in4mation HUB

Fundamentos de Programação

Structs como Parâmetros de Funções

in4mation HUB

CÓPIA DE VALORES E MODIFICAÇÕES

```c
#include <stdio.h>

struct Ponto {
  int x;
  int y;
};

void modificarPonto(struct Ponto p) {
  p.x = 10;
}

int main() {
  struct Ponto p1 = {5, 3};

  modificarPonto(p1);

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

  return 0;
}
```

Ao passar uma struct como parâmetro por valor para uma função, uma cópia dos valores da struct é criada na memória da função. Modificações feitas na struct dentro da função não afetam a struct original na memória da função chamadora.

in4mation HUB

MODIFICANDO A STRUCT ORIGINAL

Ao passar uma struct como parâmetro por referência para uma função, o endereço de memória da struct original é passado para a função. Modificações feitas na struct dentro da função afetam a struct original na memória da função chamadora.

```c
#include <stdio.h>

struct Ponto {
  int x;
  int y;
};

void modificarPonto(struct Ponto *p) {
  p->x = 10;
}

int main() {
  struct Ponto p1 = {5, 3};

  modificarPonto(&p1);

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

  return 0;
}
```

in4mation HUB

PONTEIROS PARA STRUCTS

```c
struct Ponto {
  int x;
  int y;
};

void modificarPonto(struct Ponto *p) {
  // ...
}

int main() {
  struct Ponto p1;
  struct Ponto *ptr_ponto = &p1;

  modificarPonto(ptr_ponto);

  // ...
}
```

Para passar uma struct por referência, é necessário utilizar um ponteiro para a struct. O operador `&` é utilizado para obter o endereço de memória de uma variável.

in4mation HUB

EXEMPLOS PRÁTICOS

Função para calcular a área de um retângulo

```c
#include <stdio.h>

struct Retangulo {
  int base;
  int altura;
};

int calcularAreaRetangulo(struct Retangulo r) {
  return r.base * r.altura;
}

int main() {
  struct Retangulo ret = {5, 10};

  int area = calcularAreaRetangulo(ret);

  printf("Área do retângulo: %d\n", area);

  return 0;
}
```

in4mation HUB

EXEMPLOS PRÁTICOS

Função para trocar os valores de duas structs

```c
#include <stdio.h>

struct Ponto {
  int x;
  int y;
};

void trocarValores(struct Ponto *p1, struct Ponto *p2) {
  struct Ponto temp = *p1;
  *p1 = *p2;
  *p2 = temp;
}

int main() {
  struct Ponto ponto1 = {5, 3};
  struct Ponto ponto2 = {10, 2};

  printf("Valores iniciais:\n");
  printf("Ponto 1: (%d, %d)\n", ponto1.x, ponto1.y);
  printf("Ponto 2: (%d, %d)\n", ponto2.x, ponto2.y);

  trocarValores(&ponto1, &ponto2);

  printf("Valores trocados:\n");
  printf("Ponto 1: (%d, %d)\n", ponto1.x, ponto1.y);
  printf("Ponto 2: (%d, %d)\n", ponto2.x, ponto2.y);

  return 0;
}
```

in4mation HUB

DESAFIO

Escreva uma função chamada `trocaPonto` que receba como parâmetros duas structs do tipo `Ponto` e troque suas coordenadas `x` e `y`. Teste essa função no programa principal.