



Estruturas de dados

Prof. Manassés Ribeiro
manasses.ribeiro@ifc.edu.br



Agenda

1. Estruturas de dados
2. Variáveis compostas homogêneas
 - a. unidimensionais (vetores)
 - i. Manipulação
 - b. bidimensionais (matrizes)
 - i. Manipulação
3. Variáveis compostas heterogêneas
 - a. declaração
 - b. manipulação



Estruturas de dados

Uma estrutura de dados é uma **coleção** tanto de **valores** (e seus relacionamentos) quanto de **operações** (sobre os valores e estruturas decorrentes). É uma **implementação concreta de um tipo abstrato de dado (TAD)** ou um tipo de dado primitivo.

O termo estrutura de dados também pode ser encontrado como sinônimo de TAD.



Variáveis compostas homogêneas

Quando uma determinada **estrutura de dados** é composta de variáveis do mesmo tipo primitivo de dados, tem-se um **conjunto homogêneo de dados**.

Neste primeiro momento, veremos dois tipos: variáveis compostas unidimensionais (**vetores**) e bidimensionais (**matrizes**)



Variáveis compostas unidimensionais (Vetores)



Variáveis compostas unidimensionais

É um tipo de **estrutura de dados** de uma única dimensão, que também são conhecidas como **vetores**.

Declaração:

```
tipo_de_dados identificador[tamanho_vetor];
```



Variáveis compostas unidimensionais

Exemplo:

```
int vet[10];
```

Vetor vet

0	1	2	3	4	5	6	7	8	9
20			35				65		



Manipulação

A **manipulação** dos vetores é realizada pelo seu **índice**. Cada “variável” é **independente e pode ser acessada por um índice** que no caso da linguagem de programação C, **inicia em zero**.

Para acessar um determinado dado é necessário saber em qual **vetor** o dado procurado se encontra (**identificador**), para então acessar a posição correspondente do vetor (**índice**).

Manipulação

Por exemplo, para
acessar o dado 27 deverá
ser acessado o vetor “c”
na posição 2: c[2]

	0	1	2	3	4	5	6	7	8	9
Vetor a	2	7	9	5	3	4	1	0	4	3
	0	1	2	3	4	5	6	7	8	9
Vetor b	3	2	3	3	3	2	5	8	3	6
	0	1	2	3	4	5	6	7	8	9
Vetor c	6	14	27	15	9	8	5	0	12	18



Manipulação

Ou seja, todas as manipulações de atribuição, leitura, etc, são realizadas da mesma forma das variáveis simples. Ex:

- `c[1] = 20;` *// Para atribuir o valor 20 no vetor “c” na posição 1*
- `scanf(“%i”, &a[5]);` *// Para ler um valor para o vetor “a” na posição 5*
- `printf(“%i”, b[6]);` *// Para escrever o valor correspondente do vetor “b” na posição 6*



Manipulação

Para realizar a **manipulação de vários dados** é necessário utilizar as estruturas de repetição.

Por exemplo, para carregar um vetor de inteiros com 50 valores diferentes, poderia ser utilizado uma estrutura de repetição.

```
#define TAM 50
```

```
int vet[TAM], j;
```

*Declaração do
vetor*



Manipulação

Para realizar a manipulação de **vários dados** é necessário utilizar as estruturas de repetição.

Por exemplo, para carregar um vetor de inteiros com 50 valores diferentes, poderia ser utilizado uma estrutura de repetição.

```
#define TAM 50
```

```
int vet[TAM], j;
```

*Declaração do
vetor*

```
for (j=0; j < TAM; j++){  
    printf("Digite um valor para vet[%i]: ", j);  
    scanf("%i", &vet[j]);  
}
```

*Procedimento
para carregar
um vetor*

Manipulação

Para realizar a manipulação de **vários dados** é necessário utilizar as estruturas de repetição.

