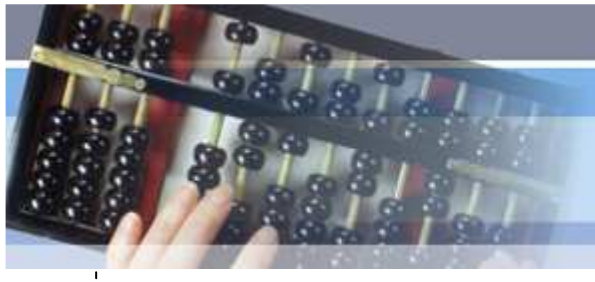
Curso de C

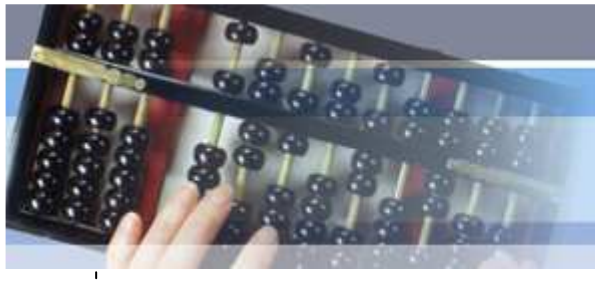
Vetores



Vetores

Roteiro:

- Declaração de vetores
- Uso correto de vetores
- Vetores com tamanho variável
- Matrizes
- Vetores de caracteres (texto)



Introdução

Tipos de Dados:

- Da linguagem C:
  - Números inteiros: `int`, `long int`, `unsigned int`, ...
  - Números fracionários: `float`, `double`, ...
  - Caracteres: `char`
- Do programador:
  - Vetores, estruturas, enumerações, etc
  - Sinônimos

Vetores

Declaração e Uso

Vetores

Conceitos:

- Seqüência de valores
- Todos do mesmo tipo
- Nome único para a variável
- Acesso por índice
- Tamanho fixo
- Numeração de 0 até *tamanho*-1
- Alocados sequencialmente na memória

int v[10]

v_0 (int)
v_1 (int)
v_2 (int)
v_3 (int)
v_4 (int)
v_5 (int)
v_6 (int)
v_7 (int)
v_8 (int)
v_9 (int)

Declaração

Declaração de vetor:

```
tipo  variavel  [tamanho]
```

- **tipo**: Qualquer tipo da linguagem C
- ***variavel***: Nome da variável
- **[tamanho]**: Número de elementos

Exemplo:

```
int vetor[10];
double medidas[100];
```

int v[10]

v_0 (int)
v_1 (int)
v_2 (int)
v_3 (int)
v_4 (int)
v_5 (int)
v_6 (int)
v_7 (int)
v_8 (int)
v_9 (int)

Acesso

Elementos com índice:

Elementos do vetor:

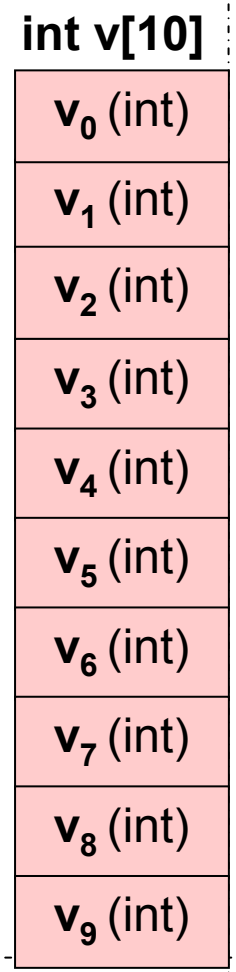
```
vetor[0], vetor[1], vetor[2], ...
```

