FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Engenharia Eletrotécnica

PROGRAMAÇÃO DE MICROPROCESSADORES

2021 / 2022

Licenciatura em Engenharia Eletrotécnica
e Computadores

1º ano
1º semestre

Trabalho nº 6
Vetores e *Strings*

http://tele1.dee.fct.unl.pt/pm

Paulo da Fonseca Pinto
Luis Bernardo

1 Introdução

Nos capítulos 6 e 7 do livro "Linguagem C" de Luís Damas, recomendado para a disciplina de Programação de Microprocessadores, são apresentados respectivamente os vetores e as ***strings***. Esta aula visa consolidar estas matérias através de um conjunto de exercícios. Faça todos os exercícios pedidos em ficheiros separados e **GUARDE O CÓDIGO desenvolvido na memória USB**. Durante a aula o docente pode pedir-lhe para mostrar o código desenvolvido.

2 Coisas práticas sobre *strings*

Os vetores e as ***strings*** podem parecer muito semelhantes e por isso esta aula reúne os dois. Veja se consegue perceber as diferenças…

O que é uma ***string***?

As ***strings*** **NÃO EXISTEM**! São apenas um **conceito**.

Uma ***string*** é um conjunto de caracteres **em que se combinou** que acaba quando se encontrar o carácter '\0'.

Baseado nisto, normalmente as funções que tratam de ***strings*** nem têm necessidade de saber o comprimento da ***string*** – "vão andando" até encontrar o carácter '\0'.

Que memória é que se vai usar para guardar uma ***string***? É que assumindo esta combinação (de acabar no \0) não se sabe o comprimento exacto…

A verdadeira resposta é que pode usar-se memória estática ou memória dinâmica. Para não complicar as coisas vamos usar em Programação de Microprocessadores o seguinte:

As ***strings*** vão estar sempre guardadas em vetores (de caracteres) que vão ter um tamanho tão grande que a ***string*** vai sempre lá caber e assim não nos traz problemas.

Por exemplo, podíamos ter um vetor de 1.000 caracteres e o carácter '\0' irá aparecer sempre antes do fim, não causando problemas. Claro que isto não é muito eficiente… E tem também o problema de a ***string*** poder ser maior do que 1.000…

A alternativa de se usar memória dinâmica é abordada no capítulo 12 do livro.

Esta aula começa com exercícios com vetores de caracteres e depois com exercícios de ***strings***.

3 Uma linha com 20 caracteres

Pretende-se fazer um programa que manipule um **vetor de caracteres**, e não uma **string**, chamado ***linha.c***. O vetor de caracteres tem 20 posições. Existem as funções seguintes:

- ***menu*** que contém o menu e devolve a opção escolhida pelo utilizador. O utilizador pode escolher uma das seguintes opções (cada uma delas uma função):
- ***limpar_vetor*** (coloca em cada elemento do vetor o carácter espaço em branco);
- ***ler_vetor*** (pede ao utilizador que escreva um texto que coloca no vetor. O texto é `lido` com `getchar` e todos os caracteres que o utilizador escrever acima do tamanho do vetor são ignorados. Se o utilizador não escreveu os suficientes ficam os que estavam antes no vetor);
- ***imprimir_vetor*** (imprime o vetor no ecrã usando `putchar`. Imprime todos os caracteres do vetor);
- ***substitui_vetor*** (substitui todas as ocorrências de um carácter 'a' pelo carácter 'b', em que 'a' e 'b' são fornecidos na lista de argumentos. Retorna o número de substituições feito que deve ser mostrado no ecrã);
- ***conta_caracter*** (conta quantas ocorrências do carácter 'a' existem no vetor, retornando esse valor, que é escrito no ecrã).
- ***main*** que é simplesmente a chamada a procedimentos;

A seguir é mostrado um exemplo do programa

/********** PROGRAMA LINHA **********/

1 – limpar o vetor
2 – ler e introduzir valores
3 – mostrar o vetor
4 – substituir carácter
5 – contar carácter
s - sair
O que pretende fazer? 2

Texto (acaba com ENTER): aqui esta um texto exemplo para ver

1 – limpar o vetor
2 – ler e introduzir valores
3 – mostrar o vetor
4 – substituir carácter
5 – contar carácter
s - sair
O que pretende fazer? 3

Texto: aqui esta um texto e

1 – limpar o vetor
2 – ler e introduzir valores
3 – mostrar o vetor
4 – substituir carácter
5 – contar carácter
s - sair
O que pretende fazer? 4

Qual o carácter a substituir: u
Por que carácter? h

Fizeram-se 2 substituições.
Texto: aqhi esta hm texto e

1 – limpar o vetor
2 – ler e introduzir valores
3 – mostrar o vetor
4 – substituir carácter
5 – contar carácter
s - sair
O que pretende fazer? 5

Que carácter quer contar: h

Existem 2 ocorrências do carácter h.

