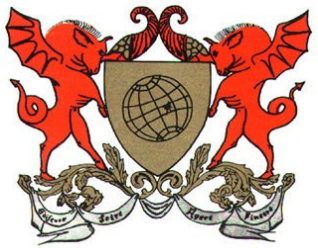
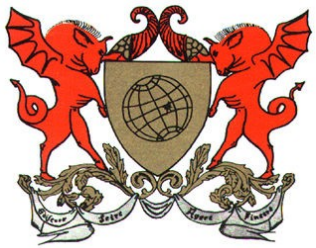


Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

SIN 110
Programação

Sistemas de Informação
Prof. Guilherme C. Pena
guilherme.pena@ufv.br



Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

Aula de Hoje

Strings

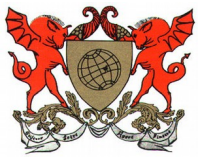
Material adaptado do Prof. Murilo Naldi

String

Cadeia de caracteres (ou String)

- É uma sequência de letras e símbolos, onde os símbolos podem ser espaços em branco, dígitos e vários outros como pontos de exclamação, interrogação, símbolos matemáticos, etc.

Ou seja, é uma sequência de caracteres.

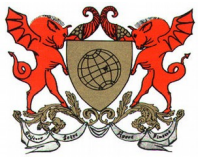


Representação de uma String

O exemplo: "a99.9b9.9c9-22", contém letras, números e símbolos.

Strings são representadas entre **aspas duplas.**

```
printf("texto");
```

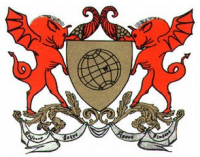


String

Basicamente uma **string** é qualquer **vetor** do tipo **char** com uma característica **IMPORTANTE**:

> O último dos caracteres sempre deve ser o caracter nulo (*caracter null*) representado pelo símbolo '\0' (contra-barra zero).

'\0' representa o fim da string.



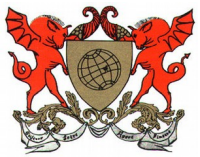
Declaração de uma String

Por essa característica, devemos sempre declarar o vetor com o **tamanho + 1**.

```
char texto[TAMANHO + 1];
```

Devemos utilizar uma posição além do tamanho desejado para que possa ser colocado o marcador **'\0'** no final da string quando essa tiver o tamanho máximo.

```
Ex: char texto[4] = {'a', 'b', 'c', '\0'};
```

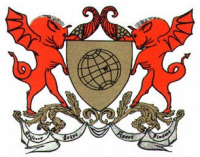


Declaração de uma String

O texto que vamos gravar em uma string não precisa ocupar todos os caracteres do vetor.

Por isso, utiliza-se o '\0' para indicar onde o texto termina.

```
char texto[10];
texto[0] = 'a';
texto[1] = 'b';
texto[2] = 'c';
texto[3] = '\0';
```



Declaração de uma String

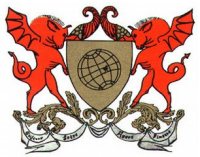
Outros modos de declarar/inicializar uma string:

```
char texto[] = {'a','b','c', 'd','\0'};

char texto[5] = "abcd";

char texto[] = "abcd";
```

Nas duas últimas declarações, o '\0' é automaticamente colocado na posição de índice 4.

No entanto, essa atribuição pode ser feita APENAS na declaração!

