MAC115 – Introdução à Computação para Ciências Exatas e Tecnologia
Instituto de Física – Outubro de 2012

Terceiro Exercício-Programa (EP3)
Data de entrega: 31/10/2012

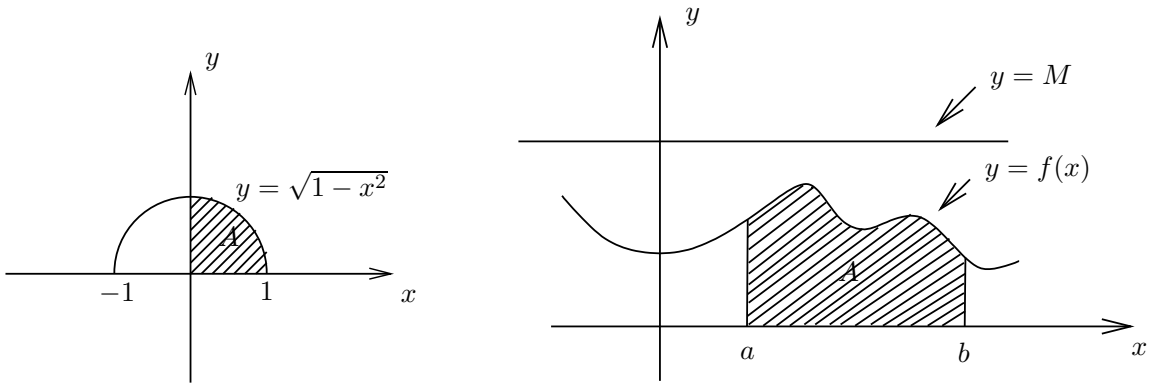
1 Cálculo do valor de π

Considere a função

$$f(x) = \sqrt{1 - x^2}, \quad x \in [-1, 1] \tag{1}$$

que é a semi-circunferência positiva de raio 1.

Sabemos que a área de uma circunferência de raio 1 é π. Logo, a área A de $f(x)$ no intervalo $[0, 1]$ (veja a área hachurada no primeiro quadrante do gráfico na Figura 1(a)) é $\pi/4$. Se soubermos calcular A, então podemos obter o valor de π.



(a) $f(x) = \sqrt{1 - x^2}$ (b) Uma função qualquer

Figura 1: Área sob o gráfico de uma função.

2 Área sob o gráfico de uma função

Seja f uma função positiva no intervalo $[a, b]$ tal que $f(x) \leq M$ para todo $x \in [a, b]$. A região hachurada na Figura 1(b) corresponde à área A de $f(x)$, no intervalo $[a, b]$. A seguir apresentamos três métodos que podem ser utilizados para calcular A.

