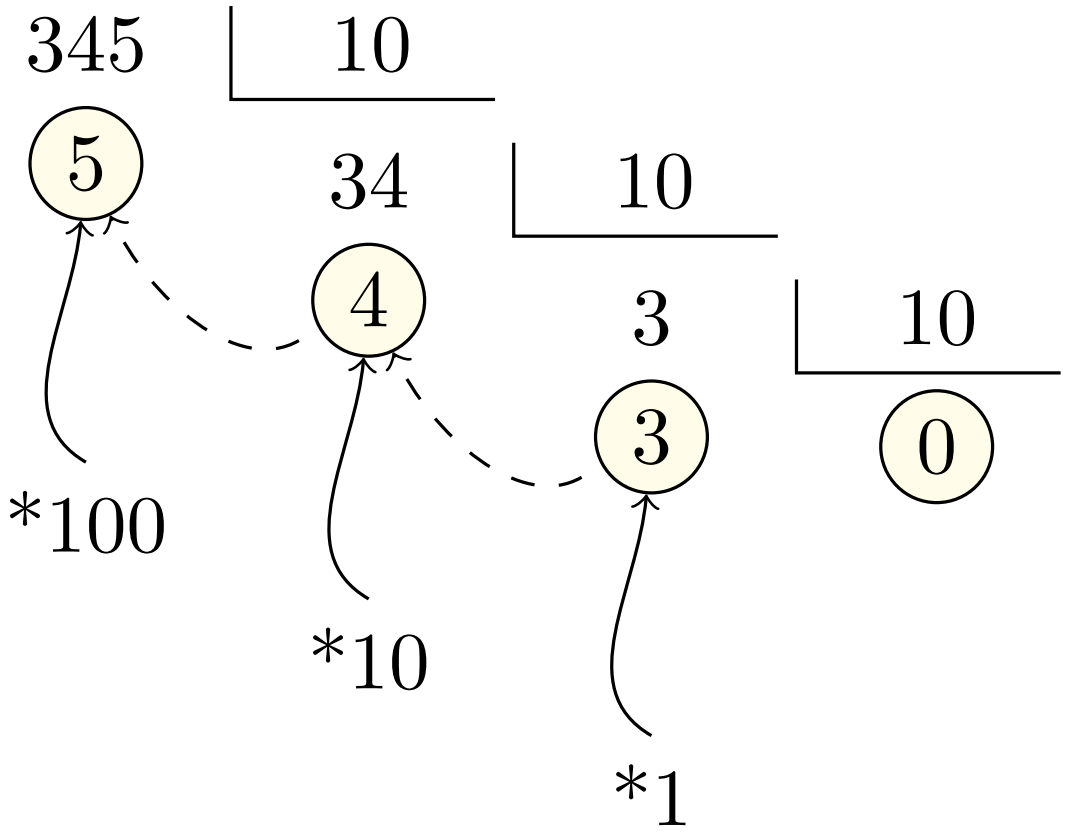
Separação de dígitos

- Divisão sucessiva por 10



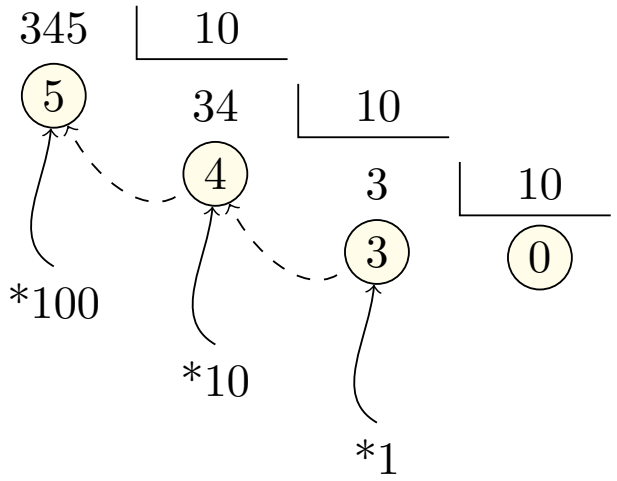
Separação de dígitos e inversão

- Divisão sucessiva por 10



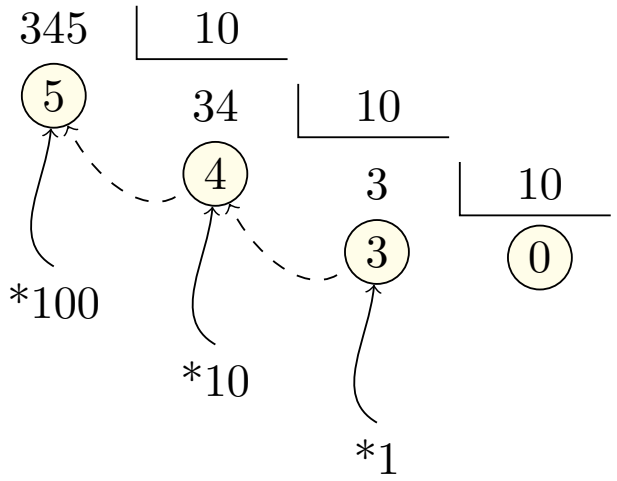
Padrão Repetitivo

num	digito	pot10
345	5	100
34	4	10
3	3	1
0		



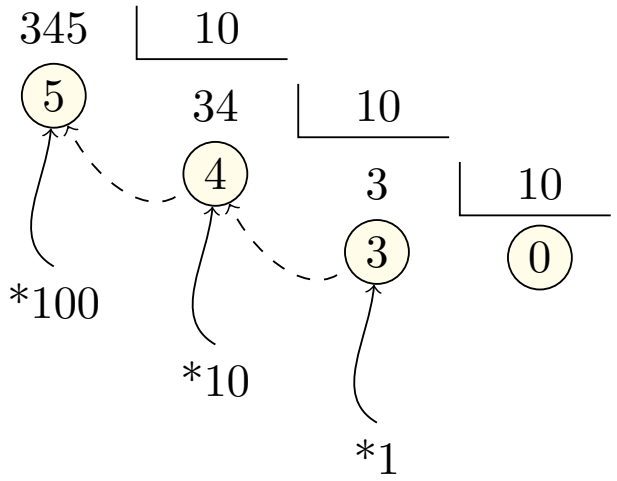
Padrão Repetitivo

num	digito	pot10
345	5	100
