Aula 13 - Structs

Priscila Delabetha

Vamos supor que você foi contratado por uma grande empresa para criar um aplicativo em C que armazena todas as informações dos funcionários, um banco de dados.
Vamos supor que tem 300 funcionários.

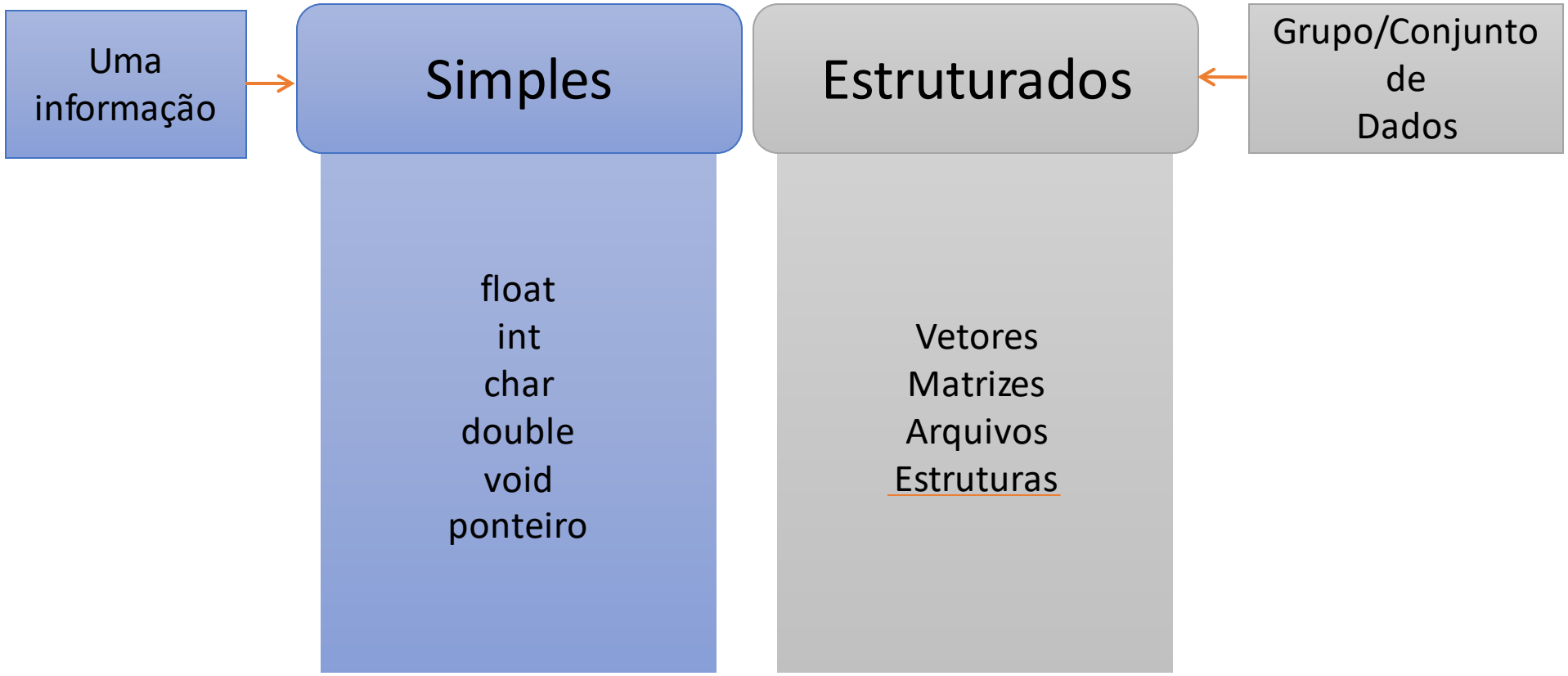
Como vai fazer isso? Com vetores?

Antes de estudar **structs**, nossos dados são armazenados em variáveis simples. Temos de escolher os tipos (char, int, float, double).