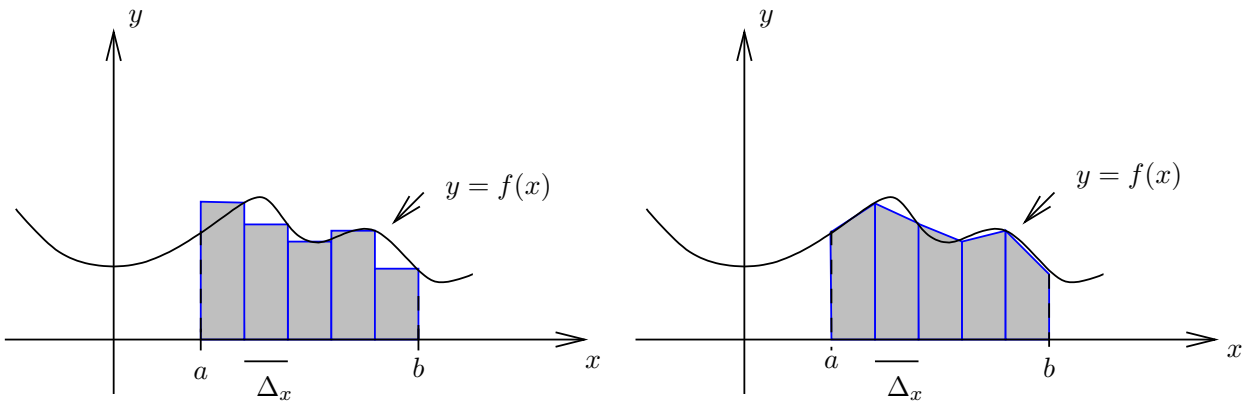
2.1 Método dos retângulos

O cálculo da área A pelo **método dos retângulos** é definido da seguinte forma

$$\begin{aligned} A &\approx f(a + \Delta_x) \cdot \Delta_x + f(a + 2 \cdot \Delta_x) \cdot \Delta_x + \cdots + f(a + k \cdot \Delta_x) \cdot \Delta_x \\ &= \Delta_x \cdot [f(a + \Delta_x) + f(a + 2 \cdot \Delta_x) + \cdots + f(a + k \cdot \Delta_x)] \end{aligned} \tag{2}$$

na qual k é o número de retângulos e $\Delta_x = (b - a)/k$ é a largura dos retângulos.

A Figura 2(a) mostra um exemplo com $k = 5$. Observe que a precisão do resultado depende de k, ou seja, quanto maior k, mais próximo o valor calculado será do valor da área.



(a) Método dos retângulos (b) Método dos trapezóides

Figura 2: Dois métodos para cálculo de área.

2.2 Método dos trapezóides

O cálculo da área A pelo método dos trapezóides é definido da seguinte forma

$$\begin{aligned} A \approx\ & ((f(a) + f(a + \Delta_x)) \cdot \Delta_x)/2 + ((f(a + \Delta_x) + f(a + 2 \cdot \Delta_x)) \cdot \Delta_x)/2 + \cdots \\ & \cdots + ((f(a + (k-1) \cdot \Delta_x) + f(a + k \cdot \Delta_x)) \cdot \Delta_x)/2 \end{aligned} \tag{3}$$

na qual k é o número de trapézios e $\Delta_x = (b - a)/k$ é a "largura" dos trapézios. A Figura 2(b) mostra um exemplo com $k = 5$.

2.3 Método Monte Carlo

O cálculo de A pelo método Monte Carlo pode ser realizado da seguinte forma. Gera-se aleatoriamente uma sequência de pontos (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n) onde $a \leq x_i \leq b$ e $0 \leq y_i \leq M$ (lembre-se que $f(x) \leq M$ para todo $x \in [a, b]$). Em seguida, calcula-se a proporção P de pontos que estão entre a curva e o eixo das abscissas, i.e., a proporção de pontos tais que $0 \leq y_i \leq f(x_i)$. Pelo método Monte Carlo temos então que:

$$A \approx P \cdot \text{área do retângulo } [a, b] \times [0, M] = P \cdot (b - a) \cdot M \tag{4}$$

É intuitivo que, quanto maior o número de pontos gerados, mais o valor obtido pelo método se aproxima do valor da área A.

3 Geração de números aleatórios

Como visto na Seção 2.3, o método Monte de Carlo requer a geração de números aleatórios (no caso, uma sequência de pontos aleatórios).

Há um certo número de métodos para gerarmos números aleatórios, dentre os quais se destaca o **método da congruência multiplicativo**. O método da congruência multiplicativo, proposto por D. W. Lehmer

em 1951, obtém o número inteiro aleatório x_i a partir do número inteiro x_{i-1}, mediante uma relação de recorrência do tipo

$$x_i = (k * x_{i-1}) \ \% \ m$$

onde k e m ($k \leq m$) são inteiros convenientemente escolhidos.

O primeiro número da sequência, x_1, será obtido a partir de um inteiro positivo qualquer, menor que m, escolhido como x_0 (também chamado de **semente**), e usando a fórmula de recorrência acima.

