MEFT - Programação

1º Ano - 1º Semestre de 2019/2020

Série 4 (04/11/2019)

1. Considere a função de Bernoulli[1] $x_{n+1} = 2\,x_n \pmod 1$.

a) Construa um programa em C que calcule as primeiras 100 iteradas da função começando com uma condição inicial fornecida pelo utilizador. O programa, para além de escrever os valores no ecran, deve igualmente escrever os resultados num ficheiro externo chamado "*Bernoulli.dat*" e o formato deve ser o seguinte: "(número da iterada) (6 espaços) (valor de x com 10 casas decimais)".

b) Calcule à mão as 30 primeiras iteradas e compare os resultados obtidos com os fornecidos pelo seu programa.

2. Pretende-se construir um programa que executa as seguintes operações com números complexos: **módulo**, **adição**, **diferença**, **produto** e **divisão**. A entrada de dados deve ser na forma:

$$./programa < Operacao > < Re\ z_1 > < Im\ z_1 > < Re\ z_2 > < Im\ z_2 >$$

A representação dos complexos no programa deve ser feita através de estruturas. Quando definir a estrutura crie também um nome alternativo usando 'typedef'. Deverá ser criada uma função para executar cada uma das operações pedidas. A transferência das estruturas para as funções (e das funções) deve ser feita por ponteiros. A impressão dos resultados deverá ser feita na função main e não nas funções que executam as operações. Caso o programa seja executado sem argumentos, deverá apresentar um pequeno menu explicativo.

Nota: No caso do módulo o programa deve receber apenas um complexo.

(v.s.f.f.)

---

[1] (mod 1) significa eliminar a parte inteira de um número real.

Para decompor um real (double) nas partes inteira e fraccionária pode usar-se a função:

double modf (double *value*, double **integer-part*);

em que *value* é o número que se pretende dividir, *integer-part* é o ponteiro para a parte inteira e o retorno é a parte fraccionária. Funções idênticas são igualmente definidas para *float* (**modff**) para *long double* (**modfl**).

3. Um número inteiro positivo de 'n' dígitos, diz-se **número de Armstrong** se é igual à soma de cada um dos seus dígitos elevado à potência 'n'. Exemplo:

$$371 = 3^3 + 7^3 + 1^3 \qquad [\ 27 + 343 + 1\ ]$$

Construa um programa que encontra os números de Armstrong até um certo valor máximo, dado pelo utilizador na linha de comandos, e escreve os números obtidos no ecran e num ficheiro, a seis colunas. O programa deverá executar, em funções separadas, as seguintes tarefas:

a) Obter o número de algarismos dum número inteiro positivo;
b) Testar se um dado número inteiro positivo é ou não número de Armstrong.

Nota: A impressão dos resultados deve ser feita fora das funções pedidas.

4. Escreva um programa que lê duas *string*'s como argumento e construa funções que reproduzam as seguintes funções de **C** (as funções pedidas só devem conter ciclos sobre vectores de caracteres ou as funções que entretanto construir):

a) '**strlen**' que retorna o comprimento de uma string:

```
size_t strlen (const char *s)
```

Aplique-a aos argumentos dados.

b) '**strcpy**' que copia 'str2' para 'str1' e retorna um ponteiro para 'str1':

```
char * strcpy (char *str1, const char *str2)
```

Aplique-a copiando o primeiro argumento para outra string.

c) '**strcat**' que acrescenta 'str2' a 'str1' e retorna um ponteiro para 'str1':

```
char * strcat (char *str1, const char *str2)
```

Aplique-a juntado o segundo argumento à copia que fez do primeiro.

d) Reescreva a função '**strcat**' utilizando apenas '**strlen**' e '**strcpy**'.

Nota: A(s) string(s) que usar para as exemplificações pedidas devem ser alocadas com o tamanho mínimo necessário.