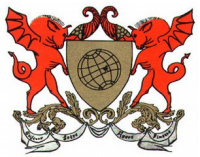
```
texto = "abcd"; //NÃO É PERMITIDO
```



Declaração de uma String

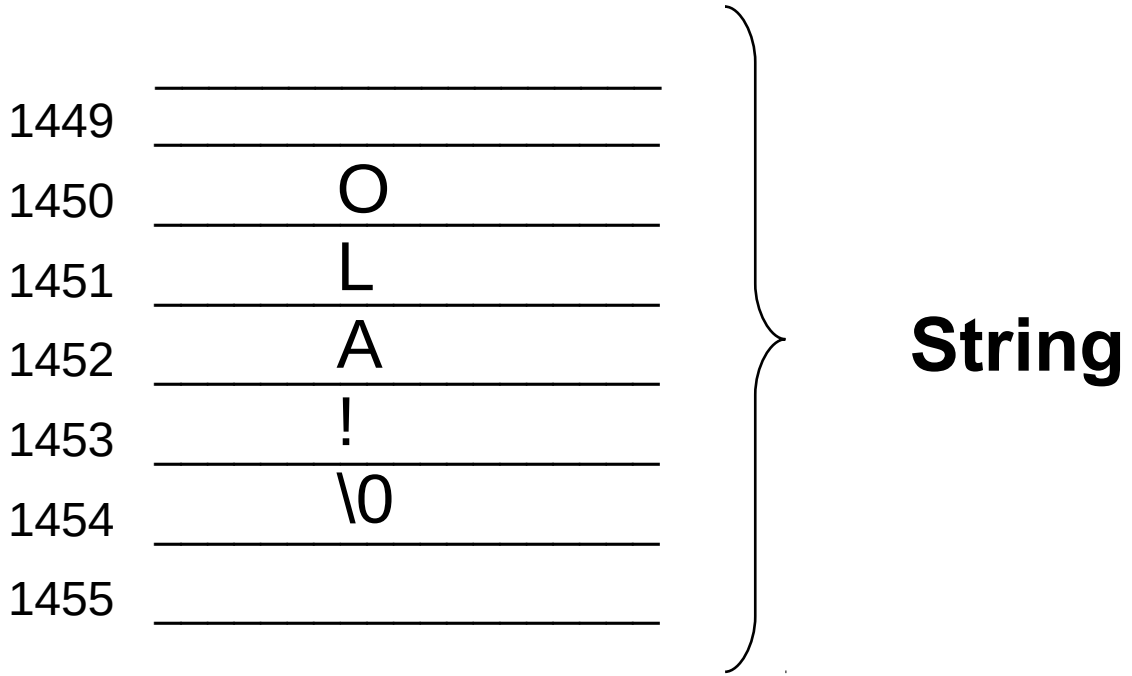
A maioria dos caracteres **alfanuméricos**, isto é, caracteres **alfabéticos e numéricos**, que podem pertencer a uma string em C estão relacionados na tabela **ASCII**.

http://pt.wikipedia.org/wiki/ASCII



String

Cada caracter de uma string ocupa 1 byte de memória e o último caracter é sempre o '\0' (null).



Lendo uma string do teclado

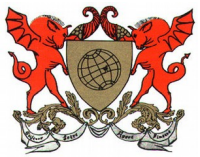
Podemos ler uma string do teclado de várias formas, porém **sempre devemos entender como cada uma delas funciona**.

Lendo a string inteira* (para num espaço ou salto de linha):

```
char str[10];
scanf("%s", str);
```

* Note que não utilizamos o e comercial (&)

Uso restrito pois **lê somente uma palavra**.



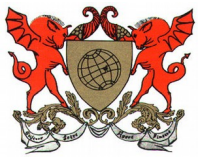
Lendo uma string do teclado

Podemos utilizar a **função "gets"** (stdio.h).

Lê uma cadeia de caracteres até encontrar um enter (fim de linha).

```
char str[10];
gets(str);
```

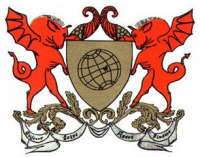
Melhor para ler frases e nomes completos.



Função Leitura - **fflush(stdin)**

As funções **scanf() e gets()** tem um detalhe importante na linguagem C ao ler **char ou strings**:

Sempre que digitamos um valor, apertamos a tecla **ENTER**, e esse **ENTER** fica gravado em um espaço, chamado *buffer* de entrada, como um caracter **'\n'**.



