Aula prática #10 – Registos e Ficheiros

Problema 1

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) Defina um registo que permita guardar um numero complexo.

b) Implemente uma função de leitura de números complexos.

```
1    complexo leComplexo()
```

c) Implemente uma função de escrita de números complexos.
Um complexo z com componentes x (parte real) e y (parte imaginária) escreve-se $z = x + iy$

```
1    void escreveComplexo(complexo c)
```

d) Implemente uma função que adicione 2 números complexos.
Dados os complexos $u = a + ib$ e $v = c + id$ a soma é $u + v = (a + c) + i(b + d)$

```
1    complexo somaComplexo(complexo c1, complexo c2)
```

e) Implemente uma função que calcule o modulo de um número complexo.
O módulo angular de z corresponde ao módulo do vetor da origem a (x, y)

```
1    double modComplexo(complexo c)
```

f) Implemente uma função que calcule o argumento angular de um numero complexo.
O argumento angular de z corresponde ao argumento angular do vector da origem a (x, y)

```
1    double argComplexo(complexo c)
```

g) Escreva um programa que permita testar a biblioteca que desenvolveu.

Problema 2

Suponha que as linhas de um ficheiro de texto contêm a designação de uma tarefa realizada durante um mês, seguida de quantos trabalhadores estiveram envolvidos nessa tarefa em cada um dos dias úteis do mês. Um valor negativo significa que a tarefa foi terminada.

Exemplo: Polimento 0 1 2 2 1 3 2 -1

Significa que a tarefa Polimento envolveu 0 trabalhadores no 1º dia útil do mês, 1 no 2ª dia útil, etc., e terminou no 7º dia útil. Escreva e teste um programa que leia o ficheiro e calcule:

- Qual o dia em que se trabalhou no maior numero de tarefas.
- Qual o dia em que se terminaram mais tarefas.
- Qual a tarefa que ocupou mais trabalhadores.

Exemplo

```
Input                Output
a 3 -1               Dia com mais tarefas: 1
b 0 0 2 3 -1         Dia com mais tarefas acabadas: 4
c 0 0 0 3 -1         Tarefa com mais trabalhadores: b
d 1 0 0 3 -1
e 2 -1
```

Problema 3

Uma empresa de produtos eletrónicos pretende informatizar os dados relativos aos artigos de que dispõe para venda.

a) Defina um registo designado `artigo`, adequado à representação de um artigo, contendo a seguinte informação:

- designação do artigo, por exemplo, "televisor"
- marca, por exemplo, "SuperTV"
- modelo, por exemplo, "S-30"
- preço
- quantidade disponível em armazém

b) Escreva uma função que leia os dados de vários artigos a partir de um ficheiro com o nome indicado no parâmetro `nomeFicheiro` para o vetor `armazem` do tipo artigo ($tamanho < 100$). A função retorna o número de artigos lidos.

```
1    int leArtigos(artigo armazem[], char *nomeFicheiro)
```

c) Escreva uma função que retorne o número total de artigos de uma certa marca e modelo a especificar.

```
1    int totalArtigos(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

```
1    void alertaArtigos(artigo armazem[], int n)
```

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

```
1    void ordenaArtigos(artigo armazem[], int n)
```

f) Implemente um programa que permita testar as funções que desenvolveu nas alíneas anteriores. Utilize o ficheiro `artigos.txt` disponibilizado no Moodle.

Problema 4

Escreva e teste um programa que para uma data completa de um certo dia (guardada num registo), indique 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 datas.zip.
Conteúdo do ficheiro `datas.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:

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

Exemplo

```
data (dia mes ano)? 3 6 2016
numero de dias? 2
5 6 2016
data (dia mes ano)? 15 12 2016
numero de dias? -2
13 12 2016
data (dia mes ano)? 25 12 2016
numero de dias? 10
4 1 2017
```

Problema 5

Pretende-se implementar um jogo de cartas com as seguintes regras: os jogadores vão retirando cartas do baralho e comparam o valor da carta. Aquele que vencer essa ronda ganha 1 ponto e quem perdeu a ronda, perde um ponto. Em caso de empate, as pontuações não se alteram. Os jogadores iniciam com 20 pontos cada um. O jogo termina quando se atingem as 100 rondas ou quando um dos jogadores fica com zero pontos.

A lógica do jogo está já parcialmente implementada no ficheiro `jogo.c`. Complete as seguintes funções que ainda não foram implementadas:

```
1    void criaBaralho(carta *baralho)
```

Cria um conjunto de cartas que respeite as estruturas de dados definidas. O baralho deve ter uma carta de tipo para todos os naipes.

```
1    void baralha(carta *baralho)
```

Baralha as posições das cartas dentro do baralho de forma aleatória.

```
1    int comparaCarta(carta c1, carta c2)
```

Verifica se uma carta é maior do que outra. O retorno da função deve ser 1, 0 ou -1 conforme a carta c1 seja maior, igual ou menor do que a carta c2, respetivamente.

Problema 6

Pretende-se implementar o jogo do Enforcado (Hangman em inglês). Para tal deverá usar o ficheiro parcialmente implementado "hangman.c"e completar as funções em falta. O ficheiro de teste usado é "palavras.txt". As funções a implementar são:

```
1  void carregaPalavras(char *ficheiro, char palavras[][MAX], int *tamanho);
```

Deve ler as palavras a partir do ficheiro de texto e criar e devolver um vetor de strings. O parâmetro tamanho deverá ser usado para devolver o tamanho desse vetor.

```
1  void revelaLetras(char letra, char *palavra, char *obfuscada);
```

Deve retornar uma nova palavra em que mostra o(s) caracter(es) da palavra obfuscada indicados pelo parâmetro letra e que estejam presentes na palavra fornecida também como parâmetro.

```
1  int joga(char *palavra, char *palavraObfuscada);
```

Implemente a lógica de jogo.