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

UNIVASF

```c
#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

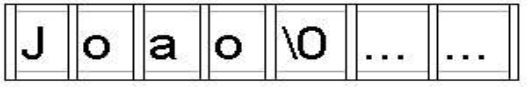
Exemplo:

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);
```

UNIVASF

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);
```

UNIVASF

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.

UNIVASF

```c
#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);
}
```

&string[0]

for (i=0;string[i];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:	Exemplo de saída:
3 eu ele eles	2 3 4

```c
#include <stdio.h>
int main()
{
    char string[100];
    int i, c, n;
    scanf("%d", &n);
    for (c=0; c<n; c++)
    {
        scanf ("%s",string);
        for (i=0;string[i];i++);
        printf("%d\n", i);
    }
}
```

Strings

2. Strings (continuação)

Exercício: Construa um programa que declare duas strings, string1 e string2, respectivamente, ambas com capacidade para armazenar 20 caracteres válidos, o programa deve ler, através da entrada padrão, uma string e colocá-la na string1, depois, o programa deve atribuir o conteúdo da string1 para a string2 e, após este processo, apresentar a string2 na saída padrão.

UNIVASF

```c
#include <stdio.h>
int main()
{
    char string1[21], string2[21];
    int i;
    printf ("Entre com uma string: ");
    scanf ("%s",string1);
    string2 = string1;
    printf(string2);
}
```

```c
#include <stdio.h>
int main()
{
    char string1[21], string2[21];
    int i;
    printf ("Entre com uma string: ");
    scanf ("%s",string1);
    for (i=0;i<21;i++)
        string2[i]=string1[i];
    printf(string2);
}
```

```c
#include <stdio.h>
int main()
{
    char string1[21], string2[21];
    int i;
    printf ("Entre com uma string: ");
    scanf ("%s",string1);
    for (i=0;i<21;i++)
    {
        string2[i]=string1[i];
        if (string1[i]=='\0')
            break;
    }
    printf(string2);
}
```

!string1[i]

```c
#include <stdio.h>
int main()
{
    char string1[21], string2[21];
    int i;
    printf ("Entre com uma string: ");
    scanf ("%s",string1);
    for (i=0; string2[i]=string1[i]; i++);
    printf(string2);
}
```

Funções para Manipulação de Strings

Strings

Um espectador atento já deve ter se perguntado:

Para armazenar uma string na memória eu efetuo a declaração de um vetor de caracteres com um determinado número de elementos, sendo que um destes elementos deverá conter o caractere '\0', o que ocorrerá se o usuário digitar uma string com o número de caracteres igual ou maior que o número de elementos no vetor?

Ocorrerá o que chamamos de falha de segmentação.

Para resolver este problema faremos o seguinte:

...

```
char str[30];
scanf("%29s", str)
```

UNIVASF

Strings

Outro detalhe que já deve ter sido percebido pelos espectadores que implementam suas soluções para os problemas propostos e efetuam testes de execução de seus programas, é o fato de ao se utilizar o %s para ler uma string através da entrada padrão, com a função scanf(), a leitura é finalizada ao se chegar ao \n (enter) ou ao se localizar um espaço.

Existe uma sequência de controle para ler strings com o scanf() mais flexível que o %s. Esta é o %[].

Com ela podemos escolher o que queremos ler. Dentro dos colchetes escrevemos os caracteres permitidos (ex: %[aeiou]) ou negados (ex: %[^aeiou]).

Para lermos uma string podemos usar, por exemplo, %[^\n], que, neste caso, vai ler todos os caracteres até encontrar o \n (**o \n não é incluído na string**).

UNIVASF

Strings

Também é possível limitar o tamanho de uma string lida com o %[].

Por exemplo, para lermos uma string com 60 caracteres no máximo utilizaremos:

```
...

char str[61];

...

scanf("%60[^\n]", str);
```

UNIVASF

Funções Básicas para manipulação de Strings

- gets

A função **gets()** lê uma string do teclado.

Sua forma geral é:

gets (nome_da_string);

