


# Estrutura de dados heterogêneas

Roberto Rocha



**E quando os atributos não forem  
homogêneos?**

# Dados Homogêneos – vetores e matrizes

Cada vetor e matriz até então só armazena tipos básicos:

Inteiro, real, caractere, lógico

Supondo necessitar armazenar os dados de um comércio

Código, descrição da mercadoria, preço

Vetor código

0	250
1	315
2	417
:	
N	999

Vetor descrição

0	tijolo
1	led
2	papel
:	
N	xxx

Vetor preço

0	10,00
1	0,50
2	15,30
:	
N	99,99

# Dados Homogêneos – vetores e matrizes

Vetor código

0	250
1	315
2	417
:	
N	999

Vetor descrição

0	tijolo
1	led
2	papel
:	
N	xxx

Vetor preço

0	10,00
1	0,50
2	15,30
:	
N	99,99

Pela associação, saberíamos que:

a mercadoria de código **250**, possui a descrição **tijolo** e o valor **10,00**

# Dados Homogêneos – vetores e matrizes

Vetor código

0	250
1	315
2	417
:	
N	999

Vetor descrição

0	tijolo
1	led
2	papel
:	
N	xxx

Vetor preço

0	10,00
1	0,50
2	15,30
:	
N	99,99

```
#include <stdlib.h>
#include <string.h>
#include <locale.h>
int main()
{
    setlocale(LC_ALL,"portuguese");
    int vet_codigo[100];
    char vet_descricao[100][30];
    float vet_preco[100];

    vet_codigo[0]=200;
    strcpy(vet_descricao[0],"Tijolo");
    vet_preco[0]=10.00;
    printf("Código...:%d\n",vet_codigo[0]);
    printf("Descrição:%s\n",vet_descricao[0]);
    printf("Preço....:%.2f\n",vet_preco[0]);
    return 0;
}
```

# Algoritmos – Registros

Ficha com campos a serem preenchidos com informações sobre uma Mercadoria

Mercadoria		
Código	Descrição	Preço

# Algoritmos – Registros

Mercadoria		
Código	Descrição	Preço

Definição de registros:

```
tipo <nome do tipo>  
= registro  
    lista de campos: <tipo 1>  
    .....  
    lista de campos : <tipo n>  
fimregistro
```

```
tipo mercadoria  
= registro  
    código :inteiro  
    descricao:caractere  
    preco :real  
fimregistro
```

# Algoritmos – Registros

**tipo** mercadoria  
**= registro**  
    **código :inteiro**  
    **descricao:caractere**  
    **preco :real**  
**fimregistro**

var  
  a,b : **inteiro**  
  m: mercadoria

**Inicio**  
  m.codigo ← 250  
  m.descricao ← "tijolo"  
  m.preco ← 10.00  
**finalgoritmo**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
struct Tmercadoria {
    int codigo;
    char descricao[50];
    float preco;
};
typedef struct Tmercadoria mercadoria;
int main()
{
    setlocale(LC_ALL,"portuguese");
    mercadoria m;
    m.codigo=250;
    strcpy(m.descricao,"Tijolo");
    m.preco=10.00;

    printf("Código...:%d\n",m.codigo);
    printf("Descrição:%s\n",m.descricao);
    printf("Preço.....:%.2f\n",m.preco);

    return 0;
}
```

```
Código...:250
Descrição:Tijolo
Preço.....:10,00
```



# Algoritmos – Registros

Escreva o registro correspondente e atribua um valor exemplo

a1



Aluno		
Nome: Ana	Matricula: 1234	
Nota1: 9.5	Nota2: 8.0	Nota3: 10.0

# Algoritmos – Registros

Escreva o registro correspondente e atribua um valor exemplo

a1

Aluno		
Nome: <b>Ana</b>	Matricula: <b>1234</b>	
Nota1: <b>9.5</b>	Nota2: <b>8.0</b>	Nota3: <b>10.0</b>

**tipo** aluno

= **registro**

matricula :**inteiro**

nome :**caractere**

nota1,nota2,nota3 :**real**

**fimregistro**

var

a1: aluno

**inicio**

a1.matricula ← 1234

a1.nome ← "Ana"

a1.nota1 ← 9.5

a1.nota2 ← 8.0

a1.nota3 ← 10.0

**finalgoritmo**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
struct Taluno {
    int matricula;
    char nome[50];
    float nota1,nota2,nota3;
};
typedef struct Taluno aluno;
int main()
{
    setlocale(LC_ALL,"portuguese");
    aluno a1;
    a1.matricula=1234;
    strcpy(a1.nome,"Ama");
    a1.nota1=9.5;
    a1.nota2=8.0;
    a1.nota3=10.0;

    printf("Matricula:%d\n",a1.matricula);
    printf("Nome.....:s\n",a1.nome);
    printf("Nota1.....:%.2f\n",a1.nota1);
    printf("Nota2.....:%.2f\n",a1.nota2);
    printf("Nota3.....:%.2f\n",a1.nota3);

    return 0;
}
```

```
Matricula:1234
Nome.....:Ama
Nota1.....:9,50
Nota2.....:8,00
Nota3.....:10,00
```

