Universidade Federal do ABC
Programação Estruturada
Fabrício Olivetti França
Lista de Exercícios 02

1 Objetivos da lista

Esta lista de exercícios tem como objetivo introduzir funções na linguagem C. Como declarar funções, passagem de parâmetros e retorno.

2 Exercícios

2.1 Crie uma função em C que:

1. Receba um valor inteiro e retorne 0 caso par e 1 caso ímpar.

```
Entrada: 2
Saida  : 0

Entrada: 3
Saida  : 1
```

2. Receba os valores em ponto flutuante de largura, comprimento e altura de um objeto e calcule seu volume (a saída deve ter apenas duas casas decimais).

```
Entrada: 2 2 2
Saida  : 8.00

Entrada: 2.1 3.5 5.6
Saida  : 41.16
```

3. O algoritmo Gauss-Legendre aproxima o valor de π com rápida convergência, com apenas 25 iterações ele consegue produzir os primeiros 45 milhões de dígitos. Começando com $a = 1, b = 1/\sqrt{2}, t = 1/4, p = 1$, repete-se a seguinte sequência de instruções até convergência:

```
a1 = (a+b)/2
b  = raiz(a*b)
t  = t - p*(a - a1)^2
p  = 2p
a  = a1
```

Ao final, $\pi = (a+b)^2/(4t)$. Implemente esse algoritmo e busque os dígitos de seu resultado após 10 repetições no site http://www.angio.net/pi/ Questões para pensar: os dígitos são os mesmos? O que pode impedir uma precisão arbitrária em seu código?

A saída do algoritmo deve ser o valor de pi encontrado com 20 casas decimais.

4. Com os conhecimentos obtidos na lista 01 faça um programa calculadora, este programa deve receber uma *operacao* e após saber a operação os valores em ponto flutuante ($float$) devem ser inseridos. A calculadora deve conter as operações de soma ($x + y$) (*opção 1*), subtração ($x - y$) (*opção 2*), multiplicação ($x * y$) (*opção 3*), divisão ($\frac{x}{y}$) (*opção 4*) e potenciação (x^y) (*opção 5*). Cada operação deve ser encapsulada em uma função. Toda saída do tipo ponto flutuante deve ter apenas duas casas decimais.

```
Entrada: 5 2 5
Saida  : 32

Entrada: 3 3 5
Saida  : 15
```

5. Faça um programa que calcule o *coeficiente binomial* $\binom{n}{k}$, o usuário deve inserir os valores inteiros n e k e o programa deve retornar o resultado de $\binom{n}{k}$. Dica: $\binom{n}{k} = \frac{n!}{k!*(n-k)!}$

Obs.: O programa deve ter suas operações modularizadas em funções, ou seja, uma função para calcular o fatorial e outra para calcular o coeficiente binomial.

```
Entrada: 4 2
Saida  : 6
```

6. Faça um programa que encontre as raízes de uma função quadrática usando a fórmula de Bhaskara. A entrada deve ser inserida como a sequência dos coeficientes a, b e c (tipo *double*) e o programa deve imprimir a equação inserida e calcular as raízes, reais ou complexas. As operações devem ser modularizadas em funções. Toda saída do tipo ponto flutuante deve ter apenas duas casas decimais.

```
Entrada: 1 3 1
Saida  : -2.61 -0.381

Entrada: 1 1 3
Saida  : -0.50 - 1.66i -0.50 + 1.66i
```

7. Crie um programa conversor, da mesma forma que o problema 2, a entrada do usuário deve ser o *tipo de conversão*, a *unidade de entrada*, a *unidade de saída* e o valor a ser convertido. Os valores usados devem ser pontos flutuantes ($float$) e as operações de conversão devem ser modularizadas em funções. Para o *tipo de conversão* teremos duas opções, (*opção 1*) temperatura e (*opção 2*) base numérica. Na conversão de temperatura teremos 3 opções de *valores de entrada e saída*: (*opção 1*) Celcius, (*opção 2*) Fahrenheit e (*opção 3*) Kelvin. Para conversão de base numérica teremos 2 opções de *valores de entrada e saída*: (*opção 1*) Decimal, e (*opção 2*) Binário. Toda saída do tipo ponto flutuante deve ter apenas duas casas decimais.

```
Entrada: 1 1 2 10
Saida  : 50.00 F

Entrada: 2 1 2 10
Saida  : 1010
```