Extensos testes estatísticos mostraram que a sequência gerada só será satisfatória se forem escolhidos valores convenientes para k, m e x_0, e que apenas umas poucas combinações de k, m e x_0 geram sequências satisfatórias. Uma dessas combinações é obtida tomando-se $k = 75$, $m = 65537$ e com x_0 podendo ser qualquer inteiro entre 0 e 65536. A sequência dos números assim gerados se repete em ciclos de comprimento 65537, o que não apresenta inconveniente para este exercício programa, pois usaremos uma quantidade de números aleatórios que será menor do que essa. Cada inteiro gerado entre 0 e 65536 aparece exatamente uma vez em cada ciclo.

Se desejamos números aleatórios entre zero e um, podemos dividir cada número obtido por $m - 1$, assim gerando números reais $x_i/(m-1)$. Também podemos obter números entre zero e uma constante $T > 0$ (real) simplesmente fazendo a operação $(x_i/(m-1)) * T$.

Tais números gerados em computador, sendo predizíveis e reprodutíveis, não são exatamente aleatórios, sendo por isso denominados frequentemente de pseudo-aleatórios.

4 Erro relativo

Dado um número x e uma aproximação y para x, o *erro* (também chamado de *erro absoluto*) da aproximação y em relação a x é definido como sendo $|y - x|$. Quando a grandeza de x não é próxima da de 1, o erro absoluto pode não ser a maneira mais adequada de medir a qualidade da aproximação y. Por exemplo, os erros absolutos de 1.01 em relação a 1 e de 0.02 em relação a 0.01 são idênticos, mas é claro que a primeira aproximação é muito melhor que a segunda.

Face à limitada avaliação de uma aproximação conferida pelo erro absoluto, tenta-se definir o *erro relativo* de y em relação a x como sendo

$$|\frac{y-x}{x}|$$

Assim, nos dois exemplos anteriores, os erros relativos são respectivamente de 0.01 (ou 1%) e 1 (ou 100%). Contudo, esta definição ainda é incompleta quando $x = 0$. Nestes casos, a divisão por 0 não pode ser realizada e adotam-se valores arbitrários para o erro relativo. No caso de também ocorrer que $y = 0$, a aproximação certamente é perfeita e adota-se que o erro é 0. No caso de $y \neq 0$, a aproximação é certamente insatisfatória e adota-se o valor arbitrário 1 para o erro relativo. (Na prática, os erros relativos procurados são sempre menores que 1.)

Assim, definimos

$$\text{errorel}(y, x) = \begin{cases} |\frac{y-x}{x}|, & \text{se } x \neq 0 \\ 0, & \text{se } x = 0 = y \\ 1, & \text{se } x = 0 \neq y. \end{cases}$$

5 O exercício programa

Você deverá escrever um programa em C que realiza vários experimentos para calcular um valor aproximado de π em termos da área da função $f(x) = \sqrt{1 - x^2}$ no intervalo $[0, 1]$ conforme descrito na Seção 1. Em

cada um dos experimentos, a área em questão será calculada utilizando-se um dos três métodos de cálculo aproximado descritos na Seção 2. A saída do seu programa deverá ser conforme especificada mais adiante.

Os experimentos a serem realizados serão descritos em um arquivo denominado `entrada.txt`, ou seja, em vez de ler os dados de entrada do teclado, desta vez o seu EP lerá os dados de entrada de um arquivo.

5.1 Formato do arquivo de entrada

No arquivo de entrada, os experimentos serão identificados por um caractere de acordo com o método que deverá ser utilizado:

- Se o identificador for `m`, um experimento consiste em repetir, pelo número especificado de vezes, o cálculo da área pelo método Monte Carlo, obtendo-se assim várias aproximações de π, das quais interessa-nos a média. São fornecidos os seguintes dados.
  - "semente" (como nos EPs anteriores) para a geração dos números pseudo-aleatórios que serão utilizados no sorteio dos pontos no método Monte Carlo. Esta semente deverá ser utilizada apenas uma única vez no experimento.
  - número de repetições do processo de cálculo da área.
  - número de pontos a serem sorteados (deve-se usar o mesmo número de pontos em cada uma das repetições).
- Se o identificador for `r`, um experimento consiste em calcular a área usando o método dos retângulos. A largura dos retângulos deve ser calculada em função do número de retângulos especificado. O seguinte dado é fornecido:
  - número de retângulos a serem utilizados no método dos retângulos.
- Se o identificador for `t`, um experimento consiste em calcular a área usando o método dos trapezóides. A largura dos trapezóides deve ser calculada em função do número de trapézios especificado. O seguinte dado é fornecido:
  - número de trapézios a serem utilizados no método dos trapezóides.

Exemplo de um arquivo de entrada

```
m 1984 10 1000
m 2003 2  10000
r 100
t 10
```

5.2 Saída do programa

Para cada experimento especificado no arquivo de entrada, o seu programa deverá imprimir pelo menos os seguintes dados.

Se o método for Monte Carlo, deverá imprimir

- a semente utilizada para o sorteio dos pontos
- o número de pontos sorteados
- o número de repetições

- Cada um dos valores aproximados de π e os respectivos erros relativos em relação a $\pi = 3.14159$
- Uma linha com
  - a sequência de caracteres `<resposta>`
  - a sequência de caracteres `<mtc>`
  - a média dos valores aproximados e o erro relativo da média em relação a $\pi = 3.14159$

Se o método for o dos retângulos ou o dos trapezóides, deverá imprimir

- número de retângulos/trapézios considerado
- valor de Δ_x
- uma linha com
  - a sequência de caracteres `<resposta>`
  - a sequência de caracteres `<ret>` para o caso de retângulos ou `<tpz>` para o caso dos trapezóides
  - o valor aproximado obtido para π e o respectivo erro relativo em relação a $\pi = 3.14159$

5.3 Funções a serem implementadas

O seu EP deverá implementar e utilizar, obrigatoriamente, todas as seguintes funções.

- Uma função com o protótipo

  ```
  float errorel(float y, float x);
  ```

  que recebe dois reais y e x em `y` e `x`, respectivamente, e devolve o erro relativo de y em relação a x, conforme descrito na Seção 4.

- Uma função com protótipo

  ```
  float raiz(float x) ;
  ```

  que recebe um real $x \geq 0$ em `x` e devolve uma aproximação da raiz quadrada de x.

  Para calcular $\sqrt{x}$ quando $x > 0$, você deve utilizar a seguinte recorrência:

$$\begin{cases} b_0 = x \\ b_i = \dfrac{1}{2}\left(b_{i-1} + \dfrac{x}{b_{i-1}}\right) & i = 1, 2, \cdots \end{cases}$$

  Por exemplo, os termos b_1 e b_2 são calculados da seguinte forma:

$$b_1 = \frac{1}{2}\left(b_0 + \frac{x}{b_0}\right) = \frac{1}{2}\left(x + \frac{x}{x}\right) = \frac{x+1}{2}$$

$$b_2 = \frac{1}{2}\left(b_1 + \frac{x}{b_1}\right) = \frac{1}{2}\left(\frac{x+1}{2} + \frac{2x}{x+1}\right).$$

  A partir de b_2, obtemos b_3 e assim por diante.

  Este processo deve ser repetido até o primeiro n tal que $\text{errorel}(b_n, b_{n+1}) < \texttt{EPS}$, onde `EPS` é um número positivo dado que representa a precisão do cálculo da raiz. A aproximação obtida para $\sqrt{x}$ será b_{n+1}. Use como `EPS` o valor `1E-6` (10^{-6}).

  Cuidado!!! Não aplique a recorrência para $x = 0$, ou ocorrerá uma divisão por zero!!!

