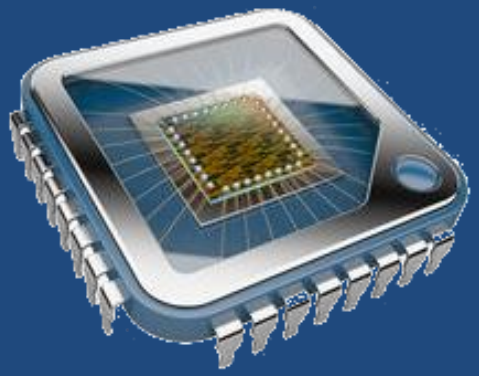


Introdução à Programação
Aula 10

Prof. Max Santana Rolemberg Farias
max.santana@univasf.edu.br
Colegiado de Engenharia de Computação

GESE
Grupo de Engenharia de Sistemas Embarcados

CEComp
Engenharia da Computação
UNIVASF

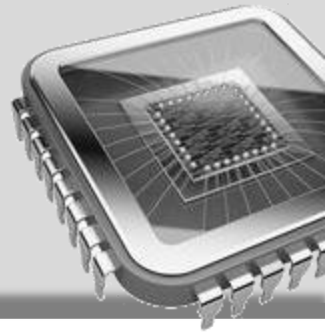
Vetores

- São estruturas quem contém um conjunto de elementos de um mesmo tipo de dado.
  - Com uma dimensão
  - Armazenados em posições da memória
  - Cada posição (índice) do vetor armazena somente um valor.
- Os vetores podem ser representados graficamente da seguinte forma:

VOCÊ ? SABIA

Na linguagem C a numeração dos índices começa sempre em zero. O seja, os dados serão indexados de 0 a 9 para um vetor de tamanho 10.

Vetores
Declaração de Vetores

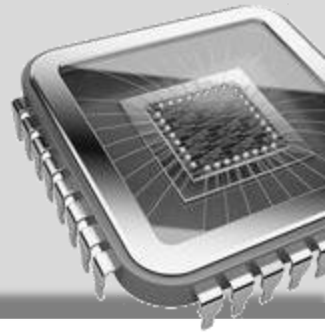


- Os vetores podem ser usados diretamente como variáveis.

<tipoDado> <nomeVetor>[<tamanho>]

Os elementos de um vetor não podem ser de um tipo função.
O tamanho do vetor deve ser observado pelo programador. Se o programador não tiver atenção o vetor pode sobrescrever variáveis do programa.

Vetores
Atribuição de valores



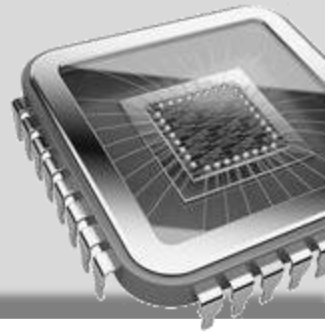
- As variáveis de tipo vetor podem ser iniciadas, mas não podem ser usadas como operando esquerdo em operações de atribuição...

```
int vetor0[10];
int vetor1[10]
vetor0 = vetor1
```

Somente os elementos de um vetor podem ser modificados. Para atribuir um valor ao elemento do vetor dever fazer:

<vetor>[<indice>] = <valor>

Vetores
Iniciação



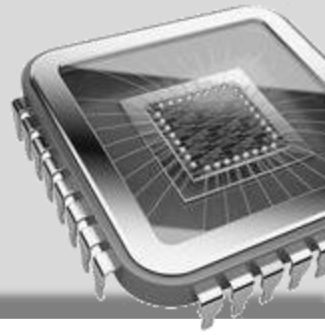
- As variáveis do tipo vetor podem ser iniciadas como lista de iniciação, que são relação de valores dispostos entre chaves e separado por vírgula.

```
int vetor0[5] = {0,1,2,3,4};
```

Que pode-se deixar em aberto o tamanho do vetor, o qual será preenchido pelo número de valores fornecidos no momento da declaração (durante a inicialização)

```
int vetor1[] = {0,1,2,3,4,5,6,7,8,9};
```

