Lista de Exercícios 6

Introdução a Linguagem de Programação (2COP005)

Prof. Sérgio Montazzolli Silva

Data da entrega: 25/06/19 (Terça), até as 23:59

Como entregar

Crie uma pasta com o nome "Lista 6" e dentro dela coloque os arquivos ".c" correspondentes a cada exercício pedido. Nomeie os arquivos como "ex1.c", "ex2.c" e assim por diante, para cada exercício. Ao final, gere um arquivo ZIP ou RAR desta pasta e envie por e-mail para ***smsilva@uel.br*** com o título "*Lista 6- 2COP005 - Seu Nome Completo*".

Obs. 1: Antes de enviar, verifique se o arquivo compactado contém todos os exercícios e se ele não está corrompido. A nota será proporcional ao número de exercícios resolvidos, e arquivos corrompidos invalidam a entrega.
Obs. 2: Não incluir arquivos ".exe" no envio, apenas arquivos com extensão ".c". Caso estes arquivos executáveis sejam incluídos, o antivírus do seu serviço de e-mail poderá impedir o envio.

Exercícios

Em todos os exercícios, quando não for explicitado, crie programas que permitam que o usuário entre com os dados de teste.

Exercício 1. Números primos são divisíveis apenas por 1 e por eles mesmos. Crie uma função que faça a checagem se um número é primo ou não. A função deve receber um número inteiro para análise, e responder com: 0 para falso (não é primo); 1 para verdadeiro (é primo).

Exercício 2. Crie uma função que receba 3 números reais a,b e c, e verifique se c está no intervalo de $[a, b]$. Caso sim, retorne o valor 1, e caso não, retorne -1.

Exercício 3. A soma dos ângulos internos de um triângulo deve sempre ser igual a 180°. Crie uma função que receba como argumentos os ângulos internos, e retorne 0 quando não for um triângulo, ou 1 quando for. A função também deve retornar 0 se qualquer um dos 3 ângulos for negativo.

Exercício 4. Ainda sobre triângulos, crie uma função que receba 2 de seus 3 ângulos internos, e retorne o valor do terceiro ângulo. O cálculo deve respeitar a restrição de que a soma dos 3 ângulos seja 180°. Utilize a função do exercício anterior para checagem. Não permita que o usuário entre com valores negativos.

Exercício 5. Dados dois números a e b, lidos pelo seu programa, crie as seguintes funções:

(a) *maior*: retorna a se $a > b$, ou b caso contrário;

(b) *menor*: retorna a se $a < b$, ou b caso contrário;

(c) *resto*: retorna o resto da divisão de a por b;

(d) *imprime*: sem retorno, apenas imprime na tela os valores de a e b. Exemplo: supondo $a = 201.2$ e $b = 10.9$, imprimir:

```
Argumento a: 201.2
Argumento b: 10.9
```

Exercício 6. A função *rand*() é usada para gerar um número inteiro aleatório. O programa abaixo mostra dois exemplos de uso desta função. No primeiro exemplo é gerado um número aleatório, e no segundo é gerado um número aleatório entre 0 e 9 (com base no resto da divisão pelo primeiro número).

```
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    int r = rand();
    printf("Numero aleatorio: %d",r);
    int rq = r%10;
    printf("Numero aleatorio entre 0-9: %d",rq);
}
```

Pede-se:

(a) Crie uma função chamada *gera_numero*, que receba como argumentos dois números inteiros a e b, e retorne um número inteiro c qualquer, no intervalo $[a, b]$;

(b) Crie um vetor de inteiros de 30 posições, e associe um número aleatório entre -100 e 100 para cada posição. Utilize para isso a função criada no item anterior. Depois, crie outra função para realizar a soma de todos os elementos deste vetor.

Exercício 7. Utilizando a função *imprime*, do exercício 5, e a função *gera_numero*, do exercício 6(a), crie um programa que gere dois números a e b aleatórios e os imprima na tela. Cada número deve estar dentro do intervalo $[0, 100]$, e o programa deve fazer esta ação repetidamente por 200 vezes.

Exercício 8. A biblioteca *math.h* possui a função $pow(x, y)$ que realiza a operação x^y, ou seja, x elevado a y. Neste caso, x e y são números reais. Crie um programa que tenha uma função chamada *minha_pow*, e que faça exatamente o que a função *pow* faz. Porém, *minha_pow* deve receber como argumentos dois números inteiros, e retornar outro número inteiro como resultado. **Não** utilize a função *pow* padrão no seu programa.

Exercício 9. A função $log(x)$, que retorna o logaritmo natural de x, está implementada na biblioteca *math.h*. Nós podemos facilmente aproximar esta função, quando x está no intervalo $[0, 2]$, através da seguinte serie de Taylor:

$$log(x) = -\sum_{n=1}^{\infty} \frac{(1-x)^n}{n}$$

Crie uma função chamada $minha\_log(x, n)$ que receba um número x, no qual pretende-se computar seu logaritmo natural, e um número n, que represente as iterações da aproximação. Compare *minha_log* com *log* para os seguintes valores:

x	n	$log(x)$	$minha\_log(x, n)$	Valor esperado ($minha\_log(x, n)$)
1.8	1			0.800000
1.8	2			0.480000
1.8	3			0.650667
1.8	4			...
1.8	5			...
1.8	10			...
1.8	100			0.587787
1.8	1000			0.587787

A implementação do aluno deve bater com os valores esperados. Os valores das células que não foram preenchidas devem ser encontrados pelo aluno, e colocados no código em forma de comentário.