# Vetor de Registros

tipo mercadoria

= registro

código :inteiro

descrição :caractere

preço :real

fimregistro

var

m: vetor[0..9] de MERCADORIA;

inicio

m[0].codigo ← 250

m[0].descrição ← “tijolo”

m[0].preço ← 10.00

m[1].codigo ← 315

m[1].descrição ← “led”

m[1].preço ← 0.50

m[2].codigo ← 417

m[2].descrição ← “papel”

m[2].preço ← 15.30

fimalgoritmo

	Código	Descrição	Preço
0	250	Tijolo	10.00
1	315	Led	0.50
2	417	Papel	15.30
:			

# Vetor de Registros

tipo mercadoria

= registro

código :inteiro

descrição :caractere

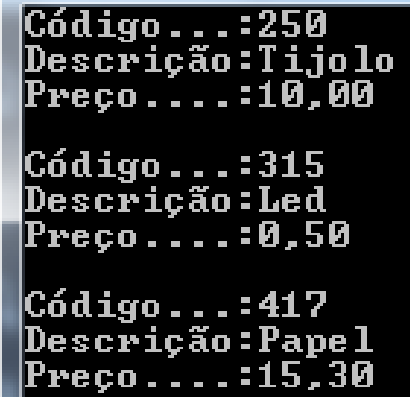
preço :real

fimregistro

var

m: vetor[0..9] de mercadoria

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
struct Tmercadoria {
    int codigo;
    char descricao[50];
    float preco;
};
typedef struct Tmercadoria mercadoria;
int main()
{
    setlocale(LC_ALL,"portuguese");
    mercadoria m[10];
    int i;
    m[0].codigo=250;
    strcpy(m[0].descricao,"Tijolo");
    m[0].preco=10.00;
    m[1].codigo=315;
    strcpy(m[1].descricao,"Led");
    m[1].preco=0.50;
    m[2].codigo=417;
    strcpy(m[2].descricao,"Papel");
    m[2].preco=15.30;
    for (i=0;i<3;i=i+1)
    { printf("Código...:%d\n",m[i].codigo);
      printf("Descrição:%s\n",m[i].descricao);
      printf("Preço.....:%.2f\n\n",m[i].preco);
    }
    return 0;
}
```



```
Código...:250
Descrição:Tijolo
Preço.....:10,00

Código...:315
Descrição:Led
Preço.....:0,50

Código...:417
Descrição:Papel
Preço.....:15,30
```

# Algoritmos – Registros – Funções

Mercadoria		
Código	Descrição	Preço

**tipo mercadoria**  
= **registro**  
**código :inteiro**  
**descrição :caractere**  
**preço :real**  
**fimregistro**

```
struct Tmercadoria {  
    int codigo;  
    char descricao[50];  
    float preco;  
};  
typedef struct Tmercadoria mercadoria;
```

Crie uma função para ler e retornar um registro de mercadoria

```
função leMercadoria():mercadoria  
var  
    m:mercadoria  
inicio  
    leia(m.código)  
    leia(m.descrição)  
    leia(m.preço)  
    retorne m  
fimfunção
```

```
29 mercadoria leMercadoria()  
30 {  
31     mercadoria m;  
32     printf("Código...:");  
33     fflush(stdin);  
34     scanf("%d", &m.codigo);  
35     printf("Descrição:");  
36     fflush(stdin);  
37     gets(m.descricao);  
38     printf("Preço....:");  
39     fflush(stdin);  
40     scanf("%f", &m.preco);  
41     return m;  
42 }
```

## Algoritmos – Registros – Funções – passagem por valor

Mercadoria		
Código	Descrição	Preço

**tipo mercadoria**  
= **registro**  
código :**inteiro**  
descrição :**caractere**  
preço :**real**  
**fimregistro**

```
struct Tmercadoria {  
    int codigo;  
    char descricao[50];  
    float preco;  
};  
typedef struct Tmercadoria mercadoria;
```

Crie um procedimento para receber e imprimir um registro de mercadoria

```
procedimento imprimeMercadoria(m mercadoria)  
var  
inicio  
    escreva(m.código)  
    escreva(m.descrição)  
    escreva(m.preço)  
fimprocedimento
```

```
43 void imprimeMercadoria(mercadoria m)  
44 {  
45     printf("Código...:%d\n", m.codigo);  
46     printf("Descrição:%s\n", m.descricao);  
47     printf("Preço...:%.2f\n", m.preco);  
48 }
```

# Algoritmos – Registros – Funções – passagem por referência

Mercadoria		
Código	Descrição	Preço

Crie um procedimento para alterar dados de mercadoria

procedimento alteraMercadoria(var m mercadoria)

var

inicio

  escreva(m.código)

  leia(m.código)

  escreva(m.descrição)

  leia(m.código)

  escreva(m.preço)

  leia(m.preço)

fimprocedimento

tipo mercadoria

= registro

código :inteiro

descrição :caractere

preço :real

fimregistro

struct Tmercadoria {

  int codigo;

  char descricao[50];

  float preco;

};

typedef struct Tmercadoria mercadoria;

