Linguagem de Programação I

Aula 7 – Estruturas de Dados Homogêneas e Heterogêneas

Prof. Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Estruturas de Dados

Homogêneas e Heterogêneas

Visão Geral

▪ Homogêneas

Coleção de variáveis do **mesmo tipo**, armazenadas em posições **contíguas** na memória. Acessadas por um **índice**.

1. **Vetores** — unidimensionais
2. **Matrizes** — bidimensionais
3. Multidimensionais (não estudaremos)

▪ Heterogêneas (Structs)

Coleção de variáveis de **tipos diferentes**, agrupadas sob um mesmo nome — referenciando o mesmo "dono".

Permite armazenar, por exemplo, *nome* (char), *idade* (int) e *peso* (float) juntos em um único registro.

▪ Motivação

Armazenar as idades de 100 pessoas com variáveis separadas (`idade1`, `idade2` ... `idade100`) é impraticável. Estruturas de dados resolvem esse problema de forma elegante.

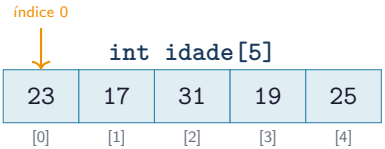
Vetores

Estruturas Homogêneas Unidimensionais

O que é um Vetor?

Definição

Um **vetor** (ou *array*) é uma estrutura que armazena uma sequência de objetos do **mesmo tipo** em posições **consecutivas** da memória RAM. Permite **acesso aleatório**: qualquer elemento pode ser acessado diretamente pelo seu índice. [Cormen et al. 2009]



Declaração

```
tipo nome[tamanho];
```

Exemplos:

```
int    idade[10];
```
10 inteiros

```
double notas[30];
```
30 reais

```
char   letras[40];
```
40 caracteres

Importante

Índices vão de **0** a **n−1**.
Se `n == 0`: vetor vazio.
Se `n == N`: vetor cheio.

Vetores — Exemplo

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

int busca (int x, int n, int v[]){
   int i;
   i = n-1;
   while (i >= 0 && v[i] != x)
      i -= 1;
   return i;
}

int main (void){
int i,j,aux;
int v1[10],v[]={8,3,4,7,6,5,1,2,9,0};


  for (i = 0; i < 10; ++i){
    printf("Vetor original v[%d] = %d\n", i, v[i]);
    v1[v[i]] = i;} //permutando aqui

  for (j = 0; j < 10; ++j) printf("Permutado v[%d] = %d\n", j, v1[j]);

  printf("Informe qual o item deseja encontrar no vetor original:");
  scanf("%d",&aux);

  i = busca(aux, 10, v);
  printf("Posição do item = %d\n", i);
}
```

Matrizes

Estruturas Homogêneas Bidimensionais

O que é uma Matriz?

Definição

Uma **matriz** é um vetor de vetores: organiza dados do mesmo tipo em **linhas e colunas**, acessados por **dois índices** [*linha*][*coluna*].

Declaração

```
tipo nome[linhas][colunas];
```

Exemplos:

```
int    notas[10][10];
double num[10][30];
char   letras[5][40];
```

`float notasAlunos[4][4]`

Aluno	P1	P2	P3	P4
José				
Liz				
Rosa				
Igor				

20

alunos $\times$ 4 notas = `float notas[20][4]`

Formas de inicializar uma matriz

```
// Por linhas explícitas:
float mat[4][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

// Sequencialmente:
float mat[4][3] = {1,2,3,4,5,6,7,8,9,10,11,12};

// Omitindo o número de linhas:
float mat[][3] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

Regra importante

O **número de colunas** deve ser sempre fornecido. O número de **linhas** pode ser omitido numa inicialização — o compilador o calculará automaticamente.

Matrizes — Exemplo: Notas de Alunos

```c
#include <stdio.h>
#define tamanho 5

void main(){
    double notaAluno[tamanho][4];
    for (int i = 0; i <= (tamanho-1); i++){
        for (int j = 0; j <= 3; j++){
            printf ("Informe Prova %d da %dª pessoa: ",j+1,i+1);
            scanf ("%lf", &notaAluno[i][j]);
        }
    }

    printf("Segue listagem de notas por aluno: \n");
    for (int i = 0; i <= (tamanho-1); i++){
        printf ("%dº aluno: ",i+1);
        for (int j = 0; j <= 3; j++){
            printf("%2.2f ",notaAluno[i][j]);
        }
        printf ("\n");
    }
}
```

Exercício: Distâncias entre Cidades

Problema

A tabela abaixo apresenta as distâncias (km) entre 5 cidades. Leia a tabela e permita ao usuário consultar a distância entre duas cidades. Encerre quando o usuário digitar 0.

