Programação em C - Exercícios sobre processamento de ficheiros

---

Ligações úteis:

- Slides de *Visão geral da linguagem C* disponibilizados no moodle;
- C - File I/O (Tutorialspoint)

Exercícios:

1. Escreva um programa **conta_caracteres** que conte o número de carateres de um ficheiro, escrevendo esse número noutro ficheiro. O nome do ficheiro a analisar e do ficheiro que conterá o resultado deverá ser passado na linha de comandos da shell.

   Exemplo:

   ```
   $ ./conta_caracteres fich1.txt fich2.txt
   $ cat fich2.txt
   O ficheiro tem ??? caracteres.
   ```

2. Escreva um programa com o nome **palavras**, que recebe da shell um parâmetro que é interpretado como um nome de um ficheiro. O programa deverá imprimir todas a palavras existentes nesse ficheiro, uma por linha. Entende-se por "palavra" uma sequência de duas ou mais letras, maiúsculas ou minúsculas.

   Exemplo de possível execução do programa:

   ```
   $ ./palavras pp.c
   define
   main
   int
   if
   ...
   ```

   O programa deverá ter a seguinte função principal:

   ```
   int main(int argc, char* argv[]){
       FILE *fich;
       char *pal;

       if (argc != 2){
   ```

```
        mensagem("Uso: ./palavras ficheiro");
    }
    fich = abre_fich(argv[1]);
    while((pal = palavra(fich)) != NULL){
        printf(" %s\n", pal);
        free(pal);
    }
}
```

e incluir as funções seguintes:

- `FILE* abre_fich(char* s)` - Tenta abrir o ficheiro s para leitura. Retorna NULL se não conseguir;
- `void mensagem(char* m)` - Escreve a mensagem m e abandona o programa (com exit);
- `char* palavra(FILE *f)` - Retorna a próxima palavra lida do ficheiro ou NULL caso não exista mais nenhuma.

3. Escreva o programa **mat2file** que pede ao utilizador que introduza os elementos (inteiros) de uma matriz, e que os armazena num ficheiro (utilizando como separador o caracter espaço). As dimensões da matriz e o nome do ficheiro são passados através da linha de comando (e.g., `mat2file 2 3 matriz.txt`), sendo os elementos pedidos, um a um, ao utilizador. No ficheiro, os dois primeiros elementos são o número de linhas e de colunas, sendo os restantes os elementos da matriz. Por exemplo, a matriz:

$$\begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \end{bmatrix}$$

deve ser armazenada como:

```
2 3
10 20 30 40 50 60
```

4. Pretende-se que implente um programa *my_ppm_rot* que faça uma rotação, segundo um eixo vertical, de uma imagem em formato de armazenamento *Plain Portable Pixel Map* (PPM). O programa deverá receber dois argumentos correspondentes ao nome ficheiro de entrada (imagem a transformar) e ao nome do ficheiro de saída (imagem transformada). A descrição do formato Plain PPM está disponível em http://netpbm.sourceforge.net/doc/ppm.html, e em https://en.wikipedia.org/wiki/Netpbm.

Considere o ficheiro *img.ppm*

```
P3
3 2
255
# The part above is the header
# "P3" means this is a RGB color image in ASCII
# "3 2" is the width and height of the image in pixels
# "255" is the maximum value for each color
# The part below is image data: RGB triplets
255   0   0  # red
  0 255   0  # green
  0   0 255  # blue
255 255   0  # yellow
255 255 255  # white
  0   0   0  # black
```

que corresponde à seguinte imagem:

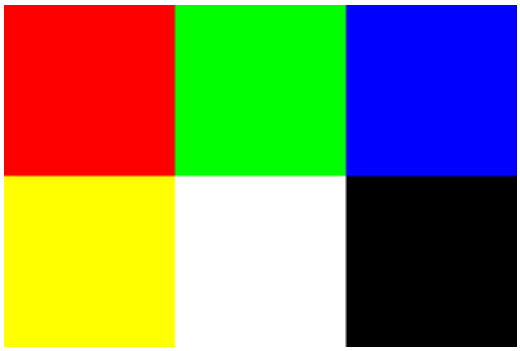

Figure 1: imagem original (ampliada)

A execução do programa:

```
$ ./my_ppm_rot img.ppm img_rot.ppm
```

deverá criar o ficheiro *img_rot.pmm*:

```
P3
3 2
255
  0   0 255
  0 255   0
255   0   0
  0   0   0
255 255 255
255 255   0
```

correspondente à imagem:

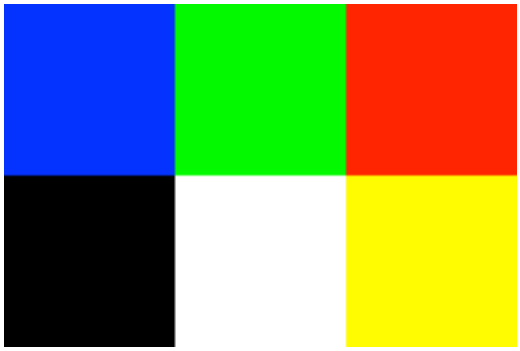

Figure 2: imagem transformada (ampliada)

Nota: Poderá converter uma imagem para o formato ppm utilizando, por exemplo, o utilitário `convert`. Exemplo:

```
convert img.jpeg  -compress none  imp.pgm
```