Exercícios de programação em C - II

Ligações úteis:

- Slides de *Visão geral da linguagem C* disponibilizados no moodle;
- Apontamentos de Programação Imperativa (CC1003), em particular:
  - Apontadores;
  - Programação com apontadores;
- The C Book;
- Everything you need to know about pointers in C;

---

1. Escreva um programa que calcule e imprima a soma de uma lista de números passados como parâmetros na linha de comando.

   Exemplo:

   ```
   $ ./soma 4 5 2 8
   19
   ```

2. Escreva um programa **ccaract** que imprima o número de caracteres de cada parâmetro passado na linha de comandos. Escreva duas versões, uma que utilize a função `strlen()`, e outra que não a utilize.

   Exemplo:

   ```
   $ ./ccaract ola 4 23 fim
   3
   1
   2
   3
   ```

3. 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.
   ```

4. *Espaços!*

- Implemente uma função com o protótipo `void esp(char *s);` que troca os caracteres de **s** que não são letras nem dígitos por espaços. Por exemplo, `"a, Ah Ah!"` fica transformada em `"a  Ah Ah "`.
- Implemente uma função com o protótipo `char* esp1(char *s);` que retorna uma "string" resultante da troca de todos caracteres de **s** que não são letras nem dígitos por espaços (**s** não é alterada). O espaço para o resultado deve ser obtido pela função `malloc()`.
- Escreva um pequeno programa para testar as funções acima.

5. 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.

6. Escreva um programa que leia uma lista de inteiros (o primeiro inteiro indica o tamanho da lista), e que escreva os inteiros da lista que aparecem repetidos pelo menos uma vez na lista dada.

   Exemplo de possível execução do programa:

```
$ ./repetidos
6
1
2
1
10
2
1

1
2
```

As duas últimas linhas constituem o resultado. Para resolver o exercício, deverá guardar num vector `a[]`, **alocado dinamicamente**, o conjunto dos inteiros que já foram lidos pelo menos uma vez. Note-se que é necessário distinguir o caso de o inteiro ter ocorrido uma vez, do de ter ocorrido duas ou mais vezes. Assim, sugere-se que cada elemento de `a[]` seja uma estrutura como indicado:

```
struct elemento{
    int valor; // um inteiro lido
    int ocorrencias; // numero de vezes que esse inteiro foi lido
};
typedef struct elemento ELEMENTOS;

ELEMENTOS *a = NULL;
```