34	4	10
3	3	1
0		



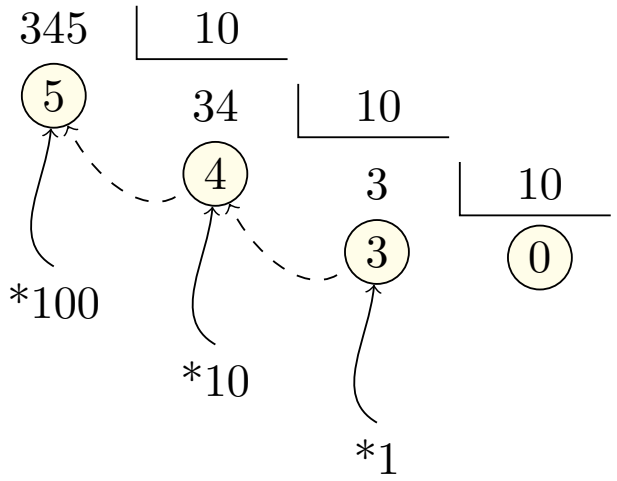
Padrão Repetitivo

num	digito	pot10
345	5	100
34	4	10
3	3	1
0		



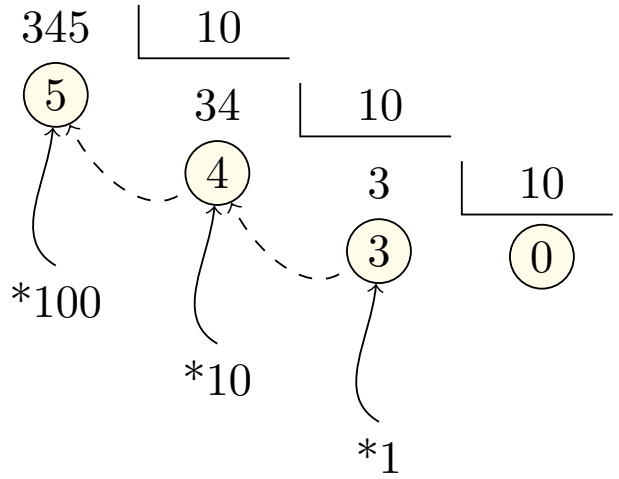
Padrão Repetitivo

num	digito	pot10
345	5	100
34	4	10
3	3	1
0		



Padrão Repetitivo com o acumulador

num	digito	pot10	inverso
			0
345	5	100	500
34	4	10	540
3	3	1	543
0			

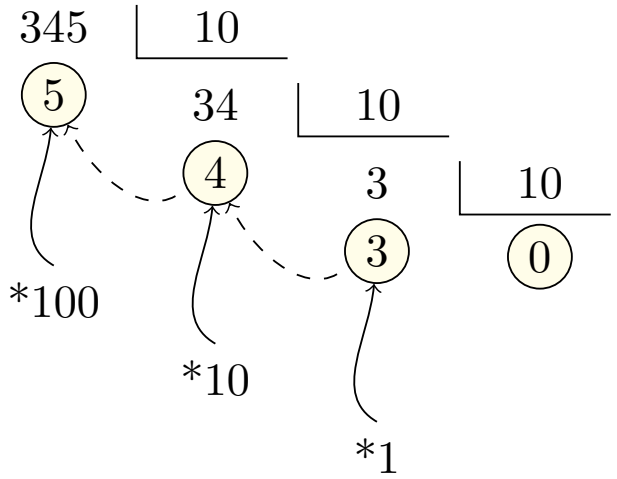


Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso + digito*pot10;
num:= num div 10;
pot10:= pot10 div 10;
```

Padrão Repetitivo com o acumulador

num	digito	pot10	inverso
			0
345	5	100	500
34	4	10	540
3	3	1	543
0			

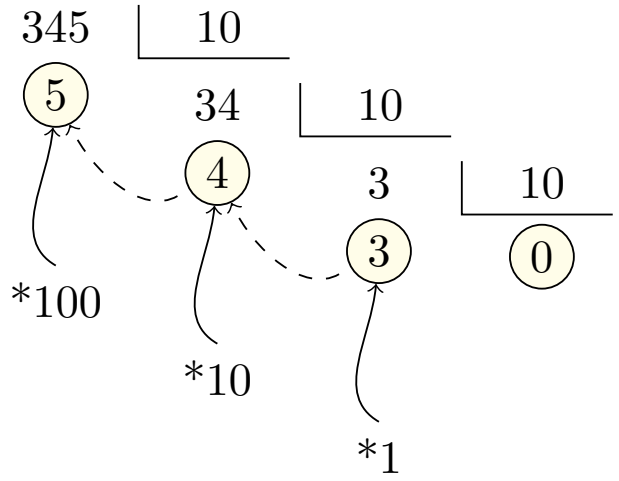


Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso + digito*pot10;
num:= num div 10;
pot10:= pot10 div 10;
```

