MAC 115 – Introdução à Ciência da Computação para Física
IF – Segundo Semestre de 2009

Primeiro Exercício-Programa — Data de entrega: até 22 de outubro de 2009

Professores: Alfredo Goldman (Java) e Marco Dimas Gubitoso (linguagem C)

Vários probleminhas

Neste exercício programa queremos familiarizar o uso do computador para a criação de programas, logo, ao invés de um problema mais complexo, serão dados vários problemas simples.

1 Problemas

1.1 Números de Fibonacci múltiplos de 3

Escreva um programa que imprima os n primeiros números de fibonnaci que sejam múltiplos de 3.
Para fazer este programa em Java, escreva um método que recebe n como parâmetro e que faça a impressão. Em C, faça um programa que leia o valor n e imprima os números pedidos.

1.2 Mágica do 9

Para este exercício será necessário imprimir uma tabela com números, para em seguida fazer um "teste" com um voluntário. Os números impressos em posições não múltiplas de nove devem ser aleatórios. Nas posições múltiplas de 9, vamos colocar o número 42. Como abaixo:

```
 1-60  2-87  3-81  4-95  5-86  6-79  7-46  8-57
 9-42 10-90 11-29 12-42 13-14 14-17 15-58 16-37
17-71 18-42 19-95 20-74 21-24 22-92 23-10 24-99
25-95 26-22 27-42 28-38 29-53 30-30 31-45 32-93
33-89 34- 5 35-30 36-42 37-86 38-93 39-45 40-30
41-58 42-91 43-14 44-27 45-42 46-23 47-30 48-13
49-68 50-29 51-92 52-35 53-45 54-42 55-66 56- 7
57-42 58- 6 59-93 60-44 61-66 62-84 63-42 ...
```

Peça para o voluntário pensar em um número (de até 2 dígitos), somar os seus dígitos e subtrair do número original. Em seguida, encontrar o número correspondente na tabela. Você como mágico pode ter certeza que ele vai encontrar o 42 :).
Para gerar números aleatórios em entre 0 e 99 em Java, use:
`Math.round ((float) (Math.random() * 100.0));`
Para fazer o mesmo em C, coloque `#include <stdib.h>` no começo do programa e use `numero = 100*drand48();`, `numero` é uma variável do tipo `int`.

1.3 Crescimento populacional

Uma possível fórmula para se modelar o crescimento de uma população é a seguinte:

$$P_{n+1} = rP_n * (1 - P_n).$$

Onde P_n corresponde a porcentagem de indivíduos (segundo um valor máximo). r representa uma constante que representa, por exemplo a quantidade de recursos disponíveis.
Faça um programa que imprima a população para diversas iterações. Observem o seguintes "fatos":

- com valores pequenos de r, a população tende a desaparecer;
- com valores maiores de r, entre 1.5 e 2.0 espera-se uma estabilização;
- o que será que acontece com valores de próximos de 4.0 ?

1.4 Constante de Brun

Dado um valor de precisão, faça um programa que calcule a constante de Brun B_2, que é dada pela seguinte fórmula:

$$B_2 = \left(\frac{1}{3} + \frac{1}{5}\right) + \left(\frac{1}{5} + \frac{1}{7}\right) + \left(\frac{1}{11} + \frac{1}{13}\right) + \left(\frac{1}{17} + \frac{1}{19}\right) + \ldots,$$

onde são somados os inversos dos primos gêmeos (diferença de dois) [1]

1.5 Série

Um exemplo de série interessante é o seguinte:

$$\sum_{i=1}^{\infty} \frac{1}{n^2}.$$

Verifique que quanto mais termos forem somados, mais próximo o número se aproxima de $\frac{\pi * \pi}{6}$.

1.6 Conjectura de Collatz

Em teoria dos números existem várias conjecturas aparentemente simples, mas que são difíceis de provar. Uma dela é a conjectura de Collatz, que diz o seguinte:
Considere a função $f(x)$:

$$f(x) = \begin{cases} x/2 & \text{se } x \text{ for par} \\ 3x+1 & \text{se } x \text{ for ímpar} \end{cases}$$

Conjectura: Para qualquer inteiro n, existe um d tal que $f^d(x) = 1$.
Em C, faça um programa que, dadas uma sequência de inteiros n_i terminada por 0, calcule e imprima o valor de d para cada n_i.
Em Java, faça um método que dado um inteiro n_i calcule e imprima o valor de d.
Se, para um certo n_i, o valor 1 não tiver sido atingido após 10000 iterações, o programa deve acusar o fato e prosseguir com o valor seguinte (n_{i+1}), se houver.

1.7 Importante

Estruture bem o seu código para que as resoluções fiquem elegantes. Em Java e em C, comente os trechos necessários. Em Java, separe a complexidade do programa em vários métodos distintos. Entregue uma classe por problema.

[1] Em 1994, Thomas Nicely descobriu o bug de ponto flutuante do Pentium ao calcular a soma dos inversos dos números primos gémeos (a constante de Brun) até 10^{14} neste processador.

2 Observações importantes

2.1 Sobre a elaboração:

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

- Os alunos devem trabalhar sempre juntos, a idéia 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 (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;
- Recomendamos fortemente que o exercício seja desenvolvido da forma descrita na observação acima, mas também pode ser feito individualmente.

2.2 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.
- 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.

importante: Para EPs feitos individualmente podem ser entregues apenas 3 dos problemas apresentados. No caso de duplas, TODOS os problemas devem ser resolvidos.

2.3 Sobre a entrega:

- O prazo de entrega é o dia 22/10/2009;
- Deverão ser entregues um arquivo por problema dentro de um arquivo zip nomeado EP1-<número USP>. No caso de EPs feitos por programação em pares, basta entregar uma versão, incluindo ambos números USP no nome do arquivo (EP1-<número USP 1>e<número USP 2>).
- No início do arquivo, acrescente um cabeçalho bem informativo, como o seguinte:

```
/*****************************************************************/
/**   MAC 115 - Introdução à Computação                         **/
/**   IF-USP - Segundo Semestre de 2009                         **/
/**   <turma> - <nome do professor>                             **/
/**                                                             **/
/**   Primeiro Exercício-Programa  --  Problema                 **/
/**   Arquivo:                                                  **/
/**                                                             **/
/**   <nome do(a) aluno(a)>                <número USP>         **/
/**   <nome do(a) aluno(a)>  (se houver)   <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.
- Guarde uma cópia do seu EP pelo menos até o fim do semestre!