Estrutura de Dados

Prof. Maurício Neto

UNIFAVIP
wyden

O que vamos ver?

01

Estrutura de Dados Heterogêneas

02

Registros

03

Estrutura Aninhada

01

Estrutura de Dados Heterogêneas

Estrutura de Dados Heterogêneas

Podemos armazenar dados de uma pessoa, como nome (string), idade (int) e salário (float). Dentro de uma struct, podemos incluir um vetor e até mesmo outras structs.

Definição: Uma struct, ou registro, é um conjunto de variáveis de diferentes tipos organizadas sob um único nome. Criar uma struct é equivalente a criar um novo tipo de dado.

Struct

A diferença entre uma struct e um vetor (ou matriz) está na maneira como cada um organiza e armazena os dados:

Struct (ou Registro):
Composição: Uma struct é uma coleção de variáveis de diferentes tipos agrupadas sob um único nome.

Flexibilidade: Pode conter tipos variados, como inteiros, strings, floats, e até outras structs.

Struct

Uso: Usada para representar uma entidade complexa com múltiplas características. Por exemplo, uma struct Pessoa pode conter um nome (string), idade (int) e salário (float).

```
struct Pessoa {
    char nome[50];
    int idade;
    float salario;
};
```

UNIFAVIP wyden

Vetor

Composição: Um vetor é uma coleção de elementos do mesmo tipo armazenados de forma sequencial na memória.

Homogeneidade: Todos os elementos em um vetor devem ser do mesmo tipo.

Vetor

Uso: Usado para armazenar uma lista de elementos do mesmo tipo. Por exemplo, um vetor de inteiros pode armazenar uma sequência de números.

```
int numeros[5] = {1, 2, 3, 4, 5};
```

Matriz

Composição: Uma matriz é essencialmente um vetor de vetores, permitindo a organização de dados em duas (ou mais) dimensões.

Homogeneidade: Todos os elementos em uma matriz também devem ser do mesmo tipo.

Matriz

Uso: Usada para representar dados tabulares ou gradeados. Por exemplo, uma matriz de inteiros pode ser usada para armazenar uma tabela de números.

```
int matriz[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

UNIFAVIP wyden

Estrutura de Dados Heterogêneas

Podemos dizer que as structs são “precursoras” da POO.

As structs e a POO estão relacionadas na medida em que as classes na POO podem ser vistas como uma evolução das structs, oferecendo funcionalidades avançadas como **métodos**, **encapsulamento** e **herança**.
No entanto, structs ainda são úteis em contextos onde uma organização simples de dados é suficiente, enquanto a POO é mais adequada para sistemas complexos que requerem modelagem de comportamento e dados.

02

Registros

Registros

Registros, contém todos os dados necessários para identificar uma entidade do mundo real, e que em computação são representados por variáveis.

Exemplo Aluno:
- Aluno
- Matrícula
- Nascimento
- Sexo
- Telefone
- Email

Registros

```c
struct Aluno {
    char nome[100];
    int matricula;
    char nascimento[11];
    char sexo;
    char telefone[15];
    char email[100];
};
```

Registros

Structs são tipos de variáveis que agrupam dados geralmente de tipos diferentes (int, double, float, char). Um exemplo de struct é uma que representa Funcionários, onde pode conter um int (para idade), uma string (para o nome), e um float (para o salário).