2 – ler e introduzir valores
3 – mostrar o vetor
4 – substituir carácter
5 – contar carácter
s - sair
O que pretende fazer? s
Adeus

4 Um texto com 20 linhas de 20 caracteres

O objectivo do próximo programa é generalizar o problema anterior. Considere um vetor a duas dimensões, uma matriz, de caracteres com a dimensão 20 x 20. Chame ao ficheiro ***padrao.c***.

Considere que se numeram

- as linhas da primeira para a última (de cima para baixo) desde o valor zero a 19
- as colunas da esquerda para a direita do mesmo modo
- as diagonais do modo como mostra a figura seguinte (a diagonal principal tem a identificação de zero; as diagonais para cima são positivas; as diagonais para baixo são negativas)

O que se pretende é que faça um programa que trabalhe sobre a matriz e que faça o seguinte:

escrever_coluna – esta função escreve o carácter 'a' (lido ao utilizador) na coluna i de todas as linhas (também parâmetro de entrada);

escrever_linha – esta função escreve o carácter 'a' (lido ao utilizador) na linha i (também parâmetro de entrada);

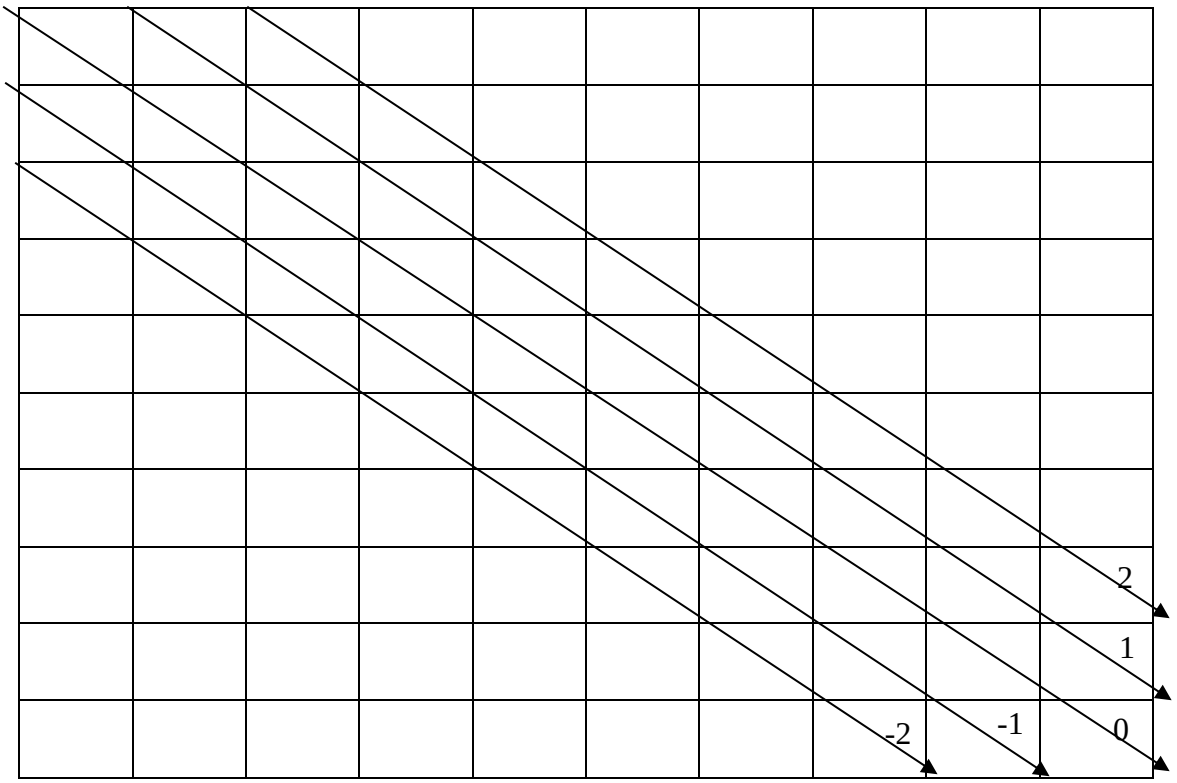
escrever_diagonal – esta função escreve o carácter 'a' (parâmetro de entrada) na diagonal i (também parâmetro de entrada);

limpa_matriz – esta função coloca todos os valores da matriz com o carácter 'a' (parâmetro de entrada)

escreve_matriz – esta função escreve a matriz no ecrã para o utilizador ver os disparates que anda a fazer.

menu – esta função contém o menu das hipóteses anteriores

main – só chama funções



4.1 FAMILIARIZAÇÃO COM FORMAS ALTERNATIVAS DE VER *ARRAYS*

Uma linha da matriz é, de facto, um vetor. Mude então a função ***escrever_linha*** para que tenha como parâmetro de entrada um vetor (cuja dimensão é dada como parâmetro) em vez de uma matriz. Será que pode mudar a função que trata as colunas do mesmo modo? Experimente…

5 Trocas e baldrocas

Vai-se passar agora para as ***strings***.

