Structs em C: Organizando o Caos

Uma forma inteligente de agrupar variáveis

O Problema do Mundo Real

Como guardar os dados de um personagem de jogo?

- **Jett (Valorant)**
  - Vida: 100
  - Escudo: 50
  - Ultimate: Pronta
  - Nome: "Jett"

Se usarmos variáveis soltas, o código vira uma bagunça...

O Código Bagunçado (O Jeito "Raiz")

```c
// Variáveis soltas para representar a Jett
float vida_jett = 100.0;
float escudo_jett = 50.0;
char nome_jett[20] = "Jett";

// Olha o problemão para passar pra uma função!
void imprimeStatus(float vida, float escudo, char nome[]) {
    // ...
}

// Chamada da função
imprimeStatus(vida_jett, escudo_jett, nome_jett);
```

A Revelação: A `struct` entra em cena!

E se pudéssemos criar nosso próprio tipo de variável, um tipo `Personagem`?

`variáveis soltas` ➡️ `struct Personagem`

É como colocar todas as peças relacionadas numa única caixa organizada.

O Código Organizado (Com Struct)

```c
// Agora temos uma "caixa" para o personagem
typedef struct {
    float vida;
    float escudo;
    char nome[20];
} Personagem;

// A função fica muito mais limpa!
void imprimeStatus(Personagem p) {
    // ...
}

// Declaração e chamada
Personagem jett = {100.0, 50.0, "Jett"};
imprimeStatus(jett);
```

Anatomia de uma `struct`

```c
typedef struct { // Palavra-chave para criar um novo tipo
    // --- Campos ou Membros ---
    char nome[50];
    float nota;
    int matricula;
    // ----------------------
} Aluno; // Nome do novo tipo que criamos

// Como usar:
Aluno joao; // 'joao' é uma variável do tipo Aluno
```

Acessando os Dados: O Ponto (`.`)

Para pegar ou alterar um campo dentro de uma `struct`, usamos o operador ponto (`.`). Ele significa "acesse o membro...".

```c
#include <stdio.h>
#include <string.h>

// ... (definição da struct Aluno)

Aluno joao;

// Atribuindo valores
strcpy(joao.nome, "Joao da Silva");
joao.matricula = 2025123;
joao.nota = 9.5;
```

Resumo da Ópera

- **Structs** são "caixas" para agrupar variáveis relacionadas.
- Elas criam **novos tipos de dados** personalizados em C.
- Usamos o **operador ponto (`.`)** para acessar os campos de uma variável `struct`.
- O principal benefício é a **organização** e a **limpeza** do código.

Próximos Passos!

Agora você já sabe organizar seus dados! Mas fica um desafio...

E se uma `struct` for GIGANTE? Copiar ela inteira toda vez que chamamos uma função pode deixar o programa lento.

Na próxima aula: Vamos resolver isso com a ferramenta mais poderosa do C: **Ponteiros**! Veremos como passar apenas o "endereço" da nossa `struct` em vez da cópia inteira.

#ateaproxima