MAC 115 – Introdução à Ciência da Computação - IF Diurno
IF/USP – Primeiro Semestre de 2006
Segundo Exercício-Programa

Introdução

Neste exercício vamos exercitar dois fundamentos básicos da computação matemática: (1) o cálculo aproximado de funções matemáticas (p.ex., funções trigonométricas) através de seqüências de operações matemáticas básicas (+, -, * e /) e (2) a programação de testes automatizados para verificar a correção dos programas que criamos.

Primeira Parte — Funções matemáticas

Você deverá escrever uma classe contendo métodos para calcular as funções matemáticas seno, cosseno, tangente, secante, ln (logaritmo na base e) e e^x. Para tanto, use a fórmula dada pela série de Taylor:

- $sen(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + \frac{(-1)^k x^{(2k+1)}}{(2k+1)!} + \ldots$
- $cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + \frac{(-1)^k x^{(2k)}}{(2k)!} + \ldots$
- $ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots + \frac{(-1)^{(k-1)} x^k}{k} + \ldots$
  Isso funciona bem sempre que $|x| < 1$.
- outras fórmulas podem ser encontradas na Tabela 1 de `http://www.haverford.edu/physics-astro/MathAppendices/Taylor_Series.pdf`.

Tente implementar estas funções de uma forma ao mesmo tempo clara (que seja fácil de entender) e eficiente (que seja rápida, i.e., que execute poucas operações). Sinta-se à vontade para utilizar métodos (e, talvez, classes) auxiliares se isso for ajudar na clareza do código.

O cálculo de ln tem uma complicação a mais. Note que a fórmula dada tem $1+x$ como parâmetro e não x. Assim, para implementar o $ln(1+x)$ deve-se criar um método `double ln(double umMaisX)` e no interior da função definir uma variável local `x = 1 - umMaisX` de modo que se possa calcular $ln(1+x)$.

A classe que contém as funções matemáticas deverá conter também um método através do qual definir-se-á a precisão do cálculo. Se este método não for chamado, a precisão padrão a ser adotada deverá ser 1.0e-8, ou seja, o erro máximo permitido no cálculo aproximado será de 8 casas decimais. Usuários que desejarem uma precisão maior ou menor poderão usar este método para alterá-la.

Segunda Parte — Testes Automatizados

Você deverá implementar também uma outra classe ou, se preferir, várias outras classes que serão responsáveis por testar se os cálculos das funções matemáticas estão corretos. Para cada uma das funções matemáticas, você deverá realizar 3 tipos de testes:

1. Testes com valores básicos cujo resultado é bem conhecido, por exemplo, $sen(0)$, $sen(\frac{\pi}{2})$, $sen(-\frac{\pi}{2})$, $sen(\frac{\pi}{4})$, etc. Neste caso, use valores bem diferentes de forma a cobrir o maior número possível de tipos de casos diferentes.
2. Testes usando propriedades conhecidas das funções em questão como, por exemplo, $cos^2 + sen^2 = 1$, $tan^2 + 1 = sec^2$, $exp(ln(x)) = x$, etc.

3. Compare suas implementações com as implementações disponíveis na biblioteca padrão de Java. A documentação de funções como `java.lang.Math.cos()` e `java.lang.Math.sin()` pode ser encontrada em `http://java.sun.com/j2se/1.4.2/docs/api/` (selecione o pacote `lang` e depois a classe `Math`). Não é necessário fazer este teste para a função secante.

Três observações importantes:

1. Note que, nos testes, não poderemos utilizar simplesmente o operador de comparação (`==`) para verificar se o resultado dos métodos é o esperado pois quase sempre haverá uma pequena diferença nas últimas casas decimais (por exemplo, se usarmos a precisão padrão de 8 casas decimais, é bem provável que haja diferenças a partir da oitava casa decimal). Portanto, você deve levar isso em consideração; uma possível solução é implementar um método `boolean igual (double x, double y, double erroAceitável)` que devolve `true` se o valor absoluto da diferença entre `x` e `y` for menor do que `erroAceitável`.

2. Organize seus testes de uma forma elegante. Não implemente apenas uma única classe com dezenas de métodos, cheios de código repetido e com nomes estranhos. Agrupe os métodos em classes diferentes de forma a melhorar a organização do código, use nomes bons para seus métodos, classes e variáveis, de forma que a sua intenção fique bem explícita e evite ter muito código repetido, agrupando as repetições em um único método de forma a evitar muita redundância no código.

3. Escreva seus testes de forma que todos eles possam ser executados através da chamada de um único método chamado `testaTudo()`. Os testes que dão certo devem imprimir uma mensagem bem sucinta e resumida do que foi testado. Os testes que dão errado devem imprimir uma mensagem bem destacada indicando claramente qual foi o erro detectado.

Sobre a avaliação:

- No caso de exercícios feitos em dupla, a mesma nota da correção será atribuída aos dois alunos do grupo;
- Não serão toleradas cópias! Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO (inclusive o original);
- Exercícios atrasados não serão aceitos;
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO;
- 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 qualidade do seu trabalho sob esse ponto de vista influenciará sua nota!
- As informações impressas pelo seu programa na tela devem aparecer da forma mais clara possível. Este aspecto também será levado em consideração no cálculo da sua nota;
- Uma regra básica é a seguinte: do ponto de vista do monitor responsável pela correção dos trabalhos, quanto mais convenientemente apresentado estiver o seu programa, melhor será a disposição dele para dar-lhe uma nota generosa.

Sobre a entrega:

- O prazo de entrega é o dia 23/05/2006;
- No início do arquivo, acrescente um cabeçalho bem informativo, como o seguinte:

```
/*****************************************************************/
/**   MAC 115 - Introdução à Computação                         **/
/**   IF-USP Diurno - Primeiro Semestre de 2006                 **/
/**   Professor Alfredo Goldman                                 **/
/**                                                             **/
/**   Segundo  Exercício-Programa                               **/
/**                                                             **/
/**   <nome do(a) aluno(a)>               <número USP>          **/
/**   <nome do(a) aluno(a)>               <número USP>          **/
/**                                                             **/
/**   <data de entrega>                                         **/
/*****************************************************************/
```

Não é obrigatório que o cabeçalho seja idêntico a esse, apenas que contenha pelo menos as mesmas informações.

- Guarde uma cópia do seu EP pelo menos até o fim do semestre!