UNIVERSIDADE
FEDERAL DO CARIRI

Vetores

Professora Dra. Luana Batista da Cruz
luana.batista@ufca.edu.br

Roteiro

01 Estruturas de dados homogêneas

02 Vetores

01

Estruturas de dados homogêneas

Estruturas de dados homogêneas

- As **estruturas homogêneas** são conjuntos de dados formados pelo mesmo tipo de dado primitivo. Elas permitem agrupar diversas informações dentro de uma mesma variável

- Este agrupamento ocorrerá obedecendo sempre ao **mesmo tipo de dado**, e é por esta razão que estas estruturas são chamadas homogêneas

- A utilização deste tipo de estrutura de dados recebe diversos nomes, como: variáveis indexadas, variáveis compostas, arranjos, vetores, matrizes, tabelas em memória ou arrays

- Os nomes mais usados e que utilizaremos para estruturas homogêneas são: **matrizes** e **vetores** (matriz de uma linha e várias colunas)

02

Vetores

Declaração de um vetor
Uso de constante em vetor
Busca sequencial
Ordenação
String

Vetores

- Correspondem a posições de memória
- São identificados por um nome
- Individualizadas por índices
- Conteúdo do mesmo tipo

- Resumindo: vetores são **posições de memória** identificadas por um mesmo **nome**, individualizadas por **índices** e cujo conteúdo é do **mesmo tipo**

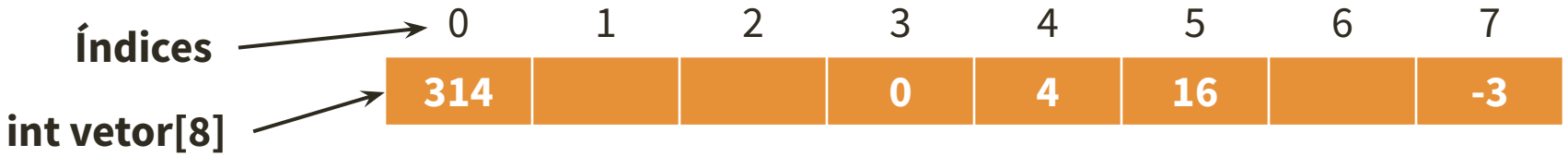
Declaração de um vetor

- <tipo> identificador [<número de posições>];
  - **Tipo**: int, float, double, etc
  - **Identificador**: é o nome da variável que identifica o vetor
  - **Número de posições**: é o tamanho do vetor

- **Exemplos**
  - int vetor[5];
  - double notas[50];
  - char palavra[20];

Declaração de um vetor

- É um arranjo de elementos armazenados em memória principal, um após o outro, todos utilizando o mesmo nome de variável



- Um vetor sempre começa com o índice de valor igual a zero e termina com o valor de seu tamanho menos um
- Um valor de um vetor pode ser acessado a partir de seu índice
  - Ex: vetor[0] = 314; printf(“%d”, vetor[0]);

Declaração de um vetor

- Ao declararmos um vetor, os seus elementos **não** são inicializados
- Mas é possível atribuir valores iniciais
- Os valores iniciais são colocados entre chaves

- **Exemplos**
  - int vetor[5] = {0, 2, 5, 3, 9};
  - double notas[5] = {0.0, 10.0, 7.5, 8.5, 9.9};

Declaração de um vetor

- **Importante**
  - A quantidade de valores entre chaves não deve ser maior que o número de elementos
  - A fim de facilitar a inicialização, C permite deixar o número de elementos em branco []
  - Neste caso, o compilador vai supor que o tamanho do vetor é igual ao número de valores especificados entre chaves

- **Exemplos**
  - int vetor[] = {0, 2, 5, 3, 9}; // tamanho = 5
  - double notas[] = {10.0, 9.5, 7.5}; // tamanho = 3

