Unidade 4

Estruturas de Dados Homogêneas

DEC0012 - Linguagem de Programação I
Curso de Engenharia de Computação
Profa. Andréa Sabedra Bordin

Definição

- Estruturas de dados **homogêneas** são estruturas que armazenam itens de dados do **mesmo tipo**.
  - Também chamada de **variável composta homogênea.**
- Essas estruturas são entidades "**estáticas**", já que permanecem do mesmo tamanho ao longo da execução do programa.
- Uma estrutura de dados homogênea é um **grupo de posições de memória consecutivas, todas de mesmo nome e mesmo tipo (tamanho)**.
- Podem ser do tipo **vetor** (*array* unidimensional) ou **matriz** (*array* multidimensional).

Vetor (*array* unidimensional)

c [0]	5
c [1]	4
c [2]	3
c [3]	6
c [4]	-3
c [5]	7
c [6]	8
c [7]	-2
c [8]	1
c [9]	9

- O vetor é uma estrutura de dados homogênea com apenas **1 dimensão (coluna).**
- Para criar um vetor de 10 elementos do tipo int precisamos reservar a memória para 10 elementos.
- Isso pode ser feito usando a declaração:
- int c[10];
- Os 10 elementos do vetor c são denominados c[0], c[1], c[2], . . . , c [9].

Vetor (*array* unidimensional)

- Para criar um vetor a de 10 elementos do tipo **int:**
  - int a[10];
- Podem ser declarados vetores de outros tipos:
  - float
  - double
  - char
- Também se usa um vetor do tipo **char** para armazenar um conjunto de caracteres (também chamado de *string*).

Vetor (*array* unidimensional)

```
int c[10];
//Atribuição de valor
c[0] = 5;
c[1] = 10;
// Impressão
printf(" %i ", c[0] );
```

Para fazer referência a uma posição particular ou elemento, especificamos o **nome da estrutura e o número da posição (índice) daquele elemento**

Vetor (*array* unidimensional)

- Operações com elementos do vetor:
- x = c[0] + 5;
- c[1] = y – 10;
- Para imprimir a soma dos valores contidos nos três
- primeiros elementos do vetor c podemos escrever:
- s = c[0] + c[1] + c[2] ;
- Para dividir por 2 o valor do sétimo elemento do vetor c e atribuir o resultado à variável x podemos escrever:
- x = c[6] / 2;

Vetor (*array* unidimensional)

Exemplo 1: Criação e Inicialização de um vetor

```
3   #include <stdio.h>
4
5   int main()
6   {
7     int c[10];
8     int i;
9
10    for ( i = 0; i < 10; i++ ) // inicializa o array
11      c[i] = 5;
12
13    printf("Elemento          Valor \n");
14
15    for ( i = 0; i < 10; i++ ) // imprime o array
16      printf(" %i               %i \n", i, c[i]);
17
18    return 0;
19  }
```

Vetor (*array* unidimensional)

- Os elementos de um vetor também podem ser inicializados na declaração do vetor na forma de uma lista de valores separados por vírgulas e entre chaves.

```
3   #include <stdio.h>
4
5   int main()
6   {
7     int n[15] = {11, 22, 33, 44, 55, 66, 77};
8     int i;
9
10    printf("Elemento     Valor \n");
11    for ( i = 0; i < 15; i++ ) // imprime o array
12      printf(" %i             %i \n", i, n[i]);
13
14    return 0;
15  }
```

Vetor (*array* unidimensional)

- Os valores dos elementos pode ser fornecidos pelo usuário.

```c
#include <stdio.h>

int main()
{
  int n[5];
  int i;

  for ( i = 0; i < 5; i++ ) // inicializa o array
   {
     printf("\n Digite elemento %i do vetor ", i);
     scanf("%i", &n[i]);
   }

  printf("\n Elemento      Valor \n");

  for ( i = 0; i < 5; i++ ) // imprime o array
    printf(" %i              %i \n", i, n[i]);

  return 0;
}
```

Vetor (*array* unidimensional)

- Somando os elementos do vetor.

```
 6    const int arraySize = 10; // tamanho do vetor
 7    float n[arraySize];
 8    float sum = 0;
 9    int i;
10  //-------------------------------------------------------
11    for ( i = 0; i < arraySize; i++ ) // inicializa o array
12     {
13       printf("\n Digite elemento %i do vetor ", i);
14       scanf("%f", &n[i]);
15     }
16  //-------------------------------------------------------
17    printf("\n Elemento      Valor \n");
18    for ( i = 0; i < arraySize; i++ ) // imprime o array
19      printf(" %i              %.2f \n", i, n[i]);
20  //-------------------------------------------------------
21    for ( i = 0; i < arraySize; i++ )
22     {
23       sum = sum + n[i];
24     }
25
26    printf("\n Soma dos elementos do vetor = %.2f \n\n", sum);
27
```