Função Leitura - **fflush(stdin)**

O grande problema desse caracter **'\n'** é que no primeiro uso de um scanf() (seja ler um **inteiro**, um **real** ou um **char**) e depois precisamos ler novamente outro dado char ou string,

```
...
scanf("%d", &numero);
scanf("%c", &letra);
gets(s);
...
```

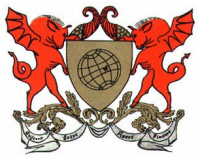
O segundo scanf() percebe o **'\n'** deixado pelo primeiro scanf() como um caracter válido e **"pega"** ele para a variável do segundo scanf().

Função Leitura - **fflush(stdin)**

A forma de solucionar este problema é usando uma função especial **fflush(stdin)** que **"limpa"** o *buffer* de entrada, tirando o caracter **'\n'** de lá.

```
...
scanf("%d", &numero);
fflush(stdin);
scanf("%c", &letra);
fflush(stdin);
gets(s);
...
```

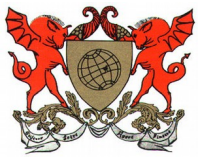
Nesse caso, com o *buffer* limpo, o segundo scanf() volta a ficar esperando que o usuário digite alguma coisa.



Função Leitura - **fflush(stdin)**

OBS: Lembre-se que isso só será necessário se o programa precisar ler um **char** ou uma **string** após um scanf() já ter sido acionado.

Se o programa ler **apenas** valores numéricos (int ou float) então não precisa usar a função especial.



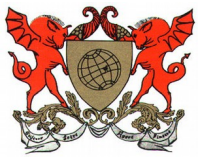
Função Leitura - **fflush(stdin)**

Exemplo:

```
#include <stdio.h>
int main( )
{
     int num;
     char letra;
     printf("Digite um numero inteiro: ");
     scanf("%d", &num);
     printf("O valor digitado foi: %d\n", num);
     printf("Digite uma letra do alfabeto: ");
     fflush(stdin);
     scanf("%c", &letra); //gets("alguma string");
     printf("A letra digitada foi: %c\n", letra);
     return 0;
}
```

O programa imprimirá na tela:

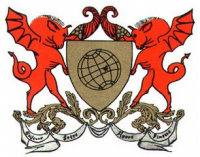
```
Digite um numero inteiro: 56
O valor digitado foi: 56
Digite uma letra do alfabeto: H
A letra digitada foi: H
```



Escrevendo uma string na tela

Podemos escrever a string inteira na tela com o printf (%s):

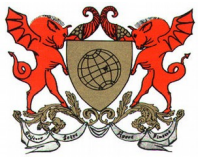
```
printf("%s", str);
```



Escrevendo uma string na tela

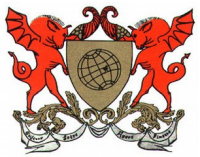
Podemos utilizar a **função "puts"** (stdio.h).

```
puts(str);
```



Exercício

Faça um programa que permita o usuário digitar uma string e no final, imprimir o tamanho da string, isto é, quantos caracteres ela possui com exceção do '\0'.

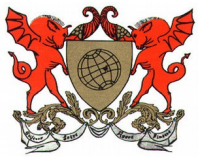


Exercício

Declare duas strings com capacidade para 20 caracteres. Leia pela entrada padrão a primeira string.

Em seguida, copie o texto da primeira string para a segunda.

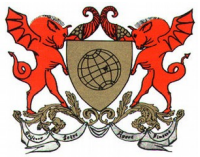
Imprima no final o conteúdo das duas strings.



Exercício

Declare duas strings com capacidade para 20 caracteres e uma para 40 caracteres. Leia pela entrada padrão a primeira e a segunda string.

Em seguida, junte as duas strings lidas na terceira e imprima no final o seu conteúdo.



