UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Vetores

Aula 09

Marcos Silvano Almeida
marcossilvano@professores.utfpr.edu.br
Departamento de Computação
UTFPR Campo Mourão

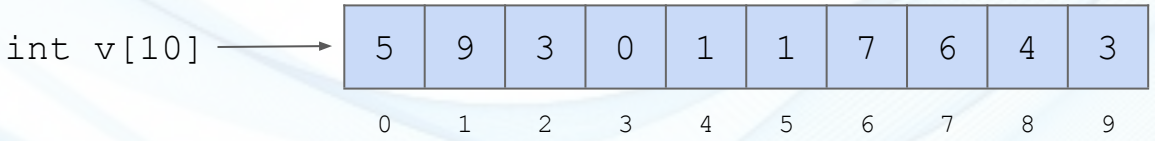
Roteiro

- Vetores
- Vetores como parâmetros de funções

- Adendo
  - Trabalhando com arquivos separados

Vetores | Arrays

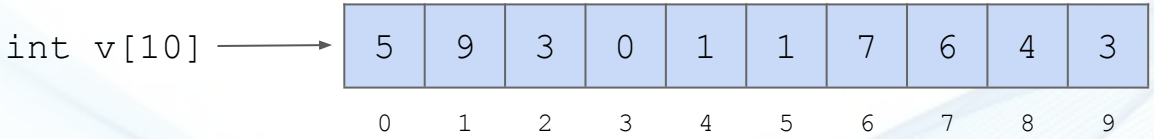
- Um vetor ou array, é uma estrutura de dados composta que permite armazenar uma **sequência** de **tamanho fixo** de valores do **mesmo tipo**
  - É o equivalente a termos uma sequência de variáveis do mesmo tipo



- O tamanho (quantidade de elementos) do vetor é definido em sua declaração.
  - `tipo nome_variável[tamanho];`
    - `int v[15];`
    - `float prices_list[50];`
    - `char vowels[5];`

Vetores | Arrays

- Considerando o vetor v:



- Em um vetor **v** de tamanho **n=10**:
  - Primeiro elemento **v**[0] ⇐ primeiro elemento está na posição 0
  - Último elemento **v**[9] ⇐ último elemento está na posição n-1

  - ❌ Inválido v[-4] ⇐ Índices negativos
  - ❌ Inválido v[12] ⇐ Acessar elementos foram do intervalo

Vetores :: *declarando e acessando*

```
int v1[6];                      << tamanho (fixo para toda execução)
                                   (posições contêm lixo de memória)
int v2[6] = {0};                << inicializa todas as posições com zero

int v3[] = {2,4,6,8,10};          << inicialização
v3[3] = 55;                       << {2,4,6,55,10}
int a = v3[1];                    << {2,4,6,55,10}
printf("3a posicao:%d", v3[2]);   << {2,4,6,55,10}

int v4[5]= {5,6,7,8,9};         << tamanho + inicialização

int n = read_int();
int v5[n];                      << tamanho varia pela execução do programa
```

❌ `v2 = {10,20,30,40,50};` << não é possível atribuição de vetor,
❌ `v1 = v2;` somente de seus elementos

UTFPR

Atividades

(1) Escreva um programa que declara um vetor de inteiros com 10 elementos, iniciados com dezenas (10, 20, …, 100). Em seguida, o programa deve:

a. Imprimir o primeiro elemento

b. Imprimir o último elemento

c. Substituir o valor do penúltimo elemento por 99

d. Imprimir o penúltimo elemento

Exemplos: acessando e manipulando vetores

Trabalhando com vetores

- Para trabalhar com vetores, os comandos de laço são essenciais

```c
#include <stdio.h>
int main() {
   printf("\nIMPRIME OS ELEMENTOS DE UM VETOR\n\n");

   int v[] = {1,2,3,4,5,6,7,8,9,10};// vetor de tamanho 10

   for (int i = 0; i < 10; i++) {   // i vai de 0 à 9
       printf(" %d", v[i]);          // usamos laço para percorrer vetor.
   }

   printf("\n");
   return 0;
}
```

Atividades

(2) Escreva um programa que declara um vetor de inteiros de tamanho 20, não inicializado. Depois, o programa deve:

a. Varrer o vetor, colocando valores múltiplos de 10 em cada posição.
Ex: 10, 20, 30, …

b. Imprimir o conteúdo do vetor em ordem reversa, isto é, do último para o primeiro elemento.

c. Modifique o programa para que o tamanho do vetor seja definido por uma variável **int n**, declarada previamente ao vetor. Escreva um código que permite ao usuário informar a quantidade de elementos do vetor, via terminal.

