2ª lista de exercícios

Controles de repetição;
Desvios incondicionais;
Funções / Funções recursivas

Introdução à Programação C (CMT012)

Prof. Ronald Souza
IC/UFRJ

Questão 1) Uma das maneiras de se calcular a raiz quadrada de um número é pelo **método de Newton**: a partir de um valor inicial, chega-se iterativamente ao valor da raiz desejada. Considere que desejamos calcular a raiz quadrada de um valor *n*. Neste método, é necessário dar um *chute inicial* para a raiz. Chamaremos este chute inicial de x_0.

Por exemplo, vamos calcular a raiz quadrada do número n = 612. Considere que o chute inicial para a raiz de 612 seja $x_0 = 10$. O algoritmo funciona da maneira mostrada abaixo. Na última coluna **os algarismos corretos estão sublinhados**. Observe que em poucos passos obtivemos a solução correta.

$$
\begin{array}{lcccccl}
x_0 & = & \text{valor inicial} & = & 10 & & \\
x_1 & = & x_0 - \frac{x_0^2 - n}{2 * x_0} & = & 10 - \frac{10^2 - 612}{2 * 10} & = & 35.6 \\
x_2 & = & x_1 - \frac{x_1^2 - n}{2 * x_1} & = & 35.6 - \frac{35.6^2 - 612}{2 * 35.6} & = & \underline{2}6.3905556 \\
x_3 & = & x_2 - \frac{x_2^2 - n}{2 * x_2} & = & 26.3905556 - \frac{26.3905556^2 - 612}{2 * 26.3905556} & = & \underline{24.7}906355 \\
x_4 & = & \vdots & = & \vdots & = & \underline{24.738}6883 \\
x_5 & = & \vdots & = & \vdots & = & \underline{24.7386338}
\end{array}
$$

→ Escreva uma **função** que recebe um racional do tipo `double` como parâmetro de entrada e retorna sua raiz quadrada, também em formato `double`**, usando o método de Newton (descrito acima)**. O cálculo deve ser interrompido quando a diferença em valor absoluto entre dois valores consecutivos de *x* for menor que 10^{-6}, ou seja, **o cálculo deve ser interrompido quando $|x_i - x_{i-1}| < 10^{-6}$.** Dica: use a função **fabs()** da biblioteca <math.h> para cálculo de valor absoluto de um número racional.

Entrada e saída: A entrada é um número racional. **A saída são 3 números:** (i) o número lido, (ii) a raiz quadrada calculada com a função **sqrt** da biblioteca matemática <math.h> e (iii) a raiz quadrada calculada com o método que você escreveu.

Questão 2) Escreva um programa em C que imprima uma figura como a mostrada abaixo. O número de linhas da figura deve ser informado pelo usuário, mas não pode ser maior que 10. **Não** crie uma função específica para isso (ou seja, simplesmente escreva toda a lógica na **main()**).

```
       *
      ***
     *****
    *******
   *********
  ***********
```

Questão 3) Agora sim, modifique o exercício anterior como segue: crie uma **função** chamada **arvore** e que possui a seguinte assinatura:

```
void arvore(int linhas, int invertida);
```

Esta função, ao ser **chamada**, deverá imprimir uma imagem como a do exercício anterior, para o total de linhas fornecido. Aqui, porém, há duas possibilidades de impressão:

1) Caso o parâmetro **`invertida`** seja igual a 0 (isto é, *falso*), imprimir como no exercício anterior.
2) Caso contrário, imprimir a imagem "de cabeça para baixo".

Alguns exemplos de chamada da função, e a saída esperada:

Chamada	**Imagem de saída**

`arvore(4, 0);`	`   *`
	`  ***`
	`*****`
	`*******`

`arvore(5, -77);`	`*********`
	`*******`
	`*****`
	`***`
	`*`

Questão 4) Vimos em aula que a avaliação de uma **expressão lógica** em C dá-se **da esquerda para a direita,** e é encerrada **tão logo seu valor de verdade seja obtido.**

→ O que será impresso após a execução de cada uma das linhas onde há um comentário de interrogação ("// ?") à direita?
Primeiro TENTE determiná-los mentalmente e ESCREVA em cada linha o que você espera como resultado. Somente depois disso VERIFIQUE em um programa se a sua resposta é correta. Algo saiu diferente do esperado? Se sim, procure ENTENDER o porquê, e efetue as correções necessárias na sua resposta final.

```
#include<stdio.h>
int f(int a);
int main() {
    int a = 2, b = 3, c = 4;

    f(a) || f(b) && f(c);                // ?
    f(a) || (f(b) && f(c));              // ?
    (f(a) || f(b)) && f(c);              // ?
    (a > b) && f(a) || f(b) && f(c);     // ?
    (b > a) && f(a) || f(b) && f(c);     // ?
    ((b > a) && f(a) || f(b)) && f(c);   // ?

    return 0;
}
int f(int a) {
    printf("%d ", a);
    return a;
}
```

Questão 5) Faça um programa que leia um número natural **n** e dois números naturais **a** e **b** diferentes de 0 e apresente na tela em ordem crescente os **n primeiros naturais que são múltiplos de a ou de b**.

Exemplo: Para n = 6, a = 2 e b = 3 deverá ser apresentada a sequência: 0 2 3 4 6 8.

O programa deverá conter as 3 seguintes funções:

`int ehMultiplo(int r, int s);` **//verifica se r é múltiplo de s (retorna 1 se sim; 0 caso contrário)**

`void multiplos(int q, int x, int y);` **//imprime os q primeiros múltiplos de x ou de y.**

`int main();` **//lê as entradas do usuário e chama a função 'multiplos'.**

Questão 6) Matematicamente, pode-se definir o Máximo Divisor Comum (MDC) entre dois inteiros positivos **a** e **b** como

```
mdc(a, b) = a           , se b = 0;
          = mdc(b, a%b) , caso contrário
```

Podemos assim escrever uma **função recursiva** para o cálculo de MDC, como segue:

```
int mdc(int x, int y){
    if (y == 0) //caso base
        return x;
    return mdc(y, x % y); //chamada recursiva!
}
```

Ou simplesmente:

```
int mdc(int x, int y){
    return (y == 0) ? x : mdc(y, x % y);
}
```

Agora é a sua vez:

a) Implemente a versão recursiva da Questão 1 desta lista (método de Newton).

b) Vimos que a quantidade de dígitos de um inteiro 'x' pode ser obtida iterativamente pela seguinte função (onde 'x' é o parâmetro de entrada):

```
int numDigitos(int x) {
    int num = 0;
    do {
       num++;    //x possui ao menos 1 dígito.
       x /= 10;
    } while (x); //Equivale a "while(x != 0)"
    return num;
}
```

Escreva uma versão recursiva da função numDigitos.

c) Vimos que a soma dos *n* primeiros termos de uma série harmônica produz o chamado **número harmônico** H_n definido abaixo

$$H_n = \sum_{k=1}^{n} \frac{1}{k}.$$

Escreva um programa em C que calcula **recursivamente** o número *Hn*, para algum N de entrada tal que N <= 100. **Teste se os valores encontrados pelo seu programa estão corretos.** São exemplos de saídas esperadas:

N	**Saída**
1	1
2	1.5
3	~1.83333
4	~2.08333
5	~2.28333
10	~2.92897
15	~3.31823
20	~3.59774