

Aula prática #10 – Strings e Funções Recursivas

Problema 1

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

```
1 void inverta(char *strOriginal, char *strInvertida);
```

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

Exemplo

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

Problema 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 arbitrário e apenas termina com "Ctrl+D". Para tal, implemente uma função com o seguinte cabeçalho:

```
1 int capicua(char * str);
```

A função deve retornar 1 se a string str é capicua e 0 se não for capicua.

Exemplo

```
1 Palavra? Programa  
2 Resultado: Programa nao e capicua.  
3 Palavra? ana  
4 Resultado: ana e capicua.  
5 Palavra? .
```

Problema 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:

```
1  int multiplicacao(int primeiroNumero, int segundoNumero);
```

Problema 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 $\text{mdc}(x, y) = x$ se $y = 0$, senão $\text{mdc}(x, y) = \text{mdc}(y, \text{mod}(x, y))$.

Nota: $\text{mod}(x, y)$ é o resto da divisão de x por y .

Para tal, implemente uma função com o seguinte cabeçalho:

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

Problema 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

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

Problema 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 decodificar strings usando este algoritmo.

[1] Wikipédia, Cifra de César

[2] Wikipédia, Criptografia

Assumindo o texto em minúsculas, escreva e teste uma função:

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

que devolve o carácter correspondente à aplicação do shift indicado. Por exemplo, o carácter '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

```
1 carater e shift? f 3
2 resultado: i
3 carater e shift? a -2
4 resultado: y
```

Problema 7

Usando a função anterior, implemente e teste o procedimento:

```
1 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 caracteres. Dica: use a função isalpha() (requer a inclusão do ficheiro ctype.h) para testar essa condição.

Exemplo

```
1 texto? criptografia
2 shift? 3
3 cifra: fulswrjudild
4
5 texto? isto funciona mesmo?
6 shift? -7
7 cifra: blmh yngvbhgt fxlfh?
```

Problema 8

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

```
1 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

```
1 cifra? fulswrjudild
2 com shift + 1: gvmtxskvejme
3 com shift + 2: hwnuytlwfknf
4 ...
5 com shift +23: criptografia
6 com shift +24: dsjquphsbgjb
7 com shift +25: etkrvqitchkc
```

Problema 9

Escreva e teste uma função

```
1 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

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

Problema 10

A sequência de Fibonacci é definida pela seguinte relação recursiva:

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

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.

O resultado do programa deverá ser igual ao seguinte:

Exemplo

```
1 Pretende usar numero maximo de valores(1) ou valor maximo(2)? 1
2 Introduza um numero maximo de valores: 13
3 Sequencia: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144.
4
5 Pretende usar numero maximo de valores(1) ou valor maximo(2)? 2
6 Introduza o valor maximo: 20
7 Sequencia: 0, 1, 1, 2, 3, 5, 8, 13.
```

Problema 11

Implemente uma versão iterativa (usando “for” / “while”) e compare o tempo de execução de ambos para valores de n superiores a 100000.

Problema 12

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:

```
1 int proximoFib();
```