Declaração de um vetor

- **Diferentes forma de declarar um vetor**

```
// declaração sem inicializar os valores do vetor (eles terão 'lixo')
int v1[3];

// declaração inicializando os valores do vetor
int v2[3] = {0, 2, 5};

// declaração alternativa inicializando os valores do vetor
int v3[] = {0, 2, 5};
```

Vetores

- **Exemplo 1**

```c
#include<stdio.h>
#include<stdlib.h>

int v[3];

int main(){
  v[0] = 4;
  v[1] = 5;
  v[2] = 6;

  printf("V[0]= %d V[1]= %d V[2]= %d \n", v[0], v[1], v[2]);
  printf("Somatorio %d \n", v[0] + v[1] + v[2]);

  system("PAUSE");
  return 0;
}
```

Declaração de uma variável vetor: o número 3 (três) indica que o vetor terá três elementos

Atribuição em um elemento de um vetor: o vetor sempre começa com índice 0 (zero)

Vetores

- **Exemplo 2**: soma dos valores de um vetor de inteiro de 8 posições

```c
#include <stdio.h>
#include <stdlib.h>

int i, soma, v[8] = {1, 23, 17, 4, -5, 100, 4, 0};

int main(){

  for(i=0; i<8; i++)
    printf("V[%d]= %d \n", i, v[i]);

  soma = 0;
  for(i=0; i<8; i++)
    soma += v[i];

  printf("Soma = %d \n", soma);

  system("PAUSE");
  return 0;
}
```

Uso de constante em vetores

- **#define**
  - Constante usada para armazenar valores que **não** podem ser modificadas durante a execução de um programa

  - É **extremamente recomendável** utilizar letras maiúsculas ao declarar uma constante, porque podemos facilmente diferenciá-las das variáveis que por convenção devem ser declaradas em letras minúsculas

  - Quando o programa é compilado, o compilador **substitui as ocorrências das constantes** definidas pelo valor declarado

Uso de constante em vetores

- **Exemplo 3**: faça um programa que armazene em um vetor a idade de 5 pessoas digitada pelo usuário. Posteriormente, apresente o resultado da soma das idades e a média

Uso de constante em vetores

- **Exemplo 3**

```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 5

int i, idades[TAM];
float soma = 0, media = 0;

int main(){

  for(i=0; i<TAM; i++){
    printf ("Digite a %da. idade: \n", i+1);
    scanf ("%d", &idades[i]);
  }

  for(i=0; i<TAM; i++)
    soma += idades[i];

  media = soma/TAM;
  printf ("Soma = %.2f \n", soma);
  printf ("Media = %.2f \n", media);

  system ("PAUSE");
  return 0;
}
```

Foi definido para a variável **TAM** do define um tamanho igual a 5

Vetores

- **Exemplo 4**: faça um programa que retorna a média dos valores de um vetor de float de tamanho 10

Vetores

- **Exemplo 4**

```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 10

int i;
float soma=0, media, vetor[TAM];

int main(){

  for(i=0; i<TAM; i++){
    printf ("Digite um valor: ", i+1);
    scanf ("%f", &vetor[i]);
    soma = soma + vetor[i];
  }

  media = soma/TAM;
  printf ("Media = %.2f \n", media);

  system ("PAUSE");
  return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 10

int i;
float soma=0, media, vetor[TAM];

int main(){

  for(i=0; i<TAM; i++){
    printf ("Digite um valor: ", i+1);
    scanf ("%f", &vetor[i]);
  }

  for(i=0; i<TAM; i++)
    soma += vetor[i];

  media = soma/TAM;
  printf ("Media = %.2f \n", media);

  system ("PAUSE");
  return 0;
}
```

Vetores

- **Exemplo 5**: faça um programa que retorna o maior e menor número em um vetor de inteiros. O tamanho do vetor deve ser definido pelo usuário