Vetores
Exemplo



```
...
int v [5];
v [2] = 13;
v [1] = 5;
...
```

0x12		0xA1	0xA2	0xA3	0xA4	0xA5	Edereços
v	->	v[0]	v[1]	v[2]	v[3]	v[4]	Referências
v	->	?	?	?	?	?	Posições
v	->	?	?	**13**	?	?	Posições
v	->	?	**5**	**13**	?	?	Posições

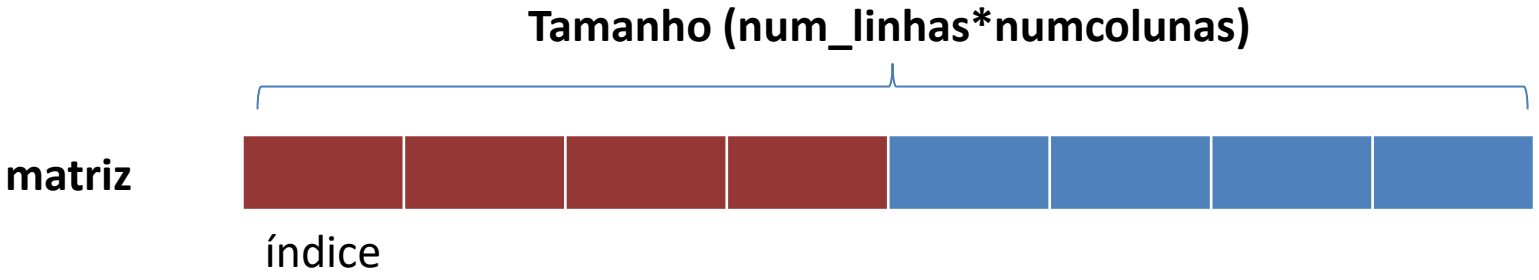
- A execução da declaração int vertor0[5] faz com que sejam alocados 5 espaços na memória para armazenar os 5 elementos do tipo int.

Vetores Multidimensionais

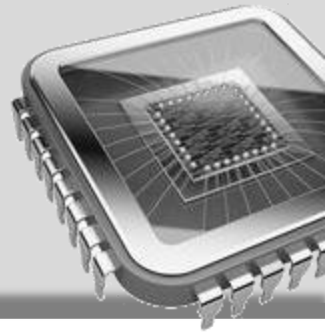
- Os vetores multidimensionais podem ser usados diretamente como variáveis.

<tipoDado> <nomeVetor>[<num_linhas>] [<num_colunas>]

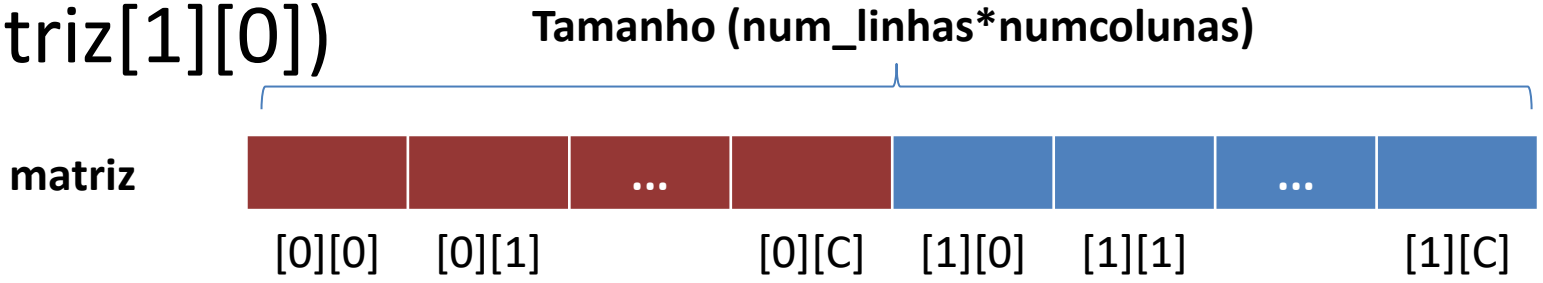
- Pode ser representados graficamente da seguinte forma: int matriz[2][4]



