Universidade Federal do ABC
MCTA028-15 - Programação Estruturada
2018.Q3

Lista de Exercícios 3

Professores Emílio Francesquini e Carla Negri Lintzmayer

16 de outubro de 2018

1. Calculando potências

   (a) Escreva uma função que computa a potência a^b para valores a (`double`) e b (`int`) passados por parâmetro (não use bibliotecas como `math.h`).

   (b) Use a função anterior e crie um programa que imprima todas as potências: $2^0, 2^1, \ldots, 2^{10}, 3^0, \ldots, 10^{10}$

2. Escreva uma função que computa o fatorial de um número inteiro n passado por parâmetro. Obs.: Caso $n \leq 0$ a função deve retornar 1. Use a função anterior e crie um programa que imprima os valores de $n!$ para $n = 1, \ldots, 20$.

3. Escreva uma função que recebe um número inteiro n passado por parâmetro e devolve o primeiro número da série de Fibonacci que é maior ou igual a n.

4. Escreva uma função que recebe um número inteiro n passado por parâmetro e devolve o maior número primo que é menor ou igual a n.

5. Escreva uma função que recebe um número ponto flutuante n passado por parâmetro e devolve a raiz quadrada de n. Use o método de Newton, encontrando o zero da função: $f(x) = x^2 - n$.

6. Considere o código em C abaixo:

```c
#include <stdio.h>

int soma1(int q, int c);
int soma2(int ra);
```

```c
int i = 10;
int j = 20;

int main() {
    int i, k, ra, p;
    p = 10;
    ra = 5;
    for (i = 0; i < 3; i++) {
        k = soma1(ra, p);
        ra = soma2(k);
        printf("%d, %d\n", ra, k);
    }
}

int soma1(int q, int c) {
    int soma = q + i + c;
    return soma;
}

int soma2(int ra) {
    int k = j;
    ra = ra + k;
    return ra;
}
```

(a) Determine quais são as variáveis locais e globais deste programa, identificando a que função pertence cada variável local.

(b) Mostre o que será impresso na tela do computador quando for executado este programa

7. Escreva uma função chamada **teste** que recebe um valor inteiro positivo n como parâmetro. Sua função deve retornar um valor inteiro b tal que $b^k = n$ para algum inteiro k, e b seja o menor possível.

8. Escreva uma função chamada **teste** que recebe um valor inteiro n (positivo ou negativo) como parâmetro. Sua função deve imprimir todos os valores a e b (inclusive negativos) tais que $a \times b = n$.

9. Escreva um algoritmo **iterativo** para avaliar $a \times b$ usando apenas adição, onde a e b são inteiros não negativos.

10. Escreva um algoritmo **recursivo** para avaliar $a \times b$ usando apenas adição, onde a e b são inteiros não negativos.

11. Faça uma representação da memória do computador considerando as chamadas das funções recursivas abaixo. Faça um modelo passo a passo como nos exemplos vistos em sala de aula:

    - fatorial(6)
    - fibonacci(5)

12. Determine o que a seguinte definição recursiva para uma função f calcula. A definição da função f (válida para $n \geq 0$) é dada abaixo:

$$f(n) = \begin{cases} 0 & \text{se } n = 0 \\ n + f(n-1) & \text{caso contrário} \end{cases}$$

13. Considere o número $\binom{n}{k}$, que representa o número de grupos distintos com k objetos que podem ser formados a partir de n objetos. Por exemplo, $\binom{4}{3} = 4$, pois com 4 objetos A, B, C e D é possível formar 4 diferentes grupos de 3 objetos: ABC, ABC, ACD e BCD. Sabe-se que

$$\binom{n}{k} = \begin{cases} n & \text{se } k = 1 \\ 1 & \text{se } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{se } 1 < k < n \end{cases}$$

    Faça um programa que lê dois inteiros positivos n e k e calcula $\binom{n}{k}$. **Não** use a fórmula fechada $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

14. Execute a função `ff` abaixo com os argumentos 7 e 0.

```
int ff(int n, int ind) {
    int i;
    for (i = 0; i < ind; i++)
        printf(" ");
        printf("ff(%d, %d)\n", n, ind);
    if (n == 1)
        return 1;
    if (n % 2 == 0)
        return ff(n/2, ind+1);
    return ff((n-1)/2, ind+1) + ff((n+1)/2, ind+1);
}
```

15. Quais os valores de `fun(3)` e `fun(7)`?

```
int fun(int n) {
    if (n < 4)
        return 3 * n;
    return 2 * fun(n-4) + 5;
}
```

16. A seguinte função calcula o maior divisor comum dos inteiros estritamente positivos m e n. Escreva uma função recursiva equivalente.

```
int euclides(int m, int n) {
   int r;
   do {
      r = m % n;
      m = n;
      n = r;
   } while (r != 0);
   return m;
}
```

17. Escreva uma função recursiva que imprima uma régua inglesa de ordem n no intervalo $[0..2n]$. Nessa régua, o traço no ponto médio deve ter comprimento n, os traços nos pontos médios dos subintervalos superior e inferior devem ter comprimento $n-1$, e assim por diante. A figura abaixo mostra uma régua inglesa de ordem 4.

```
.
. -
. --
. -
. ---
. -
. --
. -
. ----
. -
. --
. -
. ---
. -
. --
. -
.
```

18. A recursividade pode ser utilizada para gerar todas as possíveis permutações de um conjunto de símbolos. Por exemplo, existem seis permutações no conjunto de símbolos `A`, `B` e `C`: `ABC`, `ACB`, `BAC`, `BCA`, `CBA` e `CAB`. O conjunto de permutações de n símbolos é gerado tomando-se cada símbolo por vez e prefixando-o a todas as permutações que resultam dos $n-1$ símbolos restantes. Consequentemente, permutações num conjunto de símbolos podem ser especificadas em termos de

permutações num conjunto menor de símbolos. Formule um algoritmo recursivo para calcular todas as permutações de n símbolos.