Padrão Repetitivo com o acumulador

num	digito	pot10	inverso
			0
345	5	100	500
34	4	10	540
3	3	1	543
0			

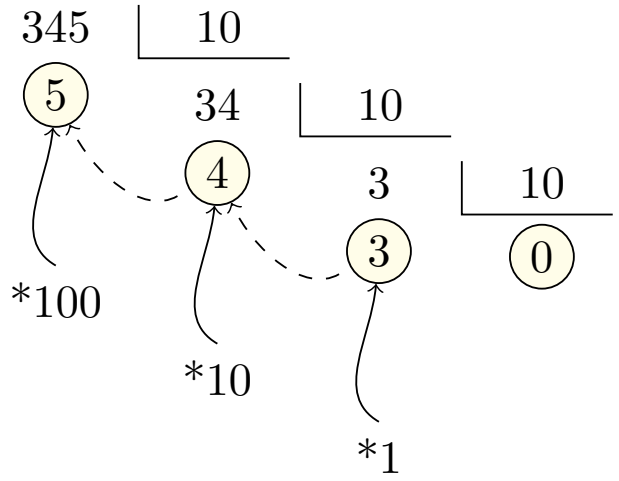


Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso + digito*pot10;
num:= num div 10;
pot10:= pot10 div 10;
```

Padrão Repetitivo com o acumulador

num	digito	pot10	inverso
			0
345	5	100	500
34	4	10	540
3	3	1	543
0			

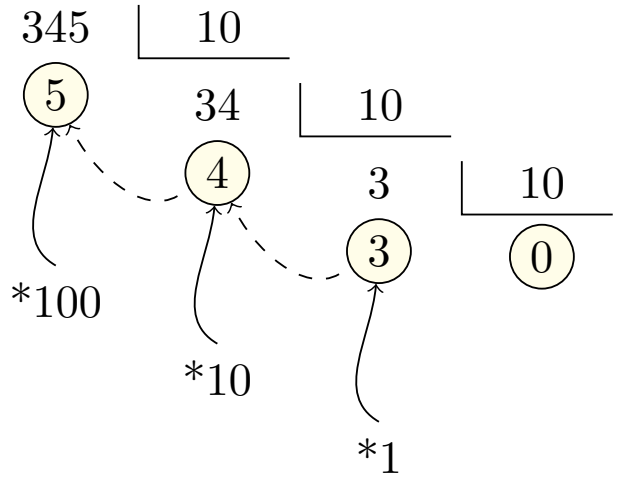


Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso + digito*pot10;
num:= num div 10;
pot10:= pot10 div 10;
```

Padrão Repetitivo com o acumulador

num	digito	pot10	inverso
			0
345	5	100	500
34	4	10	540
3	3	1	543
0			



Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso + digito*pot10;
num:= num div 10;
pot10:= pot10 div 10;
```

Inverter um número de três dígitos - V2

```
program inv_v2;
var i, num, digito, inverso, pot10: integer;
begin
      read( num );
      inverso:= 0;
      pot10:= 100;
      i:= 1;
      while i <= 3 do
      begin
            digito:= num mod 10;
            inverso:= inverso + digito*pot10;
            num:= num div 10;
            pot10:= pot10 div 10;
      end;
      writeln( inverso );
end.
```

Inverter um número de quatro dígitos - V2