Por exemplo, para carregar um vetor de inteiros com 50 valores diferentes, poderia ser utilizado uma estrutura de repetição.

```
#define TAM 50
```

```
int vet[TAM], j;
```

*Declaração do
vetor*

```
for (j=0; j < TAM; j++){
```

```
    printf("Digite um valor para vet[%i]: ", j);
```

```
    scanf("%i", &vet[j]);
```

```
}
```

*Procedimento
para carregar
um vetor*

```
for (j=0; j < TAM; j++){
```

```
    printf("%i", vet[j]);
```

```
}
```

*Procedimento
para escrever
um vetor*

Manipulação

Exemplo 2:

Encontrar, no vetor “vet”, e escrever os valores que são pares e maiores do que 20.

```
#define TAM 50
```

```
int vet[TAM], j;
```

*Declaração
do vetor*

```
for (j=0; j < TAM; j++){
```

```
    printf("Digite um valor para vet[%i]: ", j);
```

```
    scanf("%i", &vet[j]);
```

```
}
```

*Procedimento
para carregar
um vetor*

```
for (j=0; j < TAM; j++){
```

```
    printf("%i, ", vet[j]);
```

```
}
```

*Procedimento
para escrever
um vetor*

Manipulação

Exemplo 2:

Encontrar, no vetor “vet”, e escrever os valores que são pares e maiores do que 20.

```
#define TAM 50
```

```
int vet[TAM], j;
```

*Declaração
do vetor*

```
for (j=0; j < TAM; j++){
```

```
    printf("Digite um valor para vet[%i]: ", j);
```

```
    scanf("%i", &vet[j]);
```

```
}
```

*Procedimento
para carregar
um vetor*

```
for (j=0; j < TAM; j++){
```

```
    printf("%i, ", vet[j]);
```

```
}
```

*Procedimento
para escrever
um vetor*

```
for (j=0; j < TAM; j++){
```

```
    if ((vet[j] % 2 == 0) && (vet[j] > 20)) {
```

```
        printf("%i, ", vet[j]);
```

```
    }
```

```
}
```

*Procedimento para
encontrar os valores
pares e > 20*



Exercício

Faça um programa em C que **carregue e escreva** um vetor de inteiros. Além disso, o programa deverá **calcular a média dos valores do vetor**, escrevê-la, e escrever também os valores do vetor que sejam maiores do que a média calculada.



Variáveis compostas bidimensionais (Matrizes)



Variáveis compostas bidimensionais

É um tipo de estrutura de dados de duas dimensões, que também são conhecidas como **matrizes**.

Declaração:

```
tipo_de_dados identificador[tam_dim_1][tam_dim_2];
```

Exemplo:

```
#define LIN 30  
#define COL 30  
float mat[LIN][COL];
```



Manipulação

Assim como nos vetores, a manipulação das matrizes é realizada pelos seus **índices**, que agora são **dois índices para serem manipulados**, um para as linhas (eixo y) e outro para as colunas (eixo x).

Para acessar um determinado dado é necessário saber em qual **matriz** o dado procurado se encontra (**identificador**), para então acessar a posição correspondente pelos índices (linha e coluna).

Manipulação

Por exemplo, para acessar os valores **35** e **88** deve ser acessada a matriz “a” nas posições **(0,3)** e **(2,1)**, respectivamente: **$a[0][3]$** e **$a[2][1]$** .

Matriz a

	0	1	2	3	...	n
0	20			35		
1		66		10		
2		88	12			
3		2		31		
⋮						
n						



Manipulação

Para todas as **manipulações** de atribuição, leitura, etc, seguem a mesma forma de acesso dos vetores, considerando, portanto, as **duas dimensões** das matrizes. Ex:

- `a[0][0] = 20;` *// Para atribuir o valor 20 na matriz “a” na posição (0,0)*
- `scanf(“%i”, &a[2][3]);` *// Para ler um valor para o vetor “a” na posição (2,3)*
- `printf(“%i”, a[3][3]);` *// Para escrever o valor correspondente do vetor “a” na posição (3,3)*

Manipulação

Assim como nos vetores, para realizar a **manipulação das matrizes** é necessário utilizar as estruturas de repetição.

