Professor: Claudio Fabiano Motta Toledo (`claudio@icmc.usp.br`)
Estagiário PAE: Jesimar da Silva Arantes (`jesimar.arantes@usp.br`)

Essa lista de exercícios tem como objetivo principal desenvolver algoritmos a partir dos conteúdos abordados em sala de aula. Todos os exercícios também devem ser implementados em linguagem C. Nos programas que pedem para implementar apenas funções desenvolva também o programa principal (main) para testá-los. Não utilize variáveis globais.

1. Desenvolva um programa para testar as funções a seguir:

   - Uma função que receba dois números a e b, em seguida, faça a troca destes dois números. Dica: a e b devem ser passados por referência.
   - Uma função que receba dois números a e b, em seguida, decremente o primeiro e incremente o segundo. Dica: a e b devem ser passados por referência.
   - Uma função que receba o raio, perímetro e área de um círculo. Em seguida, calcule e retorne nas variáveis perímetro e área o respectivo perímetro e área do círculo. Dica: perímetro e área devem ser passados por referência.
   - Uma função que receba o lado, perímetro e área de um quadrado. Em seguida, calcule e retorne nas variáveis perímetro e área o respectivo perímetro e área do quadrado. Dica: perímetro e área devem ser passados por referência.
   - Uma função que receba os valores a, b e c passados por valor, receba também dois valores $x1$ e $x2$ passados por referência. Em seguida, calcule e retorne as duas raízes da equação do segundo grau nas variáveis $x1$ e $x2$.

2. Desenvolva um programa que receba coordenadas cartesianas digitadas pelo usuário e imprima seus valores convertidos para coordenadas polares. O usuário poderá digitar quantas coordenadas ele desejar. Utilize uma função que converte de coordenada cartesiana (x, y) para coordenada polar (r, s).

3. Desenvolva um programa que primeiro utilize uma função para gerar um vetor com tamanho definido pelo usuário e valores aleatórios dentro de um intervalo $[min, max]$ determinado pelo usuário. Em seguida, uma outra função deve receber o vetor gerado, o tamanho do vetor e dois números *maior* e *menor*. Retorne nas variáveis *maior* e *menor* o menor e maior elemento do vetor. Dica1: *menor* e *maior* devem ser passados por referência. Dica2: use apenas uma estrutura de repetição para achar o menor e maior elemento.

4. Desenvolva um programa que utilize uma função para inverter os elementos de um vetor (com valores gerados usando a função do exercício anterior) sem utilizar um vetor auxiliar. Apenas o vetor fornecido à função deverá ser utilizado.

5. Desenvolva um programa para testar uma função com protótipo *void somabit(int b1, int b2, int *vaium, int *soma);*. A função recebe três bits (inteiros 0 ou 1) b1, b2 e *vaium. A função retorna um bit soma que armazena o resultado da soma dos três primeiros e o bit "vai-um" em *vaium.

6. Desenvolva um programa que leia a quantidade total de segundos e converta para Horas, Minutos e Segundos. Imprima o resultado da conversão no formato HH:MM:SS. Para isso, utilize a função com protótipo *void converteHora(int total_segundos, int *hora, int *min, int *seg).*

7. Desenvolva um programa que utilize os parâmetros argv e argc. Logo, o programa principal (*main*) deverá receber o dia, mês e ano. Utilizando funções, o programa deverá verificar se os valores fornecidos formam uma data válida. Se for válida, o programa principal imprime a data como ilustrado a seguir:

   Entrada: 01/11/2011

   Saída: 01 de novembro de 2011

8. Desenvolva um programa que utilize os parâmetros argv e argc para informar à função principal a dimensão MxN de uma matriz e os valores min e max. Em seguida, o programa principal utiliza uma função para gerar a matriz MxN com valores aleatórios dentro de um intervalo $[min, max]$. Uma outra função deve receber a matriz gerada e retornar o maior elemento da matriz, o menor elemento da matriz e o valor médio das entradas da matriz.

9. Desenvolva uma função recursiva que receba um número n, calcule e retorne o somatório de 1 até n.

10. Desenvolva uma função recursiva que calcule e retorne o fatorial de um número n.

11. Desenvolva uma função recursiva que calcule e retorne o enésimo termo de fibonacci.

12. Desenvolva uma função recursiva que calcule e retorne o valor de $a \times b$ usando apenas somas, onde a e b são inteiros não-negativos.

13. Desenvolva uma função recursiva que calcule e retorne o valor de $a + b$ usando apenas incremento de uma unidade, onde a e b são inteiros não-negativos.

14. Desenvolva uma função recursiva que calcule e retorne o maior elemento em um vetor.

15. Desenvolva uma função recursiva que calcule e retorne o menor elemento em um vetor.

16. Desenvolva uma função recursiva que calcule e retorne a média dos elementos de um vetor.

17. Desenvolva uma função recursiva que calcule e retorne a soma dos dígitos decimais de um inteiro positivo. Por exemplo, a soma dos dígitos de 315 é 9.

18. Desenvolva uma função recursiva que calcule e retorne a conversão um inteiro na base decimal para a base binária.

19. Desenvolva uma função recursiva que verifique se uma determinada palavra é palíndromo ou não.

20. Desenvolva uma biblioteca usando arquivos de cabeçalhos para o seu uso pessoal. Essa biblioteca deverá possuir uma série de funções matemáticas como:
    (a) Calcular o fatorial de forma iterativa: int factorialI(int n);
    (b) Calcular o fatorial de forma recursiva: int factorialR(int n);
    (c) Calcular o fibonacci de forma iterativa: int fibonacciI(int n);
    (d) Calcular o fibonacci de forma recursiva: int fibonacciR(int n);
    (e) Calcular a distância entre dois pontos: double distancia(double x1, double y1, double x2, double y2);
    (f) Calcular a raiz quadrada de um número: double raiz(double n);
    (g) Gerar um número aleatório entre 0 e 1: double randDouble();

21. Desenvolva uma biblioteca usando arquivos de cabeçalhos para o seu uso pessoal. Essa biblioteca deverá possuir uma série de funções que manipulam vetores como:
    (a) Inicializar um vetor com valores aleatórios entre [min, max]: void init_vector(int vet[], int size, int min, int max);
    (b) Imprimir um vetor qualquer: void print_vector(int vet[], int size);
    (c) Trocar duas posições de um vetor de lugar: void swap_vet(int vet[], int pos_a, int pos_b);
    (d) Verificar se o vetor está em ordem crescente: int ascending_order(int vet[], int size);
    (e) Verificar se o vetor está em ordem crescente: int descending_order(int vet[], int size);