8. (Problema) Retornando ao problema **16** da lista 1, faça um programa que modularize a solução deste problema, ou seja, uma função para calcular os descontos e outra para calcular os acréscimos, assim como uma função para clacular os salários baseado nos descontos e acréscimos.

```
Entrada: 1 2 10
Saida  : 10025

Entrada: 5 1 20
Saida  : 2950
```

2.2 Resolva os problemas a seguir usando recursão:

9. (Problema) Faça um programa que calcule a potência (x^y) de inteiros inseridos na forma x y.

```
Entrada: 2 6
Saida  : 64

Entrada: 3 3
Saida  : 27
```

10. (Problema) Dada uma array de *int* de tamanho 1000 chamada *fibMem* que é acessível em qualquer ponto do programa. Modifique o algoritmo recursivo de Fibonacci visto em aula para verificar se a array contém o valor de Fibonacci requisitado e, caso não tenha, preencha com tal valor.

```
#define <stdio.h>

int fibMem[1000];

int fib (int n)
{
   if (n==0) return 0;
   if (n<=2) return 1;
   /* altere a linha abaixo */
   return fib(n-1) + fib(n-2);
}

int main ()
{
    int n;

    fibMem[1] = 1;
    fibMem[2] = 1;

    scanf("%d", &n);
    printf("%d\n", fib(n));
```

```
    return 0;
}
```

```
Entrada: 4
Saida  : 3

Entrada: 14
Saida  : 610
```

11. (Problema) Faça um programa que converta um número decimal para binário e vice-versa (decimal para binario *opção 1*, binario para decimal *opção2*).

```
Entrada: 4 1
Saida  : 100

Entrada: 10000 2
Saida  : 16
```

12. (Problema) Faça um programa que calcule o *coeficiente binomial* $\binom{n}{k}$, o usuário deve inserir os valores inteiros n e k e o programa deve retornar o resultado de $\binom{n}{k}$. Faça o calculo utilizando triângulo de Pascal.

```
Entrada: 4 2
Saida  : 6
```

13. (Problema) Faça um programa para achar o máximo divisor comum entre dois inteiros inseridos pelo usuário.

```
Entrada: 3 6
Saida  : 3

Entrada: 10 50
Saida  : 10
```

14. (Problema) Faça um programa que diga se um número inteiro inserido pelo usuário é palídromo ou não.

```
Entrada: 1221
Saida  : sim

Entrada: 1234
Saida  : nao
```

2.3 Desafio

15. (Desafio) A Torre de Hanoi é um quebra-cabeça matemático no qual temos três bastões e n discos. O objetivo deste quebra-cabeça é mover a pila de discos de um bastão para o outro seguindo as 3 regras a seguir:

1. Somente um disco pode ser movido por vez;
2. Somente os discos do topo da pilha podem ser movidos e só podem ser colocados sobre o topo de outra pilha (ou em um bastão que não tenha nenhum disco).
3. Discos não podem ser colocados sobre discos menores, o tamanho dos discos decresce da base ao topo da pilha.

Faça um programa que receba o número de discos e retorne os movimentos realizados para resolver o quebra-cabeça.

```
Entrada: 2
Saida  : A–B, A–C, B–C

Entrada: 3
Saida  : A–C, A–B, C–B, A–C, B–A, B–C, A–C
```

Dica: tente resolver primeiro sem o uso de programação

http://www.dynamicdrive.com/dynamicindex12/towerhanoi.htm