## Prof. Msc. Elias Batista Ferreira Prof. Dr. Gustavo Teodoro Laureano Profa. Dra. Luciana Berretta Prof. Dr. Thierson Rosa Couto

## Sumário

1	Lê número (+)	2
2	Fatorial (++)	3
3	Número de dígitos (++)	4
4	Cálculo da raiz quadrada (+++)	5
5	Próxima potência (+++)	6
6	Cálculo dos digitos do CPF (++++)	7
7	Cálculo de PI (++++)	9

# 1 Lê número (+)



(+)

Escreva um programa que leia *n* números e apresente sua média. Você deve implementar uma função, denominada le\_numero, que leia um número via terminal e retorne seu valor. A leitura dos dados deve ser realizada via a função le\_numero. A função deve seguir o protótipo:

```
1 /**
2 * Lê um número double via terminal e retorna seu valor.
3 * @return retorna o valor lido.
4 */
5 double le_numero();
```

### **Entrada**

O programa deve ler um número n e em seguida n números reais.

#### Saída

O programa deve apresentar a média dos números lidos com 2 casas decimais.

Entrada	Saída
3 2 2 2	2.00

# **2** Fatorial (++)



Dado um número inteiro n, calcule seu fatorial n!. O fatorial de um número é dado pela equação: n! = n(n-1)(n-2)...1. Por definição, 0! = 1.

Você deve implementar a função:

```
1 /**
2 * Funcao que calcula o fatorial de um numero n
3 * @param n um numero inteiro positivo
4 * @return o fatorial de n
5 */
6 unsigned long int fat( unsigned int n);
```

#### **Entrada**

O programa deve ler um número inteiro n.

#### Saída

O programa deve apresentar uma linha com a mensagem: "n! = f", onde n é o número lido e f o seu fatorial.

### Observações

O fatorial de um número é resultado de uma operação de produtório que pode levar a valores incrivelmente grandes. Lembre-se de usar tipos de dados apropriados ao problema proposto.

Entrada		
2		
Saída		
2!	= 2	

Entrada		
4		
Saída		
4!	= 24	

# 3 Número de dígitos (++)



Escreva um programa que leia um número n e apresente a quantidade de dígitos que ele possui. Você deve implementar a função:

```
1 /**
2 * Funcao que calcula a quantidade de digitos de um numero inteiro
3 * @param n um numero inteiro
4 * @return quantidade de digitos de n
5 */
6 int digit_count(long int n);
```

### **Entrada**

O programa deve ler um número inteiro n.

#### Saída

O programa deve apresentar uma linha com a mensagem: "Numero de digitos: c", onde c é a quantidade de dígitos de n.

Entrada	Saída
123	Numero de digitos: 3

## 4 Cálculo da raiz quadrada (+++)



Os Babilônios utilizavam um algoritmo para aproximar uma raiz quadrada de um número qualquer, da seguinte maneira:

Dado um número n, para calcular  $r = \sqrt{n}$  assume-se uma aproximação inicial  $r_0 = 1$  e calcula-se  $r_k$  para  $k = 1, \ldots, \infty$  até que  $r_k^2 \approx n$ . O algoritmo deve realizar a aproximação enquanto  $|n - r_k^2| > e$ . O método babilônico é dado pela seguinte equação:

$$r_k = \frac{r_{k-1} + \frac{n}{r_{k-1}}}{2} \tag{1}$$

Funções a serem implementadas:

```
1
2 /**
3 * Função que calcula a raiz quadrada de n.
4 * @param n um numero real qualquer
5 * @param e um numero real correspondente ao erro
6 * @return a raiz quadrada de n
7 */
8 double raiz( double n, double e );
9
10 /**
11 * Valor absoluto de um numero qualquer
12 * @param n um número real qualquer
13 * @return o valor absoluto de n
14 */
15 double absoluto( double n );
```

#### Entrada

O programa deve ler um número **double** n, cuja raiz quadrada deseja-se obter, e o erro e que deverá ser considerado pelo algoritmo.

#### Saída

A saída deve apresentar cada iteração do algoritmo, sendo cada linha composta pelo valor aproximado da raiz quadrada de *n* com 9 casas decimais, seguido do erro, também com 9 casas decimais.

```
Entrada

2
0.00001

Saída

r: 1.5000000000, err: 0.250000000
r: 1.416666667, err: 0.006944444
r: 1.414215686, err: 0.000006007
```

## 5 Próxima potência (+++)



Escreva um programa que leia dois números inteiros, n e p, e calcule a número mais próximo de n que seja uma potência inteira de p. Por exemplo, para o número n=5 e p=2 o número mais próximo de p0 que é uma potência inteira de p0 e p1 a p3 e p4 a p5 e p5 o número mais próximo é p5 e p6 de p6 de p7 e p8.

Você deve implementar a função:

```
1 /**
2 * Função que calcula a potencia de p mais próximo a n.
3 * @param n valor inteiro
4 * @param p valor da potencia
5 * @return retorna o valor da potencia mais proxima.
6 */
7 int next_power( int n, int p );
```

#### **Entrada**

O programa deve ler os números inteiros  $n \in p$ .

#### Saída

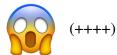
O programa deve apresentar uma linha com o valor da potência mais próxima a n no formato: " $n \rightarrow k$   $\hat{p} = x$ ", onde k é a base da potência e x o valor mais próximo de n.

_

Entrada	Saída
10 3	10 -> 2^3 = 8

Entrada	Saída
40 4	40 -> 2^4 = 16

## 6 Cálculo dos digitos do CPF (++++)



Codifique uma função que cálcule os digitos verificadores do CPF. Verifique a forma de calcular os digitos verificadores do CPF na internet.

Você deve implementar a função:

```
1 /**
2 * Função que calcula os digitos do CPF
3 * @param n número de até 9 digitos correspondendo ao número do CPF do qual se quer descobrir os digitos
4 * @return os digitos do cpf
5 */
6 *int calcular_digitos_cpf( int n );
```

#### **Entrada**

O programa deve ler um número de, no máximo, 9 digitos.

#### Saída

O programa deve apresentar uma linha com o valor dos 2 digitos do CPF. Sempre mostrar exatamente 2 casas.

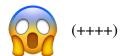
Entrada	Saída
689184370	72

Entrada	Saída
756810245	98

Entrada	Saída
210043620	13

Entrada	Saída
974804937	03

## 7 Cálculo de PI (++++)



Na história da ciência, muitas foram as tentativas de se encontrar o número  $\pi$  com a maior precisão possível. Uma dessas tentativas foi a do matemático John Wallis, que desenvolveu a série infinita da Equação 2 em 1655.

$$\frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \frac{8}{9} \cdot \dots = \frac{\pi}{2}$$
 (2)

Faça um programa que calcula o valor de *pi* usando a série proposta, permitindo o usuário definir a quantidade de termos da série.

Você deve implementar a função:

```
1 /**
2 * Função que calcula o valor de pi usando a série proposta por John Wallis
3 * @param n quantidade de termos da série
4 * @return o valor aproximado da constante pi
5 */
6 double compute_pi( int n );
```

#### **Entrada**

O programa deve ler a quantidade de termos *n*.

#### Saída

O programa deve apresentar uma linha com o valor de  $\pi$  com 12 casas decimais.

Entrada	Saída
3	3.55555555556

Entrada	Saída
13	3.255721745232

Entrada	Saída
89	3.159094156315