Aula prática 11 – Strings e Funções Recursivas

1 - Escreva um programa capaz de ler uma frase (*string*) introduzida pelo utilizador e imprima essa string invertida. Implemente o seguinte procedimento:

```
void inverte(char *strOriginal, char *strInvertida);
```

No primeiro argumento do procedimento é passada a *string* original e no segundo argumento é devolvida a *string* invertida.

Exemplo

```
Escreva uma frase: Programacao 1 e' divertido!
A frase invertida e' "!oditrevid 'e 1 oacamargorP".
```

2 - Escreva um programa que determina se uma palavra introduzida pelo utilizador é capicua. O programa deve permitir ao utilizador testar o número de *strings* que pretender e apenas termina quando o utilizador introduz "." (ponto final).

Exemplo

```
Palavra? Programa
Resultado: Programa nao e' capicua.
Palavra? ana
Resultado: ana e' capicua.
Palavra? .
```

3 - Sem usar as estruturas de controlo de fluxo "**for**" e "**while**" e usando apenas a operação de **soma (+)**, implemente um algoritmo recursivo para calcular a multiplicação de dois números inteiros. Para tal, implemente uma função com o seguinte cabeçalho:

```
int multiplicacao(int primeiro_numero, int segundo_numero);
```

4 - Sem usar as estruturas de controlo de fluxo "**for**" e "**while**", implemente um programa que que recorra a uma função recursiva para calcular o máximo divisor comum (mdc) entre dois números inteiros, sabendo que mdc(x, y) = x se y = 0, senão mdc(x, y) = mdc(y, mod(x, y)). Nota: mod(x, y) é o resto da divisão de x por y. Para tal, implemente uma função com o seguinte cabeçalho:

```
int mdc(int x, int y);
```

5 - Escreva um programa que pede ao utilizador para escrever uma frase e apresenta no ecrã quantas palavras constituem a frase, a palavra de maior comprimento e o comprimento médio das palavras.

Exemplo

```
Frase? O jornal de hoje tem na capa uma fotografia interessante
Numero de palavras: 10
Palavra maior: interessante
Comprimento medio: 4.7
```

6 - A cifra de César[1] é uma das técnicas mais simples e conhecidas de Criptografia[2]. Pretende-se a implementação de um programa capaz de codificar e descodificar *strings* usando este algoritmo.

[1] Wikipédia, Cifra de César
[2] Wikipédia, Criptografia

6.1 - Assumindo o texto em minúsculas, escreva e teste uma função

```
char converte(char c, int shift);
```

que devolve o caráter correspondente à aplicação do *shift* indicado. Por exemplo, o caráter 'm' com um *shift* de +3 deverá ser convertido em 'p', ou em 'k' com um *shift* de -2. A função deverá ainda garantir que o resultado pertence sempre à gama [a-z].

Exemplo

```
caráter e shift? f 3
resultado: i
caráter e shift? a -2
resultado: y
```

6.2 - Usando a função anterior, implemente e teste o procedimento

```
void desloca(char texto[], char* cifra, int shift);
```

que aplica o *shift* indicado ao texto fornecido e gera a cifra pretendida. O algoritmo só deve ser aplicado a letras, mantendo os restantes carateres. Dica: use a função isalpha() (requer a inclusão do ficheiro *ctype.h*) para testar essa condição.

Exemplo

```
texto? criptografia
shift? 3
cifra: fulswrjudild

texto? isto funciona mesmo?
shift? -7
cifra: blmh yngvbhgt fxlfh?
```

6.3 - Alguém que não conheça a frase original, mas que mesmo assim pretenda desencriptá-la, tem vários mecanismos para o fazer. Um deles chama-se "ataque por força bruta" e consiste em tentar todas as combinações possíveis de desencriptação até ("por força bruta") se obter a frase original. Implemente o procedimento

```
void ataque(char cifra[]);
```

que revela o texto original através de um "ataque por força bruta". Este procedimento descodifica a cifra obtida aplicando-lhe cada um dos *shifts* possíveis e imprimindo o resultado no ecrã. O utilizador poderá então analisar os 25 resultados e encontrar a *string* inicial.

Exemplo

```
cifra? fulswrjudild
com shift + 1: gvmtxskvejme
com shift + 2: hwnuytlwfknf
(…)
com shift +23: criptografia
com shift +24: dsjquphsbgjb
com shift +25: etkrvqitchkc
```

6.4 - Descubra o texto original de ambas as cifras.

```
cifra? emu pm rdqzfq sgqpqe!
cifra? f jvxleuf dzezkvjkv mrz jvi trear...
```

7 - Escreva e teste uma função

```
int conta(char frase[],char palavra[])
```

que recebe uma frase e uma palavra e retorna quantas vezes essa palavra aparece na frase. Utilize esta função num programa permite o utilizador testar várias frases e palavras até que a frase seja "." (ponto final).

Nota: Palavras escritas com maiúsculas ou minúsculas devem ser consideradas palavras diferentes.

Exemplo

```
Frase? A Ana foi ao cinema e Ao parque
Palavra? ao
Resultado: A palavra ao apareceu 1 vez na frase.
```

8 - Sem usar as estruturas de controlo de fluxo "**for**" e "**while**", implemente um algoritmo recursivo para calcular e imprimir a sequência de Fibonacci até um valor máximo ou até um número máximo de valores.

A sequência de Fibonacci é definida pela seguinte relação recursiva:

$$F_n = F_{n-1} + F_{n-2}, \text{ com } F_0=0 \text{ e } F_1=1$$

O resultado do programa deverá ser igual ao seguinte:

```
Pretende usar numero maximo de valores(1) ou valor máximo(2)? 1
Introduza um numero maximo de valores: 13
Sequencia: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144.

Pretende usar numero maximo de valores(1) ou valor máximo(2)? 2
Introduza o valor máximo: 20
Sequencia: 0, 1, 1, 2, 3, 5, 8, 13.
```

9 - Implemente uma função com base em variáveis estáticas que retorne o próximo número da sequência de Fibonacci, começando em F_2. Isto é, a primeira chamada à função deverá retornar 1, a segunda deverá retornar 2, a terceira deverá retornar 3, a quarta deverá retornar 5, etc. Implemente a função com o seguinte cabeçalho:

```
int proximoFib();
```

Outros exercícios recomendados do livro *C How To Program*: 8.7, 8.16, 8.17 e 8.21

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