3. Linguagem de Programação C

Prof. Renato Tinós

Departamento de Computação e Matemática
(FFCLRP/USP)

Principais Tópicos

3.5. Vetores

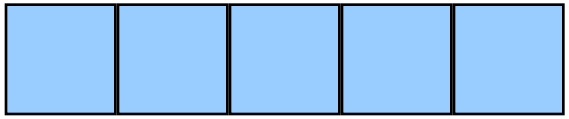
- 3.5.1. Vetores
- 3.5.2. Strings
- 3.5.3. Matrizes

3.5.1. Vetores

- **Definição: coleção de elementos com as seguintes características**
  - Contém dados homogêneos
    - Todo elemento armazenado em um mesmo vetor deve ser do mesmo tipo
      - Ex.: vetor de inteiros só pode ter elementos do tipo inteiro
  - Contém dados que podem ser ordenados
    - os elementos de um vetor podem ser organizados de uma forma pré-estabelecida

3.5.1. Vetores

- **Pode-se pensar em um vetor como uma seqüência de dados atômicos**



  - Os dados atômicos em um vetor são chamados de elementos

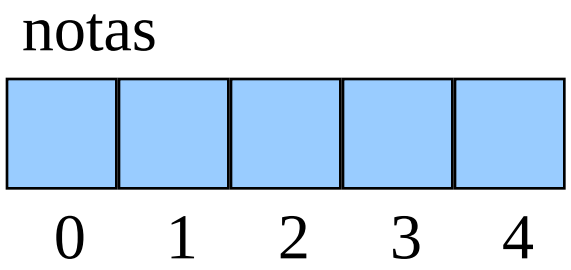
- **Vetor possui duas propriedades fundamentais**
  - Tipo de elemento
  - Tamanho do vetor

3.5.1. Vetores

- **Declaração**
  - *tipo nome[tamanho]*
    - Ex.: *int vetor[10];*
  - Tamanho do vetor pode ser especificado como uma constante
    - Facilita mudança do tamanho
    - Ex.: *#define NElementos 10*
      int vetor[NElementos];

3.5.1. Vetores

- **Nome**
  - É aconselhável que o nome do vetor indique que tipo de valor está sendo armazenado
    - Ex. *#define NJuizes 5*
      double notas[NJuizes];



3.5.1. Vetores

- **Indexação**
  - Cada elemento de um vetor é identificado por um índice

Em C, o primeiro elemento tem índice igual a 0 e o último tem índice igual ao número de elementos – 1

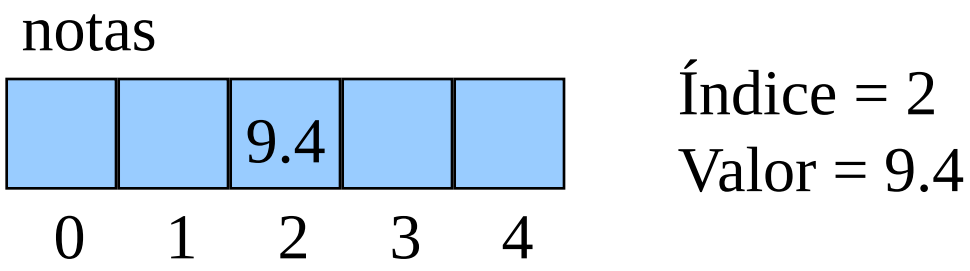
  - Exemplo: vetor de 4 elementos possui os índices:
    - » 0, 1, 2, 3

3.5.1. Vetores

- **Para se referir a um elemento específico de um vetor, devem ser fornecidos**
  - **Nome** do vetor
  - **Índice** correspondente à posição do elemento dentro do vetor
    - Ex.: A nota do segundo juiz é dada por *notas[1]*

3.5.1. Vetores

- **Expressão com seleção**
  - Funciona como uma simples variável
    - Ex.: nota[2] = 9.4;
  - É importante distinguir entre índice de um elemento e valor de um elemento



3.5.1. Vetores

- **Expressão com seleção**
  - É possível mudar os valores em um vetor, mas nunca o seu tamanho
  - Valor do índice não precisa ser uma constante
    - Pode ser qualquer expressão cujo resultado é um tipo escalar (Ex.: int, short, long)

```
...
for (i = 0; i < NJuizes; i++){
        notas[i] = 0.0;
}
...
```