Armazenamos também conjuntos de valores relacionados em vetores que tem a capacidade de conter vários elementos do mesmo tipo.

Uma **struct** é um conjunto de uma ou mais variáveis (às quais também chamamos de campos) agrupados em um único nome, para facilitar a referência.

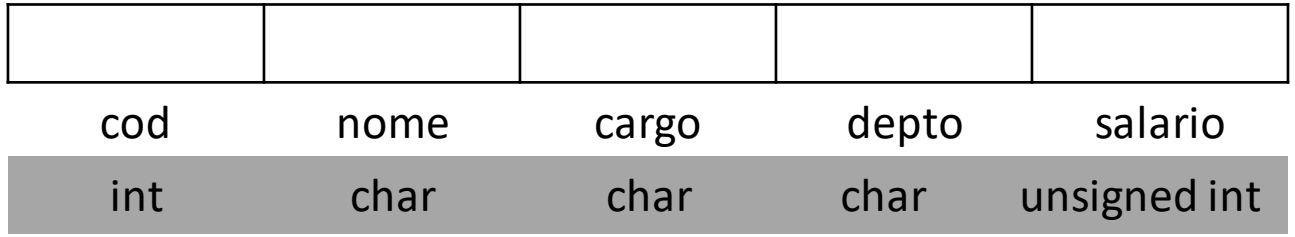
Tipos de Dados em C



Estruturas

- Tipo de dado estruturado que permite agrupar informações com características diferentes, isto é, elementos com tipos de dados diferentes.
- Cada unidade de informação é um campo do registro