```
struct Funcionario {
    char nome[100];
    int idade;
    float salario;
};
```

Registros

Uma struct é considerada um novo tipo de dado. Com ela, informamos ao compilador o nome, o tamanho em bytes e a forma como esses dados devem ser armazenados e recuperados na memória.

No exemplo anterior criamos o tipo de dado **Funcionário.**

Registros

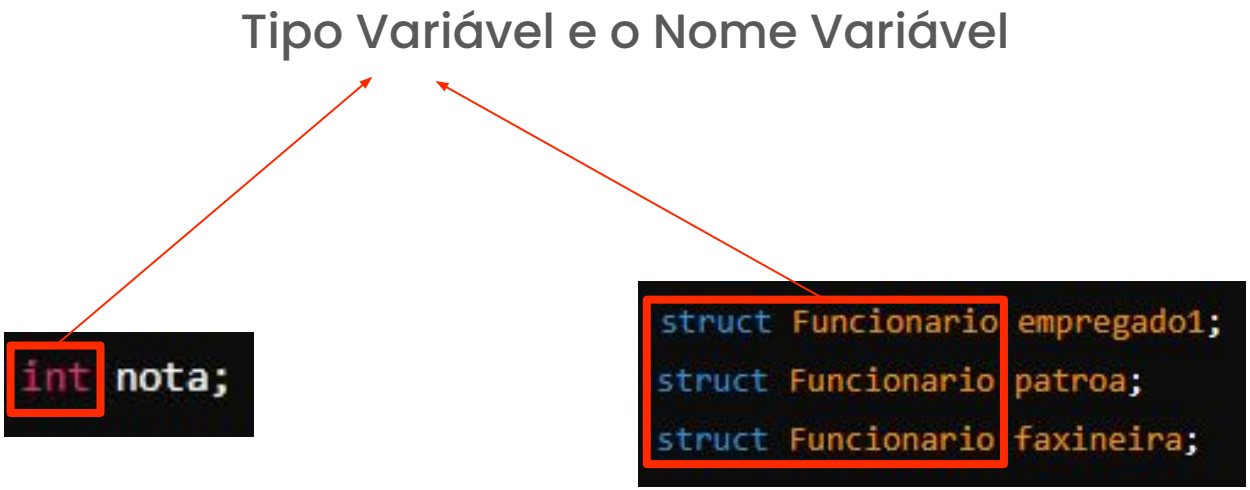
Como se trata de um novo tipo de dado, a declaração de uma struct segue a mesma sintaxe de declaração de variável. Por exemplo, para declarar um tipo inteiro usamos:

```
int nota;
```

No caso da struct Funcionario, a declaração de variáveis é feita da mesma forma:

```
struct Funcionario empregado1;
struct Funcionario patroa;
struct Funcionario faxineira;
```

Registros


Tipo Variável e o Nome Variável
int nota;
struct Funcionario empregado1;
struct Funcionario patroa;
struct Funcionario faxineira;

Atividade

Faça um código que recebe o título do livro, o autor, o ano e o preço, utilizando Struct, e que o retorno seja igual ao de baixo:

Input:

```
Digite o título do livro: Estrutura de Dados
Digite o autor do livro: Maurício
Digite o ano de publicação do livro: 2024
Digite o preço do livro: 199
```

Output:

```
Informações do livro:
Título: Estrutura de Dados
Autor: Maurício
Ano: 2024
Preço: R$ 199.00
```

03

Estrutura Aninhada

Estrutura aninhada

Uma estrutura aninhada é, essencialmente, uma estrutura de dados que contém outra dentro de si, ou seja, uma estrutura de dados que pode ser acessada por meio de outra.

Estrutura aninhada

```c
#include <stdio.h>

struct Motor {
    int potencia;
    char tipo[20];
};

struct Carro {
    char modelo[50];
    int ano;
    struct Motor motor;
};

int main() {
    struct Motor motor1 = {200, "V8"};
    struct Carro carro1 = {"Mustang", 1968, motor1};

    printf("Modelo: %s\n", carro1.modelo);
    printf("Ano: %d\n", carro1.ano);
    printf("Potência do Motor: %d\n", carro1.motor.potencia);
    printf("Tipo de Motor: %s\n", carro1.motor.tipo);

    return 0;
}
```

Estrutura aninhada

```c
struct Motor {
    int potencia;
    char tipo[20];
};

struct Carro {
    char modelo[50];
    int ano;
    struct Motor motor;
};
```

Motor tem dois membros: potência (um inteiro) e tipo (uma string).

Carro tem três membros: modelo (uma string), ano (um inteiro), e motor (uma instância da struct Motor).

Estrutura aninhada

```c
int main() {
    struct Motor motor1 = {200, "V8"};
    struct Carro carro1 = {"Mustang", 1968, motor1};
```

Foi criado uma instância de Motor chamada motor1 com valores iniciais.

Criamos uma instância de Carro chamada carro1, incluindo motor1 como o valor para o membro motor.

Estrutura aninhada

```c
printf("Modelo: %s\n", carro1.modelo);
printf("Ano: %d\n", carro1.ano);
printf("Potência do Motor: %d\n", carro1.motor.potencia);
printf("Tipo de Motor: %s\n", carro1.motor.tipo);
```

Acessamos os membros da struct Carro diretamente (carro1.modelo, carro1.ano).

Acessamos os membros da struct aninhada Motor através da instância de Carro (carro1.motor.potencia, carro1.motor.tipo).

Estrutura aninhada

