

Vetores, Strings e Matrizes

Sumário

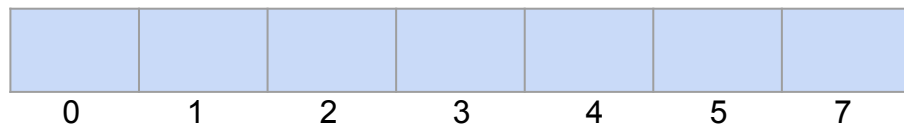
- Vetores
- Strings
- Matrizes

Sumário

- **Vetores**
- Strings
- Matrizes

Vetores

- Um vetor (*array*) é uma variável capaz de armazenar um conjunto de valores em uma área contígua de memória
 - quando declaramos um vetor, será alocada uma área contígua de memória baseada no tamanho do vetor
 - em C, todos os elementos ou posições do vetor armazenam o mesmo tipo de dado
 - cada posição é identificada por um índice numérico, começando por zero



Vetores

- Como declarar:

`tipo nome[tamanho];`

- Exemplo:

```
3  int main(){
4
5      int vetor[10];
6      float vet[5000];
7
8      return 0;
9  }
```

The diagram shows the same C code as the previous block, but with colored boxes highlighting parts of the declarations and callouts explaining them:

- A red box highlights the word `int` in `int vetor[10];`, with a callout bubble labeled "Tipo de dado" (Data type).
- A light blue box highlights the word `vetor` in `int vetor[10];`, with a callout bubble labeled "Nome do vetor" (Vector name).
- A light blue box highlights the number `10` in `int vetor[10];`, with a callout bubble labeled "Tamanho, ou número de posições" (Size, or number of positions).

```
3  int main(){
4
5      int vetor[10];
6      float vet[5000];
7
8      return 0;
9  }
```

Vetores

- Em C a contagem do índice inicia em 0 (primeiro elemento)
- O índice do último elemento será `tamanho - 1`
- O acesso é feito colocando-se o índice entre colchetes

```
3  int main(){
4      int vetor[4];
5
6      vetor[0] = 1;
7      vetor[1] = 1;
8      vetor[2] = 1;
9      vetor[3] = 1;
10
11     return 0;
12 }
```

```
3  int main()
4  {
5      int i;
6      int vetor[4];
7      for(i=0; i<4; i++){
8          scanf("%d", &vetor[i]);
9      }
10     return 0;
11 }
```

Vetores

- Em C a contagem do índice inicia em 0 (primeiro elemento)
- O índice do último elemento será **tamanho - 1**
- O acesso é feito colocando-se o índice entre colchetes

```
3  int main(){
4      int vetor[4];
5
6      vetor[0] = 1;
7      vetor[1] = 1;
8      vetor[2] = 1;
9      vetor[3] = 1;
10
11     return 0;
12 }
```

Acesso a posições individuais do vetor.

```
3  int main()
4  {
5      int i;
6      int vetor[4];
7      for(i=0; i<4; i++){
8          scanf("%d", &vetor[i]);
9      }
10     return 0;
11 }
```

Vetores

- O C não verifica se a posição acessada de um vetor é válida, porém não há garantias nos dados fora do intervalo declarado
 - Você pode acessar a posição 11 de um vetor de tamanho 10, mas nesta área haverá lixo.
- Um vetor pode ser inicializado das seguintes maneiras:

```
3  int main(){
4      int vetor[] = {1,2,3,4};
5
6      int vetor[4] = {1,2,3,4};
7
8      return 0;
9  }
```


Exercícios

1. Faça um programa que declara e inicializa um vetor com os seguintes dados: {1,2,3,3,4,6,6,7,5,7,1,3}. Em seguida, o programa deve imprimir todo o conteúdo do vetor na tela.
2. Crie um programa que lê 8 valores inteiros e armazene-os em um vetor. Na sequência, percorra o vetor substituindo cada valor pelo seu quadrado. Por fim, mostre na tela todos valores presentes no vetor.
3. Baseado no vetor lido na questão anterior, crie um código que imprima o maior e o menor valores contidos no vetor.

Sumário

- Vetores
- **Strings**
- Matrizes

Strings

- A linguagem C não possui um tipo string explícito.
- Uma string pode ser declarada como um vetor de caracteres:

```
char nome_string[tamanho];
```

- A string pode ser inicializada no momento da declaração:

```
char nome_string[tamanho] = "texto da string"
```

- Será colocado ao final da cadeia o caractere nulo, indicado por '\0'.

