Ficha de Trabalho N.º 8

Objetivos: Conceção e codificação de soluções para problemas, usando recursividade.

1) Calcule o valor do somatório apresentado a seguir, usando uma função recursiva (o valor de *n* deve ser pedido ao utilizador).

$$\sum_{i=1}^{n} i$$

2) Escreva uma função recursiva que permita calcular o fatorial de um valor inteiro *n* (pedido ao utilizador). Faça um programa para testar a sua função.

3) Escreva um programa em C que inclua uma função recursiva chamada potencia que permita calcular x^n (*x* real, *n* inteiro positivo).

4) Calcule o valor do somatório apresentado a seguir, usando uma função recursiva.

$$\sum_{i=1}^{n} \frac{r^i}{i!}$$

Os valores de r (real) e n (inteiro) devem ser pedidos ao utilizador.

5) Elabore funções recursivas que permitam calcular o valor de cada um dos somatórios das alíneas a seguir. Faça a simulação da função elaborada.

a) $x = \sum_{i=1}^{n} [(i)!]^i$

b) $x = \sum_{i=1}^{n} i * (i+1)!$

c) $x = \sum_{i=1}^{n} \frac{(i!)}{i}$

d) $x = \sum_{i=2}^{n} \frac{(i!)}{(i-1)!}$

e) $x = \sum_{i=1}^{n} \frac{(i!)^2}{i+1}$

6) Elabore várias funções para calcular o Fibonacci de um número:

a) uma, recursiva simples, aplicando diretamente a própria definição de número de Fibonacci;

b) outra, iterativa;

c) ainda outra, recursiva pro, que usa um vetor para armazenar os números de Fibonacci anteriores que depois vão ser utilizados para o cálculo do seguinte.

Para cada uma das funções criadas, calcule e mostre o tempo de execução para o cálculo do Fibonacci de um número elevado (e.g. 35 ou 40), permitindo efetuar uma avaliação comparativa do tempo de execução e consequentemente do desempenho do algoritmo respetivo.

Possivelmente, vários fatores externos podem alterar o tempo de execução (outros processos que estejam a correr no processador), pelo que, um melhoramento que se pode implementar é fazer várias medições para o tempo de execução de cada função e tomar valores médios e mostrar o desvio máximo também.

Obs. Para calcular o tempo de execução de cada tipo de implementação pode usar-se:

```
#include <time.h>
#include <sys/time.h>
```

Depois, colocar a definição dos dados:

```
struct timeval t1, t2;
double elapsedTime;
```

Quando se pretender iniciar a contagem, colocar o código:

```
// iniciar o cronómetro
gettimeofday(&t1, NULL);
```

Colocar depois o código de que se pretende calcular o tempo de execução, neste caso a chamada à função:

```
//
// Código....
//
```

Quando se pretender finalizar a contagem e mostrar o tempo decorrido:

```
// Calcular e imprimir o tempo decorrido em milisegundos
elapsedTime = (t2.tv_sec - t1.tv_sec) * 1000.0;
elapsedTime += (t2.tv_usec - t1.tv_usec) / 1000.0;

printf("Texto informativo do processo que foi executado Terminado em %.9f milisegundos. \n\n", elapsedTime);
```