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	2
eu	3
ele	4
eles	

```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);
    }
}
```

```c
#include <locale.h>
#include <stdio.h>
int main()
{
   char string[100];
   int i, c, n;
   setlocale(LC_CTYPE,"Portuguese_Brazil");
   printf("Quantas strings serão fornecidas? ");
   scanf("%d", &n);
   for (c=0; c<n; c++)
   {
      printf("Forneça a %dº string: ", c+1);
      scanf ("%s",string);
      for (i=0;string[i];i++);
      printf("A %dº string %s possui %d caracteres.\n", c, string, 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 pergunt

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");
    puts (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**.

```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()**.

```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.

```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

```c
#include<string.h>
int main()
{
    char string1[100],string2[100];
    printf("\nEntre com a  primeira string: ");
    scanf ("%99[^\n]", string1);
    printf("\nEntre com a  segunda string:  ");
    scanf ("%99[^\n]", string2);
    if (strlen(string1)==strlen(string2))
    {
        if (!strcmp(string1,string2))
            printf("\nAs strings sao iguais!\n");
    } else {
        strcat(string1,string2);
        puts (string1);
    }
}
```

```c
#include<stdio.h>
#include<string.h>
int main(){
    char string1[100],string2[100];
    printf("\nEntre com a  primeira string: ");
    scanf("%99[^\n]",string1);
   printf("\nEntre com a  segunda string:  ");
   setbuf(stdin,NULL);
   scanf("%99[^\n]",string2);
    if (strlen(string1)==strlen(string2)){
        if (!strcmp(string1,string2))
           printf("\nAs strings sao iguais!\n");
    } else {
        if (strlen(string1)+strlen(string2)<=99){
            strcat(string1,string2);
            puts (string1);
        }else
            puts ("\nAs strings concatenadas nao podem ser armazenadas na variavel declarada!\n");
    }
}
```

Funções Básicas para manipulação de Strings

3. Strings (continuação)

Exercício:

Com base no que vimos, construa um programa em C que leia duas *strings,* fornecidas pelo usuário, através da entrada padrão, e verifique se a segunda *string* lida está contida no final da primeira, retornando o resultado da verificação na saída padrão.

UNIVASF