```c
/*Exemplo*/
#include <stdio.h>
int main ()
{
    char string[100];
    printf ("Digite o seu nome: ");
    gets (string);
    printf ("\n Ola %s!",string);
}
```

A utilização da função gets() pode gerar falha de segmentação!

Funções Básicas para manipulação de Strings

- puts

A função **puts()** escreve uma string na saída padrão.

Sua forma geral é:

puts (nome_da_string);

Obs.: A função *puts()* efetua automaticamente uma mudança de linha após a impressão da string na saída padrão.

```c
/*Exemplo*/
#include <stdio.h>
int main ()
{
    char string[100];
    puts ("Digite o seu nome: ");
    gets (string);
    puts ("\n Ola");
    printf (string);
}
```

Funções Básicas para manipulação de Strings

- strcpy

Sua forma geral é:

strcpy (string_destino, string_origem);

A função **strcpy()** copia o conteúdo da *string_origem* para a *string_destino*. A partir deste ponto as funções para manipulação de strings apresentadas neste tópico estão no arquivo cabeçalho **string.h**.

UNIVASF

```c
/*Exemplo*/
#include <stdio.h>
#include <string.h>
int main ()
{
   char str1[100],str2[100],str3[100];
   printf ("Entre com uma string: ");
   gets (str1);
   strcpy (str2,str1);
   strcpy (str3, "\nVoce digitou a string ");
   printf ("\n%s",str3);
   puts (str2);
}
```

Funções Básicas para manipulação de Strings

- strlen

Sua forma geral é:

strlen (string);

A função **strlen()** retorna o comprimento da string fornecida. O terminador nulo não é contado. Isto quer dizer que, de fato, o comprimento do vetor que contém a string deve ser, ao menos, uma unidade maior que o inteiro retornado por **strlen()**.

UNIVASF

```c
/*Exemplo*/
#include <stdio.h>
#include <string.h>
int main ()
{
   int size;
   char str[100];
   printf ("Entre com uma string: ");
   gets (str);
   size=strlen (str);
   printf ("\nA string que voce digitou tem tamanho %d",
   size);
}
```

Funções Básicas para manipulação de Strings

- strcat

A função strcat() tem a seguinte forma geral:

strcat (string_destino,string_origem);

A função strcat() concatena a *string_destino* com a *string_origem.* A *string_origem* permanecerá inalterada e será anexada ao fim da *string_destino*.

```c
/*Exemplo*/
#include <stdio.h>
#include <string.h>
int main ()
{
   char str1[50],str2[100];
   printf ("Entre com uma string: ");
   scanf ("%49[^\n]", str1);
   strcpy (str2,"Voce digitou a string ");
   strcat (str2,str1);
   printf ("\n\n%s\n",str2);
}
```

Funções Básicas para manipulação de Strings

- strcmp

Sua forma geral é:

strcmp (string1, string2);

A função strcmp() compara a string1 com a string2. Se as duas forem idênticas a função retorna zero. Se elas forem diferentes a função retorna não-zero.

UNIVASF

```c
/*Exemplo*/
#include <stdio.h>
#include <string.h>
int main ()
{
   char str1[100],str2[100];
   printf ("Entre com uma string: ");
   scanf ("%99[^\n]", str1);
   printf ("\n\nEntre com outra string: ");
   scanf ("%99[^\n]", str2);
   if (strcmp(str1,str2))
      printf ("\nAs duas strings são diferentes!");
   else
      printf ("\nAs duas strings são iguais!");
}
```

Exercício:

Construa um programa que leia duas strings fornecidas pelo usuário, através da entrada padrão, verifique se estas possuem o mesmo tamanho, caso possuam, as compare. Se forem iguais, retorne uma mensagem na saída padrão indicando este fato. Caso não possuam o mesmo tamanho, concatene-as e retorne o resultado desta operação na saída padrão.

UNIVASF