```c
struct Endereco {
    char rua[50];
    int numero;
    char cidade[50];
    char estado[3];
    char cep[10];
};

struct Departamento {
    char nome[50];
    struct Endereco endereco;
};

struct Universidade {
    char nome[100];
    struct Endereco endereco;
    struct Departamento departamentos[5];
};
```

Endereço: Contém informações sobre um endereço (rua, número, cidade, estado, cep).

Departamento: Contém o nome do departamento e um Endereço.

Universidade: Contém o nome da universidade, um Endereço, e um array de Departamentos.

UNIFAVIP wyden

Estrutura aninhada

```c
int main() {
    struct Endereco end_uni = {"Av. Principal", 1000, "Cidade Universitária", "SP", "123
    struct Endereco end_dep1 = {"Rua Secundária", 200, "Cidade Universitária", "SP", "12
    struct Endereco end_dep2 = {"Rua Terciária", 300, "Cidade Universitária", "SP", "123

    struct Departamento dep1 = {"Departamento de Computação", end_dep1};
    struct Departamento dep2 = {"Departamento de Matemática", end_dep2};

    struct Universidade universidade = {"Universidade Exemplo", end_uni, {dep1, dep2}};
```

São criadas instâncias de **Endereço** para a universidade e dois departamentos.
São criadas instâncias de **Departamento** para "Computação" e "Matemática" com seus respectivos endereços.
É criada uma instância de **Universidade** com seu endereço e os dois departamentos.

Estrutura aninhada

```
printf("Universidade: %s\n", universidade.nome);
printf("Endereço: %s, %d, %s, %s, %s\n", universidade.endereco.rua, universidade.ende

for (int i = 0; i < 2; i++) {
    printf("Departamento %d: %s\n", i + 1, universidade.departamentos[i].nome);
    printf("Endereço: %s, %d, %s, %s, %s\n", universidade.departamentos[i].endereco.r
}

return 0;
```

Os membros da struct **Universidade** são acessados diretamente (**universidade.nome, universidade.endereco**).
Os membros da struct aninhada **Endereço** dentro de **Universidade** são acessados (**universidade.endereco.rua, universidade.endereco.numero**).
Um loop **for** é usado para acessar os membros da array de structs departamentos dentro da Universidade.

Estrutura aninhada

As structs (estruturas) em C são úteis para agrupar variáveis de diferentes tipos sob um único nome, permitindo organizar e manipular dados de maneira estruturada.

1. Agrupar Dados Relacionados

Quando você tem um conjunto de dados relacionados que devem ser tratados como uma unidade lógica, struct é uma boa escolha.

2. Modelagem de Objetos do Mundo Real

Ao modelar objetos do mundo real, como um carro, um livro ou uma pessoa, as structs permitem encapsular todas as propriedades relevantes desses objetos.

Estrutura aninhada

3. Melhorar a Legibilidade do Código

Usar struct melhora a legibilidade do código ao permitir que variáveis relacionadas sejam agrupadas, tornando o código mais intuitivo e fácil de entender.

4. Trabalhar com Dados Complexos

Quando você está lidando com dados complexos que têm várias propriedades, como registros em um banco de dados, struct ajuda a organizar essas propriedades de maneira clara.

Estrutura aninhada

5. Passagem de Dados para Funções

struct pode ser usada para passar múltiplos dados relacionados para funções de uma maneira organizada, em vez de passar vários argumentos individuais.

6. Implementação de Estruturas de Dados

Algumas estruturas de dados, como listas ligadas, árvores e grafos, são naturalmente representadas usando struct.

Atividade

Crie um programa em C para gerenciar informações sobre uma biblioteca. O programa deve definir as structs necessárias para armazenar o endereço da biblioteca, que inclui rua, número, cidade, estado e CEP, e para representar um livro, que inclui título, autor, ano de publicação e ISBN. Além disso, crie uma struct **Biblioteca** que contenha o nome da biblioteca, o endereço (utilizando a struct **Endereco**) e um único livro (utilizando a struct **Livro**).

Na função main, crie e preencha uma instância de Biblioteca com um nome, um endereço e um livro, e por fim, imprima as informações da biblioteca e do livro armazenado.