Vetores

- **Exemplo 5**

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int qntd;
    printf("Digite a quantidade de elementos: ");
    scanf("%d", &qntd);

    int vetor[qntd], maior, menor;

    for (int i = 0; i < qntd; i++) {
        printf ("Digite um numero: ");
        scanf ("%d", &vetor[i]);

        if(i == 0){
            maior = vetor[i];
            menor = vetor[i];
        }else if(vetor[i] > maior)
            maior = vetor[i];
        else if(vetor[i] < menor)
            menor = vetor[i];
    }

    printf("Menor: %d \n", menor);
    printf ("Maior: %d \n", maior);

    system("PAUSE");
    return 0;
}
```

Busca sequencial

- **Exemplo**: busca sequencial de um número dentro de um vetor de tamanho 8

```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 8

int i, num, achou, v[TAM] = {1, 23, 17, 4, 5, 100, 6, 0};

int main(){
  printf ("Digite um numero:\n");
  scanf ("%d", &num);
  achou = 0;

  for(i=0; i<TAM; i++)
    if(v[i] == num){
      achou = 1;
      break;
    }

  if(achou)
    printf ("Numero %d encontrado!\n", num);
  else
    printf ("Numero %d nao encontrado!\n", num);

  system ("PAUSE");
  return 0;
}
```

Busca sequencial

- **Exemplo**: busca sequencial de um número dentro de um vetor de tamanho 8

Usando while

```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 8

int i, num, achou, v[TAM] = {1, 23, 17, 4, 5, 100, 6, 0};

int main(){
  printf ("Digite um numero:\n");
  scanf ("%d", &num);
  achou = 0;

  while(i<TAM && !achou){
    if(v[i] == num){
        achou = 1;
    }
    i++;
  }

  if(achou)
    printf ("Numero %d encontrado!\n", num);
  else
    printf ("Numero %d nao encontrado!\n", num);

  system ("PAUSE");
  return 0;
}
```

Ordenação

- **Exemplo**: ordene em ordem crescente um vetor de tamanho 6

```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 6

int i, j, aux, v[TAM] = {23,17,4,5,100, 6};

int main(){

    for( i = 0; i < TAM; i++ ){
        for( j = i + 1; j < TAM; j++ ){
            if( v[i] > v[j] ){
                aux= v[i];
                v[i] = v[j];
                v[j] = aux;
            }
        }
    }

    for( i = 0; i < TAM; i++ )
        printf("%d\n", v[i]);

    system("PAUSE");
    return 0;
}
```

String

- Uma string é um vetor (cadeia) de caracteres
- Em C, uma string é terminada com o caractere nulo (código igual a 0 em ASCII) e ele pode ser escrito como ‘\0’

String

- Uma string é um vetor (cadeia) de caracteres
- Em C, uma string é terminada com o caractere nulo (código igual a 0 em ASCII) e ele pode ser escrito como ‘\0’

```c
#include <stdio.h>
#include <conio.h>

#define MAX 20

int main(){

    char ch, frase[MAX];
    int i=0;

    printf("Informe a frase: ");
    while ( ((ch=getche()) != '\r') && (i < MAX - 1) ){
        frase[i]=ch;
        i++;
    }

    frase[i] = '\0';
    printf("\nFrase: %s", frase);

    system("pause");
    return 0;
}
```

‘\r’ representa a tecla enter relacionado a função **getche()**

String

- Uma string é um vetor (cadeia) de caracteres
- Em C, uma string é terminada com o caractere nulo (código igual a 0 em ASCII) e ele pode ser escrito como '\0'

string.h é a biblioteca que contém funções de manipulação de strings

strcpy é uma função que copia uma string para uma variável

Strings podem ser atribuídas diretamente na sua declaração

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main (){
  char frase[100] = "Estrutura de Dados I";
  printf ("Frase = %s\n", frase);
  strcpy (frase, "Estrutura de Dados II");
  printf ("Frase = %s\n", frase);

  system ("PAUSE");
  return 0;
}
```

Strings usam aspas duplas

String

- **gets()**
  - Lê uma string do teclado

```c
#include <stdio.h>
#include <stdlib.h>

int main (){
  char string[100];
  printf("Digite o seu nome: ");
  gets(string);
  printf("\n\n Ola %s",string);

  system("PAUSE");
  return 0;
}
```

scanf("%[^\n]", string);
Pega todos os caracteres até o \n
setbuf(stdin, NULL);
Limpa o buffer

