**Lista de exercícios 1**

**Disciplina**: Estruturas de Dados I
**Professora**: Juliana Pinheiro Campos Pirovani

1) Implemente uma função para testar se um número inteiro é primo ou não. Um número é primo quando ele tem somente dois divisores: o número 1 e ele mesmo. Essa função deve obedecer ao protótipo a seguir e ter como valor de retorno 1 se n for primo e 0 caso contrário.

```
int primo (int n);
```

2) Implemente uma função que retorne a soma dos n primeiros números naturais ímpares. Essa função deve obedecer ao protótipo:

```
int soma_impares (int n);
```

Ex: Para n = 2 → soma = 1 + 3 = 4. Para n = 3 → soma = 1 + 3 + 5 = 9, etc.

3) Implemente uma função que retorne uma aproximação do valor de pi de acordo com a fórmula de Leibniz:

$$\pi \simeq 4 * (1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots)$$

Isto é

$$\pi \simeq 4 * \sum_{i=0}^{n} \frac{-1^i}{2 * i + 1}$$

A função deve obedecer ao seguinte protótipo, em que n indica o número de termos que devem ser usados para avaliar o valor de pi:

```
double pi(int n);
```

DICA: Utilize a função pow da biblioteca math.h.

4) Vamos supor que um número real seja representado por uma estrutura em C, como esta:

```
struct realtype {
        int left;
        int right;
      };

typedef struct realtype Real;
```

onde left e right representam os dígitos posicionados à esquerda e à direita do ponto decimal, respectivamente. Se left for um inteiro negativo, o número real representado será negativo.

a) Escreva uma função para ler um número real (solicitando a parte inteira e a parte fracionaria desse número) e armazenar em uma estrutura Real passada como parâmetro. Seu protótipo é:

void leReal(Real * r);

b) Escreva uma função que receba uma estrutura real como parâmetro e imprime o número real representado por ela no formato “xx.xx”. Seu protótipo é:

void imprimeReal(Real * r);

c) Escreva uma função chamada soma que recebe duas estruturas Real como parâmetros e defina o valor de uma terceira estrutura para representar o número que seja a soma das duas estruturas de entrada. Seu protótipo é:

Real soma(Real* r1, Real * r2);

5) Implemente uma função que receba como parâmetro um vetor de números inteiros v de tamanho n e inverta a ordem dos elementos armazenados nesse vetor. Essa função deve obedecer ao protótipo:

void inverte (int n, int *v);

6) Implemente uma função que receba uma string como parâmetro e substitua todas as letras por suas sucessoras no alfabeto. Por exemplo, a string “Casa” seria alterada para “Dbtb”. Essa função deve obedecer ao seguinte protótipo:

void shift_string (char* str);

A letra z deve ser substituída pela letra a ( e Z por A). Caracteres que não forem letras devem permanecer inalterados.

7) Reimplemente a função do exercício 7 para que retorne uma nova string alocada dentro da função, com o resultado esperado, preservando a string original str inalterada. O novo protótipo da função deve ser:

char* shift_string (char* str);

OBS: Se necessário, utilize a função strlen da biblioteca string.h.

8) Implemente uma função que indique se um ponto (x,y) está localizado dentro ou fora de um retângulo. O retângulo é definido por seus vértices inferior esquerdo e superior direito. A função deve retornar 1, se o ponto estiver dentro do retângulo, e 0 caso contrário, obedecendo ao protótipo:

int dentro_ret (Ponto *ie, Ponto *sd, Ponto *p);

Sendo ie o ponteiro para o vértice inferior esquerdo, sd o ponteiro para o vértice superior direito e p o ponteiro para o ponto que deseja saber se está dentro ou fora do retângulo.

OBS:
- Deve ser definida uma estrutura para o ponto.
- Crie o nome de tipo Ponto para a estrutura criada.
- Uma função principal (main) para testar o programa deve solicitar as coordenadas dos 3 pontos.
- A função dentro_ret tem como parâmetros um ponteiro para cada um dos 3 pontos definidos.

9) Escreva uma função recursiva, potencia(x,y), que retorne x elevado a potência y.

10) Escreva uma função recursiva, SomaSerie(i,j,k: inteiro): inteiro, que devolva
a soma da série de valores do intervalo [i,j], com incremento k. OBS: i e j sempre devem ser somados a serie.