Programação Imperativa 2022/2023 (CC1003), DCC/FCUP

Folha 3

3.1 Escreva um programa que lê um valor x em vírgula flutuante, calcula a expressão $3x^5 + 2x^4 - 5x^3 - x^2 + 7x - 6$ e imprime o resultado. Como não existe em C uma operação pré-definida para potências, deve usar multiplicações repetidas.

3.2 Modifique o programa da questão anterior para calcular a expressão usando a seguinte fórmula equivalente (método de Horner): $((((3x + 2)x - 5)x - 1)x + 7)x - 6$. Note que desta forma não usamos potências e, portanto, o programa efetua menos multiplicações.

3.3 (**Plataforma codex**) Dado um número inteiro positivo n, escreva uma definição da função `int soma_divisores(int n)` que calcula a soma dos divisores positivos de n inferiores a ele próprio. Por exemplo: para 12 a função retorna 16.

3.4 Escreva um programa que lê dois inteiros não negativos, que representam o numerador e denominador de uma fração, e imprime a fração correspondente simplificada. Exemplo:

```
Numerador: 56
Denominador: 32
A fração 56/32 é equivalente a 7/4
```

3.5 (**Plataforma codex**) Escreva um programa que lê dois valores inteiros positivo da entrada padrão e que imprime o mínimo múltiplo comum destes valores, i.e., o menor inteiro positivo que seja simultaneamente múltiplo de ambos os valores lidos da entrada padrão. O programa deve imprimir apenas o valor do mínimo múltiplo comum e uma mudança de linha.

3.6 Usando uma função de teste de primalidade, escreva um programa que imprime uma lista de primos até um limite superior especificado pelo utilizador. Escreva a lista de primos numa linha separados por espaços. Exemplo:

```
Limite superior? 50
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

3.7 (**Plataforma codex**) Dois números primos são gémeos se diferem em duas unidades. Por exemplo, $(3, 5)$, $(5, 7)$, $(11, 13)$, $(17, 19)$ são pares de números primos gémeos. Escreva um programa que lê um número inteiro positivo da entrada padrão e imprime o primeiro par de números primos gémeos maiores ou iguais ao número dado.

3.8 Um número inteiro positivo n é um quadrado perfeito se é quadrado de algum inteiro positivo. Pode-se mostrar que n é um quadrado perfeito se e só se $n = 1 + 3 + 5 + \cdots, +(2k + 1)$ para algum natural

k. Escreva uma função `int quadrado(int n)` que utilize esta propriedade para verificar se n é um quadrado perfeito. O resultado da função é 1, no caso de n ser um quadrado perfeito, e 0, caso contrário.

3.9 Modifique a implementação do algoritmo de Euclides para mostrar os passos da resolução, considerando a versão que efetua subtrações sucessivas. Deve imprimir uma linha de texto com os valores de (a, b) em cada iteração e, no fim, o valor do m.d.c. e ainda o número de iterações efetuadas, como se mostra nos exemplos. Exemplos para `mdc(12,18)` e `mdc(36,21)`:

```
mdc(12,18) = mdc(12,6) = mdc(6,6) = 6
3 iterações
mdc(36,21) = mdc(15,21) = mdc(15,6) = mdc(9,6) = mdc(3,6) = mdc(3,3) = 3
6 iterações
```

32 bits

- 32 bits + 32 bits 32 bits ⇒

3.10 Indique qual o menor dos tipos numéricos `short`, `int` ou `long` é suficiente para armazenar as seguintes quantidades; assumo os limites na arquitetura `X86` vistos nas aulas teóricas.

1. número de dias num ano short (365)
2. número de horas num ano short (8760)
3. número de segundos num dia short (86400)
4. número de segundos num mês (31 dias) int (2 678 400)
5. número de segundos desde 1 de janeiro de 1900 long

3.11 A sequência de Fibonacci é uma sequência de números inteiros, que começa por 0 e 1, na qual cada termo subsequente corresponde à soma dos dois anteriores. A sequência é definida recursivamente segundo às seguintes formulas:

$$F_0 = 0 \quad (1)$$
$$F_1 = 1 \quad (2)$$
$$F_n = F_{n-1} + F_{n-2}. \quad (3)$$

Escreva uma definição recursiva da função `int fibonacci(int n)` que calcula o valor na posição n da sequência de Fibonacci.

3.12 Escreva uma definição recursiva da função `int soma_digitos(int n)` que calcula a soma dos dígitos dum inteiro n. Por exemplo, para o número 1234, a função deve retornar o valor 10.

3.13 (**Plataforma codex**) Pretende-se calcular o logaritmo natural (i.e., de base e) usando a série de Taylor:

$$\log(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}.$$

Escreva uma função `double serie_log(double x, int n)` que calcula aproximadamente a série acima somando os termos até à potência n de x. Pode assumir que $-1 < x < 1$ e que $n \geq 1$. Tenha o cuidado de evitar o cálculo desnecessário de potências sucessivas de x.