in4mation HUB

Fundamentos de Programação

Introdução às Structs

in4mation HUB

O QUE SÃO STRUCTS?

Uma struct, ou estrutura em português, é um tipo de dado composto em C que permite agrupar diferentes tipos de dados em uma única variável.

in4mation HUB

VANTAGENS DE USAR STRUCTS

01 Organização

Permitem agrupar dados relacionados de forma lógica, tornando o código mais organizado e legível.

02 Reutilização

Permitem criar tipos de dados personalizados que podem ser reutilizados em diferentes partes do código.

03 Eficiência

Permitem criar tipos de dados personalizados que podem ser reutilizados em diferentes partes do código.

in4mation HUB

DIFERENÇA ENTRE STRUCTS E UNIONS

- Structs: Permitem agrupar dados de diferentes tipos em uma única variável, cada um com seu próprio nome e tipo de dado.

- Unions: Permitem armazenar diferentes tipos de dados em um único espaço de memória, mas apenas um dos tipos de dados pode ser acessado por vez.

in4mation HUB

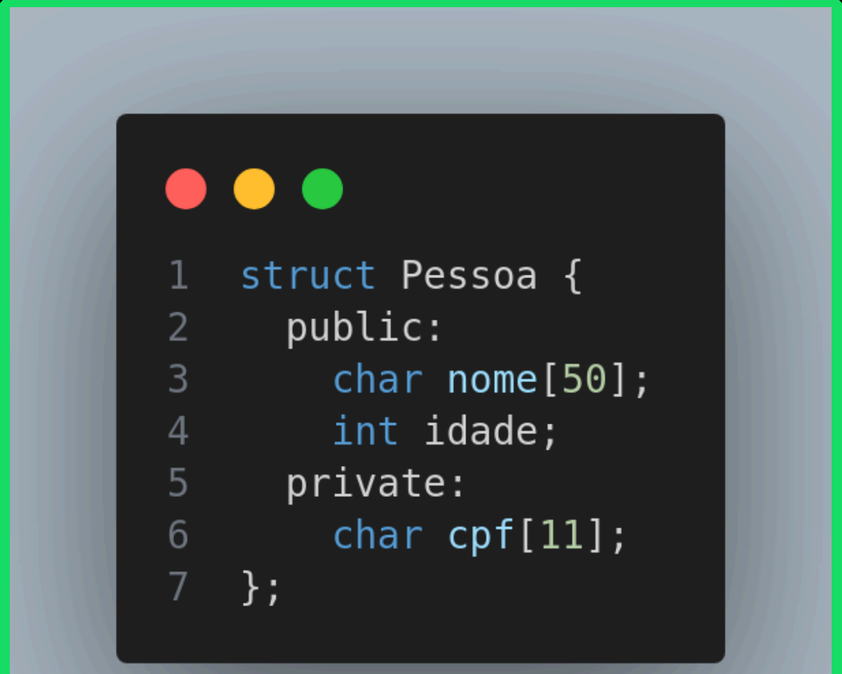
DECLARAÇÃO E ESTRUTURA DE UMA STRUCT

```
1  struct nome_da_struct {
2    tipo_de_dado1 membro1;
3    tipo_de_dado2 membro2;
4    ...
5  };
```

```
1  struct Ponto {
2    int x;
3    int y;
4  };
```

in4mation HUB

TIPOS DE DADOS E MODIFICADORES DE ACESSO



```c
struct Pessoa {
  public:
    char nome[50];
    int idade;
  private:
    char cpf[11];
};
```

Modificadores de acesso podem ser usados para controlar o acesso aos membros de uma struct:

- *public: acesso livre de qualquer lugar do código.*
- *private: acesso restrito à struct em que são declarados.*

in4mation HUB

DECLARAÇÃO E INICIALIZAÇÃO DE MEMBROS

```
1  struct Ponto p1 = {10, 20};
```

Os membros de uma struct podem ser declarados e inicializados da mesma forma que variáveis normais

