

Profa Ma. Luciane Y. H. Kanashiro

Introdução

- Vetores e matrizes:
 - Estruturas de dados homogêneas
 - Armazenam vários valores, mas todos de um mesmo tipo (todos int, todos double, todos float, todos char)

Introdução

- Problemas reais Temos coleções de dados que são de tipos diferentes
- Exemplo: passagem de ônibus, que contém número da passagem: int Data: char[], origem: char[], destino:char[], horário:char[], poltrona:int, Idade: int, nomechar[]

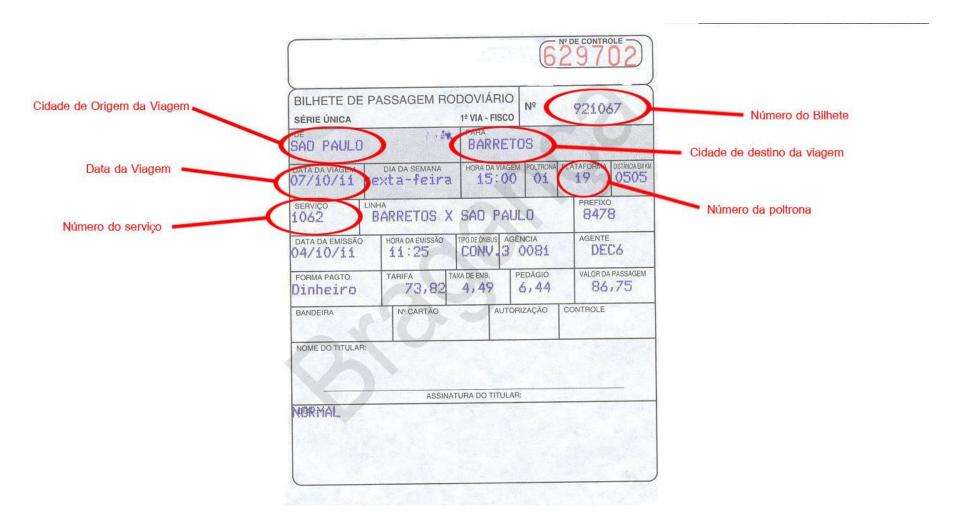
| De: | Para: | | |
|----------|-----------|--------|--|
| Horário: | Poltrona: | Idade: | |

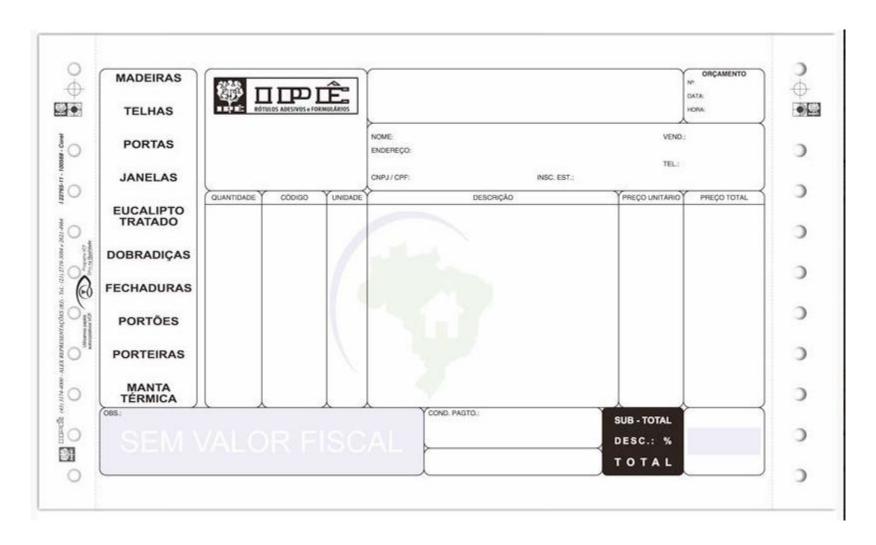
- Tipo de dado heterogêneo
- Conjunto de informações de tipos primitivos diferentes

- Agregam vários dados acerca de uma mesma entidade
- Cada dado em um registro é chamado de campo
- Exemplos:
 - carro: cor, marca, ano, placa, chassi
 - pessoa: nome, idade, endereço

 Mundo real: existe coleção de dados com tipos diferentes

| FÓRUM DE ARQUITETURA DE MOSSORÓ/RN | | | | | |
|---|-----------|-----------|--|--|--|
| FICHA DE INSCRIÇÃO Nome: Endereço: | | | | | |
| Profissão: Fone.: E-mail: PROFISSIONAL ASSOCIADO IAB ESTUDANTE | | | | | |
| R\$ 60,00 | R\$ 40,00 | R\$ 30,00 | | | |
| REALIZAÇÃO: PromoExpo resultate analysis as estables. | PROMOÇÃO: | APOIO: | | | |





