

## 2º. LABORATÓRIO DE ALGORITMOS

**Objetivo:** Entender a estrutura de um programa, trabalhando com organização do código em funções e procedimentos.

**Tarefa:** Escrever um algoritmo/programa que receba dois valores inteiros positivos ( $x$  e  $y$ ) e informar se  $x$  é divisível por  $y$ . Vamos focar no problema quando  $y$  é 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15 ou 25.

Fontes:

- [1] <http://www.somatematica.com.br/fundam/critdiv.php>
- [2] <http://www.mundoeducacao.com/matematica/regras-divisibilidade.htm>
- [3] <http://www.brasilecola.com/matematica/criterios-divisibilidade.htm>
- [4] [http://pt.wikipedia.org/wiki/Critérios\\_de\\_divisibilidade](http://pt.wikipedia.org/wiki/Critérios_de_divisibilidade)

### Requisitos:

Obs: **Em vermelho estão as entradas do usuário.**  
**Em azul está o resultado gerado pelo programa.**

1. Para testar “se  $a$  é divisível por  $b$ ” bastaria verificar se  $a \bmod b = 0$ , ou na linguagem de programação,  $a \% b == 0$ . Mas não vamos usar isso em nenhum teste de divisibilidade deste lab. Vamos usar as regras teóricas sobre divisibilidade.
2. É proibido usar no código: variáveis globais, `break` e `goto`.
3. Todas as funções devem ter apenas um único ponto de retorno. Ou seja, só é permitido ter um `return` ou `return` no final da função.
4. O programa deve conter as funções aqui especificadas, respeitando os protótipos abaixo:

*Considere os protótipos de tipo void como procedimentos e os de tipo bool como funções que retornam um valor lógico*

```
void imprimirObjetivoLab();
bool testarDivisibilidade(int dividendo, int divisor);
bool divisibilidade2(int num);
bool divisibilidade3(int num);
bool divisibilidade4(int num);
bool divisibilidade5(int num);
bool divisibilidade6(int num);
bool divisibilidade7(int num);
bool divisibilidade8(int num);
bool divisibilidade9(int num);
bool divisibilidade10(int num);
bool divisibilidade11(int num);
bool divisibilidade12(int num);
bool divisibilidade15(int num);
bool divisibilidade25(int num);
```

5. A função principal (main) deve usar o procedimento `void imprimirObjetivoLab()` para imprimir a mensagem inicial abaixo:

```
Programa TESTE DE DIVISIBILIDADE
<linha vazia>
O programa tem por objetivo informar se um determinado
numero eh ou nao divisivel por outro.
<linha vazia>
Os testes de divisibilidade sao validos para os seguintes
divisores: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15 e 25.
<linha vazia>
```

6. A função principal (main) deve solicitar o dividendo e o divisor. Considere que o usuário irá fornecer valores inteiros maiores que zero para dividendo e divisor.

```
<linha vazia>
Dividendo: 1400
Divisor: 19
```

7. Enquanto o divisor não for 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15 ou 25, a função principal (main) deve imprimir a mensagem abaixo e solicitar a entrada de novos valores (como no item 6).

```
Divisor invalido! Favor informar novos valores.
```

8. Se o divisor for válido, a função main deve usar a função `bool testarDivisibilidade(int dividendo, int divisor)` para verificar se o dividendo é divisível pelo divisor.

- A função `testarDivisibilidade` retorna Verdadeiro (true) caso o dividendo seja divisível pelo divisor. Caso contrário, retorna Falso (false).
- A função `testarDivisibilidade` reusa, de acordo com o valor do divisor, as funções específicas de divisibilidade por 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15 ou 25.
- Para outros valores de divisor, a função `testarDivisibilidade` retorna false diretamente. Na verdade, isso nem precisava fazer, pois já estamos garantindo no bloco do programa principal (main) que o divisor é válido. Mas é melhor fazer para a função ficar mais completa.

