# MAC0115 – Introdução à Computação para Ciências Exatas e Tecnologia

Instituto de Física — Turma 21 — Segundo Semestre de 2008

Segundo Exercício-Programa — Data de entrega: até **22 de outubro de 2008.**

## Seno e Cosseno

Neste exercício-programa, você deverá escrever um programa em `C` para imprimir juntos os gráficos das funções seno e cosseno. Mais especificamente, o seu programa deverá ter como entrada três números: dois números reais $a$ e $b$, $a < b$, representando o intervalo em que queremos que as funções sejam calculadas e o terceiro número, um inteiro $n > 0$, que representará a qualidade do gráfico impresso conforme explicado abaixo.

Para imprimir o gráfico de cada uma destas funções, você deverá calcular o valor de cada uma delas em $n$ pontos equidistantes do intervalo $[a, b]$. Seu programa deve imprimir os eixos e, no gráfico, assinalar o valor de *sen(x)* com `+` e o valor de *cos(x)* com `x`, como no exemplo que está na última página deste enunciado. Quando os valores de *sen(x)* e *cos(x)* coincidem entre si ou com o eixo, seu programa deve assinalar o valor com `*`, como no exemplo da última página.

O seu programa deve ter pelo menos as seguintes funções:

(1) Uma função seno, com protótipo

```
float seno(float x, float epsilon);
```

Essa função recebe como parâmetros dois números reais $x$ e $\varepsilon > 0$. Ela devolve o valor de seno de $x$, utilizando uma aproximação da série

$$\frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^i \frac{x^{2i+1}}{(2i+1)!} + \cdots,$$

Devem ser incluídos na aproximação todas as parcelas até a primeira com valor absoluto menor que $\varepsilon$.

(2) Uma função cosseno, com protótipo

```
float cosseno(float x, float seno_x);
```

Essa função recebe como parâmetros dois números reais $x$ e *seno_x*. Ela devolve o valor do cosseno de $x$, calculado com o uso da identidade $\cos^2 x + \operatorname{sen}^2 x = 1$.

(3) Uma função raiz quadrada, com protótipo

```
float raiz_quadrada(float x, float epsilon);
```

Essa função recebe dois números reais $x >= 0$ e $\varepsilon > 0$. Ela calcula a raiz quadrada de $x$ usando o método de Newton, descrito a seguir.

Escolhe-se como chute inicial para $\sqrt{x}$ o número $r_0 = x$ e calcula-se a seguinte seqüência de números:

$$r_{n+1} = \frac{1}{2}\left(r_n + \frac{x}{r_n}\right) \qquad n = 0, 1, 2, \ldots$$

(Ou seja: obtemos

$$r_1 = \frac{1}{2}\left(r_0 + \frac{x}{r_0}\right) = \frac{1}{2}\left(x + \frac{x}{x}\right) = \frac{x+1}{2},$$

a partir de $r_1$ obtemos $r_2$ e assim por diante.)

Esse processo deve ser repetido enquanto $|r_{n+1} - r_n| \geq \varepsilon$. A aproximação de $\sqrt{x}$ será o primeiro valor $r_{n+1}$ para o qual $|r_{n+1} - r_n| < \varepsilon$.

Nas chamadas às funções `seno` e `raiz_quadrada`, passe como `epsilon` a constante $\varepsilon = 10^{-8}$. Para definir essa constante, coloque no ínicio do seu programa este `#define`:

```
#define EPSILON 1.0e-8
```

**Dica:** Enquanto estiver escrevendo e testando versões intermediárias do seu programa, você poderá usar as funções seno, cosseno e raiz quadrada da biblioteca matemática da linguagem `C`. Para tanto, coloque a diretiva `#include <math.h>` no início do programa. As funções se chamam `sin`, `cos` e `sqrt`.

A versão final do programa não deve usar a biblioteca matemática.

## Observações Importantes sobre os Exercícios-Programa

1) Todos os exercícios-programa devem ter um cabeçalho como o seguinte:

```
/*****************************************************/
/* Aluno: Fulano de Tal                              */
/* Número USP: 12345678                              */
/* Exercicio-Programa 2 -- Data de Nascimento        */
/* MAC115 -- 2008 -- IFUSP, turma 21 -- Prof. Reverbel */
/* Compilador: ... (gcc ou DevC++) versão ...        */
/*****************************************************/
```

2) O exercício-programa é estritamente individual. Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO.
3) Exercícios atrasados não serão aceitos.
4) Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO. Seu programa deve ser compilável sem erros ou *warnings*, com o compilador no modo em que todos os *warnings* possíveis são emitidos. Caso você use o `gcc`, passe ao compilador (na linha de comando) as opções "`-Wall -ansi -pedantic -O2`". Caso você use o `DevC++`, clique em "Ferramentas" (ou "Tools") e "Opções do Compilador" (ou "*Compiler Options*") e, na tela de opções do compilador, marque como selecionada a opção "Adicione os seguintes comandos quando chamar o compilador" (ou "*Add the following commands when calling compiler*"). Na caixa de texto que aparece logo depois dessa opção, digite "`-Wall -ansi -O2`". (Não use `-pedantic` com o DevC++.)
5) É muito importante que seu programa tenha comentários e esteja bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa (conforme visto em aula). A avaliação dos exercícios-programa levará isto em conta.
6) Cada programa deve ter sido executado tantas vezes quantas forem necessárias para testar os vários casos possíveis para as entradas.
7) Você entregará seu exercício-programa através do sistema Paca/Moodle (`http://paca.ime.usp.br`).
8) Entregue apenas o programa fonte em C, num arquivo com nome `ep2-`<*seu-número-USP*>`.c`. (Exemplo: se seu número USP for 12345678, você deverá entregar um arquivo `ep2-12345678.c`.)
9) Enquanto o prazo de entrega não expirar, você poderá entregar várias versões do mesmo exercício-programa. Apenas a última versão entregue será guardada pelo sistema. Encerrado o prazo, o sistema não aceitará mais a entrega de exercícios-programa. Não deixe para entregar seu exercício na última hora!
10) Guarde uma cópia do seu exercício-programa pelo menos até o final do semestre.

# <u>Exemplo de execução do programa</u>

```
Digite a e b: 0 7
Digite n: 50
 -1                    0                     1
 -+--------------------+---------------------+->
                       *                   x
                       | +                x
                       |    +             x
                       |       +         x
                       |         +     x
                       |           +  x
                       |            x+
                       |          x    +
                       |       x         +
                       |     x            +
                       |  x               +
                       *                  +
                    x  |                  +
                  x    |                  +
               x       |                 +
            x          |                +
          x            |              +
        x              |            +
      x                |          +
     x                 |        +
    x                  |     +
   x                   |   +
   x                   |+
   x                 + |
   x              +    |
    x          +       |
     x       +         |
      x    +           |
        *              |
       +  x            |
     +       x         |
    +          x       |
   +              x    |
   +                 x |
   +                   *
   +                   |  x
    +                  |     x
     +                 |        x
      +                |          x
        +              |            x
          +            |              x
            +          |                x
               +       |                 x
                 +     |                  x
                    +  |                  x
                       *                  x
                       |  +               x
                       |    +             x
                       |       +         x
```