Exemplo :: *vetor com valores randômicos*

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
   int n;
   printf("\nVETOR DE VALORES ALEATORIOS\n ");
   printf("Tamanho > ");
   scanf("%d", &n);
   int v[n];                          // atribui valores aleatórios às posições do
vetor
   for (int i = 0; i < n; i++) { // i vai de 0 à n-1
       v[i] = rand() % 10 + 1;    // rand() sorteia valores de 1 à 10
   }
   for (int i = 0; i < n; i++) { // i vai de 0 à n-1
       printf(" %d", v[i]);
   }
   printf("\n\n");
   return 0;
```

Vetores como parâmetros de funções

Vetores como parâmetros de funções

- Vetores são sempre **parâmetros por endereço** em funções
  - O parâmetro aponta para o vetor passado como argumento à função
  - Qualquer alteração no vetor, dentro da função, refletirá no vetor passado

```
void vector_set(int n, int v[]) { // necessário passar o tamanho do vetor
    for (int i = 0; i < n; i++) {  // i vai de 0 à n-1
        v[i] = i+1;                // atribui valores de 1 à n
    }
}
int main() {
    int vec[10];                // não iniciado (contém lixo de memória)

    // função vector_set modifica os valores no vetor
    vector_set(10, vec);        // após a chamada, vec={1,2,3,4,5,6,7,8,9,10}

    return 0;
}
```

Vetores como parâmetros :: *3 opções de sintaxe*

```
void vector_set(int n, int* v) { // v terá o endereço do primeiro elemento do vetor
    for (int i = 0; i < n; i++)
        v[i] = i+1;
}

void vector_set(int n, int v[]) { // v terá o endereço do primeiro elemento do vetor
    for (int i = 0; i < n; i++)
        v[i] = i+1;
}

void vector_set(int n, int v[n]) { // v terá o endereço do primeiro elemento do vetor
    for (int i = 0; i < n; i++)
        v[i] = i+1;
}
```

Estas 3 formas de sintaxe têm o mesmo efeito.

UTFPR

Exemplo :: *funções definem operações sobre vetores*

```c
void vector_set(int n, int v[]) {
    for (int i = 0; i < n; i++) { // i vai de 0 à n-1
        v[i] = i+1;               // atribui valores de 1 à n
    }
}
void vector_print(int n, int v[]) {
    for (int i = 0; i < n; i++) { // i vai de 0 à n-1
        printf(" %d", v[i]);
    }
    printf("\n\n");
}
int main() {
    int vec[10];
    vector_set(10, vec);
    vector_print(10, vec);
    return 0;
}
```

Criamos funções para realizar operações específicas sobre vetores.

Atividades

(3) Escreva uma função para imprimir o conteúdo de um vetor de inteiros em ordem reversa. A função deve receber como parâmetros o tamanho e o vetor. Declare um vetor já inicializado na função main() e faça uma chamada à função criada para testá-la.

(4) Escreva uma função para imprimir o conteúdo de um vetor de inteiros. Ela deve imprimir somente os elementos que contenham valores positivos. A função deve receber como parâmetros o tamanho e o vetor. Declare um vetor já inicializado na função main() e faça uma chamada à função criada para testá-la.

ADENDO

Organizando o código em arquivos distintos

Criando e incluindo arquivos de código

- É comum organizarmos o código em módulos
  - Módulos, por sua vez, são colocados em arquivos distintos

- Vamos colocar alguns utilitários de vetores em um arquivo

```
void vector_set(int n, int* v, int step);
void vector_print(int n, int *v);
...
```

- Em C, nós devemos criar dois arquivos:
  - Arquivo com as implementações: extensão .c (source file | implementação)
  - Arquivo com as definições: extensão .h (header file | cabeçalho)

Arquivo vectors.c (implementação)

```c
#include <stdio.h>

void vector_set(int n, int* v, int step) {
    for (int i = 0; i < n; i++) {
        v[i] = step * (i + 1);
    }
}

void vector_print(int n, int *v) {
    for (int i = 0; i < n; i++) {
        printf("%d ", v[i]);
    }
    printf("\n");
}
```

Arquivo vectors.h (cabeçalho)

```c
#ifndef VECTORS_H
#define VECTORS_H
/**
* @brief Define o conteúdo do vetor como uma PA de razão step, iniciando em step.
*
* @param n Tamanho do vetor.
* @param v Endereço do vetor.
* @param step Razão da PA (e primeiro elemento).
*/
void vector_set(int n, int* v, int step);
/**
* @brief Imprime um vetor de n elementos.
*
* @param n Tamanho do vetor.
* @param v Endereço do vetor.
*/
void vector_print(int n, int *v);
#endif // VECTORS_H
```

Compilando e usando a biblioteca

- Para usar a biblioteca vectors.h, devemos incluí-la em nosso programa:

```
#include "vectors.h"
```

OBS:

> Quando o arquivo está na pasta local, usamos aspas duplas.
> Incluímos o arquivo de cabeçalho.

- Para compilar:

```
$ gcc main.c vectors.c        ⇇ com arquivo fonte .c

$ gcc -c vectors.c            ⇇ gera um binário da biblioteca, .o
$ gcc main.c vectors.o
```

Habitualmente, a bibliotecas são distribuídas compiladas (lib.o). De todo modo, é necessário prover o arquivo de cabeçalho (lib.h).

Referências

- Algoritmos e Programação
  - Marcela Gonçalves dos Santos
  - Disponível pelo Moodle

- Estruturas de Dados, Waldemar Celes e José Lucas Rangel
  - PUC-RIO - Curso de Engenharia
  - Disponível pelo Moodle

- Linguagem C, Silvio do Lago Pereira
  - USP - Instituto de Matemática e Estatística
  - Disponível pelo Moodle

- Curso Interativo da Linguagem C
  - https://www.tutorialspoint.com/cprogramming