in4mation HUB

ACESSANDO MEMBROS DE STRUCTS

```
1  int x = p1.x;
```

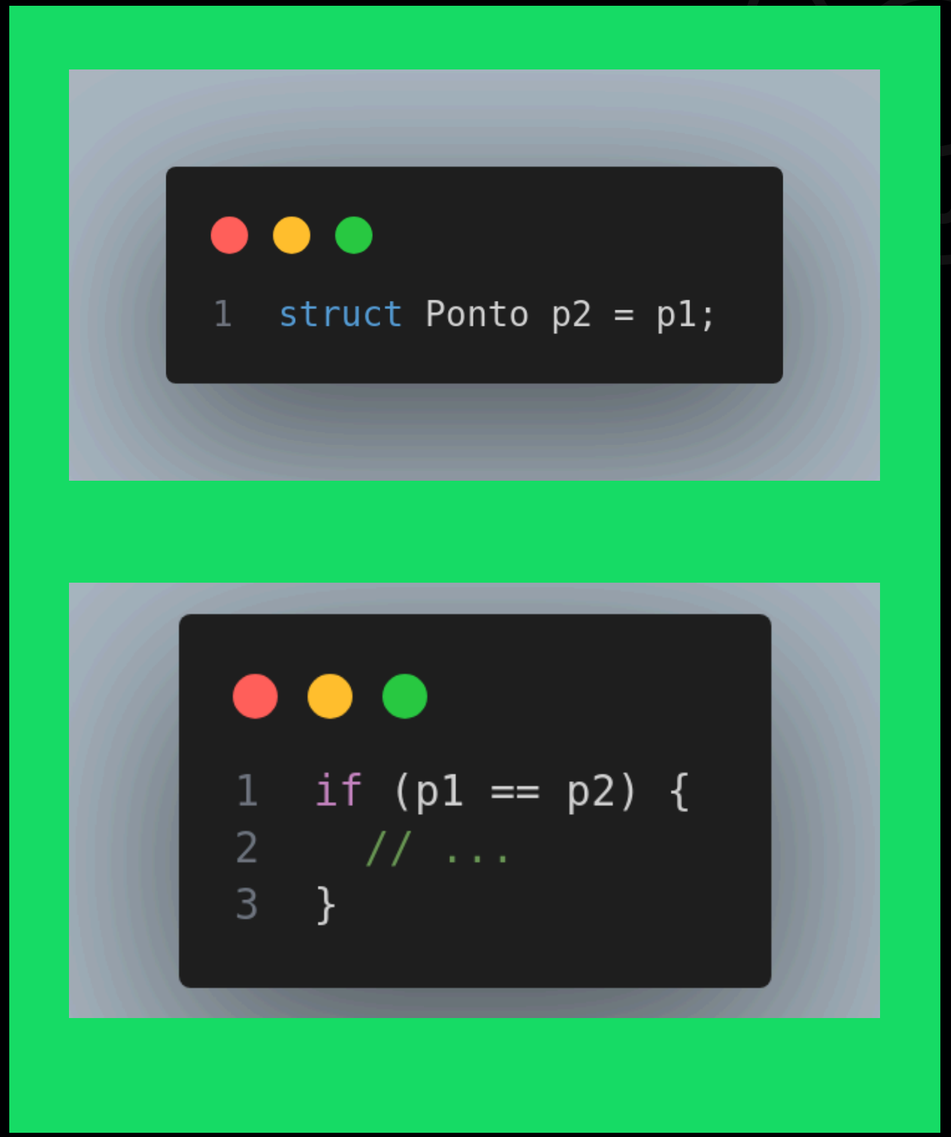
Os membros de uma struct podem ser acessados usando o operador `.`

in4mation HUB

ATRIBUIÇÃO E COMPARAÇÃO

Structs podem ser atribuídas umas às outras, copiando os valores de todos os seus membros

Também podem ser comparadas usando os operadores relacionais (`==`, `!=`, `<`, `>`, `<=`, `>=`). A comparação é feita membro a membro



in4mation HUB

CÓPIA

01

A função `memcpy()` copia o conteúdo bruto da memória, sem levar em conta os tipos de dados dos membros.

02

É importante usá-la com cautela e apenas quando se sabe exatamente o tamanho e a disposição dos membros na memória.

03

Para cópias seguras e que respeitem os tipos de dados, é recomendável criar uma função específica para cada struct que realize a cópia membro a membro.

in4mation HUB

EXEMPLOS PRÁTICOS

Representando um ponto no plano

```c
#include <stdio.h>

struct Ponto {
  int x;
  int y;
};

int main() {
  struct Ponto p1; // Declara uma variável do tipo Ponto

  // Acessando e modificando membros
  p1.x = 5;
  p1.y = 3;

  printf("Coordenadas do ponto: (%d, %d)\n", p1.x, p1.y);

  return 0;
}
```

in4mation HUB

EXEMPLOS PRÁTICOS

Representando uma data

```c
#include <stdio.h>

struct Data {
  int dia;
  int mes;
  int ano;
};

int main() {
  struct Data hoje;

  // Inicializando membros
  hoje.dia = 26;
  hoje.mes = 3;
  hoje.ano = 2024;

  printf("Data: %d/%d/%d\n", hoje.dia, hoje.mes, hoje.ano);

  return 0;
}
```

in4mation HUB

EXEMPLOS PRÁTICOS

Representando o cálculo da área de um retângulo

```c
#include <stdio.h>

struct Retangulo {
  int base;
  int altura;
};

int calcularArea(struct Retangulo ret) {
  return ret.base * ret.altura;
}

int main() {
  struct Retangulo ret1;

  ret1.base = 10;
  ret1.altura = 5;

  int area = calcularArea(ret1);

  printf("A area do retangulo e: %d\n", area);

  return 0;
}
```

in4mation HUB

DESAFIO

Crie uma struct chamada `Ponto` para representar um ponto no plano cartesiano, com membros `x` e `y`. Escreva um programa que declare uma variável do tipo `Ponto`, leia os valores `x` e `y` do usuário e imprima na tela.