

Sistemas Operacionais I

Laboratório 03

Linguagem C (Tipos Agregados Homogêneos e String)

Agenda

- Vetores
- Matrizes
- String

Variáveis Compostas

- São um conjunto de variáveis identificadas por um **mesmo nome**.
- **Homogêneas** (vetores e matrizes)
- **Heterogêneas** (estruturas)

Variáveis Compostas Homogêneas

- Correspondem a posições da memória:
 - identificadas por um **único nome**
 - individualizadas por **índices**
 - cujo conteúdo é de um **mesmo tipo**

Notas:

6,1	2,3	9,4	5,1	8,9	9,8	10	7,0	6,3	4,4
-----	-----	-----	-----	-----	-----	----	-----	-----	-----

Posição:

0 1 2 3 4 5 6 7 8 9

Variáveis Compostas Homogêneas
:: Exemplo

		Posição do livro				
		0	1	2	...	n-1
Prateleira	0	788	598	265	...	156
	1	145	258	369	...	196
	2	989	565	345	...	526
	⋮	⋮	⋮	⋮	⋮	⋮
	m-1	845	153	564	892	210

Arranjos unidimensionais

- Utilizados para armazenar conjuntos de dados cujos elementos podem ser endereçados por um único índice.
- Também são conhecidos como vetores.

Arranjos multidimensionais

- Utilizados para armazenar conjuntos de dados cujos elementos necessitam ser endereçados por mais de um índice.
- Também são conhecidos como arrays ou matrizes.

Arranjos multidimensionais :: Exemplos

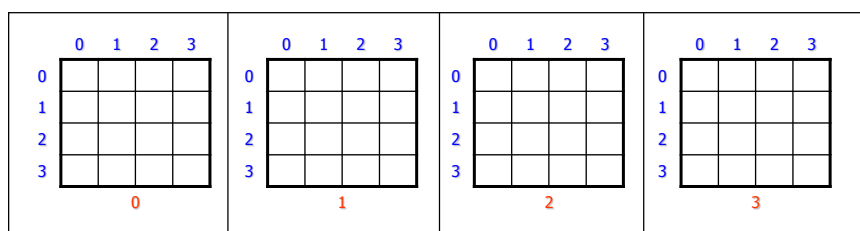
- Arranjos de 2 dimensões

	0	1	2	...	n-1
0	788	598	265	...	156
1	145	258	369	...	196
2	989	565	345	...	526
⋮	⋮	⋮	⋮	⋮	⋮
m-1	845	153	564	892	210

Arranjos multidimensionais

:: Exemplos

- Arranjo de 3 dimensões



Vetores e Matrizes na Linguagem C

- Como declarar:

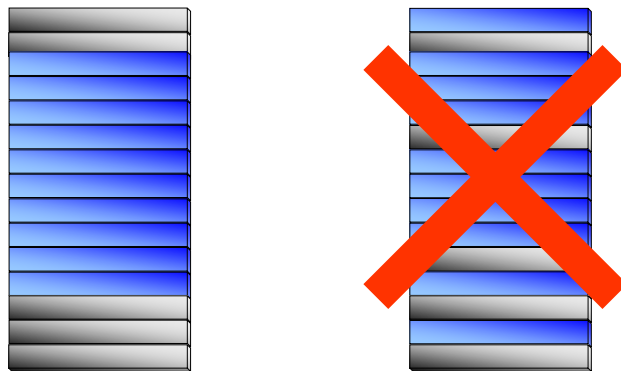
```
<tipo> <nome> [<tamanho1>][<tamanho2>]...;
```

1

```
float VetReais[100];
int Vetor[5][9];
char Nome_cliente[50];
float cubo[20][12][7];
```

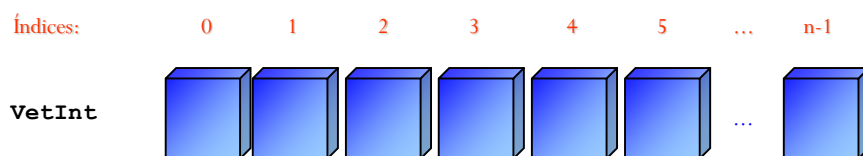
Vetores e Matrizes na Linguagem C

- O compilador C aloca uma porção **contígua** da memória para armazenar os elementos das matrizes e vetores.