```
program inv_v2;
var i, num, digito, inverso, pot10: integer;
begin
      read( num );
      inverso:= 0;
      pot10:= 1000;
      i:= 1;
      while i <= 4 do
      begin
            digito:= num mod 10;
            inverso:= inverso + digito*pot10;
            num:= num div 10;
            pot10:= pot10 div 10;
      end;
      writeln( inverso );
end.
```

Melhorando a solução

- A inicialização de `pot10` e a condição do `while` ainda dependem da quantidade de dígitos
- Como generalizar para um número **qualquer** de dígitos?

Ideia

- Se você tem 2 inteiros:

  5 4

  Como transformar no inteiro:

  $54 = 5 * 10 + 4$

- Se você tem 2 inteiros:

  54 3

  Como transformar no inteiro:

  $543 = 54 * 10 + 3$

Então é possível **ir construindo o número inverso** à medida que obtém os dígitos!!

Ideia

- Se você tem 2 inteiros:

  $5 \qquad 4$

  Como transformar no inteiro:

  $54 \quad = 5 * 10 + 4$

- Se você tem 2 inteiros:

  $54 \qquad 3$

  Como transformar no inteiro:

  $543 \quad = 54 * 10 + 3$

Então é possível **ir construindo o número inverso** à medida que obtém os dígitos!!

Ideia

- Se você tem 2 inteiros:

  $$5 \qquad 4$$

  Como transformar no inteiro:

  $$54 \quad = 5 * 10 + 4$$

- Se você tem 2 inteiros:

  $$54 \qquad 3$$

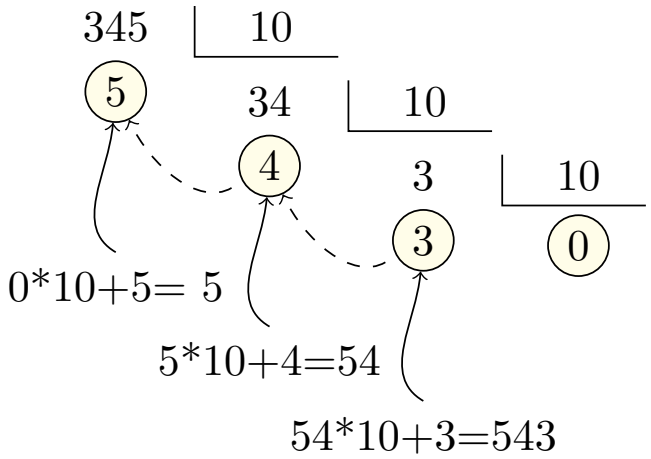
  Como transformar no inteiro:

  $$543 \quad = 54 * 10 + 3$$

Então é possível **ir construindo o número inverso** à medida que obtém os dígitos!!

Padrão Repetitivo com o acumulador - V3

num	digito	inverso
		0
345	5	5
34	4	54
3	3	543
0		

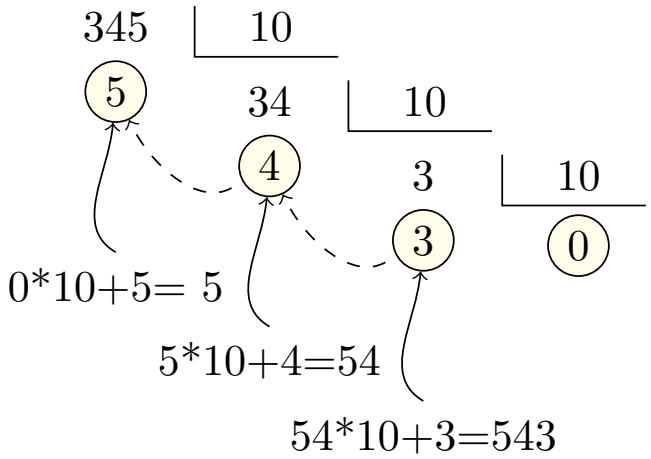


Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso*10 + digito;
num:= num div 10;
```

Padrão Repetitivo com o acumulador - V3

num	digito	inverso
		0
345	5	5
34	4	54
3	3	543
0		

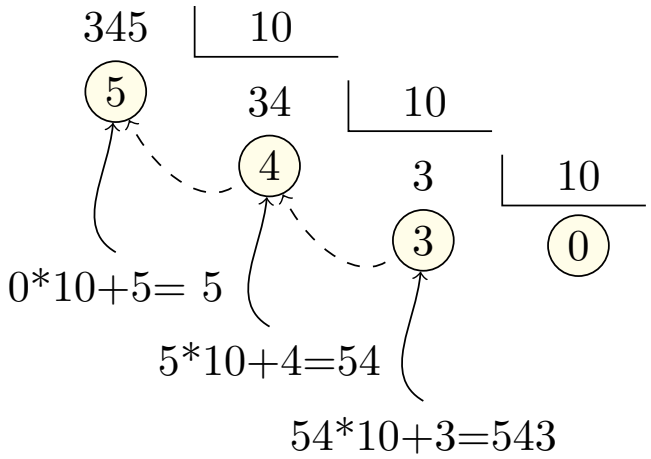


Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso*10 + digito;
num:= num div 10;
```

Padrão Repetitivo com o acumulador - V3

num	digito	inverso
		0
345	5	5
34	4	54
3	3	543
0		

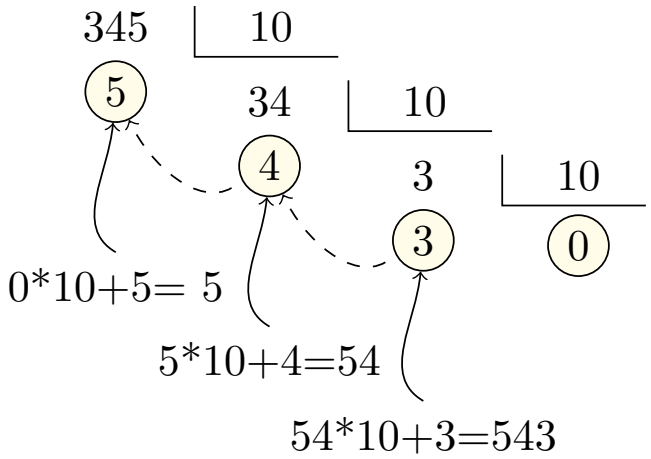


Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso*10 + digito;
num:= num div 10;
```