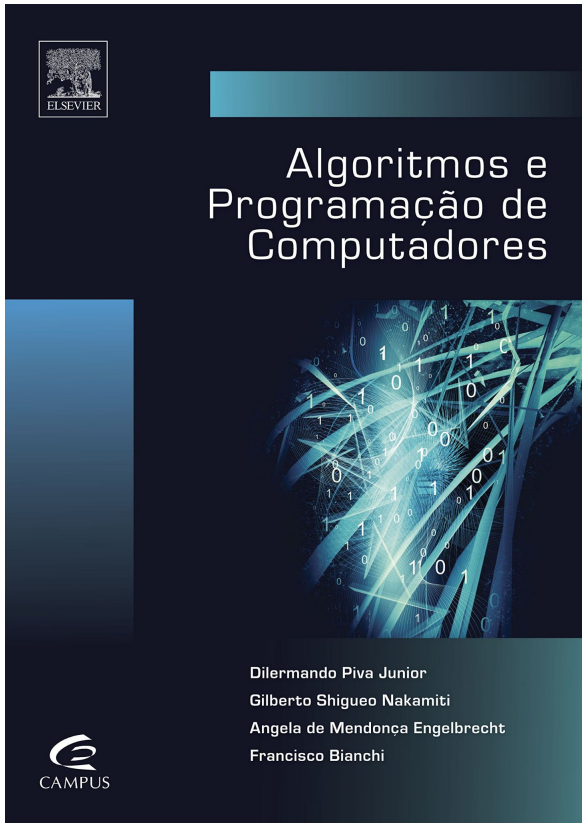
String

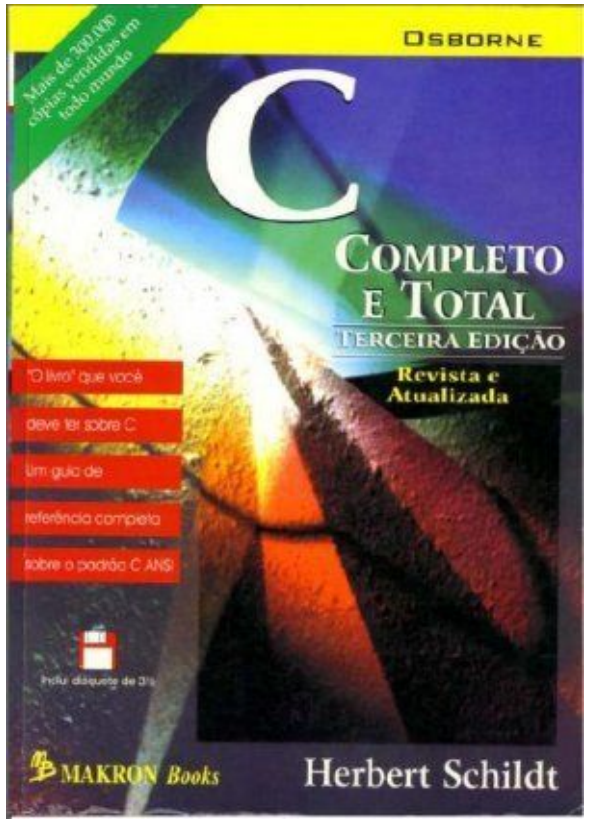
- **Alguns funções da biblioteca string.h**
  - strcmp: compara se duas strings são iguais
  - strcat: concatenação de duas strings, colocando uma ao final da outra
  - Atividade: procure uma função em C que ache uma string dentro de uma string

Resumindo..

- Estruturas de dados homogêneas
- Vetores
- Strings
- Exemplos

Referências





PIVA, D. J. et al. **Algoritmos e programação de computadores**. Rio de Janeiro, RJ: Elsevier, 2012.

SCHILDT, Herbert. **C completo e total**. Makron, 1997.