funcionário

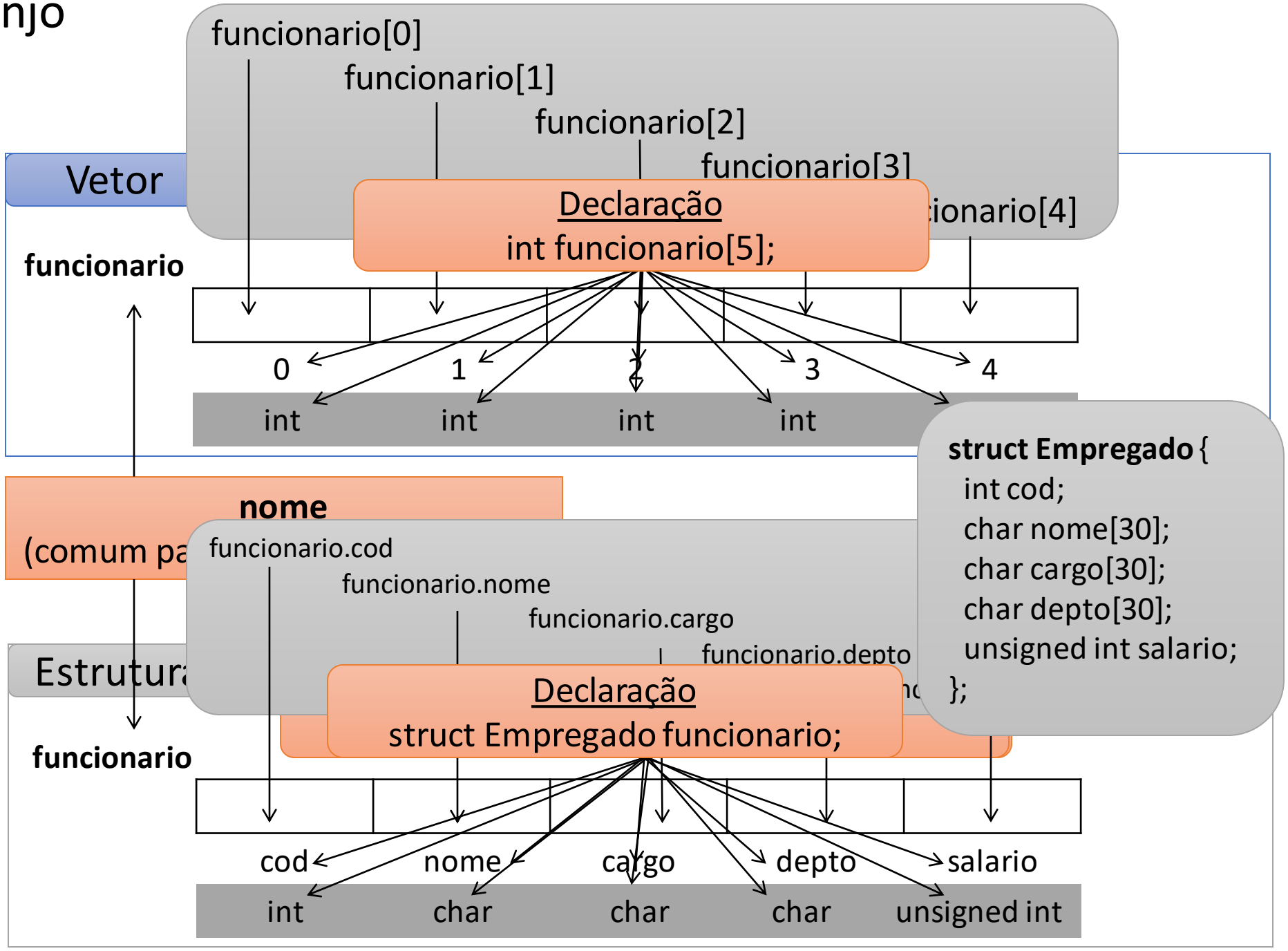


Diferença entre arranjo (vetor ou matriz) e estrutura

- **Vetores** (homogêneo)
  - Todos os elementos do mesmo tipo.
  - Um só nome, com índice para identificar cada elemento.

- **Estrutura**(heterogêneos)
  - Elementos podem ser de tipos diferentes
  - Cada elemento é identificado através de um nome único ou campo.

Diferença entre arranjo (vetor ou matriz) e estrutura



Sintaxe

```
struct identificador {
        tipo1 campo1, campo2;
        tipo2 campo3;
        ...
        tipo n campo n;
};
```

```
struct Empregado {
  int cod;
  char nome[30];
  char cargo[30];
  char depto[30];
  int salario;
};
```

Declaração de tipo struct

- Permite a declaração de novos tipos de variáveis (estruturadas);
- Caracteriza o conteúdo de possíveis variáveis (não alocam memória na declaração deste novo tipo);
- Costumam ser declaradas como globais para que possam ser utilizadas como parâmetros de funções;

- PROGRAMAÇÃO ESTRUTURADA: quando globais, não devem incluir a **declaração** de variáveis junto a do tipo, uma vez que, por questões de bom estilo de programação, programas não devem fazer uso de variáveis globais.