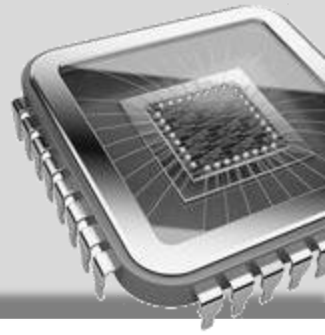
Vetores Multidimensionais Armazenamento



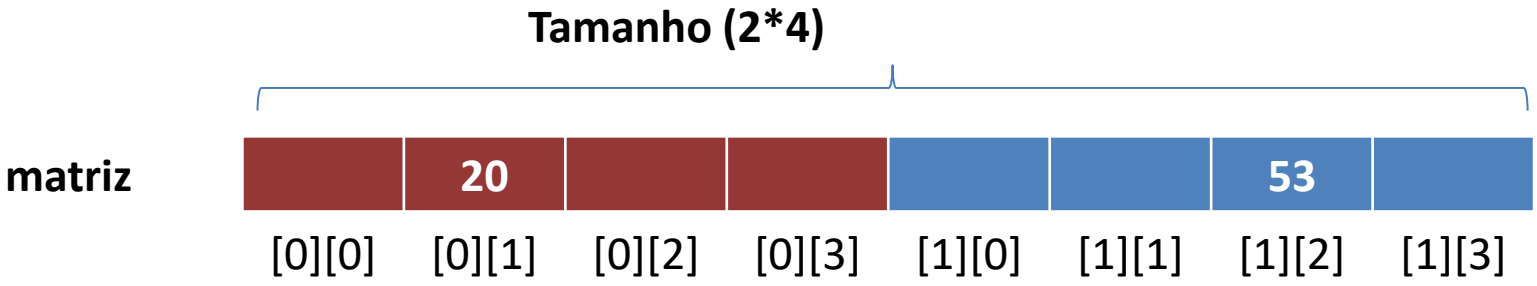
- Na primeira posição todos os índices são zero (matriz[0][0])
- Na próxima posição o índice da direita (coluna) é incrementado (matriz[0][1])
- Quando o índice da coluna chegar ao valor máximo é incrementado o índice da linha (matriz[1][0])



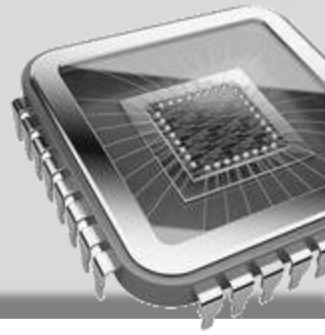
Vetores Multidimensionais Exemplo



```
int matriz[2][4];
matriz[0][1] = 20;
matriz[1][2] = 53;
```



Vetores Multidimensionais Iniciação

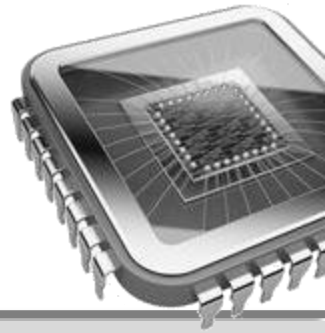


- Assim como os vetores unidimensionais os vetores multidimensionais também podem ser inicializados na declaração.

```
int matriz0[2][4] = {0,1,2,3,4,5,6,7};
```

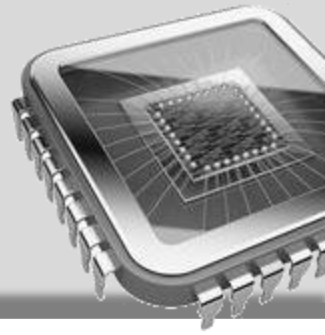
Que pode-se deixar em aberto uma e somente uma dimensão do vetor multidimensional, o qual será preenchido pelo número de valores fornecidos no momento da declaração (durante a inicialização)

```
int matriz1[][5] = {0,1,2,3,4,5,6,7,8,9};
```



STRING

String

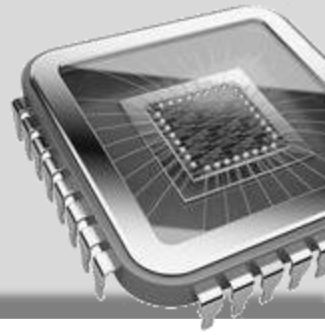


- Na linguagem C uma string é um vetor de caracteres.
- Para declarar uma string, podemos usar o seguinte formato geral:

char <nome da string[<tamanho>]

O vetor de caractere precisa, obrigatoriamente, ter o caractere nulo (‘\0’) em um dos seus elementos. O caractere nulo deve suceder o último caractere válido da string.