9. Se a função `testarDivisibilidade` retornar true, então a função main deve imprimir a mensagem:

```
1400 eh divisivel por 4
```

10. Se a função `testarDivisibilidade` retornar false, então a função main deve imprimir a mensagem:

```
1400 NAO eh divisivel por 4
```

11. Em seguida, o bloco de programação principal (a função main) deve imprimir a mensagem:

```
<linha vazia>
Deseja realizar novo teste (s/n)?
```

- Leia a resposta do usuário [Na linguagem C, usuário o comando `getche()`].
- Considere que o usuário pode responder com qualquer caractere.
- Enquanto a resposta for diferente de 's' e 'n' (maiúsculas ou minúsculas), o bloco de programa principal (a função `main`) deve perguntar novamente:

```
<linha vazia>
Opcao invalida! Deseja realizar novo teste (s/n)?
```

- Se a resposta for 'n' (maiúsculas ou minúsculas), a função `main` deve encerrar.
- Se a resposta for 's' (maiúsculas ou minúsculas), a função `main` deve solicitar a entrada de novos valores (como no item 6).

12. Fazer uma função que testa a divisibilidade por 2 usando o protótipo `bool divisibilidade2(int num)`.

Retornar `true` se `num` for divisível por 2. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 2 quando é par. Para checar se é par, verificar se o último dígito é: 0, 2, 4, 6 ou 8.

13. Fazer uma função que testa a divisibilidade por 3 usando o protótipo `bool divisibilidade3(int num)`.

Retornar `true` se `num` for divisível por 3. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 3 quando a soma dos seus algarismos for divisível por três.

Exemplo: O número 274 não é divisível por 3 porque:

$$2+7+4 = 13 = 1+3 = 4$$

O número 25848 é divisível por 3 porque

$$2+5+8+4+8 = 27 = 2+7 = 9$$

**Importante:** A função deverá repetir o processo do somatório dos algarismos dos resultados obtidos até que o somatório seja um número com um dígito. Se este dígito for igual a 3, 6 ou 9, então o número original é divisível por 3.

14. Fazer uma função que testa a divisibilidade por 4 usando o protótipo `bool divisibilidade4(int num)`.

Retornar `true` se `num` for divisível por 4. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 4 quando for divisível duas vezes por 2. Nesse caso, reusar a função `divisibilidade2`.

15. Fazer uma função que testa a divisibilidade por 5 usando o protótipo `bool divisibilidade5(int num)`.

Retornar `true` se `num` for divisível por 5. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por cinco quando terminar em 0 ou 5.

16. Fazer uma função que testa a divisibilidade por 6 usando o protótipo  
`bool divisibilidade6(int num)`.

Retornar `true` se `num` for divisível por 6. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 6 quando for divisível por 2 e por 3. Nesse caso, reusar as funções `divisibilidade2` e `divisibilidade3`.

17. Fazer uma função que testa a divisibilidade por 7 usando o protótipo  
`bool divisibilidade7(int num)`.

Retornar `true` se `num` for divisível por 7. Caso contrário, retornar `false`.

Regra de divisibilidade:

Processo	Exemplo para a entrada 7203
i. Obtenha o último algarismo do número	Último algarismo: 3
ii. Multiplique o último algarismo por 2	$3 \times 2 = 6$
iii. Obtenha o valor do número inicial, sem o último algarismo	720
iv. Subtraia o valor inicial sem o último algarismo do resultado da multiplicação	$\text{fabs}(720 - 6) = 714$
v. Repita o processo até que o resultado seja um valor menor que ou igual a 70	Último algarismo: 4 $4 \times 2 = 8$ 71 $\text{fabs}(71 - 8) = 63$
vi. Compare o resultado com a tabuada (0, 7, 14, 21, 28, 35, 42, 49, 56, 63, 70) para determinar se o número é ou não divisível por 7	63 está na tabuada do 7, logo o número inicial 7203 é divisível por 7

Outro exemplo: 77 é divisível por 7. Obs: `fabs` é uma função da `math.h`

18. Fazer uma função que testa a divisibilidade por 8 usando o protótipo  
`bool divisibilidade8(int num)`

Retornar `true` se `num` for divisível por 8. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 8 quando for divisível três vezes por 2. Nesse caso, reusar a função `divisibilidade2`.

