Bing

Ok, eu vou tentar te dar mais detalhes técnicos e abstratos sobre STRUCTS em linguagem C. Aqui estão alguns tópicos avançados que você pode se interessar:

- Structs com ponteiros para si mesmas: Uma STRUCT pode ter um membro que é um ponteiro para o mesmo tipo da STRUCT, criando assim uma referência recursiva. Isso é útil para implementar estruturas de dados que têm nós que apontam para outros nós do mesmo tipo, como listas encadeadas, árvores binárias e grafos . Por exemplo:

```
struct no {
    int dado;
    struct no *prox; // ponteiro para o mesmo tipo da struct
};
```

- Structs com arrays de ponteiros: Uma STRUCT pode ter um membro que é um array de ponteiros para qualquer tipo de dado, inclusive para o mesmo tipo da STRUCT. Isso é útil para implementar estruturas de dados que têm nós que apontam para vários outros nós do mesmo ou de diferentes tipos, como árvores n-árias, tabelas hash e matrizes esparsas . Por exemplo:

```
struct aluno {
    char nome[50];
    int matricula;
    float *notas; // array de ponteiros para float
    struct aluno **amigos; // array de ponteiros para struct aluno
};
```

- Structs com funções: Uma STRUCT pode ter um membro que é um ponteiro para uma função, que pode ser usada para executar alguma operação sobre os dados da STRUCT ou sobre outras STRUCTS. Isso é útil para implementar o conceito de polimorfismo, que permite que uma função se comporte de forma diferente dependendo do tipo de dado que recebe como argumento . Por exemplo:

```
struct animal {
    char nome[50];
    int idade;
    void (*falar)(); // ponteiro para uma função que faz o animal falar
};

void falar_cachorro() {
    printf("Au au!\n");
}

void falar_gato() {
    printf("Miau!\n");
}

struct animal a1 = {"Rex", 5, falar_cachorro}; // cria um cachorro
struct animal a2 = {"Luna", 3, falar_gato}; // cria um gato

a1.falar(); // chama a função falar_cachorro
a2.falar(); // chama a função falar_gato
```

Espero que esses tópicos tenham sido esclarecedores para você. Se você quiser saber mais sobre STRUCTS em linguagem C, você pode consultar os seguintes sites:

- Struct em C - linguagem C
- Struct em C: O que é, para que serve e como usar na prática?
- C Progressivo: Structs em linguagem C - Tutorial