String
Inicialização



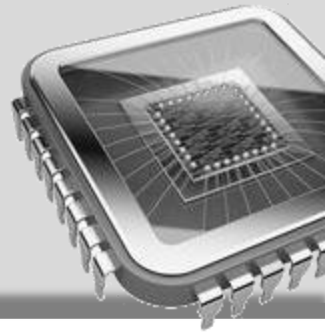
- Forma de inicialização de uma String

```
char s1[7] = “JOAO”;
char s2[] = “JOSE”;
char s3[] = {‘M’,’A’,’R’,’I’,’A’,’\0’};
char s4[10];
s4[0] = ‘M’;
s4[1] = ‘A’;
s4[2] = ‘X’;
s4[3] = ‘\0’;
```

Que não é possível atribuir uma string a um vetor de caracteres...

s4 = “MARIA”;

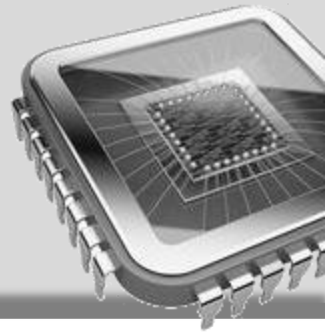
String
Como Ler



- Podemos utilizar a função scanf com o código %s para ler uma string através da entrada padrão.
- Para isso devemos fornecer o endereço de memória onde a string deve ser armazenada.
  - O endereço de onde deve-se iniciar o armazenamento da string.

```
char str[20];
char str1[20];
scanf("%s", str);
scanf("%s",&str1[0]);
```

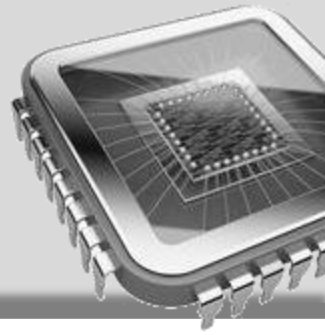
String
Como Escrever



- Podemos utilizar a função printf com o código %s para ler uma string através da entrada padrão.
- Devemos forencer o endereço de memória onde a string está armazenada
  - O endereço de onde encontra-se armazenado o primeiro caractere da string.

```
char str[20];
char str1[20];
printf("String str: %s \n", str);
printf("String str1: %s \n",&str1[0]);
```

String
Função gets

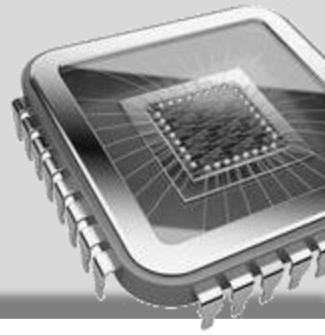


- A função gets lê uma string do teclado. Sua forma geral é:

gets(<string>);

```
char nome[100];
printf(“Nome: ”);
gets(nome);
printf (“\n Seja bem vindo %s! \n”,nome);
```

String
Função puts

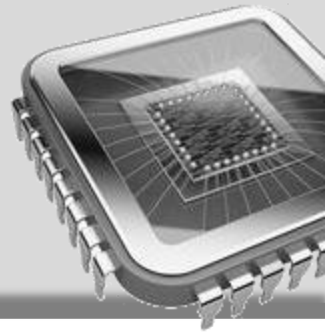


- A função puts escreve uma string na saída padrã. Sua forma geral é:

puts(<string>);

```
char nome[100];
printf(“Nome: ”);
gets(nome);
puts (“\n Seja bem vindo”);
puts(nome);
puts(”!\n”);
```

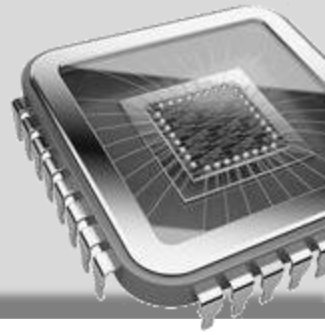
String
Função strcpy



- A função strcpy copia o conteúdo de uma string para outra. Sua forma geral é:

