

Vetores e Strings

```
#include <stdio.h>
main ()
{
    char strings [5][100],string [100];
    int count,count2;
    for (count=0;count<5;count++)
    {
        printf ("\n\nDigite a string[%d]: ",count+1);
        scanf ("%s",strings[count]);
    }
    printf ("\nEntre com a string que voce deseja ");
    printf ("saber se pertence ao vetor:");
    scanf ("%s", string);
```

Vetores e Strings

```
for (count=0;count<5;count++)
  for (count2=0;count2<100;count2++)
  {
    if (strings[count][count2]!=string[count2])
      break;
    if (string[count2]=='\0')
    {
      printf ("\nA string %s esta na posicao %d do%s",
        string, count+1, " vetor de strings.\n");
      exit(0);
    }
  }
  printf ("\nA string %s nao esta contida no vetor de %s",
    string, "strings.\n");
}
```

Vetores e Strings

4. Funções Básicas para manipulação de Strings

- gets

A função **gets()** lê uma string através do teclado. Sua forma geral é:

```
gets (nome_da_string);
```

Obs.: A função `scanf()` finaliza a leitura de uma string quando encontra o caracter ' ' ou o '\n', já a função `gets()` finaliza a leitura de uma string apenas quando encontra o caracter '\n'.

Vetores e Strings

- gets (continuação)

Exemplo:

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

Vetores e Strings

4. Funções Básicas para manipulação de Strings

A função **gets()** pode gerar um estouro da área (vetor) reservado para o armazenamento da *string*.

Este problema pode ser resolvido com a formatação da entrada através da utilização da função *scanf()*.

Exemplo:

```
...  
char str[31];  
scanf ("%30s", str);  
...
```

Vetores e Strings

4. Funções Básicas para manipulação de Strings (continuação)

- **strcpy**

Sua forma geral é:

```
strcpy (string_destino, string_origem);
```

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

Vetores e Strings

- strcpy (continuação)

Exemplo:

```
#include <stdio.h>
#include <string.h>
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%s\n", str3, str2);
}
```

Vetores e Strings

4. Funções Básicas para manipulação de Strings (continuação)

- **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 número de caracteres contidos no vetor que compõem a *string* deve ser um a mais que o inteiro retornado por **strlen()**.

Vetores e Strings

Exemplo:

```
#include <stdio.h>
#include <string.h>
main ()
{
    int size;
    char str[100];
    printf ("Entre com uma string: ");
    gets (str);
    size=strlen (str);
    printf ("\nA string que voce digitou ocupa %d %s",
        size+1, "caracteres do vetor.");
```

```
}
```

Vetores e Strings

4. Funções Básicas para manipulação de Strings (continuação)

- **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*.

Vetores e Strings

- strcat (continuação)

Exemplo:

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

Vetores e Strings

4. Funções Básicas para manipulação de Strings (continuação)

- **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**.

Vetores e Strings

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

Vetores e Strings

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 contrário, concatene-as e retorne o resultado desta operação na saída padrão.

```

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

```

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 contrário, concatene-as e retorne o resultado desta operação na saída padrão.

```

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

```


Funções

Funções são as estruturas que permitem ao usuário separar seus programas em módulos. Para fazermos programas grandes e complexos temos de construí-los módulo a módulo.

Uma função no C tem a seguinte forma geral:

```
tipo_de_retorno nome_da_função (declaração_de_parâmetros)
{
    corpo_da_função
}
```

Funções

O tipo-de-retorno é o tipo de variável que a função vai retornar. O *default* é o tipo **int**, ou seja, o tipo-de-retorno assumido por omissão.

A declaração de parâmetros é uma lista com a seguinte forma geral:

tipo nome1, tipo nome2, ... , tipo nomeN

Repare que o tipo deve ser especificado para cada uma das N variáveis de entrada. É na declaração de parâmetros que informamos ao compilador quais serão as entradas da função (assim como informamos a saída no tipo-de-retorno).

É no corpo da função que as entradas são processadas, saídas são geradas ou outras coisas são feitas.

Funções

- Comando return

Forma geral:

return valor_de_retorno; ou return;

Quando se executa uma declaração **return** a função é encerrada imediatamente e, se o valor de retorno é informado, a função retorna este valor. É importante lembrar que o valor de retorno fornecido tem que ser compatível com o tipo de retorno declarado para a função.

- Comando return (exemplo 1)

```
#include <stdio.h>
int Quadrado (int a)
{
    return (a*a);
}
main ()
{
    int num;
    printf ("\nEntre com um numero: ");
    scanf ("%d",&num);
    num=Quadrado(num);
    printf ("\n\nO seu quadrado vale: %d\n",num);
}
```

- Comando return (exemplo 1)

```
#include <stdio.h>
int Quadrado (int a)
{
    return (a*a);
}
main ()
{
    int num;
    printf ("\nEntre com um numero: ");
    scanf ("%d",&num);
    printf ("\n\nO seu quadrado vale: %d\n",
    Quadrado(num));
}
```

Funções

Observação:

Devemos lembrar agora que a função **main()** é uma função e como tal devemos tratá-la. A função **main()** retorna um inteiro. Isto pode ser interessante se quisermos que o sistema operacional receba o valor de retorno da função **main()**. Se assim o quisermos, devemos nos lembrar da seguinte convenção: se o programa retornar zero, significa que ele terminou normalmente, e, se o programa retornar um valor diferente de zero, significa que o programa teve um término anormal.

```
#include <stdio.h> /*Exemplo*/
int EPar (int a)
{
    if (a%2)
        return 0;
    return 1;
}
int main ()
{
    int num;
    printf ("Entre com numero: ");
    scanf ("%d",&num);
    if (EPar(num))
        printf ("\n\nO numero e par.\n");
    else
        printf ("\n\nO numero e impar.\n");
    return 0;
}
```

Funções

Exercício

Construa um programa que possua a função “EhDivisivel(int **a**, int **b**)”, escrita por você. A função deverá retornar 1 se **a** for divisível por **b**. Caso contrário, a função deverá retornar zero. O programa deve ler dois números fornecidos pelo usuário (**a** e **b** respectivamente), e utilizar a função EhDivisivel para retornar uma mensagem dizendo se **a** é ou não divisível por **b**.


```

#include <stdio.h>
EhDivisivel(int a, int b)
{
    if (a%b)
        return 0;
    return (1);
}
main() {
    int a,b;
    printf ("\nPrograma que retorna se \"a\" eh divisivel por \"b\"");
    printf ("\n\nEntre com o valor de \"a\": ");
    scanf("%d",&a);
    printf ("\nEntre com o valor de \"b\": ");
    scanf("%d",&b);
    if (b)
    {
        if (EhDivisivel(a,b))
            printf("\n\"a\" eh divisivel por \"b\"");
        else
            printf("\n\"a\" nao eh divisivel por \"b\"");
        return 0;
    }
}

```

```
else
{
    printf("\nTentativa de divisao por zero!");
    return 1;
}
}
```