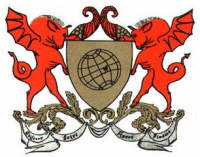
Vetor de Strings

Cada string é um vetor de caracteres:

```
char nome0[tam_string];
char nome1[tam_string];
char nome2[tam_string];
char nome3[tam_string];
char nome4[tam_string];
```

Desta forma, para acessar a i-ésima string usamos:

```
nomenum;
```



Vetor de Strings

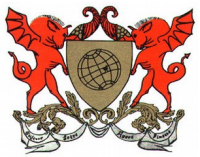
Se cada string é um vetor de caracteres, um vetor de strings é:

Uma matriz de caracteres

```
char nome[num_strings][tam_strings];
```

Desta forma, para acessar a i-ésima string usamos:

```
nome[i];
```



Vetor de Strings

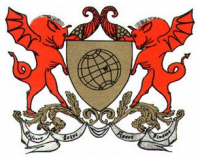
Cada string é um vetor de caracteres:

```
char nome[5][tam_string];
```

Para acessar a i-ésima string usamos:

```
nome[i];
```

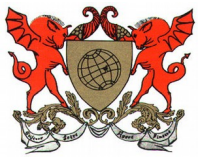
onde **i** varia de **0 até 4**.



Exemplo

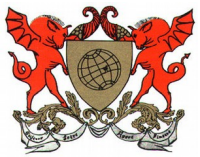
Ler 5 strings e depois imprimir cada uma delas.

```
char strings[5][50];
int i;
for(i=0; i<5; i++){
   printf("Digite uma string:");
   scanf("%s", strings[i]);
}
for(i=0; i<5; i++)
   printf("%s\n", strings[i]);
```

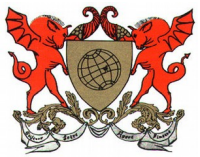


Exercício

Depois de ler as 5 strings, leia mais uma string e verifique se ela está no vetor. Caso esteja, indique o índice no vetor.



Funções da biblioteca padrão para manipulação de strings

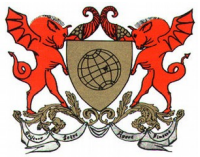


UFV - Campus Rio Paranaíba
Sistemas de Informação

Manipulando cadeias de caracteres

Em C, existe uma biblioteca de funções só com comandos para trabalhar com strings.

Biblioteca **<string.h>**

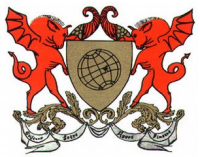


Manipulando cadeias de caracteres

Biblioteca **<string.h>**

Algumas funções:

- **strlen**
- **strcat**
- **strcmp**
- **strcpy**

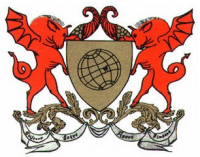


Função **strlen()**

A função strlen() retorna o tamanho da string fornecida sem retornar o '\0'.

Ex: strlen(“casa”) == 4

O tamanho retornado é sempre um número inteiro.



Função **strlen()**

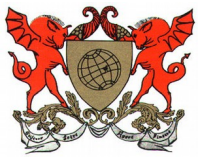
Formato geral da função: strlen(nome_do_vetor);

O comprimento de uma string é definido como o número de caracteres entre o início da string até o '\0', sem contá-lo.

Isso não deve ser confundido com o tamanho do array que contém a string.

Ex:

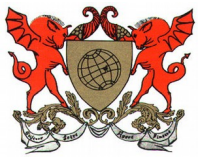
```
char texto[50] = "Sistemas";
strlen(texto) == 8;
```