Vetores e Matrizes na Linguagem C

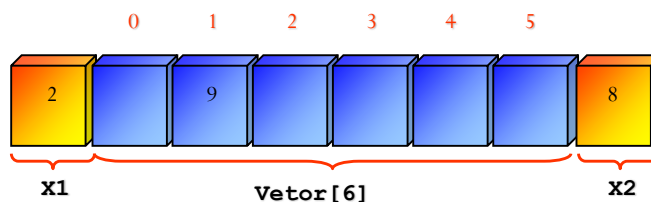
```
int VetInt[n];
```



Índice do **primeiro** elemento: zero
 Índice do **último** elemento: $n - 1$
 Quantidade de elementos: n

Vetores e Matrizes na Linguagem C

- Índices fora dos limites podem causar comportamento **anômalo** do código.



```
int x1;
int Vetor[6];
int x2
```

```
Vetor[1] = 9;
Vetor[-1] = 2;
Vetor[6] = 8;
```

Vetores e Matrizes na Linguagem C

- O tamanho de um vetor ou matriz é **pré-definido**, ou seja, após a compilação, não pode ser mudado.
- Portanto, vetores e matrizes são chamadas **estruturas de dados estáticas**, pois mantêm o **mesmo tamanho** ao longo de toda a execução do programa.

Vetores e Matrizes na Linguagem C

:: Exemplos

- Atribuir valores na declaração do vetor:

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

- Atribuir valores na declaração da matriz:

```
float matriz[2][3] = {{1,2,3},{4,5,6}};
```

Vetores e Matrizes na Linguagem C

:: Exemplos

- Colocar os números de 1 a 5 num vetor:

```
for (i=0; i<5; i++)  
    Vetor[i] = i + 1;
```

- Colocar os números de 5 a 1 num vetor:

```
for (i=0; i<5; i++)  
    Vetor[i] = 5 - i;
```


Vetores e Matrizes na Linguagem C :: Exemplos

- Preencher uma matriz $n \times m$ com zeros:

```
for (i=0; i < N; i++)  
for (j=0; j < M; j++)  
    Matriz[i][j] = 0;
```

Vetores e Matrizes na Linguagem C :: Exemplos

- Copiar dados de um vetor para outro:

```
#define TAM_MAX 10  
double VetReais[TAM_MAX], VetCopia[TAM_MAX];  
for (i=0; i<TAM_MAX; i++)  
    VetCopia[i] = VetReais[i];
```

- Boa prática de programação:
 - Definir o tamanho de vetores com **constantes** flexibiliza a manutenção do código.

Vetores e Matrizes na Linguagem C :: Exemplos

- Leitura dos dados de um vetor:

```
for (i=0; i<TAM_MAX; i++)  
{  
    printf("Digite um número: ");  
    scanf("%f", &Vet[i]);  
}
```

Strings

- Uma string é um vetor de caracteres, cujo final é indicado com um **caractere nulo** (valor inteiro zero).
- O terminador nulo também pode ser escrito como `'\0'`.
- Ao definir uma string, deve-se levar em consideração, além do número de caracteres da string, o caractere nulo que termina a string.

