Prova de Algoritmos e Estruturas de Dados I

30 de Maio de 2011

1. (6 pontos) Escreva um programa que leia, repetidamente, um valor inteiro n e imprima o valor de $\sum_{i=1}^{n}((-i)+1)/(i*i)$, se $n > 0$, e 0 caso contrário.

   O programa deve ler os valores de n da entrada padrão e imprimir na saída padrão. Ele deve funcionar para entradas não interativas. Ou seja, a entrada pode estar em arquivo, especificado por redirecionamento da entrada padrão. O programa deve terminar com o fim dos dados de entrada (i.e. o término da entrada é indicado pelo valor retornado por `scanf`).

2. (6 pontos) Escreva um programa que leia, repetidamente, três valores positivos n, d_1, d_2 e imprima (para cada vez que três valores forem lidos) todos os valores menores que n que são divisores de d_1 ou de d_2. O programa deve terminar quando um dos três valores lidos (n ou d_1 ou d_2) for menor ou igual a zero.

   Os valores devem ser lidos do dispositivo de entrada padrão e impressos no dispositivo de saída padrão.

   Cada sequência de valores impressos (para cada três valores n, d_1, d_2) deve estar em uma linha distinta, e cada valor de cada sequência deve ser separado do seguinte por um espaço.

3. (13 pontos) Um número é *primo* se ele tem dois divisores distintos: `1` e ele próprio. Alternativamente, um número é primo se ele é maior ou igual a `2` e não tem nenhum divisor próprio (um divisor próprio é um divisor maior que `1` e menor que o próprio número).

   Escreva um programa que leia um número inteiro positivo e imprima uma mensagem dizendo se o número lido é um número primo ou não.

   O seu programa deve definir e chamar uma função para determinar se um número é primo ou não. Essa função deve, para um argumento n, testar, para valores entre 2 e o maior inteiro menor ou igual a $\sqrt{n}$, se o resto da divisão por n é igual a zero, até encontrar tal valor ou até que o divisor fique maior que $\sqrt{n}$ (em outras palavras, n não é primo se existir um divisor de n entre 2 e o maior inteiro menor ou igual a $\sqrt{n}$).

   Observação: o teste pode ser feito até o maior inteiro menor ou igual a $\sqrt{n}$ (e não até $n-1$) porque o menor divisor próprio de n, se existir, é menor ou igual a $\sqrt{n}$.

O valor de $\sqrt{n}$ deve ser obtido usando o método de aproximações sucessivas de Newton. O método de Newton especifica que, se y_i é uma aproximação para $\sqrt{x}$, então uma aproximação melhor é dada por:

$$y_{i+1} = (y_i + x/y_i)/2$$

Por exemplo, sejam $x = 2$ e $y_0 = 2$. Então:

$$\begin{array}{ll} y_1 = (2 + 2/2)/2 & = 1.5 \\ y_2 = (1.5 + 2/1.5)/2 & = 1.4167 \\ y_3 = (1.4167 + 2/1.4167)/2 & = 1.4142157\ldots \end{array}$$

A sua função deve repetir esse processo até que a diferença entre aproximações sucessivas $(y_{i+1} - y_i)$ obtidas por esse método seja menor que um valor fixo *minDif*. O seu programa deve especificar *minDif* como sendo igual a 5 centésimos.