Padrão Repetitivo com o acumulador - V3

num	digito	inverso
		0
345	5	5
34	4	54
3	3	543
0		

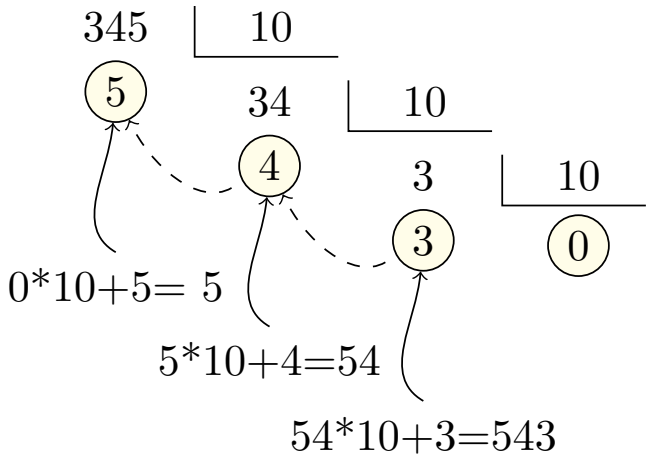


Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso*10 + digito;
num:= num div 10;
```

Padrão Repetitivo com o acumulador - V3

num	digito	inverso
		0
345	5	5
34	4	54
3	3	543
0		



Padrão repetitivo

```
digito:= num mod 10;
inverso:= inverso*10 + digito;
num:= num div 10;
```

Inversão de um número - Versão 3

```
program inv_v3;
var num, digito, inverso: integer;
begin
      read( num );
      inverso:= 0;
      while ??? do
      begin
            digito:= num mod 10;
            inverso:= inverso * 10 + digito;
            num:= num div 10;
      end;
      writeln( inverso );
end.
```

Pergunta: Qual a condição do `while` para que o programa funcione para um número com **qualquer** quantidade de dígitos?

Conversão de decimal para binário

- Interpretação de um número decimal

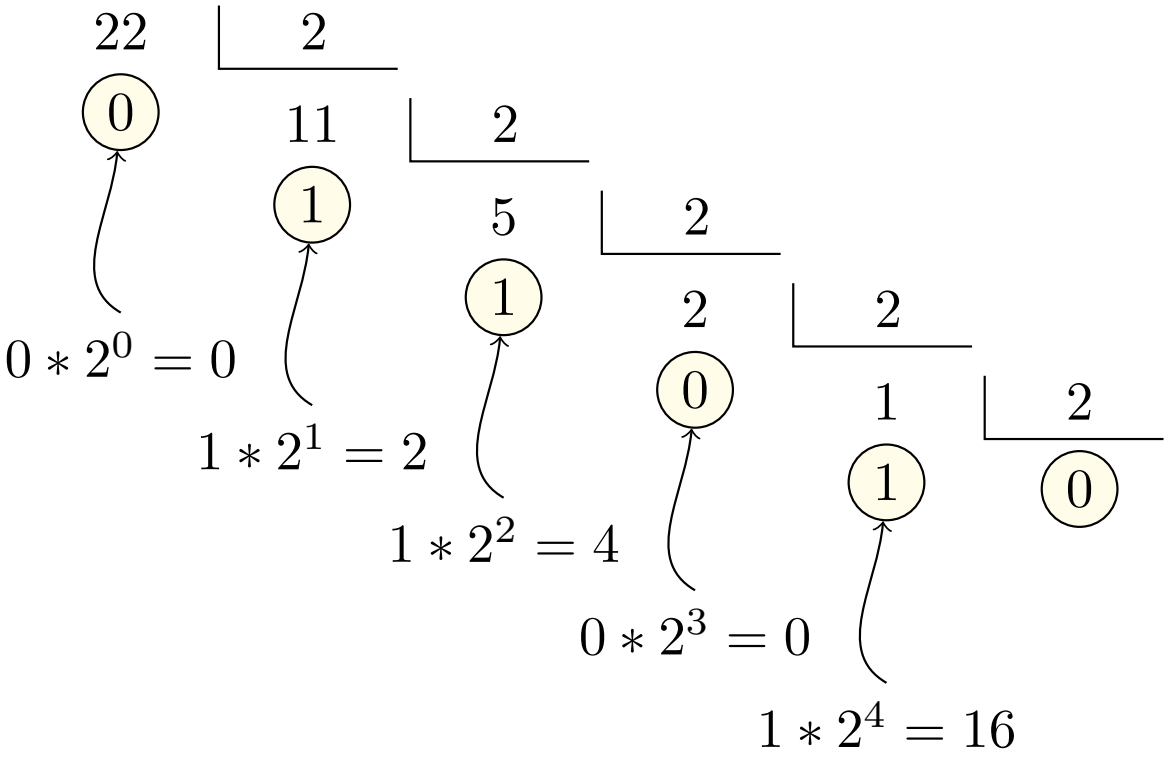
$$3865_{10} = 3 * 10^3 + 8 * 10^2 + 6 * 10^1 + 5 * 10^0$$

- Interpretação de um número binário

$$11011_2 = 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 27_{10}$$

Conversão de decimal para binário - Versão 1

- Com divisões sucessivas por 2: $22_{10} = 10110_2$



Programa - Versão 1