Função **strlen()**

Exemplo:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char nome[81];
    printf("Digite seu nome: ");
    gets(nome);
    printf("Nome: %s", nome);
    printf("\nNumero de caracteres: %d",strlen(nome));
    return 0;
}
```

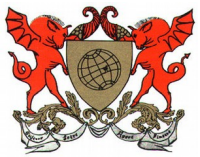


Função **strcpy()**

A função strcpy() copia a string do vetor no campo origem para o vetor no campo destino incluindo o caracter '\0'.

Sua forma geral é:

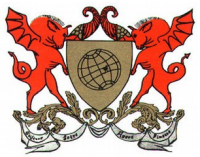
```
strcpy (string destino,string origem);
```



Função **strcpy()**

Esta função **não** verifica se a segunda string cabe no espaço livre da primeira string.

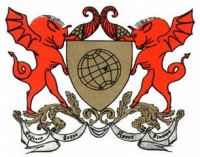
Então, para evitar problemas, o tamanho do vetor destino deve ser grande o bastante para conter a string da origem includindo o caracter '\0'.



Função **strcpy()**

Exemplo:

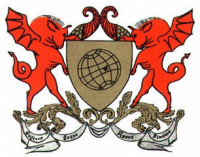
```
#include <stdio.h>
#include <string.h>
int main ()
{
  char str1[]="String de exemplo";
  char str2[40];
  char str3[40];
  strcpy (str2,str1);
  strcpy (str3,"Copia bem-sucedida");
  printf ("str1: %s\nstr2: %s\nstr3: %s\n",str1,str2,str3);
  return 0;
}
```



Função **strcpy()**

Exemplo (saída):

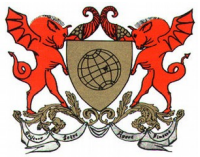
```
str1: String de exemplo
str2: String de exemplo
str3: copia bem-sucedida
```

Como você resolveria o seguinte problema?

Comparação de strings:

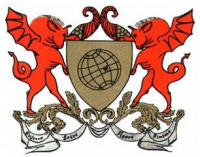
Escreva um programa em C que receba duas palavras e verifique se elas são iguais ou não.



Comparação de strings

A comparação em destaque funciona?

```
int main()
{
        char palavra1[10], palavra2[10];
        printf("Digite uma palavra: ");
        gets(palavra1);
        printf("Digite outra palavra: ");
        gets(palavra2);
        if(palavra1 == palavra2)
           puts("\nPalavras iguais");
        else
           printf("\nPalavras diferentes");
}
```

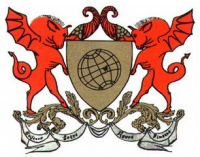


Comparação de strings

O programa anterior não trabalha corretamente, pois **palavra1** e **palavra2** são **vetores**.

Daí, **palavra1** e **palavra2** armazenam endereços distintos de memória, que não são os mesmos.

Portanto, palavra1 != palavra2



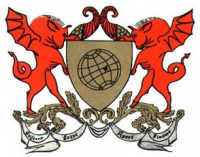
Função **strcmp()**

A função strcmp() compara uma string1 com uma string2.

Sua forma geral é:

```
strcmp (string1,string2);
```

Ela inicializa comparando o primeiro caracter de cada string. Se eles são iguais ela passa para o par seguinte. Quando um par for diferente, ela avalia qual dos caracteres é maior, seguindo a **ordem lexicográfica**.



Função **strcmp()**

Ordem Lexicográfica

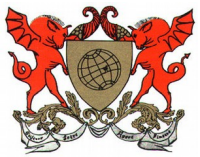
Em se tratando de strings, a ordem lexicográfica segue os valores da **Tabela ASCII**.

Bin	Oct	Dec	Hex	Sinal
0100 0000	100	64	40	@
0100 0001	101	65	41	A
0100 0010	102	66	42	B
0100 0011	103	67	43	C
0100 0100	104	68	44	D
0100 0101	105	69	45	E
0100 0110	106	70	46	F

Bin	Oct	Dec	Hex	Sinal
0110 0000	140	96	60	`
0110 0001	141	97	61	a
0110 0010	142	98	62	b
0110 0011	143	99	63	c
0110 0100	144	100	64	d
0110 0101	145	101	65	e
0110 0110	146	102	66	f

Ex: “ABC” está antes de “abc” lexicograficamente

“AAa” está antes de “AaA” lexicograficamente

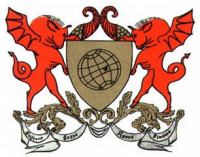


Função **strcmp()**

