String

Em C, uma string corresponde sempre a um vetor de carateres (mas um vetor de carateres pode não ser uma string!) e a sua declaração obedece à sintaxe de declaração de vetores de carateres.

A linguagem C apresenta algumas limitações no que respeita ao tratamento de vetores de strings, não fazendo o seu processamento diretamente.

O C não disponibiliza um tipo de dados string e não é possível, por exemplo, concatenar uma string a outra, utilizando os sinais de atribuição **=** e soma **+**, respetivamente, como é possível noutras linguagens. No entanto o C possui uma poderosa biblioteca de funções que permitem realizar, praticamente, todas as operações necessárias sobre strings.

Em C, como já vimos, as strings são representadas entre aspas ‘’ e os carateres entre plicas ‘.

Exemplo: “ Luís “ “a minha escola” ‘A’

nota: ‘A’ é o caráter A (1 byte), “A”, string que ocupa 2 bytes e cujo 1º elemento é o caráter A. Isto porque, em C é necessário na declaração de uma string contar-se, obrigatoriamente, com o espaço para o caráter terminador **‘\0’**, que indica o fim da string,

Dever-se-á, incluir, sempre, mais uma posição para o referido caráter.

Atenção: O caráter ‘\0’, cujo código ASCII é zero, nada tem a ver com o caráter ‘0’ (de código ASCII 48).

Exemplo: Declaração de um vetor que contém uma string com 20 carateres úteis.

```
char s [ 20 ] ;
```

Inicialização automática de strings

A declaração e inicialização de strings segue a sintaxe apresentada sobre vetores.

Exemplos:

```
char nome [ 20 ] = “André” ;
char nome [ 20 ] = {‘A’, ‘n’, ‘d’, ‘r’, ‘é’ } ;
char nome [  ] = “André” ;   /* equivalente a char nome [ 6 ] = “André” ;  */
char *nome = “André” ;       /* Idem */
```

- Os elementos sem inicialização são colocados a zero;
- O compilador calcula o nº de carateres existente na string que faz a inicialização, adiciona um caráter (nulo '/0'), sendo para isso criado um vetor com a dimensão exata para conter a string "André".

Na declaração **char nome [ ] = "André" ;** o vetor nome é inicializado com a string "André", logo considera-se uma string, pois o compilador coloca, automaticamente, o caráter terminador '/0', ficando o vetor com 6 carateres de comprimento.

Na declaração **char nome [ ] = {'A', 'n', 'd', 'r', 'é' } ;** estamos perante um vetor de carateres que não é uma string, pois não há necessidade de colocar o caráter terminador. Trata-se de um vetor com 5 carateres que irão ser inicializados individualmente e não como um todo.

Vetores bidimensionais de string

Para o armazenamento de várias string é necessário definir o nº de string (LINHAS) e o número máximo de carateres que cada uma delas terá (COLUNAS)

Exemplo: Declaração de uma matriz de 3 string, com 3 carateres úteis.

```
char s [3] [4] ;
```

Suponhamos que as string a armazenar seriam: "Ana", "Dia" e "Zé". Ficariam, esquematicamente, representadas da seguinte forma:

		j → 0	1	2	3
i →	0	'A'	'n'	'a'	'/0'
	1	'D'	'I'	'a'	'/0'
	2	'Z'	'É'	'/0'	

Leitura e escrita de strings: printf, puts, scanf e gets

A função printf recebe como formato uma string, que pode ser escrita diretamente.

printf(" Hello World!");

No entanto uma string pode ser escrita, tal como qualquer outra variável, utilizando um formato próprio: **%s,** dentro da função printf.

Exemplos:

```
char nome [ 50 ] = "Sílvia" ;
printf("Nome: %s \n", nome);
```

A função puts permite, unicamente, a escrita de strings, sejam elas constantes ou armazenadas em variáveis, fazendo automaticamente a mudança de linha.

puts("Hello World!"); é equivalente a printf("\nHello World!");

A função scanf permite a leitura de strings utilizando o formato **%s**.

- No entanto a variável que recebe a string não é precedida de um **&** (não é necessário, mas também não gera nenhum erro), ao contrário do que acontece com todos os outros tipos de variáveis que são enviadas para a função;
- Esta função apenas realiza a leitura de uma única palavra. Lê todos os carateres até encontrar um espaço, tab ou enter. Quando isso acontece termina a leitura e coloca todos os carateres lidos até essa altura na variável que lhe foi passada. Em seguida coloca o carater terminador.

Exemplo:

```
# include<stdio.h>
main()
{       char nome [ 50 ], apelido [ 50] ;
        printf("NOME? "); scanf ("%s", nome);
        printf("\nAPELIDO? "); scanf ("%s", apelido);
        printf(" \nNome completo: %s %s ", nome, apelido);
}
```

A função gets (get string) permite colocar, na variável que recebe por parâmetro, todos os carateres introduzidos pelo utilizador. Esta função não está limitada à leitura de uma única palavra.

Exemplo:

```
# include<stdio.h>
main()
{       char nome [ 100 ] ;
        printf("NOME COMPLETO? ");
        gets (nome);
        printf(" \nNome completo: %s ", nome);
}
```

Exercício resolvido: Escreve um programa que leia nomes e os apresente no ecrã até que um nome vazio seja introduzido.

```
# include<stdio.h>
main()
{       char nome [ 100 ] ;
        while ( 1 )
        {
                puts("NOME? ") ;  gets (nome) ;
                if (nome [ 0 ] == '\0' )        /* se a string for vazia */
                        break ;          /* termina o ciclo */
                else
                        printf("NOME INTRODUZIDO: %s\n ", nome) ;
        }
}
```

LEITURA E ESCRITA VÁRIOS VETORES DE STRING

Apesar de na declaração se dever definir o nº de string e o nº máximo de carateres das mesmas, para proceder à sua leitura e escrita, usando o formato **%s**, **basta usar um índice relativo ao nº de strings a ler ou imprimir.**

Exercício resolvido: Escreve um programa que leia 5 nomes, com no máximo 20 carateres, e os apresente no ecrã.

```
# include<stdio.h>
main()
{
        char nome [ 5 ] [ 21 ] ;
        printf("INSIRA 5 NOMES:\n);
        for (int i=0; i<5; i++)
        {
            printf("%d .º NOME? ", i+1);
            scanf ("%s", nome[i]);
        }
        printf("NOMES INSERIDOS:\n);
        for (int i=0; i<5; i++)
        {
            printf("%d .º NOME: %s", i+1, nome[i]));
        }
```

EXERCÍCIOS

1- Existe alguma diferença entre um vetor de carateres e uma string? Dá exemplos.

2- Escreve um programa que devolva o nº de carateres existente numa string.