```
1  program converteParaBin_v1;
2  var n: integer;
3  begin
4      write( 'Entre com um numero entre 0 e 255: ' );
5      read( n );
6      while n <> 0 do
7      begin
8          write( n mod 2 );
9          n:= n div 2;
10     end;
11 end.
```

Qual o problema deste programa?

Exercícios

1. Obter um **inteiro** que representa o número binário invertido (ao invés de escrever os dígitos).
2. Utilizar o programa que inverte o número para obter o **inteiro** em binário na ordem correta.

Conversão para binário - Versão 2

- Dado um decimal n, obter a maior potência de 2 que é menor ou igual a n.

Exemplo: Converter 22 para binário

- Maior potência de 22 que é ≤ 22

decimal	**potência de 2**	**binário**
22	$2^4 = 16$	1 | | | | | 2^4 | 2^3 | 2^2 | 2^1 | 2^0

- Sobra 22-16=6 para representar em binário

decimal	**potência de 2**	**binário**
6	$2^2 = 4$	1 | | 1 | | 2^4 | 2^3 | 2^2 | 2^1 | 2^0

- Agora falta 6-4=2 para representar em binário

decimal	**potência de 2**	**binário**
2	$2^1 = 2$	1 | | 1 | 1 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0

- Resultado:

1	0	1	1	0

Considerando conversão para 8 dígitos binários

decimal	pot2	binário
22	$2^7 = 128$	0 | | | | | | | |
22	$2^6 = 64$	0 | 0 | | | | | | |
22	$2^5 = 32$	0 | 0 | 0 | | | | | |
22	$2^4 = 16$	0 | 0 | 0 | 1 | | | | |
6	$2^3 = 8$	0 | 0 | 0 | 1 | 0 | | | |
6	$2^2 = 4$	0 | 0 | 0 | 1 | 0 | 1 | | |
2	$2^1 = 2$	0 | 0 | 0 | 1 | 0 | 1 | 1 | |
0	$2^0 = 1$	0 | 0 | 0 | 1 | 0 | 1 | 1 | 0

Padrão Repetitivo

decimal	pot2	binário
22	$2^7 = 128$	0
22	$2^6 = 64$	0 0
22	$2^5 = 32$	0 0 0
22	$2^4 = 16$	0 0 0 1
6	$2^3 = 8$	0 0 0 1 0
6	$2^2 = 4$	0 0 0 1 0 1
2	$2^1 = 2$	0 0 0 1 0 1 1
0	$2^0 = 1$	0 0 0 1 0 1 1 0