Vetor (*array* unidimensional)

- Programa que recebe as escolhas de 10 eleitores e calcula os votos para 5 candidatos.

```
 int candidato[5]; //declara o vetor de cinco posições
for (int i=0;i<5;i++)  // //inicializa o vetor com 0
        candidato[i] = 0;
 //entrada de dados
 int voto;
printf("Vote de 1 a 5");
for (int i=0;i<10;i++){
        do
            scanf("%d", &voto);
        while (voto <1 || voto > 5);
        candidato[voto-1] = candidato[voto-1] +  1;
 }
//saida de dados
 for (int i=0;i<5;i++)
        printf("Candidado %d teve %d votos \n", i+1, candidato[i]);

  return 0;
```

Matriz (*array* multidimensional)

- Matriz é uma estrutura de dados homogênea com **várias dimensões (colunas).**
- Em uma matriz pode ser armazenado um conjunto de itens de dado de mesmo tipo que são alocados sequencialmente na memória.
- Todos os dados possuem o mesmo **identificador** (mesmo nome). O que distingue cada dado é um **índice** que referencia sua localização dentro da estrutura.
- **Uma variável do tipo matriz precisa de um índice para cada uma das suas dimensões.**

Matriz (*array* multidimensional)

- Para identificar um elemento específico de uma matriz com duas dimensões (**bidimensional**), precisamos especificar dois índices:
  - o primeiro (por convenção) identifica a **linha** do elemento.
  - o segundo (por convenção) identifica a **coluna** do elemento.

	0	1	2
0	a[0][0]	a[0][1]	a[0][2]
1	a[1][0]	a[1][1]	a[1][2]
2	a[2][0]	a[2][1]	a[2][2]

Matriz (*array* multidimensional)

- As estruturas multidimensionais em C/C++ podem ter vários índices
- A quantidade dos índices pode chegar até 12 (dependendo da versão da linguagem).

Matriz (*array* multidimensional)

- Para criar uma matriz 2 x 2, i.e. uma matriz de 4 elementos do tipo int, podemos escrever:
- int m[2][2];
- Para declarar e ao mesmo tempo inicializar (i.e. atribuir os valores iniciais) podemos escrever:
- int m[2][2] = { { 1, 2}, { 3, 4} };
- Os valores são agrupados por linha e colocados entre chaves.
- Se não houver inicializadores suficientes para uma determinada linha, os elementos restantes daquela linha são inicializados com 0.

Matriz (*array* multidimensional)

- Atribuição de valor:
- m[0][0] = 5;
- m[1][0] = 10;
- Impressão do elemento m[0][0]:
- printf(" %i ", m[0][0]);

Matriz (*array* multidimensional)

- Operações com elementos de uma matriz:
- x = m[0][0] + 5;
- m[0][0] = 5 – 10;
- Para calcular a soma dos valores contidos na primeira **linha** da matriz m podemos escrever:
- soma_linha = m[**0**][0] + m[**0**][1];
- Para calcular a soma dos valores contidos na primeira **coluna** da matriz m podemos escrever:
- soma_coluna = m[0][**0**] + m[1][**0**];

Matriz (*array* multidimensional)

- Criação e inicialização de uma matriz 2x2

```c
// Exemplo 1: matrizes
#include <stdio.h>

int main()
{
  int i, j;
  int m[2][2] = { { 1, 2}, { 3, 4} };
  //int m[2][2] = { { 1, 2} };

  printf("\n Programa imprime uma matriz 2x2 com valores pre definidos: \n\n");
  for ( i=0; i<2; i++)
   {
     for ( j=0; j<2; j++)
          printf("%i   ", m[i][j] );
     printf("\n");
   }

  return 0;
}
```

Matriz (*array* multidimensional)

- Leitura de dados para uma matriz 2 x 2

```c
#include <stdio.h>

int main()
{
  const int line = 2, column = 2;
  int i, j;
  int m[line][column];
//------------------------------------------------------------------
 printf("\n Leitura de dados para matriz 2x2: \n\n");
 for ( i=0; i < line; i++)
   {
     for ( j=0; j < column; j++)
     {
       printf("\n Digite elemento [%i][%i] da matriz: ", i, j);
       scanf("%i", &m[i][j]);
     }
   }
//------------------------------------------------------------------
  printf("\n\n\n Matriz m: \n\n");
  for ( i=0; i < line; i++)
   {
     for ( j=0; j < column; j++)
          printf("%i   ", m[i][j] );
     printf("\n");
   }

  return 0;
}
```

Referências

- Adaptação do material de aula da profa. Olga Yevseyeva