- Uma função com o protótipo

  ```
  float retangulo(float a, float b, int k);
  ```

  que recebe dois reais em `a` e `b` (os extremos de um intervalo) e um inteiro em `k` (o número de retângulos a serem utilizados no método dos retângulos). A função deve devolver a área aproximada da função $f(x) = \sqrt{1-x^2}$ no intervalo com extremos `a` e `b`, calculada de acordo com o método dos retângulos.

- Uma função com o protótipo

  ```
  float trapezoide(float a, float b, int k);
  ```

  que recebe dois reais em `a` e `b` (os extremos de um intervalo) e um inteiro em `k` (o número de trapezóides a serem utilizados no método dos trapezóides). A função deve devolver a área aproximada da função $f(x) = \sqrt{1-x^2}$ no intervalo com extremos `a` e `b`, calculada de acordo com o método dos trapezóides.

- Uma função com o protótipo

  ```
  void sorteia(float a, float b, float maxf, long *xi, float *x, float *y) ;
  ```

  que recebe dois reais em `a` e `b` (os extremos de um intervalo), um real em `maxf` (uma constante M como a definida na Seção 2.3) e um inteiro longo `*xi` (uma semente ou um número da sequência pseudo-aleatória gerada a partir de uma semente). A partir deste último, sorteia as coordenadas x e y de um ponto no retângulo $[a, b] \times [0, M]$, que serão devolvidas em `*x` e `*y`, respectivamente. Além disso, devolve em `*xi` o valor do último inteiro longo da sequência aleatória usado no sorteio. Veja detalhes sobre o sorteio de pontos na Seção 5.4.

- Uma função com protótipo

  ```
  float monte_carlo(float a, float b, long *xi, int n)
  ```

  que recebe dois reais em `a` e `b` (os extremos de um intervalo), um inteiro longo `*xi` (para ser usado no sorteio dos pontos), um inteiro em `n` (o número de pontos a serem sorteados). A função deve devolver a área aproximada da função $f(x) = \sqrt{1-x^2}$ no intervalo com extremos `a` e `b`, calculada de acordo com o método Monte Carlo. Deve devolver em `*xi` o valor do último inteiro longo da sequência pseudo-aleatória usado nos sorteios que, se necessário, será utilizado no próximo cálculo de área pelo método Monte Carlo.

5.4 Observações importantes

Sobre comparações de números reais

Os números reais representados em computador podem ser imprecisos uma vez que eles são representados em uma quantidade finita de *bits*. Portanto, cálculos envolvendo números reais estão sujeitos aos erros de precisão. O valor de uma expressão aritmética que envolve várias operações com números reais pode variar ligeiramente, dependendo da ordem em que as operações são realizadas no computador. Por exemplo, sabemos que

$$\frac{a}{b} * y = \frac{a * y}{b}$$

No entanto, no computador, o valor de $\frac{a}{b} * y$ pode diferir do valor de $\frac{a*y}{b}$.

Por causa da imprecisão, nem sempre dois números que são esperados serem iguais são iguais. No seu programa, para verificar se dois números reais são iguais, defina e utilize a função

```
int sao_iguais(float x, float y) {
   if(x-y < EPS && y-x < EPS)
     return 1;
   else
     return 0;
}
```

que devolve 1 se $|x - y| <$ `EPS` e 0 se $|x - y| \geq$ `EPS`. Use EPS $= 10^{-6}$.

Sobre o sorteio de pontos

Neste EP, você utilizará o método da congruência multiplicativo para gerar uma sequência de números pseudo-aleatórios, necessários para a implementação do método Monte Carlo. A Seção 3 explica como podemos obter o próximo número aleatório x_i usando x_{i-1} (o último número aleatório gerado no programa ou uma semente, caso nenhum número aleatório tenha sido gerado ainda).

As coordenadas x e y dos pontos sorteados no método Monte Carlo devem ser obtidas a partir dessa sequência. Note que, desta vez, os números a serem sorteados são números reais num intervalo $[0, 1]$.