```
strcmp (string1,string2);
```

Portanto, o retorno da função strcmp() é sempre um número inteiro:

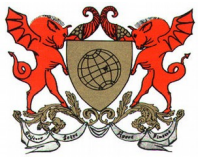
< 0	O primeiro caracter diferente tem um valor menor na string1 do que na string2.
= 0	Todos os caracteres de ambas strings são idênticos.
> 0	O primeiro caracter diferente tem um valor maior na string1 do que na string2.



Função **strcmp()**

Exemplo:

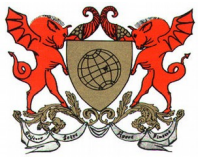
```
strcmp("casa", "carro") == 1
strcmp("casa", "casa") == 0
strcmp("carro", "casa") == -1
```



Função **strcmp()**

Exemplo:

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char cor[] = "azul";
  char texto[80];
  do {
     printf("Adivinhe minha cor favorita?\n");
     gets(texto);
  } while (strcmp (cor,texto) != 0);
  puts ("Resposta certa!");
  return 0;
}
```

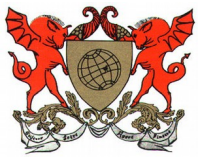


Função **strcat()**

A função strcat() **concatena** a string origem na string destino. O caracter '\0' da string destino é sobrescrito pelo primeiro caracter da origem e um novo '\0' é inserido no final da nova string destino.

Sua forma geral é:

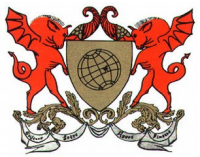
```
strcat (string destino,string origem);
```



Função **strcat()**

Concatenar significa unir duas partes. No caso das strings origem e destino, a união de ambas ficará na string destino.

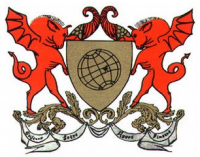
Após a concatenação, a string de origem permanecerá inalterada e apenas uma cópia dela é ligada à string de destino.



Função **strcat()**

Esta função **não** verifica se a segunda string cabe no espaço livre da primeira string.

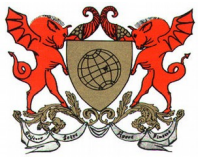
Então, para evitar problemas, o tamanho do vetor destino deve ser grande o bastante para conter o tamanho da string da origem **+** o tamanho da string destino includindo o caracter **'\0'**.



Função **strcat()**

Exemplo:

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char str[80];
  strcpy (str,"Todas ");
  strcat (str,"strings ");
  strcat (str,"estao ");
  strcat (str,"concatenadas.");
  puts (str);
  return 0;
}
```



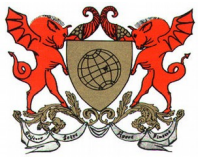
Outras Funções (string.h)

- **strupr(str)**: Converte uma string para maiúsculas.

- **strlwr(str)**: Converte uma string para minúsculas.

- **strrev(str)**: Inverte o conteúdo de uma string.

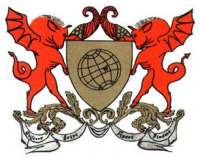
- **strset(str, char)**: Substitui todos os caracteres de uma string pelo caractere especificado.



Convertendo String em Números

Biblioteca **stdlib.h**

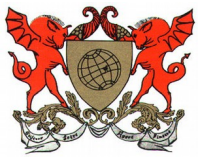
Função	Converte
`atoi(<str>)`	String para int
`atof(<str>)`	String para float
`itoa(<int>)`	Int para string
`strtod(<str>)`	String para double



Convertendo String em Números

Exemplo:

```
int main()
{
 char string1[3],string2[3];
 strcpy(string1, "123");
 strcpy(string2, "123");
  printf("string1 + string2 = %d",atoi(string1) + atoi(string2));
 return 0;
}
```

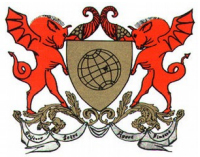


Atenção

Jamais usar o comando de atribuição para atribuir um valor a uma string.

Jamais use o operador de comparação == para comparar duas strings.

Para acessar um elemento da string basta acessá-lo através de seu índice (Igual como no vetor).



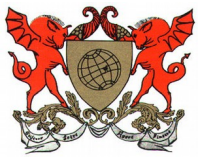
Perguntas

- O que acontece?

```
char s1[30],s2[30];
if (s1 != s2)
    printf("Diferentes");
```

- O que acontece?

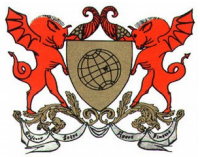
```
char m[10] ="dog";
puts(m);
strcpy(m+1,m);
puts(m);
```



Exemplo

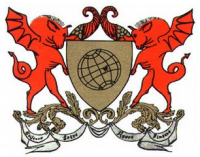
Crie um programa que o usuário possa digitar todos os seus nomes separadamente, para isso, o programa deverá perguntar quantos nomes o usuário possui. Em seguida, crie uma string com o nome completo do usuário.

Dica: Use um vetor de strings.



Exemplo

Crie um programa que o usuário digite três palavras e por fim ele deve imprimi-las em ordem alfabética.



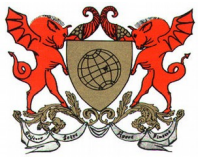
Dicas para estudar

Como estudar o que aprendemos até hoje?

Dica: Faça perguntas a si mesmo que se relacione com a matéria.

Por exemplo:

Como eu faço um código que lê um número (ou letra) e mostra na tela?

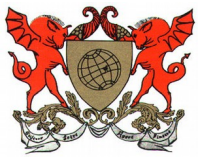


Dicas para estudar

É simples mas faça. A partir daí melhore esse código ainda se questionando:

Como mostrar agora 2 números na tela? E 10 números? E somar 100 números?

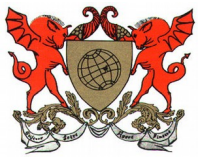
À medida que você **abrir sua mente** para **possibilidades de programas mais difíceis** você precisará recorrer às "**ferramentas**" mais úteis para seu problema (**if-else, for, while...**).



Dicas para estudar

A única maneira de aprender programação é

PROGRAMANDO!

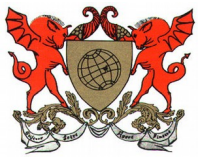


Exercícios

1) Faça um programa para receber um nome e imprimir as 4 primeiras letras do nome.

2) Faça um programa para receber uma string e imprimi-la de trás para frente, sem usar e usando strrev().

3) Receber uma string, conte e imprima quantas vogais (a, e, i, o, u) ela possui. Entre com um caractere (vogal ou consoante) e substitua todas as vogais da palavra dada por esse caractere.

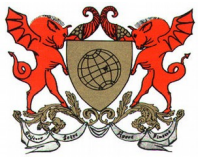


Exercícios

4) Escreva um programa que recebe uma string s e inteiros não-negativos i e j e devolve o segmento s[i..j].

5) Implemente um programa que leia duas strings, **str1** e **str2**, e um valor inteiro positivo N. Concatene N caracteres da string **str2** à string **str1** e termina **str1** com NULL.

6) Leia duas cadeias de caracteres A e B. Determine quantas vezes a cadeia A ocorre na cadeia B.



Exercícios

7) Leia N strings em um vetor de strings (MAX 100) e para cada uma delas imprima a string e o seu tamanho.

8) Escreva um programa que recebe o nome e o valor de 5 produtos e imprime o nome do produto mais caro e o nome do produto mais barato. Use vetor de strings.