Aula prática 12

1 – Escreva e teste uma biblioteca de rotinas chamada *polinomios* para operar com polinómios. A biblioteca deve conter as seguintes rotinas: leitura e escrita de um polinómio; adição de polinómios; valor de um polinómio para um certo valor da variável independente. Considere que os coeficientes dos polinómios são inteiros e que o grau máximo é 10.

```
#define MAX_GRAU_POLI 10
int ler_polinomio(int poli[]);
void escreve_polinomio(int poli[], int grau);
void adiciona_polinomios(int pol_A[], int pol_B[], int pol_R[],
int maior_grau);
float calc(int pol_A[], int grau, float x);
```

2 – Escreva e teste um programa que para uma data completa de um certo dia (guardada num registo), forneça a data de *k* dias à frente ou atrás do dia especificado. Utilize a biblioteca "datas" e a estrutura sugerida para o programa disponível no arquivo prob2_12.zip.

Conteúdo prob2_12.zip

```
datas.h: Cabeçalho (header file) com protótipo de todas as funções
necessárias para este exercício.

datas.c: Implementação de grande parte das funções da biblioteca.
Deverá implementar as funções diaSeguinte, diaAnterior e somaDias.

programa.c: Ficheiro onde reside o programa principal (função
main()), que deve ser completado para resolução do exercício.

Makefile: ficheiro que permite compilar automaticamente todos os
ficheiros no programa final.

Como compilar o programa:
Opção 1:
        1. gcc -c datas.c
        2. gcc programa.c datas.o -o programa
Opção 2 (usando o ficheiro Makefile):
        1. make
```

3 – Os números complexos são representados por pontos no plano, designando-se uma componente (normalmente segundo a coordenada horizontal) por parte real, e a outra componente por parte imaginária do complexo.

Escreva uma biblioteca de rotinas chamada *complexos* para operar com complexos. A biblioteca deve conter rotinas de leitura e escrita de complexos, soma de complexos, determinação do módulo e argumento angular de um complexo.

```
complexo le_complexo();
void escreve_complexo(complexo c);
complexo soma_complexo(complexo c1, complexo c2);
double mod_complexo(complexo c);
double arg_complexo(complexo c);
```

Escreva um programa que permita testar a biblioteca que desenvolveu.

Notas: um complexo z com componentes x (parte real) e y (parte imaginária) escreve-se z = x + iy; dados os complexos u = a + ib e v = c + id a soma é u + v = (a + c) + i(b + d); o módulo e argumento angular de z correspondem ao módulo e argumento angular do vector da origem a (x, y).

4 – Uma empresa de electrodomésticos pretende informatizar os dados relativos aos artigos de que dispõe para venda.

a) Defina um registo, *`artigo`*, adequado à representação de um artigo, contendo a seguinte informação: designação do artigo (televisor, rádio, máquina de lavar roupa, etc.), marca (por exemplo, Blabla), modelo (por exemplo, S-30), preço e quantidade disponível em armazém.

b) Escreva uma função que leia um elemento do tipo artigo, passado por parâmetro. Os dados do artigo são lidos a partir da informação introduzida pelo utilizador.

```
void le_artigo(artigo *item)
```

c) Considere que a informação sobre os artigos da empresa, em número n < 10000, foi lida para um vetor (`armazem`) de elementos do tipo artigo. Escreva uma função que retorne o número total de artigos de uma certa marca e modelo a especificar.

```
int total_artigos(artigo armazem[], int n, char marca[], char
modelo[]);
```

d) Implemente um procedimento que liste todos os produtos cuja existência em *stock* é inferior a 10 unidades

```
void alerta_artigos(artigo armazem[], int n);
```

e) Implemente um procedimento que ordene todos os produtos em *stock* pela sua quantidade.

```
void ordena_artigos(artigo armazem[], int n);
```

Outros exercícios recomendados do livro *C How To Program*: 10.5, 10.6

Pode aceder à versão electrónica através da rede da FEUP (link no Moodle)