Sintaxe em C

- A palavra reservada struct indica ao compilador que está sendo criada uma estrutura
- Uma estrutura deve ser declarada após incluir as bibliotecas e antes da main

```
struct <identificador_struct> {
   tipo <nome_variável_campo1>;
   tipo <nome_variável_campo2>;
   :
} <variáveis_estrutura>;

struct <identificador_struct> <var1>, <var2>;
```

Sintaxe em C

- Se o compilador C for compatível com o padrão C ANSI
 - Informação contida em uma struct pode ser atribuída a outra struct do mesmo tipo
 - Não é necessário atribuir os valores de todos os elementos/campos separadamente
 - Por exemplo: <var1> = <var2>;
 - Todos os campos de <var1> receberão os valores correspondentes dos campos de var2

Sintaxe em C

- Para acessar os campos da struct
 - Utiliza-se o nome da variável struct, seguido de ponto, seguido do nome do campo
 - Por exemplo: <var1>.<nome_variavel_campo>;

Representação em C

Registro

```
struct participante{
    char *nome;
    char endereço[100];
    char *profissão;
    char fone[20];
    char email[50];
```



Manipulação

Registro

 Comando para declarar variável com a struct (registro)

```
struct participante p;
```

- Já vimos que para acessar os membros de uma struct deve-se usar nome_variável.nome_membro
- Setando informações contidas no registro

```
p.nome = "José";
strcpy(p.endereco, "rua das flores")
```

Imprimindo informações contidas no registro

```
printf("%s",p.nome);
```

Manipulação

 Para obter o nome do participante e colocar na string nome da struct se poderia utilizar:

```
/*Exemplo com ponteiro*/
    p.nome = malloc(5);
    printf("Digite o nome do participante: ");
    gets(p.nome);
    printf("Nome: %s \n", p.nome);
/*Exemplo com string de char*/
    printf("Digite o endereco do participante: ");
    gets(p.endereco);
    printf("Endereco: %s \n", p.endereco);
```

Registro de Conjuntos

- Campos podem ser compostos por outros tipos construídos.
- Ex: suponha que o registro tenha um campo que indique o valor pago por tipo de profissional, sendo que cada posição corresponde a um tipo de profissional

tipo vetorTipo[3];



Representação

Registro de Conjuntos

```
    Criar conjunto (Registro)
    typedef struct tipo_categoria {
        char categoria[50];
        float valor;
}tipo;
```

Manipulação

Registro de Conjuntos

```
public class Participante{
      char *nome;
      char *endereço;
      char *profissão;
      char *fone ;
      char *email;
//Criando ponteiro para armazenar os registros tipo
  tipo *t;
               typedef struct tipo_categoria {
                      char categoria[50];
                      float valor;
                }tipo;
```

```
tipo ttipo;
```

Criando vetor para armazenar

```
tipo vetorTipo[3];
```

Atribuindo valor aos campos do Conjunto

```
ttipo. Categoria("Estudante");
ttipo.Valor(30);
vetorTipo[0] = ttipo;
```

Imprimindo

```
printf("%s\n", p.t[0].categoria);
printf("%f\n", p.t[0].valor);
```

Atribuindo valor ao campo no Registro

```
p.t = vetorTipo;
```

Conjunto de Registros

 As estruturas homogêneas podem armazenar as estruturas heterogêneas



0 1 ••• N-1

Manipulação

Conjunto de Registros

Criando vetor para armazenar os registros

```
struct participante vetor_participantes[100];
```

Atribuindo valor aos campos do Registro

```
p.nome = "Maria";
p.email="maria@gmail.com";
p.endereco = "Rua XV";
p.fone = "41 99993233";
p.profissao = "arquiteta";
p.t = vetorTipo;
```

Atribuindo valor aos campos do vetor

```
Vetor participante[0] = p;
```

Imprimindo valor dos campos do vetor

```
printf("%s\n", vetor_participantes[0].nome);
printf("%f\n", vetor_participantes[0].t[0].valor);
```

Referências

- [1] ASCENCIO, A. CAMPOS, E. **Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ e Java.** 3 ed. São Paulo: Pearson Prentice Hall, 2007.
- [2]FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. 2ª Ed. São Paulo: Makron Books, 2000.
- [4]XAVIER, Gley Fabiano Cardoso. Lógica de Programação. 11ª Ed. São Paulo: SENAC São Paulo, 2007MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo de. Algoritmos: Lógica para Desenvolvimento de Programação de Computadores. 12ª Ed. São Paulo: Érica, 2001.
- https://www.inf.pucrs.br/~cnunes/lapro/aulas/structs.pdf