```
1  if decimal < pot2 then
2      write( 0 )
3  else
4  begin
5      write( 1 );
6      decimal:= decimal - pot2;
7  end;
8  pot2:= pot2 div 2;
```

Conversão de decimal para binário - Versão 2

```
1  program converteParaBin_v2;
2  const MAX = 128;
3  var decimal, pot2: integer;
4  begin
5      write( 'Entre com um numero entre 0 e 255: ' );
6      readln( decimal );
7      pot2:= MAX;
8      while pot2 > 0 do
9      begin
10         if decimal < pot2 then
11             write( 0 )
12         else
13         begin
14             write( 1 );
15             decimal:= decimal - pot2;
16         end;
17         pot2:= pot2 div 2;
18     end;
19 end.
```

Cálculo do Máximo Divisor Comum (MDC)

- $mdc(a, b)$: é o maior valor que divide a e b
- Método 1:
  - fatoração dos números como um produto de números primos
  - Exemplo: mdc(72, 135)
    - 72 = 3 * 3 * 2 * 2 * 2
    - 135 = 5 * 3 * 3 * 3
  - o mdc é o produto dos fatores comuns: 3 * 3 = 9

MDC: Método de Euclides

- Obter o resto da divisão do maior pelo menor até que o resto seja zero
- Exemplo: mdc(135, 72)

135	63	63	0		
72	72	9	9		

MDC: Método de Euclides

- Obter o resto da divisão do maior pelo menor até que o resto seja zero
- Exemplo: mdc(135, 72)

135	63	63	0		
72	72	9	9		

$$135 \bmod 72 = 63$$

MDC: Método de Euclides

- Obter o resto da divisão do maior pelo menor até que o resto seja zero
- Exemplo: mdc(135, 72)

135	63	63	0		
72	72	9	9		

$$72 \bmod 63 = 9$$

MDC: Método de Euclides

- Obter o resto da divisão do maior pelo menor até que o resto seja zero
- Exemplo: mdc(135, 72)

135	63	63	0		
72	72	9	9		

$$63 \bmod 9 = 0$$

Padrão Repetitivo

a	135	63	63	0		
b	72	72	9	9		

```
1  if a > b then
2      a:= a mod b
3  else
4      b:= b mod a;
```

MDC - Programa

```
1  program mdc;
2  var a, b : integer;
3  begin
4      readln( a, b );
5      while (a <> 0) and (b <> 0) do
6      begin
7          if a > b then
8              a:= a mod b
9          else
10             b:= b mod a;
11     end;
12     if a = 0 then
13         writeln( 'mdc = ', b )
14     else
15         writeln( 'mdc = ', a );
16 end.
```

Tabuada

- Tabuada do número 7

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

```
1  program tabuada_do_7;
2  var j: integer;
3  begin
4      j:= 1;
5      while j <= 10 do
6      begin
7          writeln( 7, 'x', j, '= ', 7*j );
8          j:= j + 1;
9      end;
10 end.
```

Tabuada de um número i

```
program tabuada_de_i;
var i, j: integer;
begin
        read( i );
        j:= 1;
        while j <= 10 do
        begin
              writeln( i, 'x', j, '= ', i * j );
              j:= j + 1;
        end;
end.
```

Tabuada dos números de 1 a 10

- Aninhamento de comandos repetitivos

```
program tabuada;
var i, j: integer;
begin
        i:= 1;
        while i <= 10 do
        begin
                j:= 1;
                while j <= 10 do
                begin
                        writeln( i, 'x', j, '= ', i * j );
                        j:= j + 1;
                end;
                i:= i + 1;
        end;
end.
```

Exercícios

- Fazer os exercícios da Seção 7.12 (Exercícios 1, 2, 3, 4, 5, 13, 14, 15, 16, 17, 18) do livro [1]

[1] `http://www.inf.ufpr.br/cursos/ci055/livro_alg1.pdf`

Fim do tópico

- o conteúdo desta aula está no livro no capítulo 7, seções 7.1-7.4

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/`