Por exemplo, para carregar uma matriz de inteiros de tamanho 10 x 10, poderia ser utilizado uma estrutura de repetição.

```
#define LIN 10
#define COL 10

int mat[LIN][COL], l, c;

for (l = 0; l < LIN; l++){
    for (c = 0; c < COL; c++){
        printf("Digite um valor para mat[%i][%i]: ", l, c);
        scanf("%i", &mat[l][c]);
    }
}

for (l = 0; l < LIN; l++){
    for (c = 0; c < COL; c++){
        printf("%i, ", mat[l][c]);
    }
}
```

Procedimento para carregar uma matriz

Procedimento para escrever uma matriz



Manipulação

Exemplo 2:

Encontrar, na matriz “mat”, e escrever os valores que são pares e maiores do que 20.

```
#define LIN 10  
#define COL 10  
int mat[LIN][COL], l, c;
```



Manipulação

Exemplo 2:

Encontrar, na matriz “mat”, e escrever os valores que são pares e maiores do que 20.

```
#define LIN 10
#define COL 10
int mat[LIN][COL], l, c;
for (l = 0; l < LIN; l++){
    for (c = 0; c < COL; c++){
        printf("Digite um valor para mat[%i][%i]: ", l, c);
        scanf("%i", &mat[l][c]);
    }
}
```

*Procedimento
para carregar
uma matriz*

Manipulação

Exemplo 2:

Encontrar, na matriz “mat”, e escrever os valores que são pares e maiores do que 20.

```
#define LIN 10
#define COL 10
int mat[LIN][COL], l, c;
for (l = 0; l < LIN; l++){
    for (c = 0; c < COL; c++){
        printf("Digite um valor para mat[%i][%i]: ", l, c);
        scanf("%i", &mat[l][c]);
    }
}
for (l = 0; l < LIN; l++){
    for (c = 0; c < COL; c++){
        printf("%i, ", mat[l][c]);
    }
}
```

Procedimento para carregar uma matriz

Procedimento para escrever uma matriz

Manipulação

Exemplo 2:

Encontrar, na matriz “mat”, e escrever os valores que são pares e maiores do que 20.

```
#define LIN 10
#define COL 10
int mat[LIN][COL], l, c;
for (l = 0; l < LIN; l++){
    for (c = 0; c < COL; c++){
        printf("Digite um valor para mat[%i][%i]: ", l, c);
        scanf("%i", &mat[l][c]);
    }
}
for (l = 0; l < LIN; l++){
    for (c = 0; c < COL; c++){
        printf("%i, ", mat[l][c]);
    }
}
for (l = 0; l < LIN; l++){
    for (c = 0; c < COL; c++){
        if ((mat[l][c] % 2 == 0) && (mat[l][c] > 20)) {
            printf("\n%i, ", mat[l][c]);
        }
    }
}
```

*Procedimento
para carregar
uma matriz*

*Procedimento
para escrever
uma matriz*

*Procedimento para
encontrar os valores
pares e > 20*



Exercício

Faça um programa em C que **carregue e escreva** uma matriz de inteiros. Além disso, o programa deverá **calcular a média dos valores da matriz**, escrevê-la, e escrever também os valores da diagonal principal que sejam maiores do que a média calculada.



Estruturas de dados: registros

Prof. Manassés Ribeiro
manasses.ribeiro@ifc.edu.br



Agenda

1. Estruturas de dados
2. Variáveis compostas homogêneas
 - a. unidimensionais (vetores)
 - i. Manipulação
 - b. bidimensionais (matrizes)
 - i. Manipulação

Aulas anteriores



Agenda

1. Estruturas de dados
2. Variáveis compostas homogêneas
 - a. unidimensionais (vetores)
 - i. Manipulação
 - b. bidimensionais (matrizes)
 - i. Manipulação
3. **Variáveis compostas heterogêneas (registro)**
 - a. **declaração**
 - b. **manipulação**

Aulas anteriores

Aula de hoje!



Recapitulando...

Uma estrutura de dados é uma coleção tanto de **valores** (e seus relacionamentos) quanto de **operações** (sobre os valores e estruturas decorrentes). É uma **implementação concreta de um tipo abstrato de dado (TAD)** ou um tipo de dado primitivo.

O termo estrutura de dados também pode ser encontrado como sinônimo de TAD.



Recapitulando...

Variáveis compostas homogêneas:

1. Quando uma determinada **estrutura de dados** é composta de variáveis do mesmo tipo primitivo de dados, tem-se um **conjunto homogêneo de dados**.
2. Neste primeiro momento, veremos dois tipos: variáveis compostas unidimensionais (**vetores**) e bidimensionais (**matrizes**)



Variáveis compostas heterogêneas (registro)

É uma das principais estruturas de dados (TADs), que é composta por um conjunto de informações logicamente relacionadas, porém de **tipos primitivos diferentes**.

