Computação para Informática - Prof. Adriano Joaquim de Oliveira Cruz
Quarta Aula Prática - 6 de maio de 2016

O objetivo desta aula prática exercitar comandos de teste e comandos de repetição.

Exercício 1: O problema $3n + 1$

O problema a seguir foi inspirado em um problema tirado do sítio `http://acm.uva.es/p`

Contexto

Problemas em Ciência da Computação são classificados como pertecendo a uma classe, por exemplo, NP, Não resolvível, recursivo etc. Neste problema, você irá analizar uma propriedade de um algoritmo cuja classificação não é conhecida para todas as possíveis entradas.

O Problema

Considere o algoritmo mostrado na listagem 1

Algoritmo 1: Algoritmo $3n + 1$

Entrada: *numero* : Numero a ser verificado.
Saída: Números que formam o ciclo $3n + 1$
início
  ler n
  2: imprimir n
  se $n == 1$ **então**
    PARE
  fim se
  se n *é ímpar* **então**
    $n \leftarrow 3n + 1$
  senão
    $n \leftarrow n/2$
  fim se
  goto 2
fim

Dada a entrada `22`, a seguinte seqüência de números será impressa `22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1`.

É conjecturado que o algoritmo acima irá terminar (quando o 1 é impresso) para qualquer número inteiro. Apesar da simplicidade do algoritmo, não se sabe se esta conjectura é verdadeira. Todavia, já se verificou que a conjectura é verdadeira para todos os inteiros n tais que $0 < n < 1,000,000$. Na verdade, a conjectura já foi confirmada para muitos números além destes.

A sua tarefa é escrever um programa (**SEM USAR goto**) que leia um número n e imprima os algarismos que formam o ciclo $3n + 1$.

A Entrada

A entrada consiste de um número inteiro $0 < n < 1,000,000$.

A Saída

O programa imprime, um por linha, os números que compõem o ciclo $3n + 1$.

Exemplos de entrada e saída:

Nos exemplos os números impressos pelo computador estão na mesma linha para economizar espaço.

22 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1	**34** 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Exercício 2: A quantidade de números impressos pelo algoritmo anterior, incluindo o 1, é chamada de comprimento do ciclo do número n. Modifique o problema anterior para que ele imprima também o comprimento do ciclo.

A Entrada

A entrada consiste de um número inteiro $0 < n < 1,000,000$.

A Saída

O programa imprime, um por linha, os números que compõem o ciclo $3n+1$ e ao final o comprimento do ciclo em uma linha separada.

Exemplos de entrada e saída:

Nos exemplos os números impressos pelo computador estão na mesma linha para economizar espaço.

22 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1 16	**34** 34 17 52 26 13 40 20 10 5 16 8 4 2 1 14

■

Exercício 3: A sua tarefa agora é modificar o programa para que ele descubra o maior comprimento de ciclo para números n que estão em um intervalo $i \leq n \leq j$

A Entrada

A entrada constará de um par de números inteiros i e j. Você pode assumir que todos os números lidos estarão no intervalo entre 0 e 1,000,000.

A Saída

A saída deve constar de três números: i, j e o maior ciclo calculado.

Exemplos de entrada e saída:

Exemplo 1: **1 10** 1 10 20 Exemplo 2: **100 200** 100 200 125	Exemplo 3: **201 210** 201 210 89 Exemplo 4: **900 1000** 900 1000 174

■

Exercício 4: Escreva o programa 1 e verifique os resultados. Este programa mostra como podemos gerar números inteiros randômicos em um intervalo. A função `srand` serve para inicializar o gerador de

números aleatórios. A função `rand` gera um número inteiro aleatório entre 0 e `RAND_MAX` (uma constante definida em `stdlib.h`).

Listing 1: Geração de números aleatórios

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>

int main (void) {
     int i, n;
     int max = 100;

     /* srand inicializa o gerador de numeros randomicos.
        ---- So precisa aparecer uma vez no inicio do programa.
        ---- time(NULL) retorna o numero de segundos que aconteceram desde
             01/01/1970
        ---- srand deve receber um inteiro para inicializar o gerador de numeros
             randomicos. Costuma-se usar o numero de segundos fornecido por time(NULL).
     */
     srand(time(NULL));

     for (i = 0; i < 10; i++) {
          n = rand();          /* gera um número inteiro entre 0 e RAND_MAX */
          n = n % max;              /* passa este numero para o intervalo 0 - max */
          printf("%d\n", n);
     }

     return 0;
}
```

Exercício 5: Vamos fazer um teste inicial para verificar se o gerador de números aleatórios tem vícios. Vamos testar se ele gosta mais dos números pares ou dos ímpares. Escreva um programa que gere 1.000.000 números aleatórios entre 0 e `RAND_MAX` e calcule quantos destes são pares e quantos são ímpares. Calcule a frequência (em percentagem) com que cada um destes tipos foi gerado. O que você pode dizer após este pequeno experimento. Rode o programa diversar vezes e verifique se os resultados levam a mesma conclusão.

Exercício 6: Vamos avançar um pouco mais, para isto vamos criar um dado eletrônico. Será que o dado é honesto ou tem alguma tendência? Escreva um programa que gere 6.000.000 números aleatórios entre 1 e 6 e calcule a frequência (em percentagem) com que cada um destes números foi gerado. Observe que o gerador de números randômicos gera números entre 0 e um número escolhido e isto pode ser um problema. Como fazer que ele gere números entre 1 e 6? O que você pode dizer sobre o dado?

Exercício 7: Vamos usar o gerador de números aleatórios para calcular π. Para isto comece considerando a Figura 1. Escreva um programa que gere 1.000.000 de pontos cujas coordenadas devem estar entre $(0 \leq x, y \leq 1.0)$. Conte o número de pontos (*dentro*) que estão dentro do quarto de círculo (Figura 1) de raio 1.0 e centro na origem.

(a) Imprima o resultado da seguinte conta:

$$X = \frac{4 \times dentro}{1000000}$$

.

(b) Faça o experimento agora com 9.999.999 pontos.

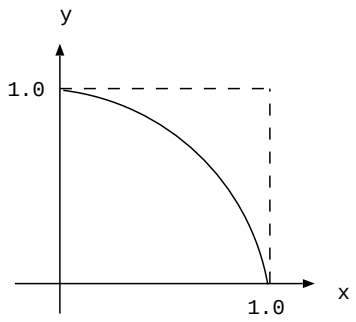

Figura 1: Figura do problema 7.