

TAD0102

Algoritmos e programação

**Prof. Dr. Josenalde Barbosa de
Oliveira**

josenalde@eaj.ufrn.br

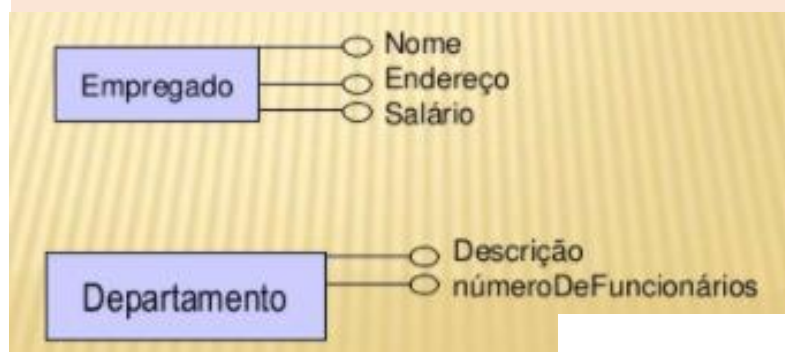
Aulas: 24T345, 6 CRDS – 90h

Tipos de Dados Heterogêneos

ESTRUTURAS

Dados heterogêneos

- Introduz conceitos e ideias de orientação a objetos
- E bancos de dados (registros), ao agrupar dados de determinada entidade (tabela ou classe) – *dados de tipos diferentes formando numa unidade*
- Pode-se ter um conjunto de estruturas (array) e manipular em funções



Estruturas – tipo definido pelo usuário - sintaxe

```
struct Produto {  
    string nome;  
    float peso;  
    float preco;  
};
```

Define um novo tipo: Produto

Membros da estrutura
(Em OO: atributos
Em BD: campos)

```
int main() {  
    Produto p1;  
    float peso = 10.5;  
    p1.nome = "joaquim";  
    p1.peso = peso;  
}
```

Em C seria necessário, neste caso, colocar **struct** antes

Acesso aos membros: (dot)
<var>.<membro>

```
Produto p1;  
vector<Produto> p(2); // Produto p[2];  
p[0].nome = "raspador de coco";  
p[1].nome = "abridor de latas";  
cout << p[0].nome << endl;  
cout << p[1].nome << endl;  
cin >> p[0].peso;
```

Estruturas

- Podem ter escopo GLOBAL ou LOCAL

```
struct Components {
    unsigned long part_number;
    float part_cost;
};
void mail();

int main() {
    struct Cars {
        int key_number;
        char car[12];
    };

    Components retrovisor;
    Cars c;
}


void mail() {
    Components pneu;
    Cars d; // ERRO não visível
}
```

Estruturas – inicialização e atribuição

```
struct Aluno {  
    unsigned long matricula;  
    std::string nome;  
};
```

O que não for inicializado, é 0 ou vazio

```
int main() {  
    Aluno a = {2020132020, "Josenalde"};  
    Aluno b = {};  
}
```



```
struct Ponto2D {  
    float x;  
    float y;  
};
```

```
int main() {  
    Ponto2D p1;  
    p1.x = 2.0;  
    p1.y = -4.0;  
}
```

Estruturas – campos com tamanho especificado

```
struct RegistradorA {  
    unsigned int SN : 4;  
    unsigned int : 4; // não utilizado  
    bool in1 : 1;  
    bool toogle : 1;  
};
```

```
struct endereço {  
    std::string rua;  
    int numero;  
};  
struct cadastro {  
    std::string nome;  
    int idade;  
    endereco ender;  
};
```

```
int main() {  
    RegistradorA reg = {14, true, false};  
    if (reg.toogle) ....  
    cadastro c;  
    c.ender.rua = "das amoras";  
}
```

Enumeração – lista de constantes

```
enum semana {Domingo, Segunda, Terca, Quarta, Quinta, Sexta, Sabado};  
enum escapes {tab='\t', novalinha='\n'};  
  
int main() {  
    enum escapes e = novalinha;  
    cout << "teste" << e;  
    e = tab;  
    cout << "ola" << e << "mundo";  
}
```

Apelido para tipos - typedef

```
typedef int inteiro;  
typedef unsigned int uint;  
inteiro x;  
uint y;
```

Exercícios

- 1) Implemente um programa que leia o nome, a idade e o endereço de uma pessoa e armazene esses dados numa estrutura. Em seguida, imprima na tela os dados da estrutura lida.
- 2) Crie uma estrutura chamada Retângulo. Ela deve conter o ponto superior esquerdo e o ponto inferior direito do retângulo. Cada ponto é definido por uma estrutura Ponto, a qual contém as posições X e Y. Faça um programa que declare e leia uma estrutura Retângulo e um Ponto e exiba:
 - 2.1) a área e o comprimento da diagonal e o perímetro desse retângulo
 - 2.2) informe se o ponto está ou não dentro do retângulo
- 3) Crie uma estrutura representando um aluno de uma disciplina, com matrícula, nome e as notas de 03 provas. Agora, escreva um programa que leia os dados de cinco alunos e os armazena nessa estrutura. Em seguida, exiba o nome as notas do aluno que possui a maior média.
- 4) Crie uma enumeração representando os meses do ano. Escreva um programa que leia um valor inteiro e exiba o nome do mês correspondente e quantos dias ele possui.