Segundo Exercício Programa

MAC 110 BMAC Noturno

Professor: Roberto Cesar
Data de engreta: 21/05/2010

1 Objetivo

Este exercício-programa terá três partes: a primeira será o cálculo aproximado de funções matemáticas através de operações matemáticas básicas ($+$, $-$, $*$ e $/$), o segunda será a implementação de testes automatizados para verificar se os cálculos aproximados implementados estão corretos, e o terceiro será a implementação de uma interface simples.

2 Primeira parte - Função matemáticas

Você deverá escrever funções em C que calculam o valor de $sen(x)$, $cos(x)$, $\sqrt{x}$, $ln(x)$ e e^x. Para tanto, o método que será utilizado é a série de Taylor:

$$sen(x) = \sum_{k=0}^{n} \frac{(-1)^k}{(2k+1)!} x^{2k+1} \tag{1}$$

$$cos(x) = \sum_{k=0}^{n} \frac{(-1)^k}{(2k)!} x^{2k} \tag{2}$$

$$\sqrt{1+x} = \sum_{k=0}^{n} \frac{(-1)^k (2k)!}{(1-2k)(k!)^2(4^k)} x^k; |x| < 1 \tag{3}$$

$$ln(1+x) = \sum_{k=1}^{n} \frac{(-1)^{k-1}}{k} x^k; |x| < 1 \tag{4}$$

$$e^x = \sum_{k=0}^{n} \frac{x^k}{k!} \tag{5}$$

Procure implementar estas funções de forma clara (que seja fácil de entender) e eficiente (que execute em poucas operações). Sinta-se livre para implementar outras funções caso acredite ser necessário.

As funções que você deverá escrever deverão receber os parâmetros: `x` (o valor para o qual função deve ser calculada) e `precisao` (precisão do valor retornado pela função, ou seja, erro máximo permitido).

Logo, elas devem ser definidas da seguinte forma:

```
float seno (float x, float precisao);
float cosseno (float x, float precisao);
float raiz_quadrada (float umMaisX, float precisao);
float log_natural (float umMaisX, float precisao);
float exponencial (float x, float precisao);
```

2.1 Funções $\sqrt{1+x}$ e $ln(1+x)$

Note que implementar as funções que calculam a raiz quadrada e o logaritmo natural têm duas complicações adicionais. A primeira delas é que a série de taylor foi expandida para calcular o valor dessas funções de $x+1$ e não de x, e a segunda é que a série apenas irá convergir para valores de $|x| < 1$.

A solução do primeiro problema é simples: quando as funções $\sqrt{1+x}$ e $ln(1+x)$ são definidas, elas recebem o parâmetro `umMaisX`, logo, dentro da função, basta apenas definir uma variável `x = umMaisX - 1` e efetuar o cálculo da série com `x`.

Para achar a solução para o segundo problema basta fazer a pergunta: se consigo calcular a série para $|x| < 1$, então será que é possível reduzir o x para o caso conhecido se a função receber $|x| \geq 1$? (Leve em consideração a função que você está calculado e pense um pouco...)

2.2 Precisão

A precisão deve ser definida pelo usuário no início do programa. Ele deverá escolher o número de casas decimais de precisão, logo, se o usúario digitar 3, a precisão deverá ser de 0.001. A precisão máxima do programa será de 5 casas decimais, portanto valores maiores que este não devem ser aceitos.

3 Testes Automatizados

Você deverá escrever também uma função que receberá como parâmetro a `precisao` e realizará três tipos diferentes de testes para cada uma das funções implementadas:

1. Teste com valores básicos cujo os resultados são bem conhecidos. Como por exemplo $sen(0) = 0$ e outros. Neste caso, use vários valores de forma a cobrir o maior número de possíveis de casos diferentes.

2. Teste utilizando identidades conhecidas. Por exemplo: $cos^2(x)+sen^2(x) = 1, e^{ln(x)} = ln(e^x) = x$ e outras. (Quanto mais melhor!)

3. Teste utilizando as funções já disponíveis na biblioteca C `math.h`[1]. Neste caso, você pode e deve realizar testes extremos, como o cálculo de $ln(x)$ para $|x| \geq 1$.

Vale reforçar que os testes devem verificar os valores retornados pelas funções implementadas para os casos fáceis e especialemente para os casos particulares e extremos.

Três observações importantes:

1. Note que nos testes é errado utilizar simplesmente o operador de comparação `==` para verificar se o resultado é o esperado, pois quase sempre haverá diferença nas últimas casas decimais (por exemplo, se for utilizada a precisão de 3 casas decimais, é bem provável que haja diferença a partir da terceira casa decimal), e isso deve ser levado em consideração. Uma possível solução é implementar uma outra função `short int igual (double a, double b, double precisao)` que devolve `TRUE` se o valor absoluto da diferença entre `a` e `b` for menor do que `precisao` e `FALSE` caso contrário. Os valores `TRUE` e `FALSE` devem ser defidinos no programa como 1 e 0 respectivamente, utilizando `#define`.

