

# Vetores

## 1. Vetores

### **Exercício:**

Construa um programa, com base no exercício anterior, que declare um vetor de reais com 10 elementos, o inicialize, com números fornecidos através da entrada padrão, e, posteriormente através de uma pesquisa nos elementos do vetor, retorne na saída padrão a posição no vetor do elemento com menor valor.

Observação: Caso o vetor apresente valores iguais deve ser informada a maior posição dentre os valores iguais.

Exemplo de entrada:

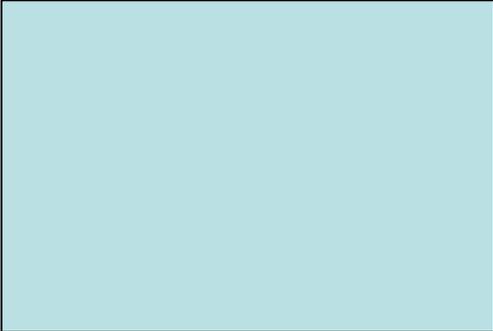
**2.6 0.0 9.2 -3.1 98.0 99.9 -3.1 9.2 6.0 1.5**

Saída para o exemplo de entrada:

**7**

```
#include <stdio.h>
int main()
{
    float vetor[10];
    int indice, ind_menor_ele;
    for (indice=0; indice<10; indice++)
        scanf("%f",&vetor[indice]);
    for (ind_menor_ele=9,indice=8;indice>=0;indice--)
        if (vetor[ind_menor_ele]>vetor[indice])
            ind_menor_ele = indice;
    printf("%d", ind_menor_ele+1);
}
```

```
#include <stdio.h>
#define num_ele 10
int main()
{
    float vetor[num_ele];
    int indice, ind_menor_ele;
    for (indice=0; indice<num_ele; indice++)
        scanf("%f",&vetor[indice]);
    for (ind_menor_ele=num_ele-1,indice=num_ele-2;indice>=0;indice--)
        if (vetor[ind_menor_ele]>vetor[indice])
            ind_menor_ele = indice;
    printf("%d", ind_menor_ele+1);
}
```



# Strings

# Strings

## 2. Strings

Na linguagem de programação C uma string é um vetor de caracteres. Porém, obrigatoriamente um dos caracteres do vetor deve ser o caractere nulo, ou seja, o '\0'. O caractere nulo sucede o último caractere válido da string em questão.

Para declarar uma string, podemos usar a seguinte forma geral:

```
char nome_da_string [tamanho];
```

# Strings

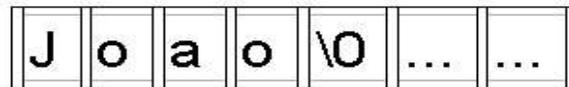
Exemplo:

*char n [7];*

Se inicializarmos a string de 7 posições declarada acima colocando nela a palavra Joao, da seguinte forma:

`char n [7]="Joao";`

Teremos na memória do computador:



# Strings

Formas de inicialização:

`char n [7]="Joao";` ou

`char n []="Joao";` ou

`char n []={'J', 'o', 'a', 'o', '\0'};` ou

`char n [7];`

`n [0]='J';`

`n [1]='o';`

`n [2]='a';`

`n [3]='o';`

`n [4]='\0';`

**Observação:**

...

`char str[10];`

...

~~`str = "Maria";`~~

# Strings

Como ler uma *string* através da entrada padrão?

Podemos utilizar a função *scanf* com o código %s.

Qual é o parâmetro que deve ser fornecido?

Devemos fornecer o endereço de memória onde a *string* deve ser armazenada, ou melhor, devemos fornecer o endereço de onde deve-se iniciar o armazenamento da *string*. Esta informação é obtida através do identificador do vetor de caracteres que conterá a *string*. Exemplo:

...

```
char n [20];
```

...

```
scanf ("%s", n);    ou    scanf ("%s", &n[0]);
```

# Strings

Como escrever uma *string* na saída padrão?

Podemos utilizar a função *printf* com o código %s.

Qual é o parâmetro que deve ser fornecido?

Devemos fornecer o endereço de memória onde a *string* está armazenada, ou melhor, devemos fornecer o endereço de memória onde encontra-se armazenado o primeiro caractere da *string*. Esta informação é obtida através do identificador do vetor de caracteres que contém a *string*. Exemplo:

...

```
char n [20];
```

...

```
printf ("%s", n);    ou    printf ("%s", &n[0]);
```

# Strings

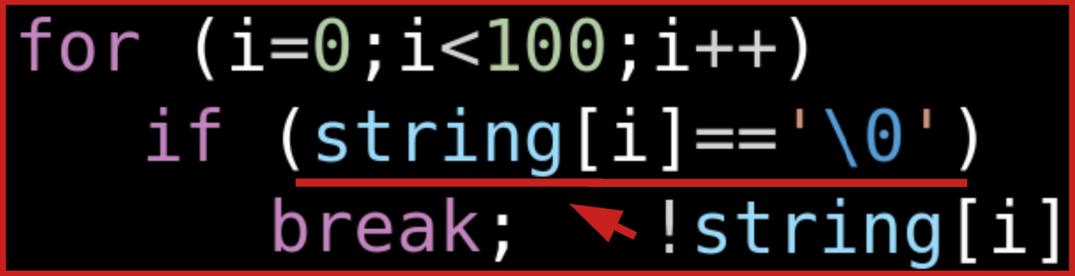
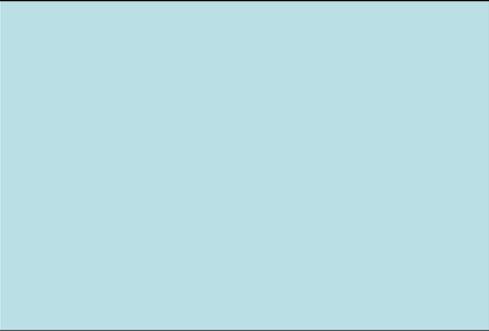
## 2. Strings



### **Exercício:**

Construa um programa que leia através da entrada padrão uma string e retorne na saída padrão o número de caracteres que a mesma possui. Considere que no máximo a string irá conter 99 caracteres válidos.

```
#include <stdio.h>
int main()
{
    char string[100];
    int i;
    printf ("Entre com uma string: ");
    scanf ("%s",string);
    for (i=0;i<100;i++)
        if (string[i]=='\0')
            break;
    printf("A string \"%s\" possui %d caracteres",
    string, i);
}
```



*for (i=0;string[i];i++);*

# Strings – Exercícios

# Strings

## 2. Strings

### Exercício:

Construa um programa que receba através da entrada padrão um número natural que indicará a quantidade de strings que serão fornecidas através da entrada padrão, o programa deve retornar na saída padrão o número de caracteres que pertencente a cada string. Considere que no máximo cada string irá conter 99 caracteres válidos.

Exemplo de entrada:

3  
eu  
ele  
eles

Exemplo de saída:

2  
3  
4