EP-01 - Aproximação de Pi

Atualização!!!

ATENÇÃO: para auxiliar na **avaliação da eficiência** da implementação escolhida, você deve contabilizar durante a execução do programa a quantidade de **operações de ponto-flutuante** efetuadas na obtenção do valor de **pi**.

Para isso, utilize um contador que será incrementado a cada operação efetuada sobre **doubles**. Contabilize apenas as operações da última execução do método (isto é, com arredondamento para cima) e, ao final, **imprima esse valor**, conforme exemplo neste enunciado.

Caso o seu programa faça uso de operações pré-definidas (por exemplo, a função **pow**), você deve **estimar a quantidade de operações** que ela realiza. Não se esqueça que a função pow opera sobre **doubles**!

Tente otimizar ao máximo o seu programa, minimizando a quantidade de operações efetuadas!!

O restante deste enunciado e o script em anexo foram atualizados conforme esta nova especificação!

Objetivo

Neste Exercício Prático (EP) você deve obter uma **aproximação para o valor de pi (π)** através do seguinte somatório:

$$\frac{\pi}{2} = \sum_{k=0}^{\infty} \frac{2^k k!^2}{(2k+1)!}$$

Os objetivos principais são:

- Exercitar a **implementação eficiente** do somatório em termos de tempo de processamento e de precisão numérica;
- Analisar os **erros** provenientes do método utilizado e da representação numérica.

Enunciado

Faça um programa chamado **pi** que leia pela **entrada padrão (stdin)** um valor **double** representando uma **tolerância** e obtenha uma aproximação de **π** de modo que o **erro absoluto aproximado** (diferença entre a aproximação obtida na última iteração **n** e na iteração anterior **n-1**) seja no máximo esta tolerância. O seu programa deve imprimir na tela

o número de iterações **n**, o erro absoluto aproximado obtido e também o erro absoluto "exato" (para isso, compare a aproximação obtida com a constante **M_PI** fornecida na biblioteca **math.h**).

Para fins de análise dos erros provenientes da representação numérica, o método acima deverá ser executado **duas vezes**: na primeira vez, todos os cálculos de ponto flutuante devem ser efetuados com **arredondamento para baixo,** enquanto na segunda vez os arredondamentos devem ser efetuados **para cima**. O seu programa deve mostrar na tela a aproximação obtida para ambas as versões e a **diferença de ULPs** entre elas, de acordo com a fórmula apresentada em sala de aula. Por fim, o seu programa deve imprimir a quantidade de operações de ponto-flutuante (FLOPs) efetuadas na aproximação de **pi**.

Os erros e FLOPs exibidos devem ser referentes à **segunda** execução do método (execução com **arredondamento para cima**).

Para setar manualmente o método de arredondamento da máquina, utilize a função **fesetround**.

Todos os valores de ponto-flutuante devem ser impressos de **duas formas**: notação científica com 15 dígitos significativos e o hexadecimal da sua representação binária.

ATENÇÃO: nada deve ser impresso além dos valores citados, pois os resultados serão verificados via *script*. Implementações que não respeitarem essa restrição receberão nota zero. Para testar se os dados estão sendo impressos de maneira correta, execute o *script* anexo a este enunciado.

Exemplo de execução

(Dados de saída impressos com "%.15e")

Entrada:

1e-15

Saída:

50
8.881784197001253e-16 3CD0000000000000
1.199040866595170e-14 3D0B000000000000
3.141592653589783e+00 400921FB54442D01
3.141592653589806e+00 400921FB54442D33
49
2600

O que deve ser entregue

Este exercício deve ser feito **individualmente**. Deve-se entregar os códigos-fonte do programa (em linguagem C), makefile e um arquivo LEIAME contendo o nome do aluno e limitações do programa (por exemplo, casos que o programa não funciona). Todos estes

itens devem estar sob o diretório **<login>** (por exemplo, **bff21/***) e este diretório deve ser salvo e compactado em um arquivo do tipo **<login>.tgz** (por exemplo, usando o comando **tar -cvxf <login>.tgz <login>/***). O *script* anexo a este enunciado deve ser utilizado para verificar se a entrega está conforme à especificação. Para isso, execute **./script.sh <login>.tgz**