```
58 void alteraMercadoria(mercadoria *m)
59 {
60     printf("Código...:%d\n",m->codigo);
61     fflush(stdin);
62     scanf("%d",&m->codigo);
63     printf("Descrição:%s\n",m->descricao);
64     fflush(stdin);
65     gets(m->descricao);
66     printf("Preço....:%.2f\n",m->preco);
67     fflush(stdin);
68     scanf("%f",&m->preco);
69 }
```

## Algoritmos – Registros – Funções

Mercadoria		
Código	Descrição	Preço

Crie um programa principal com as seguintes opções:

- a – incluir mercadorias
- b – alterar dados das mercadorias
- c – imprimir mercadorias
- d – sair do programa



# Algoritmos – Registros – Funções

Crie um programa principal com as seguintes opções:

- a – incluir mercadorias
- b – alterar dados das mercadorias
- c – imprimir mercadorias
- d – sair do programa

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <locale.h>
5  #include <conio.h>
6  struct Tmercadoria
7  {
8      int codigo;
9      char descricao[50];
10     float preco;
11 };
12 typedef struct Tmercadoria mercadoria;
13 mercadoria leMercadoria();
14 void imprimeMercadoria(mercadoria m);
15 void alteraMercadoria(mercadoria *m);
```

```
16 int main()
17 {
18     setlocale(LC_ALL, "portuguese");
19     mercadoria m[3];
20     int i, op;
21     do
22     {
23         system("cls");
24         printf("Escolha:\n");
25         printf("a - incluir mercadorias\n");
26         printf("b - alterar mercadorias\n");
27         printf("c - imprimir mercadorias\n");
28         printf("d - sair do programa\n");
29         op=getch();
30         switch (op)
31         {
32             case 'a':
33                 for (i=0; i<3; i=i+1)
34                     {m[i]=leMercadoria();}
35                 break;
36             case 'b' :
37                 for (i=0; i<3; i=i+1)
38                     {alteraMercadoria(&m[i]);}
39                 break;
40             case 'c':
41                 for (i=0; i<3; i=i+1)
42                     {imprimeMercadoria(m[i]);}
43                 break;
44             }
45             if (op!='d')
46                 {system("pause");}
47         }
48     } while (op!='d');
49     return 0;
50 }
51
52
53
54
```

# Algoritmos – Registros

Definição de registros:

Funcionário		
Código	Nome	
Cidade		Estado
Salário		
Endereço		
Rua	Nr	Cep

# Algoritmos – Registros

Funcionário		
Código	Nome	
Cidade		Estado
Salário		
Endereço		
Rua	Nr	Cep

tipo f = registro

código :inteiro

nome :caractere

endereco :ender

cidade,estado :caracter

salario :real

fimregistro

Esta incompleta pois  
contem o tipo **ender** ,  
ainda não definido ( não  
é tipo básico). Portanto,  
deve-se defini-lo:

tipo **ender** = registro

rua :caractere

nro,cep :inteiro

fimregistro

# Algoritmos – Registros

tipo **ender** = registro

rua :caractere

nro,cep :inteiro

fimregistro

tipo **f** = registro

código :inteiro

nome :caractere

endereco :ender

cidade,estado :caracter

salario :real

fimregistro

var           Atribuição de valores:

func:f

inicio

func.código <- 1234

func.nome <- "Fulano de tal"

func.endereco.rua <- "Rua das abobóras"

func.endereco.nro <- 1001

func.endereco.cep <- 30000111

func.cidade <- "Lindo Horizonte"

func.estado <- "MG"

func.salario <- 47500.00

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <locale.h>
5
6  struct Tender
7  {
8      char rua[50];
9      int nro, cep;
10 };
11 typedef struct Tender ender;
12 struct Tf
13 {
14     int codigo;
15     char nome[50];
16     ender endereco;
17     char cidade[30], estado[2];
18     float salario;
19 };
20 typedef struct Tf f;
21 int main()
22 {
23     setlocale(LC_ALL, "portuguese");
24     f func;
25     func.codigo=1234;
26     strcpy(func.nome, "Fulano de Tal");
27     strcpy(func.endereco.rua, "Rua das abobóras");
28     func.endereco.nro=1001;
29     func.endereco.cep=30000111;
30     strcpy(func.cidade, "Lindo Horizonte");
31     strcpy(func.estado, "MG");
32     func.salario=47500.00;
33     return 0;
34 }
```

# Algoritmos – Registros – Funções

Aluno		
Nome: <b>Ana</b>		Matricula: <b>1234</b>
Nota1: <b>9.5</b>	Nota2: <b>8.0</b>	Nota3: <b>10.0</b>

Faça as funções para incluir, excluir, alterar e imprimir dados de uma turma de X alunos

Desafio:

Crie um arquivo para armazenar o tamanho de alunos da turma e logo em seguida os seus dados.

No final do sistema ele irá gravar os dados atuais, no início do programa irá ler os dados e preencher as estruturas de dados correspondentes.



**PUC Minas**  
**Virtual**