Strings

- Apesar de ser declarada como um vetor de caracteres, para ler ou imprimir uma string, basta utilizar **%s** (ou seja, não é necessário informar o índice)

```
1  #include <stdio.h>
2
3  int main(){
4      char string[20];
5      scanf("%s", string);
6
7      printf ("Você digitou: %s\n", string);
8      return (0);
9  }
```

Strings

- Apesar de ser declarada como um vetor de caracteres, para ler ou imprimir uma string, basta utilizar **%s** (ou seja, não é necessário informar o índice)

```
1  #include <stdio.h>
2
3  int main(){
4      char string[20];
5      scanf("%s", string);
6
7      printf ("Você digitou:
8      return (0);
9  }
```

Limitando o tamanho

Não se usa o operador de endereço **&** na leitura de strings

Strings

- Apesar de ser declarada como um vetor de caracteres, para ler ou imprimir uma string, basta utilizar **%s** (ou seja, não é necessário informar o índice)

```
1  #include <stdio.h>
2
3  int main(){
4      char string[255];
5      scanf("%s", string);
6
7      printf ("Você digitou: %s\n", string);
8      return (0);
9  }
```

Lê até o primeiro " " (espaço) ou ENTER

Para ler strings com mais de uma palavra, utilize gets(string) no lugar do scanf

Strings

- Para manipular strings em C utilizamos a biblioteca “string.h”
 - Ela apresenta uma série de funções úteis que permitem a manipulação de strings
 - Para utilizá-la, basta adicionar:
 - `#include <string.h>`
 - Funções úteis:
 - **strcpy(destino, origem)**: copia uma string **origem** para uma string **destino**;
 - **strcmp**: compara duas strings; retorna 0 se forem iguais, -1 se a primeira vier antes da segunda e 1 se a segunda vier antes da primeira;
 - **strlen(st)**: retorna o comprimento da string **st**.

Exercícios

4. Crie um programa que inverte uma string.
5. Crie um programa que calcula o tamanho de uma string (sem usar `strlen`).

Sumário

- Vetores
- Strings
- **Matrizes**

Matrizes

- Matrizes são vetores de duas dimensões, ou seja, um vetor onde cada uma das posições também é um vetor.
- O C permite a criação de matrizes de várias dimensões, porém, o mais usual é que uma matriz apenas possua duas dimensões.
- Assim como um Vetor, uma Matriz é um conjunto de valores de apenas um tipo.

	0	1	2	3	4	5	7
0							
1							
2							
3							

Matrizes

- Como declarar:

`tipo nome[nro_linhas][nro_colunas];`

- Exemplo:

```
3  int main(){
4      int matriz[10][10];
5      float matriz2[10][10];
6
7      int vetor[1][200];
8
9      return 0;
10 }
```

```
3  int main(){
4      int matriz[10][10];
5      float matriz2[10][10];
6
7      int vetor[1][200];
8
9      return 0;
10 }
```

The diagram illustrates the structure of array declarations in the example code. It uses callouts to identify the components of each declaration:

- Tipo** (Type): Points to the data type (e.g., `int`, `float`, `int`).
- Nome** (Name): Points to the variable name (e.g., `matriz`, `matriz2`, `vetor`).
- Linhas** (Rows): Points to the first dimension (number of rows) in the brackets (e.g., `10`, `10`, `1`).
- Colunas** (Columns): Points to the second dimension (number of columns) in the brackets (e.g., `10`, `10`, `200`).

Matrizes

- Os índices das linhas variam de 0 a `nro_linhas-1`
- Os índices das colunas variam de 0 a `nro_colunas-1`
- O acesso é feito colocando-se o índice da linha seguido do índice da coluna (ambos entre colchetes)

```
3  int main()
4  {
5      int matriz[3][3];
6      matriz[0][0] = 1;
7      matriz[0][1] = 1;
8      //...
9      matriz[2][2] = 1;
10
11     return 0;
12 }
```

```
3  int main()
4  {
5      int matriz[3][3];
6      int i, j;
7      for (i = 0; i < 3; i++){
8          for(j = 0; j < 3; j++){
9              scanf("%d", &matriz[i][j]);
10             }
11         }
12     return 0;
13 }
```

Matrizes

- Os índices das linhas variam de 0 a `nro_linhas-1`
- Os índices das colunas variam de 0 a `nro_colunas-1`
- O acesso é feito colocando-se o índice da linha seguido do índice da coluna (ambos entre colchetes)

```
3  int main()
4  {
5      int matriz[3][3];
6      matriz[0][0] = 1;
7      matriz[0][1] = 1;
8      //...
9      matriz[2][2] = 1;
10
11     return 0;
12 }
```

Acesso às posições
individuais da matriz

```
3  int main()
4  {
5      int matriz[3][3];
6      int i, j;
7      for (i = 0; i < 3; i++){
8          for(j = 0; j < 3; j++){
9              scanf("%d", &matriz[i][j]);
10          }
11      }
12     return 0;
13 }
```

Matrizes

- Uma matriz pode ser inicializada da seguinte forma:

```
3  int main(){
4      int matriz[3][3] = {
5          |   |   |   |   |   |   |   |   |   |   {00, 01, 02},
6          |   |   |   |   |   |   |   |   |   |   {10, 11, 12},
7          |   |   |   |   |   |   |   |   |   |   {20, 21, 22}
8          |   |   |   |   |   |   |   |   |   |   };
9
10     return 0;
11 }
```

Exercícios

6. Leia uma matriz 4x4 e imprima a diagonal principal.
7. Leia uma matriz 4x4 e escreva a localização (linha e a coluna) do maior valor.
8. Declare uma matriz 5x5. Preencha com 1 a diagonal principal e com 0 os demais elementos. Escreva ao final a matriz obtida.
9. Faça um programa que preenche uma matriz 5x5 com o produto do valor da linha e da coluna de cada elemento. Em seguida, imprima na tela a matriz.