3.5.1. Vetores

- **Tamanho efetivo e tamanho alocado**
  - Tamanho do vetor deve ser constante
  - Muitas vezes não se sabe quantos elementos o vetor vai conter
    - Estratégias
      - Usar alocação dinâmica (será visto em outras disciplinas)
      - Declarar tamanho como o número máximo de elementos possível (tamanho alocado) e então declarar um inteiro para indicar número de elementos utilizados (tamanho efetivo)

3.5.1. Vetores

- **Inicialização de vetores**
  - Valores iniciais podem ser atribuídos a uma variável do tipo vetor quando da sua declaração
    - Ex.: *int digitos[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }*
    - Neste caso, o tamanho do vetor pode ser omitido
      - Ex.: *int digitos[ ] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }*
      - Compilador conta o número de inicializadores e reserva a mesma quantidade de elementos para o vetor
      - Facilita a manutenção do programa

3.5.1. Vetores

- **A Linguagem C não checa se você passou dos limites de um vetor**
  - Se passar do fim de um vetor, você pode:
    - Escrever em cima dos conteúdos das posições nos extremos do vetor
    - Escrever no espaço reservado para outras variáveis
    - Escrever no espaço reservado para o código do programa

3.5.1. Vetores

```
/* Programa: Vetores*/
# include <stdio.h>


main(){
    int val[100];
    int i;

    for (i = 0; i < 100; ++i){
        val[i] = i;
    }
}
```

3.5.1. Vetores

Exercício 3.5.1. Escreva um programa em C em que o usuário possa entrar com o tamanho de um vetor e com seus elementos e que imprima os valores e os índices do maior e do menor elemento.

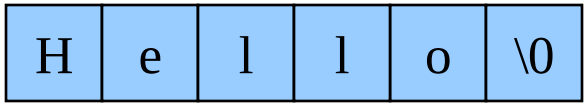
Exercício 3.5.2. Escreva um programa em C em que o usuário possa entrar com os elementos de dois vetores (colunas) de mesmo tamanho e que imprima o valor do produto interno entre eles.

3.5.1. Vetores

Exercício 3.5.3. Escreva um programa em C em que o usuário possa entrar com os elementos de um vetor e que imprima o valor da norma euclidiana deste vetor.

3.5.2. Strings

- ***Strings* são representados internamente como vetores de caracteres**
  - Caracteres são armazenados em bytes consecutivos
  - Final de *string* é representado por ‘\0’
    - Ex.: Compilador C reserva 6 bytes para o string “Hello”
    - Declaração: char hello = {‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’};
      ou char str[6] = “Hello”;



3.5.2. Strings

- **Como *strings* são vetores, elementos individuais podem ser selecionados e manipulados**

```
...
    int i, n_espacos;

    n_espacos = 0;
    for (i = 0; str[i] != '\0'; i++){
        if (str[i] == ' ')
            n_espacos++;
    }
...
```

3.5.2. Strings

- **Comando gets( )**
  - Permite entrar com uma *string* via teclado
  - Faz a leitura dos caracteres até que a tecla ENTER seja pressionada
    - A tecla ENTER não é armazenada.
    - Em seu lugar, o terminador nulo é armazenado

```
/* Programa: funcao gets*/
# include <stdio.h>

main(){
   char nome[100];

   printf("Digite o seu nome: ");
   gets(nome);
   printf("\n Ola %s  \n",nome);
}
```

3.5.2. Strings

- **Bibliotecas de operações sobre *strings***
  - Exporta operações que permitem a manipulação de *strings*
  - Interface ANSI *string.h* para manipular *strings*
    - Biblioteca padrão da linguagem C
    - Fornece um conjunto de operações avançadas
    - Permite trabalhar com o string inteiro utilizando uma simples chamada de função

3.5.2. Strings

- Funções mais comuns de *string.h* :

Nome	**Função**
strcpy (s1, s2)	Copia s2 em s1
strcat (s1, s2)	Concatena s2 ao final de s1
strlen (s1)	Retorna o tamanho de s1
strcmp (s1, s2)	Retorna 0 se s1 ==s2; menor que 0 se s1<s2; maior que 0 se s1>s2
strchr (s1, ch)	Retorna um ponteiro para a primeira ocorrência de ch em s1
strstr (s1, s2)	Retorna um ponteiro para a primeira ocorrência de s2 em s1