O registro é uma variável **composta** (possui elementos que são partes que especificam cada uma das informações do todo) e **heterogênea** (cada elemento pode ser de um tipo primitivo de dados diferente).

TPD: inteiro

Exemplo: passagem de ônibus

Número da passagem: _____ Data: _____
De: _____ Para: _____
Horário: _____ Poltrona: _____ Idade: _____
Nome do passageiro: _____



Exemplo: passagem de ônibus

TPD: caracteres

Número da passagem: _____ Data: _____
De: _____ Para: _____
Horário: _____ Poltrona: _____ Idade: _____
Nome do passageiro: _____



Variáveis compostas heterogêneas (registro)

Sintaxe de declaração de registro em C:

```
struct identificador_registro {  
    tipo_de_dados identificador1;  
    tipo_de_dados identificador2[tam_dim_1];  
    tipo_de_dados identificador3[tam_dim_1][tam_dim_2];  
};
```

Variáveis compostas heterogêneas (registro)

Sintaxe de declaração de registro em C:

Ex. variável comum

```
struct identificador_registro {  
    tipo_de_dados identificador1;  
    tipo_de_dados identificador2[tam_dim_1];  
    tipo_de_dados identificador3[tam_dim_1][tam_dim_2];  
};
```

Variáveis compostas heterogêneas (registro)

Sintaxe de declaração de registro em C:

```
struct identificador_registro {  
    tipo_de_dados identificador1;  
    tipo_de_dados identificador2[tam_dim_1];  
    tipo_de_dados identificador3[tam_dim_1][tam_dim_2];  
};
```

Ex. variável comum

Ex. vetor

Variáveis compostas heterogêneas (registro)

Sintaxe de declaração de registro em C:

```
struct identificador_registro {  
    tipo_de_dados identificador1;  
    tipo_de_dados identificador2[tam_dim_1];  
    tipo_de_dados identificador3[tam_dim_1][tam_dim_2];  
};
```

Ex. variável comum

Ex. vetor

Ex. matriz



Variáveis compostas heterogêneas (registro)

Sintaxe de declaração de registro em C:

```
struct identificador registro{  
    tipo_de_dados identificador1;  
    tipo_de_dados identificador2[tam_dim_1];  
    tipo_de_dados identificador3[tam_dim_1][tam_dim_2];  
};  
  
struct identificador registro identificador_variavel;
```

Declaração para
uso do registro
(TAD)

Declaração de registro em C: exemplo

Registro de estoque de produto contendo os elementos **nome do produto**, **código**, **preço** e um elemento **baixa** (numérico) que indica as baixas do produto por dias da semana (vetor de 6 posições):

Nome: _____

Código: _____ Preço: _____

	1	2	3	4	5	6
Baixa:						

Exemplo em C

Nome:							
Código:				Preço:			
	1	2	3	4	5	6	
Baixa:	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	

```
struct sProduto{  
    int codigo;  
    char nome[50];  
    float preco;  
    int baixa[6];  
};
```

*Declaração
do registro
(struct) em C*

Exemplo em C

Nome:							
Código:				Preço:			
	1	2	3	4	5	6	
Baixa:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

```
struct sProduto{  
    int codigo;  
    char nome[50];  
    float preco;  
    int baixa[6];  
};
```

```
struct sProduto feijao;
```

*Declaração
do registro
(**struct**) em C*

*Declaração da
variável “feijao”
como o TAD
sProduto*



Manipulação de registros

A manipulação do registro acontece **acessando seus elementos internos** por meio de cada identificador individual, que é precedido pelo identificador do registro e um ponto (“.”). O ponto, neste caso, é o elo de ligação entre o registro e seus elementos.