Vamos fazer um programa que tem duas versões: uma mais simples e outra mais complicada. Programe tudo sem pensar em comprimentos. Se conseguir, já está a caminhar para perceber as ***strings***.

5.1 DESCRIÇÃO GERAL

O programa começa por inicializar uma ***string***

Por exemplo:

`Esta é uma longa string de caracteres`

Depois pretende-se mudar uma (ou mais) palavra(s) por outra(s).

Na primeira versão apenas se muda uma palavra. O utilizador escreve a palavra a desaparecer, e a que a deve substituir.

1ª versão – mudar "**`longa`**" por "**`comprida`**"

`Esta é uma comprida string de caracteres`

Na segunda versão vai-se mudar três palavras (ou texto) por outras três palavras (ou texto). Primeiro têm de ser dadas as palavras (ou texto) e depois mostra-se o resultado.

2ª versão –

Palavras a desaparecer	**`longa`** **`string`** **`de caracteres`**
Palavras a colocar	**`transformação`** **`muita`** **`louca`**

`Esta é uma transformação muita louca`

Assuma que o máximo do texto é de 8.000 caracteres (menos um por causa do '\0'). Não existem palavras (ou texto) com mais de 80 caracteres. Como deve estar a perceber estas limitações servem para usarmos vetores para guardar as ***strings***.

Existe uma matriz chamada `desaparecer`, e outra `colocar`. Na primeira versão (mudar apenas uma palavra) só são usadas as primeiras linhas de cada uma.

O código da página seguinte mostra um começo de programa em que o número de substituições pode ir até ao máximo de 10. Use este código para o seu programa. Não se esqueça que se pode substituir texto e não simplesmente palavras.

```c
#include <stdio.h>
#include <ctype.h>  /* define isspace */
#include <string.h>

#define bool int
#define TRUE 1
#define FALSE 0

#define MAX_WORD_LENGTH 81
#define MAX_WORD 10
#define MAX_TEXT_LENGTH 8000

int quantas_palavras () {
    int num;
    do {
        scanf(" %d", &num);
        if ((num < 1) || (num > MAX_WORD))
            printf("O número de palavras deve ser > 0 e < %d!\n",
                   MAX_WORD);
    } while ((num < 1) || (num > MAX_WORD));
    return num;
}

void ler_substituicoes (int num, char chave[][MAX_WORD_LENGTH], char
colocar[][MAX_WORD_LENGTH]) {
}

bool le_texto(char * text, int maximo) {
}

void substitui_texto(int num, char retirar[][MAX_WORD_LENGTH], char
colocar[][MAX_WORD_LENGTH], char * text, char * text_changed) {
}

main() {
    int  n_word= 0;           // Número de palavras no array word
    char desaparecer[MAX_WORD][MAX_WORD_LENGTH]; // a substituir
    char colocar[MAX_WORD][MAX_WORD_LENGTH];  // Substituições
    char text[MAX_TEXT_LENGTH];            // Texto inicial
    char text_changed[MAX_TEXT_LENGTH];    // Texto modificado

    printf("Quantas palavras tem a lista de substituicao?\n");
    n_word = quantas_palavras ();

    printf("Introduza os textos a retirar e a colocar\n");
    ler_substituicoes(n_word, desaparecer, colocar);

    printf("Introduza o texto inicial. ");
    printf("Termine com uma linha apenas com um ponto final:\n");
    if (!le_texto(text, MAX_TEXT_LENGTH)) {
        printf("Não foi introduzido texto\n");
        return;
    }
    substitui_texto(n_word, desaparecer, colocar,
                                          text, text_changed);
    printf("O texto inicial eh:\n%s\n", text);
    printf("O texto substituido eh:\n%s\n", text_changed);
}
```

5.2 DICAS

Tente programar sem usar as funções de ***strings*** da biblioteca de C.

Talvez a função mais complicada seja a de substituição. Para não ficar muito complicado pense que fazer ***n*** substituições pode ser feito fazendo ***n*** vezes ***uma*** substituição. Assim a função ***substitui_texto*** recebe uma matriz mas pode chamar outra que faz a substituição vetor a vetor.

É claro que isto tem o problema de uma substituição posterior voltar a substituir um texto já substituído (no caso de haver palavras iguais), mas não se preocupe com este tipo de coisas neste momento.

Note que ao fazer comparações, muitas coisas podem acontecer:

A função tem de começar por ver se o texto existe. Faça a função em vez de usar uma já existente nas bibliotecas do C. Vá sempre comparando o texto de `chave` com o texto de `text`. Se começarem a ser iguais uma de quatro coisas pode acontecer para o próximo:

1. os caracteres continuam a ser iguais
2. os caracteres são diferentes;
3. atingiu-se o '\0' no texto a desaparecer (linha da matriz `desaparecer`);
4. atingiu -se o '\0' só no texto principal (`text`).

Para não estragar toda a beleza de programar, não se dão mais dicas.