A seguir, mostramos como pode ser gerado um ponto `(x,y)` no retângulo $[a, b] \times [0, M]$ a partir de um inteiro longo `xi_ant`. Uma vez que os números da sequência pseudo-aleatória são números entre 0 e 65536, podemos fazer

```
xi = (75 * xi_ant) %  65537;   /* sorteia novo valor de xi */
x = a + ((float)xi / 65536) * (b-a);
xi_ant = xi;
xi = (75 * xi_ant) %  65537;   /* sorteia novo valor de xi */
y = ((float)xi / 65536) * M;
```

6 Exemplo de entrada e saída

```
ENTRADA                     RESPECTIVA SAÍDA

m 1984 5 1000               Metodo Monte Carlo, semente=1984  repeticoes=5  pontos=1000
m 2003 3 10000               Estimativa 1:  pi=3.16000  erro=0.00586
m 2003 3 100000              Estimativa 2:  pi=3.15600  erro=0.00459
r 10                         Estimativa 3:  pi=3.08000  erro=0.01960
r 100                        Estimativa 4:  pi=3.21600  erro=0.02369
r 1000                       Estimativa 5:  pi=3.11600  erro=0.00815
t 10                        <resposta> <mtc>   3.14560   0.00128
t 100
t 1000                      Metodo Monte Carlo, semente=2003  repeticoes=3  pontos=10000
                             Estimativa 1:  pi=3.12520  erro=0.00522
                             Estimativa 2:  pi=3.13880  erro=0.00089
                             Estimativa 3:  pi=3.15240  erro=0.00344
                            <resposta> <mtc>   3.13880   0.00089

                            Metodo Monte Carlo, semente=2003  repeticoes=3  pontos=100000
                             Estimativa 1:  pi=3.13888  erro=0.00086
                             Estimativa 2:  pi=3.13912  erro=0.00079
                             Estimativa 3:  pi=3.14068  erro=0.00029
                            <resposta> <mtc>   3.13956   0.00065

                            Metodo dos retangulos, retangulos=10  deltax=0.10000
                            <resposta> <ret>   2.90452   0.07546

                            Metodo dos retangulos, retangulos=100  deltax=0.01000
                            <resposta> <ret>   3.12042   0.00674

                            Metodo dos retangulos, retangulos=1000  deltax=0.00100
                            <resposta> <ret>   3.13956   0.00065

                            Metodo dos trapezios, trapezios=10  deltax=0.10000
                            <resposta> <tpz>   3.10452   0.01180

                            Metodo dos trapezios, trapezios=100  deltax=0.01000
                            <resposta> <tpz>   3.14042   0.00037

                            Metodo dos trapezios, trapezios=1000  deltax=0.00100
                            <resposta> <tpz>   3.14155   0.00001
```

7 Leitura de arquivos

A seguir apresentamos uma receita que você pode usar no seu programa para leitura de arquivos em disco. O nome do arquivo de entrada deve ser, **obrigatoriamente, "entrada.txt"**.

Você deverá criar, no mesmo diretório onde se encontra o seu programa, um arquivo chamado `entrada.txt`, com o formato de conteúdo como especificado na Seção 5.1. Um arquivo com a entrada de exemplo mostrada na Seção 6 está disponível para download na página da disciplina no sistema Paca (`http://paca.ime.usp.br/course/view.php?id=631`), junto com o enunciado do EP.

O programa a seguir lê os experimentos a serem realizados de `entrada.txt` e simplesmente imprime os dados lidos na tela do monitor.