2. Organize seus testes de uma forma elegante. Agrupe os testes de uma mesma função de forma a melhorar a organização do código.

3. A função teste deve informar ao usuário os testes que foram negativos, quando os testes terminarem e o número de testes que foram negativos.

4 Interface

Você deve criar uma interface simples para que o usuário possa utilizar o programa. Esta interface deve permitir que o usuário escolha qual função deseja calcular, mudar a precisão e sair do programa. Vale dizer também que o programa não pode ser iniciado sem que `precisao` já esteja definida.

Um exemplo da saída do programa está a seguir:

```
Digite o numero de casas decimais de precisao: 3
************************************************
1 - Seno
2 - Cosseno
3 - Raiz Quadrada
4 - Logaritmo Natural
5 - Exponencial
6 - Mudar Precisao
7 - Executar Testes
8 - Sair do Programa
```

[1] Acesse o site http://www.acm.uiuc.edu/webmonkeys/book/c_guide/ para descobrir quais funções fazem parte da biblioteca `math.h`.

```
Digite a sua opcao: 2
***********************************************
Insira um valor real (graus): 180
cos(3.1415) = -0.999843
***********************************************
1 - Seno
2 - Cosseno
3 - Raiz Quadrada
4 - Logaritmo Natural
5 - Exponencial
6 - Mudar Precisao
7 - Executar Testes
8 - Sair do Programa
Digite a sua opcao: 7
***********************************************
Falha: sen(x) -> x = 180 || Obtido: 0.001438 | Esperado: 0.000000
Os testes foram finalizados com sucesso!
Numero de testes negativos: 1
***********************************************
1 - Seno
2 - Cosseno
3 - Raiz Quadrada
4 - Logaritmo Natural
5 - Exponencial
6 - Mudar Precisao
7 - Executar Testes
8 - Sair do Programa
Digite a sua opcao: 8
***********************************************
```

5 Observações importantes

5.1 Sobre a elaboação

Este EP pode ser elabarado por equipes de dois alunos, desde que as seguintes regras sejam respeitadas:

- Os alunos devem trabalhar sempre juntos. A ideia é que deve existir uma cooperação.
- 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. (denomidamos 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.
- Recomendamos fortemente que o exercício seja desenvolviado da forma descrita nos itens acima, mas ele também pode ser feito individualmente.

5.2 Sobre a avaliação

- É sua responsabilidade manter o código do seu EP em sigilo, ou seja, apenas você e seu par podem ter acesso ao código.
- No caso de EPs feitos em dupla, a mesma nota será atribuída aos dois alunos do grupo.
- **Não serão toleradas cópias!** Exercícios copiados (com eventuais disfarces) levarão à reprovação da disciplina e o encaminhamento do caso para a Comissão de Graduação.
- Exercícios atrasados não serão aceitos.
- Exercícios com erro de sintaxe (ou seja, erros de compilação) receberão nota ZERO.
- O código deve ser escrito utilizando a linguagem C e o programa padrão para a execução do código será Dev-C++ no ambiente Windows.
- É muito importante que seu programa tenha comentários e esteja bem identado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa. Isto irá influenciar a sua nota!
- As informações impressas pelo seu programa na tela devem aparecer da forma mais clara possível. Este aspecto também será avaliado.
- Uma regra básica é a seguinte: do ponto de vista do monitor responsável pela correção dos trabalhos, quanto mais convenientemente aprensentado estiver o seu programa, melhor avaliado ele será.

5.3 Sobre a entrega

- O prazo de entrega é até o dia 21/05/2010.
- Entregar apenas um arquivo com o nome ep2.c.
- Para a entrega, utilize o Paca. Você pode entregar várias versões de um mesmo EP até o término do prazo, mas somente a última versão que permanecerá armazenada pelo sistema.
- Caso o EP tenha sido feito em dupla, ambos os alunos devem enviar o arquivo no no Paca!

- Não serão aceitas submissões por e-mail 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 final do semestre.
- No início do código, acrescente o seguinte cabeçalho:

```
/*****************************************************************/
/** IME-USP - Primeiro Semestre de 2010                         **/
/** MAC 110 - BMAC Noturno                                      **/
/** Prof: Roberto Cesar                                         **/
/**                                                             **/
/** Segundo Exercicio Programa -- Series de Taylor              **/
/**                                                             **/
/** <nome do(a) aluno(a)> <numero USP>                          **/
/** <nome do(a) aluno(a)> <numero USP>                          **/
/*****************************************************************/
```