MAC 115 – Introdução à Ciência da Computação para Física
IME – Segundo Semestre de 2009
Segundo Exercício-Programa
Data de entrega: até 1/novembro/2009

Introdução

Este exercício-programa terá três partes: (1) o cálculo aproximado de funções matemáticas, tais como funções trigonométricas, através de sequências de operações matemáticas básicas (+, -, * e /), (2) a programação de testes automatizados para verificar a correção dos métodos que criamos e (3) a medição do desempenho do cálculo das funções matemáticas.

Primeira Parte — Funções matemáticas

Você deverá escrever uma classe contendo métodos para calcular as funções matemáticas seno, cosseno, raiz quadrada, 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 em bons livros de cálculo ou em `http://en.wikipedia.org/wiki/Taylor_series`.

Procure 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 do método definir uma variável local `x = umMaisX - 1` 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 dois 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, $cos^2+sen^2 = 1$ e $exp(ln(x)) = x$, etc.

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.

Terceira Parte — Avaliação de desempenho

Agora, para terminar, vamos avaliar o desempenho de nossa implementação, medindo quanto tempo ela demora para calcular cada uma das funções matemáticas implementadas. O cálculo do tempo pode ser feito através do método

```
long System.nanoTime();
```

que devolve o número de nanossegundos do relógio interno do computador. Para saber quanto tempo demora a execução de um método, basta chamar `nanoTime` antes e depois do método e subtrair um tempo pelo outro.

Você deverá escrever uma classe medidora de desempenho que imprimirá, de uma forma clara, precisa e fácil de ler, um relatório sobre o desempenho das funções. Este relatório deve mostrar o tempo médio (em milissegundos) e o respectivo desvio padrão[1] de 100 execuções de cada uma das funções matemáticas.

Finalmente, para a função seno, você deve ainda escrever um método que calcula (e imprime) qual o ganho de desempenho (em termos percentuais) ao se usar o tipo `float` ao invés do tipo `double` para realizar todos os cálculos do seno.

Observações finais

Sobre a elaboração:

Este EP pode ser elaborado por equipes de um ou dois alunos, desde que sejam respeitadas as seguintes regras.

- Os alunos devem trabalhar sempre juntos cooperativamente.

[1]Para obter o desvio padrão, calcule a raiz quadrada da (média dos quadrados menos o quadrado da média). Para tanto, você poderá usar a função de biblioteca Math.srqt(x), para não "contaminar" a análise com a sua própria função implementada.

- Caso em um grupo exista um aluno com maior facilidade, este deve explicar as decisões tomadas. E o seu par deve participar e se esforçar para entender o desenvolvimento do programa (chamamos isso de *programação em pares*, que é uma excelente prática que vocês devem se esforçar para adotar).
- Mesmo a digitação do EP deve ser feita em grupo, enquanto um digita, o outro fica acompanhando o trabalho.

Sobre a avaliação:

- É sua responsabilidade manter o código do seu EP em sigilo, ou seja, apenas você e seu par devem ter acesso ao código.
- **Não serão toleradas cópias!** Exercícios copiados (com ou sem eventuais disfarces) levarão à reprovação da disciplina e o encaminhamento do caso para a Comissão de Graduação do aluno.
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota zero.
- É muito importante que seu programa seja elegante, claro e 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 avaliado será seu trabalho.

Sobre a entrega:

- O prazo de entrega é 01/11/2009;
- Entregar apenas um arquivo de nome `Matematica.zip` contendo todas as classes de seu exercício. Se quiser entregar também um arquivo texto com uma explicação sobre o seu programa (opcional), dê o nome de `LEIAME` ao seu texto.
- No início de cada arquivo, acrescente um cabeçalho bem informativo, como o seguinte:

```
/*****************************************************************/
/**   MAC 115 - Introdução à Computação                         **/
/**   IME-USP - Segundo Semestre de 2009                        **/
/**   Professor: <nome do professor>                            **/
/**                                                             **/
/**   Segundo Exercício-Programa                                **/
/**   Arquivo: MedidorDeDesempenho.java                         **/
/**                                                             **/
/**   <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.

- Para a entrega, utilize o Paca. Você pode entregar várias versões de um mesmo EP até o prazo, mas somente a última será armazenada pelo sistema.
- Não serão aceitas submissões por email ou atrasadas. Não deixe para a última hora, pois o sistema pode ficar congestionado e você poderá não conseguir enviar.
- Guarde uma cópia do seu EP pelo menos até o fim do semestre e, novamente, você é responsável por manter o sigilo de seu código-fonte.