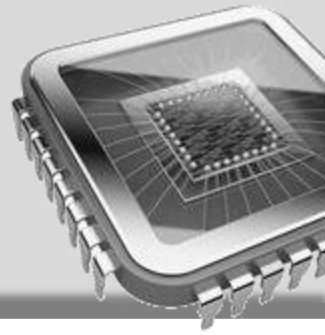
strcpy(<string_nova>,<string_original>);

String
Função strcpy



```
char nome[100];
char nome1[100];
printf(“Nome: ”);
gets(nome);
Strcpy(nome1,nome);
puts (“\n Você digitou: ”);
puts(nome);
puts(”\n Cópia: ”);
puts(nome1);
puts(”\n”);
```

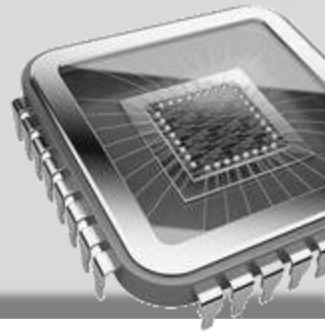
String
Função strlen



- A função strlen retorna o comprimeiro da string fornceida. O terminador nulo não é contato. Sua forma geral é:

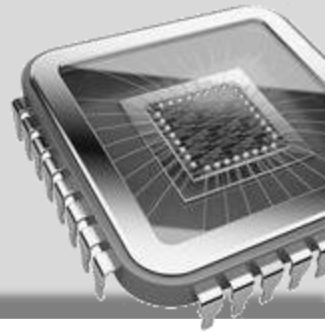
strlen(<string>);

String
Função strlen



```
char nome[100];
int tamanho;
printf(“Nome: ”);
gets(nome);
Tamanho = strlen(nome);
printf(“\n A string tem tamanho %d”, tamanho);
```

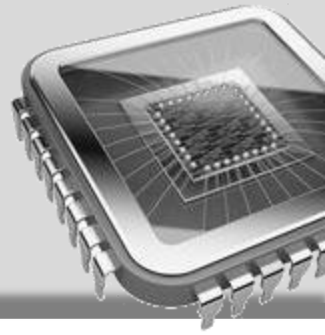
String
Função strcat



- A função strcat concatena a string_destino com a string_origem. A string_origem permanencerá inalterada e será anexada ao final da string destino. Sua forma geral é:

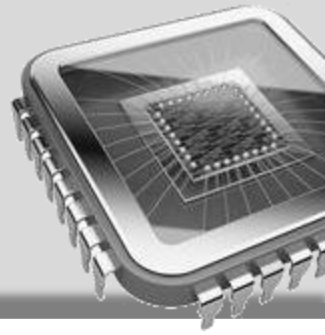
strcat(<string_destino>, <string_origem>);

String
Função strcat



```
char nome[100];
char sobrenome[100];
Char nomeCompleto[100];
printf(“Nome: ”);
gets(nome);
printf(“Sobrenome: ”);
gets(sobrenome);
strcpy(nomeCompleto,nome);
Strcat(nomeCompleto,sobrenome);
printf(“\n %s”, nomeCompleto);
```

String
sprintf e sscanf

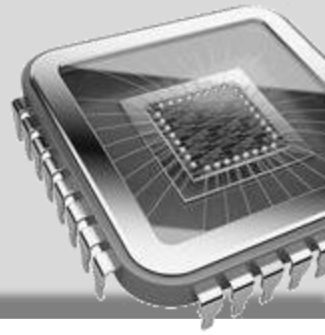


- As funções sprintf e sscanf são semelhantes a printf e scanf. Porém, ao invés de escreverem na saída padrão ou lerem da entrada padrão, escrevem e leem em uma string. Suas forma geral é:

sprintf(<string_dest>, <string_cont>,<lista_argumentos>);

sscanf(<string_orig>,<string_cont>,<lista_arqgumentos>);

String
sprintf e sscanf



Essas funções são muito utilizadas para fazer a conversão entre dados na forma numérica e sua representação na forma de strings e vice-versa.

```
int i;
char string[20];
printf(“Digite um inteiro: ”);
scanf(“%d”,&i);
sprintf(string,”valor de i - %d”, i);
puts(string);
```

```
int i, j; float w, k;
char string[] = “1 1.5 2 2.5”;
sscanf(string,“%d %f %d %f”,
        &i, &w, &j, &k);
printf(”valores: %d %f %d %.2f”,
        i, w,  j, k);
```