Strings

:: **Leitura** a partir do teclado

- Função **gets ()**
 - Lê string até o primeiro enter
- Função **scanf ()**
 - Lê string até o primeiro espaço em branco

Strings

:: Atribuição de valores

```
char curso[15] = "Engenharia"; // Valido somente na
                                // declaracao!
```

```
char curso[15];

strcpy(curso, "Engenharia"); // Requer biblioteca
                             // string.h
```

Manipulação de caracteres

- A Linguagem C possui algumas funções especiais para análise e manipulação de caracteres.
- Tais funções estão definidas na biblioteca **ctype.h**
- A biblioteca **ctype.h** possibilita a manipulação de caracteres, não de strings inteiras.

Manipulação de caracteres

:: Funções **tolower** e **toupper**

- Função **toupper**
 - Converte seu argumento para uma letra **maiúscula**:

```
<var1> = toupper(var2);
```

- Função **tolower**
 - Converte seu argumento para uma letra **minúscula**:

```
<var1> = tolower(var2);
```

Manipulação de caracteres

:: Outras funções

Função	Testa se seu argumento é um
isalnum	caractere alfanumérico
isalpha	caractere alfabético
isascii	caractere ASCII (0 a 127)
iscntrl	caractere de controle (0-0x1F ou 0x7F)
isgraph	caractere imprimível na tela (não leva espaço em consideração)
isprint	caractere imprimível na tela (leva espaço em consideração)

Manipulação de caracteres

:: Outras funções

Função	Testa se seu argumento é um
islower	caractere minúsculo
isupper	caractere maiúsculo
ispunct	caractere de pontuação
isspace	caractere de espaço, tabulação
isdigit	caractere numérico (0-9)
isxdigit	dígito hexadecimal (0-9, a-f ou A-F)

Manipulação de strings

:: Biblioteca **string.h**

- A Linguagem C possui funções especiais para análise e manipulação de **strings**.
- Tais funções estão definidas na biblioteca **string.h**.
- A biblioteca **string.h** possibilita a manipulação de strings completas (sem considerar caractere a caractere).

Manipulação de strings

:: Biblioteca **string.h**

- **strcat(str1, str2)**
Concatena str2 ao final de str1
- **int tam = strlen(str1);**
Retorna o tamanho de str1

Manipulação de strings

:: Biblioteca **string.h**

- **int valor = strcmp(str1, str2);**
 - valor = 0, se str1 e str2 são iguais;
 - valor < 0, se str1 < str2;
 - valor > 0, se str1 > str2;
- **int valor = strcmpi(str1, str2);**

Mesmo que **strcmp**, mas não é sensível ao caso

Manipulação de strings

:: Biblioteca **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.

Manipulação de strings

:: Conversão para números

- As seguintes funções fazem parte da biblioteca `stdlib.h`

Função	Converte
<code>atoi(<str>)</code>	String em int
<code>atof(<str>)</code>	String em float
<code>itoa(<int>)</code>	Int em string

Exercícios

1. Armazene num vetor de 5 posições o salário de 5 pessoas. Se o salário for menor que 1000 reais, forneça um aumento de 10% e sobrescreva o valor antigo. Ao final, mostre a lista de salários atualizada.
2. Faça um programa que leia n valores reais. Armazene estes valores num vetor. Ao final, imprima a média aritmética destes valores.
3. Faça um programa que pergunte ao usuário o número de alunos a ser lido. O tamanho dos vetores será o número informado pelo usuário. Armazene num vetor as notas G1 destes alunos; num outro vetor, armazene as notas G2 destes alunos. Ambas notas, G1 e G2, são informadas pelo usuário. Calcule a média aritmética destes alunos e armazene num terceiro vetor. Ao final, mostre as 3 notas dos alunos.

Exercícios

1. Leia e armazene num vetor de 10 posições o nome dos alunos aprovados no vestibular para Ciência da Computação em 2007/1. A seguir, leia um nome de entrada e verifique se este aluno consta na listagem dos aprovados e mostre sua classificação (ordem de aprovação, representada pelo índice do vetor +1). Caso o nome informado não exista na lista dos aprovados, mostre uma mensagem 'Nome informado não consta na lista dos aprovados'.