Exemplo em C

Nome:							
Código:				Preço:			
	1	2	3	4	5	6	
Baixa:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

```
struct sProduto{  
    int codigo;  
    char nome[50];  
    float preco;  
    int baixa[6];  
};
```

```
struct sProduto feijao;
```

```
feijao.codigo = 1;
```

*Declaração
do registro
(**struct**) em C*

*Declaração da
variável "feijao"
como o TAD
sProduto*

Exemplo em C

Nome:							
Código:				Preço:			
	1	2	3	4	5	6	
Baixa:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

```
struct sProduto{  
    int codigo;  
    char nome[50];  
    float preco;  
    int baixa[6];  
};
```

Declaração
do registro
(*struct*) em C

```
struct sProduto feijao;
```

Declaração da
variável
“feijao” como o
TAD *sProduto*

```
feijao.codigo = 1;  
feijao.nome = “Feijão preto kg”;  
feijao.preco = 5.00;
```

Exemplo de
manipulação do TAD
sProduto

```
for (int i=0; i < 6; i++){  
    feijao.baixa[i] = 0;  
}
```

Exemplo em C

typedef

O comando **typedef** é usado para criar um “alias” para tipos de dados existentes, inclusive os registros (**struct**).

```
typedef struct sProduto{  
    int codigo;  
    char nome[50];  
    float preco;  
    int baixa[6];  
} Produto;
```

*Este
identificador
pode ser
suprimido!*

```
Produto feijao;  
feijao.codigo = 1;  
feijao.nome = “Feijão preto kg”;  
feijao.preco = 5.00;  
for (int i=0; i < 6; i++){  
    feijao.baixa[i] = 0;  
}
```



Exemplo

Faça um programa em C que crie uma **variável composta heterogênea (registro)** para representar uma **pessoa**, contendo nome, cpf e data de nascimento.



Exemplo em C

As estruturas de dados também podem ser usadas de modo combinadas, como por exemplo um vetor de um tipo de dados abstrato (registro). Utilizando o exercício anterior como base, pode-se criar um vetor que os armazene:

```
#define TAM 30
```

```
typedef struct {  
    int cpf;  
    char nome[50];  
    char nasc[10];  
} Pessoa;
```

*Declaração do
registro para
representar uma
pessoa*

Exemplo em C

As estruturas de dados também podem ser usadas de modo combinadas, como por exemplo um vetor de um tipo de dados abstrato (registro). Utilizando o exercício anterior como base, pode-se criar um vetor que os armazene:

```
#define TAM 30
```

```
typedef struct {  
    int cpf;  
    char nome[50];  
    char nasc[10];  
} Pessoa;
```

```
Pessoa pessoas[TAM];  
int i;
```

Declaração do registro para representar uma pessoa

*Declaração do vetor pessoas do tipo de dados **Pessoa***



Exemplo em C

As estruturas de dados também podem ser usadas de modo combinadas, como por exemplo um vetor de um tipo de dados abstrato (registro). Utilizando o exercício anterior como base, pode-se criar um vetor que os armazene:

```
for (i = 0; i < TAM; i++){  
    printf("Digite o cpf da pessoa %i: ", i);  
    scanf("%i", &personas[i].cpf);  
  
    printf("Digite o nome da pessoa %i: ", i);  
    scanf("%s", &personas[i].nome);  
  
    printf("Digite o nascimento da pessoa %i: ",  
i);  
    scanf("%s", &personas[i].nasc);  
}
```

*Procedimento
para carregar
um vetor de
registro*



Exemplo em C

As estruturas de dados também podem ser usadas de modo combinadas, como por exemplo um vetor de um tipo de dados abstrato (registro). Utilizando o exercício anterior como base, pode-se criar um vetor que os armazene:

```
for (i = 0; i < TAM; i++){  
    printf("Pessoa %i: ", i);  
    printf("\tCPF: %i: ", pessoas[i].cpf);  
    printf("\tNome: \"%s\", pessoas[i].nome);  
    printf("\tNascimento: \"%s\", pessoas[i].nasc);  
    printf("\n");  
}
```

*Procedimento
para escrever
um vetor de
registro*



Exercício

Faça um programa em C que crie uma **variável composta heterogênea (registro)** para representar um **aluno**, contendo matrícula, nome, cpf e idade.



Agenda

1. Estruturas de dados
2. Variáveis compostas homogêneas
 - a. unidimensionais (vetores)
 - i. Manipulação
 - b. bidimensionais (matrizes)
 - i. Manipulação
3. **Variáveis compostas heterogêneas (registro)**
 - a. **declaração**
 - b. **manipulação**