Declaração de variáveis de um tipo struct

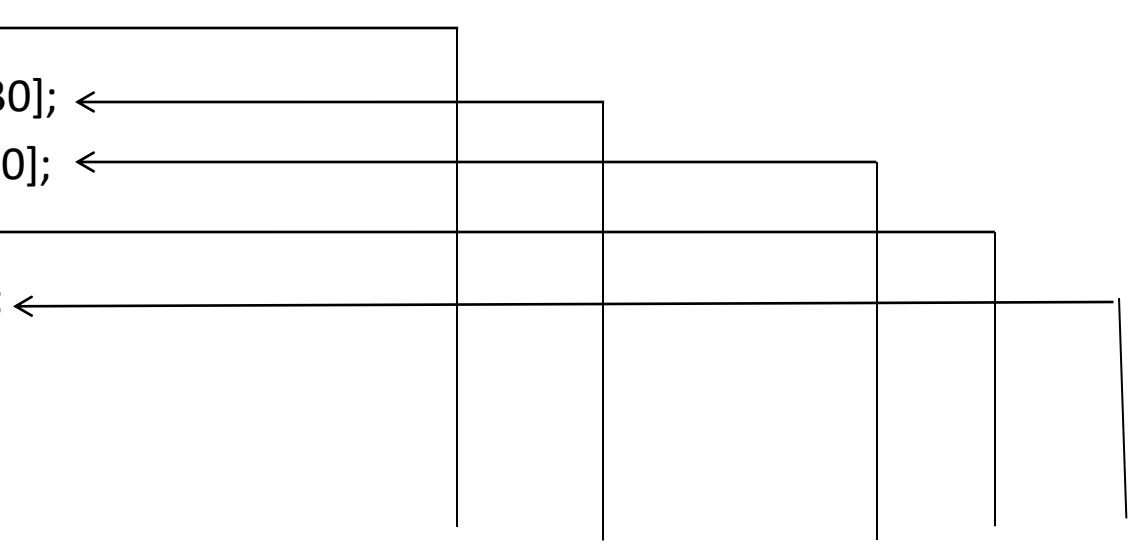
```
struct Empregado {
  int cod;
  char nome[30];
  char cargo[30];
  char depto[30];
  unsigned int salario;
};

struct Empregado func1, func2, func3;
```

Declaração da struct

Declaração de variáveis do tipo struct Empregado

Inicialização de variável tipo estrutura

```
#include <stdio.h>
#include <stdlib.h>
struct Empregado{
  int cod;
  char nome[30];
  char cargo[30];
  int depto;
  float salario;
};

int main(){
  struct Empregado funcionario ={ 1432, “Jose Costa”, “Gerente”, 2, 4656.00 };
……
```



```c
struct str_funcionario{// estrutura
  int cod;
  char nome[30];
  char cargo[10];
  int depto;
  float salario;
};

int main ( ){
  struct str_funcionario funcionario; // variavelvel: funcionario
  printf("Forneca os dados do(a) funcionario(a):");
  scanf ("%d", &funcionario.cod);
  fflush(stdin);
  printf(": ");
  gets(funcionario.nome); //scanf para no 1 branco encontrado!!!
  fflush(stdin);
  printf(": ");
  gets(funcionario.cargo);
  printf(": ");
  scanf ("%d", &funcionario.depto);
  printf(": ");
  scanf ("%f", &funcionario.salario);
  }
```

```c
struct str_funcionario{
  int cod;
  char nome[30];
  char cargo[10];
  int depto;
  float salario;
};

int main ( ){
  str_funcionario funcionario;
  printf("do(a) funcionario(a) de codigo %d:",  funcionario.cod);
  printf (": %s", funcionario.nome);
  printf (": %s",funcionario.cargo);
  printf (": %d", funcionario.depto);
  printf (": %6.2f", funcionario.salario);
}
```

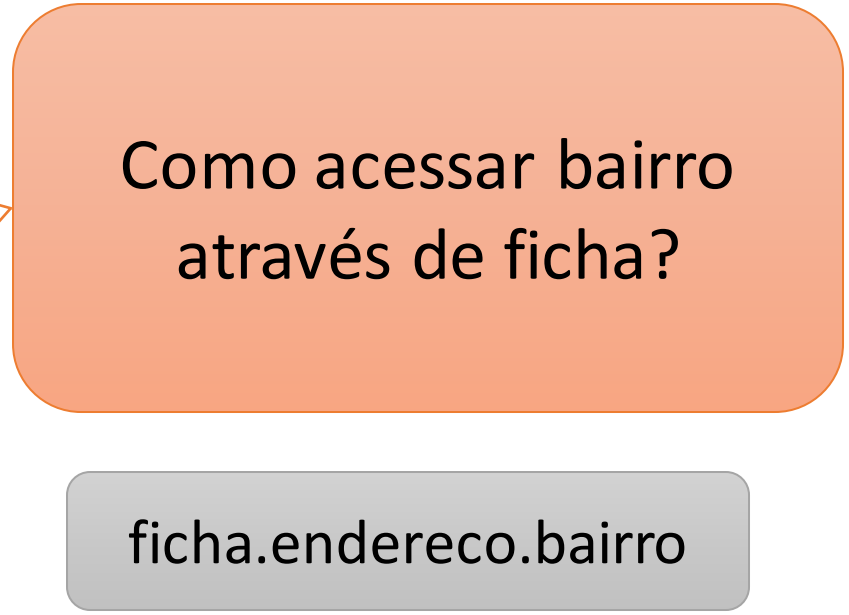
Atribuição

```
struct Empregado {
  int cod;
  char nome[30];
  char cargo[30];
  char depto[30];
  unsigned int salario;
};

struct Empregado func1, func2, func3;
```

```
func1 = func2;
```