Atribuição:

```
vetor[indice] = valor;
```

Exemplo:

```
int vetor[10];
vetor[5] = 3;
vetor[0] = vetor[1] + vetor[2];
```



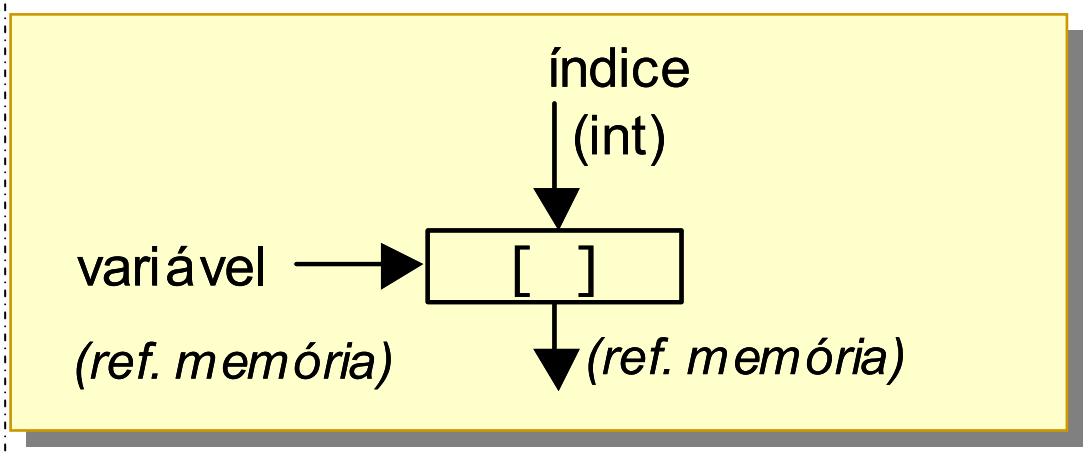
Acesso

Exemplo:

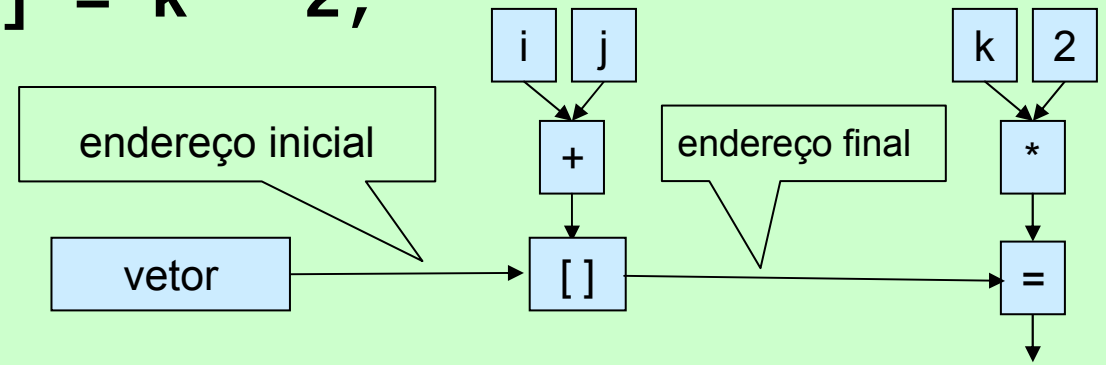
```
int vetor[100];
...
for (indice = 0;
     indice < 100;
     indice++) {

    printf("%d, ", vetor[indice]);

}
```

Acesso

Operador de índice:




```
vetor[i+j] = k * 2;
```



Acesso

Imprimir uma listra de trás para frente:

```
int main(int argc, char *argv[]) {
      int valores[10];
      int indice;
      printf("Escreva 10 números inteiros: ");
      for (indice = 0; indice < 10; indice++) {
            scanf("%d", &valores[indice] );
      }
      printf("Valores em ordem reversa:\n");
      for (indice = 9; indice >= 0; indice--) {
            printf("%d ", valores[indice]);
      }
      return 0;
}
```

Vetores\Reverso01\Reverso01.vcproj

Conteúdo inicial

Declaração com conteúdo inicial:

tipo vetor [n] = {$elem_0$, $elem_1$, ..., $elem_{n-1}$}

Tamanho do vetor

Lista de *n* valores

Exemplo:

```
int impares[5] = {1, 3, 5, 7, 9};
```

Conteúdo inicial

Declaração com conteúdo inicial :

```
tipo vetor [ ] = {elem0, elem1, ..., elemn-1}
```

Tamanho do vetor omitido

Lista de *n* valores

Exemplo:

```
int impares[] = {1, 3, 5, 7, 9};
```

Vetores

Regras para uso correto

Vetores

Cuidado com os índices:

Errado:

```
int vetor[10];
...
vetor[-2] = 3;
vetor[20] = 6;
```

Efeitos imprevisíveis!