	1	2	3	4	5
1	0	15	30	5	12
2	15	0	10	17	28
3	30	10	0	3	11
4	5	17	3	0	80
5	12	28	11	80	0

```c
#include <stdio.h>

void main(){
    int distancias[5][5];
    int c1,c2,i,j;

    for(i=0; i<=4; i++) distancias[i][i] = 0;
    distancias[0][1] = distancias[1][0] = 15;
    distancias[0][2] = distancias[2][0] = 30;
    distancias[0][3] = distancias[3][0] = 5;
    distancias[0][4] = distancias[4][0] = 12;
    distancias[1][2] = distancias[2][1] = 10;
    distancias[1][3] = distancias[3][1] = 17;
    distancias[1][4] = distancias[4][1] = 28;
    distancias[2][3] = distancias[3][2] = 03;
    distancias[2][4] = distancias[4][2] = 11;
    distancias[3][4] = distancias[4][3] = 80;

    do{
      printf("Informe as cidades que deseja buscar:");
      scanf("%d %d",&c1,&c2);

      if((c1!=0)&&(c2!=0))
          printf("A distância entre as cidades é %d\n",distancias[c1-1][c2-1]);

    }while((c1!=0)&&(c2!=0));
}
```

Distâncias — Código Completo

```c
#include <stdio.h>

void main(){
    int distancias[5][5] = {{00,15,30,05,12},
                            {15,00,10,17,28},
                            {30,10,00,03,11},
                            {05,17,03,00,80},
                            {12,28,11,80,00}};
    int c1,c2,i,j;

    do{
      printf("Informe as cidades que deseja buscar:");
      scanf("%d %d",&c1,&c2);

      if((c1!=0)&&(c2!=0))
          printf("A distância entre as cidades é %d\n",distancias[c1-1][c2-1]);

    }while((c1!=0)&&(c2!=0));
}
```

Estruturas Heterogêneas

Structs em C

O que é uma Struct?

▪ Definição

Permite agrupar variáveis de **tipos diferentes** sob um mesmo nome, fazendo referência a um mesmo "dono".

Em C, é possível criar **vetores e matrizes de structs**. [Schildt e Mayer 1997]

▪ Sintaxe

```
typedef struct nome_struct {
  tipo  campo1;
  tipo  campo2;
  ...
} alias;
```

▪ Struct aluno de academia

```
typedef struct alunoAcademia {
  char  nome[30];
  int   idade;
  double altura;
  double peso;
  char  end[50];
  char  tel[11];
} aluno;
```

▪ Acesso com .

```
aluno maria;
maria.idade = 23;
maria.peso  = 56.8;
```

Motivação: Por que usar Struct?

Sem Struct (problemático)

```
int idade[100];
float altura[100];
float peso[100];
char nome[100][30];
char end[100][50];
char tel[100][11];
```

Variáveis dispersas, difíceis de gerenciar. E se precisar de 200 pessoas?

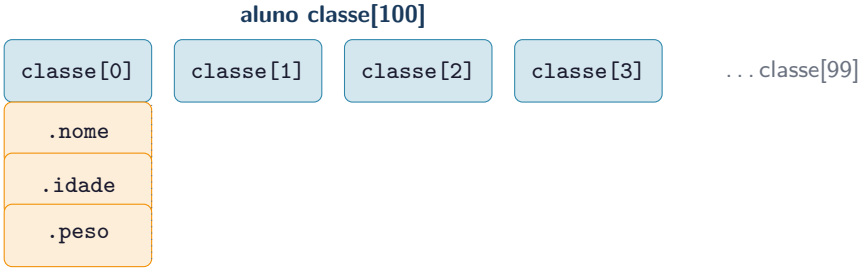
Com Struct (elegante)

```
aluno classe[100];

for(int i=0; i<100; i++) {
  scanf("%s", classe[i].nome);
  scanf("%d", &classe[i].idade);
  scanf("%f", &classe[i].peso);
  ...
}
```

Tudo agrupado. Fácil de escalar, ler e manter.

Motivação: Por que usar Struct?



Referências I

CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262032937.

SCHILDT, H.; MAYER, R. *C completo e total*. [S.l.]: Pearson University, 1997. ISBN 9788534605953.