Struct dentro de struct

```
struct tipo_endereço {
  char rua [50];
  int numero;
  char bairro [20];
  char cidade [30];
  char sigla_estado [3];
  char cep[10];
};

struct tipo_ficha {
  char nome [50];
  char telefone[16];
  struct tipo_endereco endereco;
};
...
struct tipo_ficha ficha //variável ficha
```



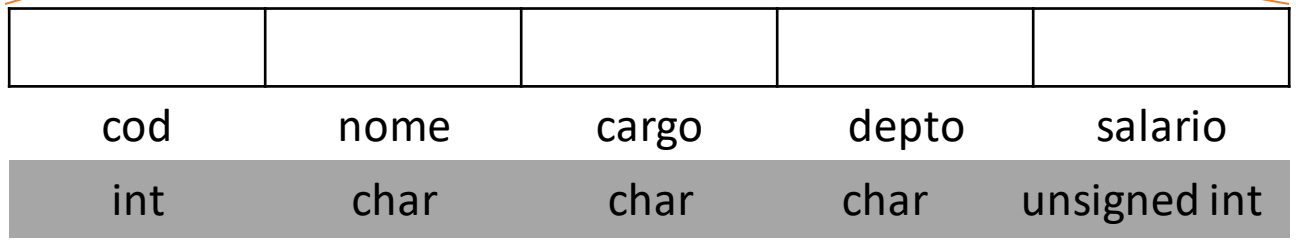
Arranjo de structs

Acesso
funcionarios[2].cod

Atribuição
funcionarios[2].cod = 0;

struct Empregado funcionarios[5]

struct Empregado funcionario




```
struct Empregado {
  int cod;
  char nome[30];
  char cargo[30];
  char depto[30];
  unsigned int salario;
};
```

typedef

A declaração de structs pode ser ainda mais simples, sem ter que declarar um tipo de struct cada vez que nos referenciarmos a uma estrutura

```
struct Funcionario f1;
```

Podemos chamar unicamente pelo nome que demos a ela, da mesma forma que as outras variáveis.

```
int codi;
Funcionario f1;
```

typedef

Podemos fazer isso se utilizarmos a palavra reservado **typedef.**

Cuja sintaxe é:

```
typedef tipo_existente apelido
```

A palavra **typedef** não cria um tipo novo de dado, apenas permite que um determinado tipo possa ser chamado de forma diferente, de acordo com a necessidade do programador.

E ela não se limita ao uso com estruturas, mas com qualquer tipo de dado!

typedef

```
typedef unsigned long int INTEIRAO_POSITIVO;

INTEIRAO_POSITIVO salario=50000;

typedef struct{
    char nome[20];
    int idade;
    char sexo;
}PESSOA;

PESSOA priscila={"Priscila", 45, 'M'};
```

Exercício

- Faça uma agenda que cadastre 10 pessoas
  - Defina uma estrutura pessoa que contenha nome, idade, endereço e telefone;
  - Endereço deve ser uma estrutura também.
  - Crie funções que façam a impressão e o cadastro de pessoas e exclusão de pessoas (a exclusão é só preencher os dados com 0).

Postar no moodle