Correto:

```
vetor[4] = 3;
vetor[7] = 6;
```

Vetores

Atribuir todos os valores:

Errado:

```
int vetor[10];
...
vetor = 0;
```

Correto:

```
int indice;
for (indice = 0; indice < 10; indice++) {
   vetor[indice] = 0;
}
```

Vetores

Copiar todos os valores:

Errado:

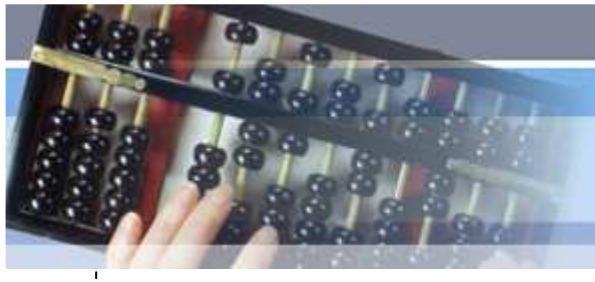
```
int vetorA[10], vetorB[10];
...
vetorA = vetorB;
```

Correto:

```
int indice;
for (indice = 0; indice < 10; indice++) {
   vetorA[indice] = vetorB[indice];
}
```

Vetores

Vetor de tamanho variável



Vetor de tamanho variável

Solução Simples:

```
int valores[100];
int numero_elementos;


int i;
for (i = 0; i < numero_elementos; i++) {
     printf("%d", valores[i]);
}
```

Limite superior para tamanho

Tamanho utilizado

Vetor de tamanho variável

Imprimir uma lista de números de trás para frente:

```
int main(int argc, char *argv[]) {
        int valores[100];
        int numero_valores;
        int i;

        printf("Quantos valores? (no máximo 100) ");
        scanf("%d", &numero_valores);
        if ( (numero_valores < 1)||(numero_valores > 100)) {
                printf(“Quantidade invalida!\n");
                return 1;
        }
...
```

Vetores

Imprimir uma lista de números de trás para frente:

```
...
        printf("Escreva os números: “);
        for (i = 0; i < numero_valores; i++) {
              scanf("%d", &valores[i] );
        }
        printf("Valores em ordem reversa:\n");
        for (i = numero_valores-1; i >= 0; i--) {
              printf("%d ", valores[i]);
        }
        printf(“\n”);
        return 0;
}
```

Vetores\Reverso02\Reverso02.vcproj

Vetores

Declaração de Matrizes

Matrizes

Conceitos:

- *Tabela* de valores
- Acesso por *dois* índices: linha e coluna
- Numeração: 0 até *linhas* – 1; 0 até *colunas* – 1

Outras propriedades:

- Valores do mesmo tipo
- Acesso aos valores através de um único nome de variável
- Dimensões fixas

```
int a[4][10];
```

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}	a_{06}	a_{07}	a_{08}	a_{09}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	a_{29}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}

>

Matrizes

Declaração:

```
tipo  variavel [linhas] [colunas]
```

- **tipo**: Qualquer tipo C
- ***variavel***: Nome da variável
- **[linhas]**: Número de linhas
- **[colunas]**: Número de colunas

Exemplo:

```
int matriz[4][10];
```

```
int a[4][10];
```

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}	a_{06}	a_{07}	a_{08}	a_{09}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	a_{29}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}

Matrizes

Acesso:

Elementos do vetor:

```
vetor[0][0], vetor[0][1], vetor[0][2], ...
vetor[linhas-1][colunas-1]
```

Atribuição:

```
vetor[linha][coluna] = valor;
```

Exemplo:

```
int vetor[6][10];
vetor[5][1] = 3;
vetor[0][2] = vetor[1][0] + vetor[2][3];
```

Matrizes

Acesso:

Exemplo:

```
int lin, col;
int matriz[4][10];
...
for (lin = 0; lin < 4; lin++) {
   for (col = 0; col < 10; col++) {
      printf("%d ", matriz[lin][col];
   }
   printf("\n");
}
```

Vetores multidimensionais

Vetor com *n* dimensões:

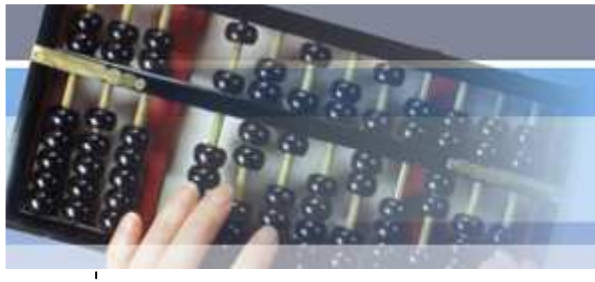
```
tipo  variavel [dim1] [dim2] ... [dimN] ;
```

Qualquer tipo C	Nome da variável	Dimensão 1	Dimensão 2	Dimensão *N*

Vetores

Vetor de caracteres

(Texto)



Texto

Conceitos:

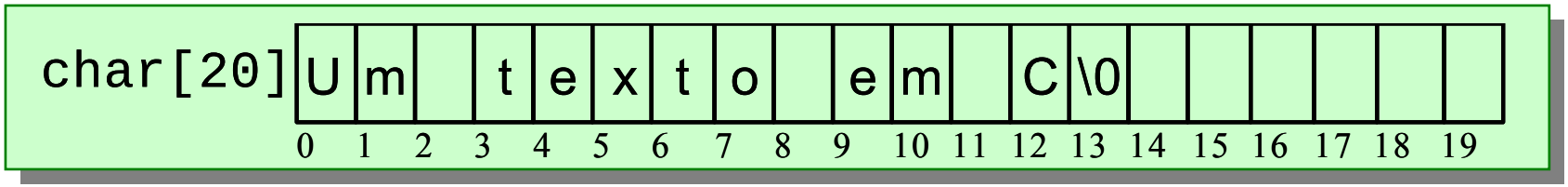
- Caracteres individuais:
  - `'a'`, `'A'`, `'f'`, `'4'`, `'.'` ← aspas simples!
  - Representam apenas um símbolo, letra ou dígito
- Texto:
  - Seqüência de caracteres
  - `"Algoritmos e Programação"`

aspas duplas!

Texto

Armazenamento:

- Vetor de caracteres
- Cada caractere do texto ⇒ um elemento do vetor
- Último caractere: nulo (‘\0’)
- Tamanho mínimo do vetor: comprimento do texto + 1



Texto

Declaração:

```
char  variavel [tamanho];
```

Tamanho máximo do texto menos um

Exemplo:

```
char texto[50];
```

Texto

Declaração:

```
char  variavel [tamanho]  = "texto";
```

Exemplo: `char texto[50] = "João da Silva";`

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

Exemplo: `char texto[] = “João da Silva“;`

Texto

Ler e imprimir:

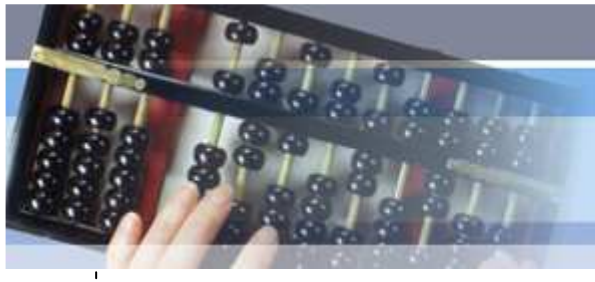
Impressão:

```
char texto[] = "Um texto em C";
printf("A variável: %s", texto);
```

Leitura:

```
char texto[20];
printf("Escreva uma palavra: ");
scanf("%s", texto);
```

OBS:
- Leitura por palavra
- Não usa &
- Cuidado com tamanho da variável



Texto

Atribuir à uma variável tipo texto:

Errado:

```
char texto[100];
...
texto = “João da Silva”;
```

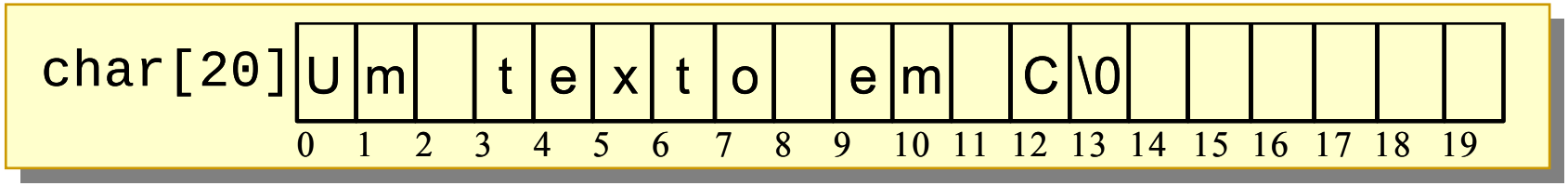
Correto:

```
#include <string.h>
...
strcpy(texto, “João da Silva”);
```

Texto

Acessar caracteres:

```
char texto[20] = "Um texto em C";
texto[12] = 'B';
printf("%s", texto);
```




```
Um texto em B
```


>

Vetores

Estatistica [calcula média e desvio padrão]
Palind [testa palindromos]
Primos [pegar primos, método da peneira]
Strings1 [teste com cadeias e funções sobre cadeias]
Concat [concatena cadeias]
Anagrama [testa anagramas]
SomaMat [soma de matrizes]
QuadradMag [testa quadrado mágico]

Vetores

Fim do Capítulo