Exemplo

```
/* Biblioteca string.h  */
# include <stdio.h>
#include <string.h>

main( )
{
     char s1[80], s2[80];

    printf(" Entre com a primeira palavra”);
    gets(s1);
    printf(“\n Primeira com a segunda palavra”);
    gets(s2);
     printf(“\n Comprimentos: %d %d\n", strlen(s1), strlen(s2));
     if(!strcmp(s1, s2)) printf ("As strings sao iguais \n");
     strcat(s1, s2);
     printf("%s \n", s1);
     strcpy(s1, "Isto eh um teste");
     printf ("%s \n", s1);
     if(strchr("alo”, “o”)) printf(”o esta  em alo \n");
     if(strstr(“alo  aqui", “alo")) printf(“alo  encontrado");
}
```

3.5.2. Strings

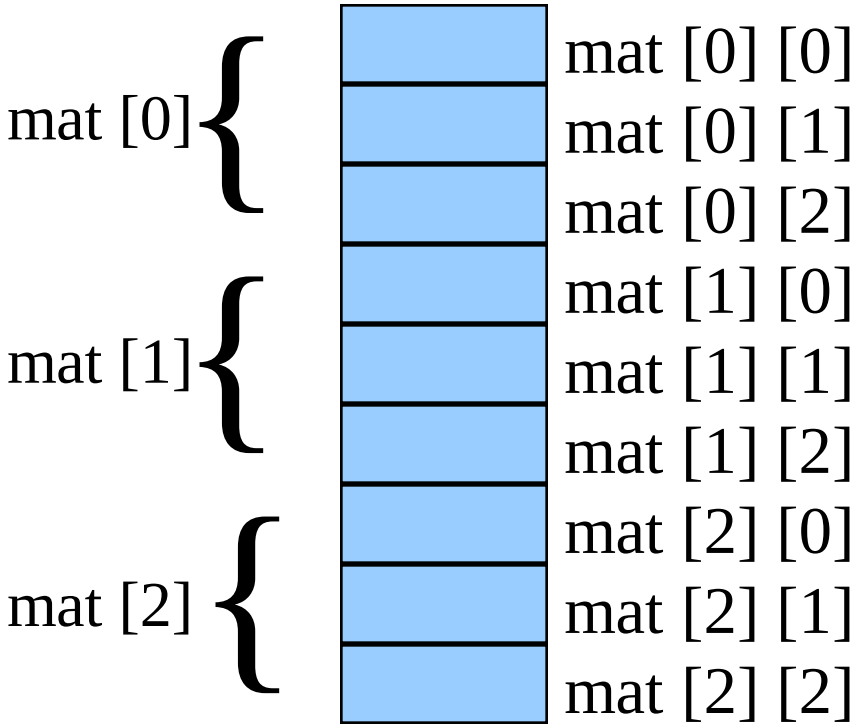
Exercício 3.5.4. Escreva um programa em C que receba um texto qualquer e conte o número de ocorrências da palavra "*de*" nesta frase.

3.5.3. Matrizes

- **São vetores multi-dimensionais**
  - Quando os elementos de um vetores são vetores
  - Ex.: *double mat [3][3];*

mat[0][0]	mat[0][1]	mat[0][2]
mat[1][0]	mat[1][1]	mat[1][2]
mat[2][0]	mat[2][1]	mat[2][2]

3.5.3. Matrizes



Internamente, C representa mat como um vetor de três elementos

Cada elemento é um vetor de três valores ponto-flutuantes

Na memória, estes nove valores formam uma lista unidimensional

3.5.3. Matrizes

- **Inicialização**
  - Como os vetores, podem ser inicializadas na declaração
    - Para enfatizar a estrutura geral, valores de cada vetor interno são inicializados entre chaves

1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0

```
double ident [3][3] = {
        {1.0,  0.0,  0.0 },
        {0.0,  1.0,  0.0 },
        {0.0,  0.0,  1.0 }
};
```

3.5.3. Matrizes

```
/* Programa: Matrizes  */
#include <stdio.h>

main() {
  int t, i, num[3][4];

  for(t=0; t<3; ++t) {
      for(i=0; i<4; ++i) {
            num[t][i] = (t*4)+i+1;
            printf("%3d ", num[t][i]);
      }
            printf("\n");
  }

}
```

3.5.3. Matrizes

Exercício 3.5.5. Escreva um programa em C em que o usuário possa entrar com os elementos de duas matrizes 3x3 e que imprima o valor da soma dos traços destas duas matrizes.

Exercício 3.5.6. Escreva um programa em C em que o usuário possa entrar com os elementos de uma matriz 2x2 e que imprima o valor do determinante desta matriz.