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

#define ENTRADA "entrada.txt"

int main()
{
  FILE *entrada;

  int  nrepet,     /* numero de repeticoes em um experimento Monte Carlo */
  npontos,         /* numero de pontos a serem sorteados em cada
                      repeticao de um experimento Monte Carlo */
  k;               /* numero de retangulos/trapezios nos metodos dos
                      retangulos/trapezios */

  long semente; /* semente a ser utilizada em um experimento Monte Carlo */

  char codigo;  /* tipo do experimento */

  entrada = fopen(ENTRADA,"r"); /* Abre entrada.txt para leitura ("r" de read) */

  if (entrada == NULL) { /* O arquivo existe? */
    printf("ERRO: arquivo de entrada nao encontrado\n");
    return -1; /* Indica que houve erro. */
  }

  while (!feof(entrada)) { /* Enquanto o fim de arquivo (end of file) não
                              foi encontrado... */
    fscanf(entrada,"%c", &codigo); /* Lê um caracter do arquivo e armazena em codigo */
    switch (codigo) {
      case 'm':
        fscanf(entrada, "%ld %d %d", &semente, &nrepet, &npontos);
        printf("semente = %ld  repeticoes = %d  pontos = %d\n", semente, nrepet, npontos);
        break;
      case 'r':
        fscanf(entrada, "%d", &k);
        printf("k = %d\n", k);
        break;
      case 't':
        fscanf(entrada, "%d", &k);
        printf("k = %d\n", k);
        break;
    }
  }

  fclose(entrada); /* Libera os recursos do sistema que estavam sendo usados
                      para controlar acesso ao arquivo. */

  return 0;        /* Terminação normal. */
}
```

O comando `switch` é uma forma alternativa de escrever um certo tipo de encadeamento do comando `if-else`. A forma equivalente usando `if-else` é o seguinte:

```
    if(codigo=='m') {
      fscanf(entrada, "%ld %d %d", &semente, &nrepet, &npontos);
      printf("semente = %ld  repeticoes = %d  pontos = %d\n", semente, nrepet, npontos);
```

```
}
else if(codigo=='r') {
  fscanf(entrada, "%d", &k);
  printf("k = %d\n", k);
}
else if(codigo=='t') {
  fscanf(entrada, "%d", &k);
  printf("k = %d\n", k);
}
```

8 Observações finais

- Não altere o protótipo das funções obrigatórias listadas acima.
- Se achar conveniente, você pode definir e usar outras funções além das obrigatórias listadas acima.
- O uso das funções da biblioteca `<math.h>` não é permitido neste exercício programa. Mas você pode utilizá-las (em uma fase de testes) para comparar os resultados das suas funções matemáticas. A versão final do seu programa (que você entregará no Paca) **NÃO** deverá conter funções dessa biblioteca.
- O seu programa não precisa fazer consistência dos dados de entrada.
- Faça seu programa com extensão "`.c`". Não serão aceitos programas com outros tipos de extensão tais como "`.cpp`", "`.exe`".
- Uma novidade na correção deste exercício programa é o desconto de pontos para programas que emitem "warnings" na compilação. Um "warning" é uma aviso do compilador de que alguma coisa pode não estar correta no seu programa. Esses avisos podem ser uma boa indicação de erros de lógica. Por exemplo,

  ```
  if (a=b) {
    [. . .]
  }
  ```

  resulta em um "warning", e provavelmente é um erro (atribuição ao invés de comparação). Dessa forma, **procure** eliminar todas as fontes de "warnings" de seu programa.

  Veja como ativar a detecção de "warnings" no compilador `CodeBlocks` nas seguintes páginas:

  - Codeblocks no Windows: `http://www.ime.usp.br/~jose/codeblocks-10.05/`
  - Codeblocks no Ubuntu: `http://www.ime.usp.br/~mac2166/ubuntu/code.html`
- Como já avisado anteriormente, neste exercício programa a indentação do código contará pontos. Uma ótima referência sobre como indentar seu código é a página do prof. Paulo Feofiloff: `http://www.ime.usp.br/~pf/algoritmos/aulas/layout.html`.
- Executáveis deste exercício programa podem ser encontrados na página da disciplina no sistema Paca (`http://paca.ime.usp.br/course/view.php?id=631`), junto com o enunciado do EP. Caso você tenha dúvidas sobre qual deve ser o comportamento do seu programa em alguma situação, veja como se comporta os executáveis.