19. Fazer uma função que testa a divisibilidade por 9 usando o protótipo  
`bool divisibilidade9(int num)`

Retornar `true` se `num` for divisível por 9. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 9 quando for divisível duas vezes por 3. Nesse caso, reusar a função `divisibilidade3`.

20. Fazer uma função que testa a divisibilidade por 10 usando o protótipo  
`bool divisibilidade10(int num)`

Regra de divisibilidade: Um número é divisível por 10 quando ele terminar em 0.

Retornar `true` se `num` for divisível por 11. Caso contrário, retornar `false`.

**Importante:** A função deverá repetir o processo até que o resultado seja um número com um dígito. Se este dígito for igual a 0, então o número original é divisível por 11.

Exemplos:

1771561 é divisível por 11?						
Ordem Ímpar	Ordem Par	Ordem Ímpar	Ordem Par	Ordem Ímpar	Ordem Par	Ordem Ímpar
1	7	7	1	5	6	1

Soma dos ímpares (Si) ..... =  $1+7+5+1 = 14$   
 Soma dos pares (Sp) ..... =  $7+1+6 = 14$   
 $S_i - S_p$  ..... =  $14-14 = 0$   
 $S_i - S_p = 0$ , logo 1771561 é divisível por 11

### 439087 é divisível por 11?

Ordem Par	Ordem Ímpar	Ordem Par	Ordem Ímpar	Ordem Par	Ordem Ímpar
4	3	9	0	8	7

Soma dos ímpares (Si)
= 3+0+7 = 10

Soma dos pares (Sp)
= 4+9+8 = 21

Si - Sp = 10 - 21 = -11 => 11 (módulo)

Repetindo o processo:

Ordem Par	Ordem Ímpar
1	1

Soma dos ímpares (Si)
= 1

Soma dos pares (Sp)
= 1

Si - Sp = 1 - 1 = 0

Si - Sp = 0, logo 1771561 é divisível por 11

22. Fazer uma função que testa a divisibilidade por 12 usando o protótipo `bool divisibilidade12(int num)`.

Retornar `true` se `num` for divisível por 12. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 12 quando for divisível por 3 e por 4. Nesse caso, reusar as funções `divisibilidade3` e `divisibilidade4`.

23. Fazer uma função que testa a divisibilidade por 15 usando o protótipo `bool divisibilidade15(int num)`.

Retornar `true` se `num` for divisível por 15. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 15 quando for divisível por 3 e por 5. Nesse caso, reusar as funções `divisibilidade3` e `divisibilidade5`.

24. Fazer uma função que testa a divisibilidade por 25 usando o protótipo `bool divisibilidade25(int num)`.

Retornar `true` se `num` for divisível por 25. Caso contrário, retornar `false`.

Regra de divisibilidade: Um número é divisível por 25 quando for divisível duas vezes por 5. Nesse caso, reusar a função `divisibilidade5`.

## Dicas:

**Dica 1: Para pegar o algarismo mais à direita de um número  $x$ , fazer:**

```
int x;  
int algarismo;  
//Código qualquer que define x como um valor inteiro positivo  
algarismo = x%10;
```

**Dica 2: Para retirar o algarismo mais à direita de um número  $x$ , fazer:**

```
int x;  
//Código qualquer que define x como um valor inteiro positivo  
x = x/10;
```

```
//Isso funciona, pois o valor é arredondado para int.  
//Ex: 13/10 = 1.3 => 1  
//Ex: 752/10 = 75.2 => 75
```

Ou você pode usar uma variável auxiliar para não “destruir” o  $x$  original. Veja:

```
int x;  
int num;  
//Código qualquer que define x como um valor inteiro positivo  
num = x/10;
```

**Entregar via MS-Forms.** Esses arquivos deverão ter o nome dos alunos e o número do lab.

Obs: Não esquecer do cabeçalho com os dados do programa.

**Prazo: 13 de outubro**