FIEP SESI SENAI IEL

Lógica de programação

Ponteiros C

Kleber Petry
kleber.petry@pr.senai.br

Conceito de ponteiro em C

Um ponteiro em C é uma variável que armazena o endereço de memória de outra variável. Em vez de armazenar diretamente um valor, um ponteiro contém a localização onde o valor está armazenado.

Declaração

Para declarar um ponteiro, utiliza-se o símbolo * antes do nome da variável. Por exemplo:

```
int *ptr;
```

Aqui, ptr é um ponteiro para um inteiro.

Atribuição de Endereço

Para atribuir um endereço a um ponteiro, usa-se o operador &, que retorna o endereço de uma variável. Por exemplo:

```
int var = 10;
ptr = &var;
```

Agora, ptr contém o endereço de var.

Acesso ao Valor Apontado

Para acessar o valor armazenado no endereço contido no ponteiro, utiliza-se o operador de desreferenciação *. Por exemplo:

printf("%d", *ptr); // Imprime o valor de var, que é 10

Benefícios do Uso de Ponteiros

- **Manipulação de Arrays e Strings:** Ponteiros são essenciais para trabalhar com arrays e strings, permitindo iteração eficiente e manipulação de dados.
- **Alocação Dinâmica de Memória:** Funções como malloc e free usam ponteiros para alocar e liberar memória dinamicamente.
- **Passagem por Referência:** Funções podem modificar diretamente as variáveis passadas como argumento, pois recebem os endereços dessas variáveis.

Exemplo

```
#include <stdio.h>
int main() {
        int var = 10;      // Variável normal
        int *ptr = &var;  // Ponteiro que armazena o endereço de var
        printf("Valor de var: %d\n", var);        // Imprime 10
        printf("Endereço de var: %p\n", &var);   // Imprime o endereço de var
```

Exemplo

```
        printf("Valor de ptr: %p\n", ptr);       //
Imprime o endereço de var (mesmo que &var)
        printf("Valor apontado por ptr: %d\n", *ptr); //
Imprime 10 (valor de var)    return 0;}
```

Atividade

- Crie um programa em C que declare uma variável do tipo int e um ponteiro para int.
- Atribua um valor à variável int e, em seguida, atribua o endereço dessa variável ao ponteiro.
- Imprima o valor da variável, o endereço da variável e o valor armazenado no ponteiro usando desferenciação.

Atividade

```
#include <stdio.h>
int main() {
        int var = 20;      // Declare uma variável do tipo int
        int *ptr = &var;  // Declare um ponteiro e atribua o endereço de var
        printf("Valor de var: %d\n", var);          // Imprime o valor de var
```

Atividade

```
        printf("Endereço de var: %p\n", &var);       // Imprime o endereço de var
        printf("Valor armazenado em ptr: %p\n", ptr); // Imprime o valor (endereço) armazenado em ptr
        printf("Valor apontado por ptr: %d\n", *ptr); // Imprime o valor apontado por ptr (desreferenciação)    return 0;}
```

Atividade

```
        printf("Endereço de var: %p\n", &var);       // Imprime o endereço de var
        printf("Valor armazenado em ptr: %p\n", ptr); // Imprime o valor (endereço) armazenado em ptr
        printf("Valor apontado por ptr: %d\n", *ptr); // Imprime o valor apontado por ptr (desreferenciação)    return 0;}
```

Atividade

- Crie um programa em C com uma função que recebe um ponteiro para int como argumento e modifica o valor da variável apontada.
- No main(), declare uma variável int e um ponteiro para int. Atribua o endereço da variável ao ponteiro e passe o ponteiro para a função.
- Imprima o valor da variável antes e depois da chamada da função para verificar a modificação.

Atividade

```
#include <stdio.h>
void modifyValue(int *ptr) {
        *ptr = 100; // Modifica o valor da variável apontada pelo ponteiro
}
int main() {
        int var = 20;      // Declare uma variável do tipo int
        int *ptr = &var;  // Declare um ponteiro e atribua o endereço de var
```

Atividade

```
    printf("Valor de var antes da função: %d\n", var); // Imprime o valor de var antes da função
    modifyValue(ptr); // Chama a função para modificar o valor de var
    printf("Valor de var depois da função: %d\n", var); // Imprime o valor de var depois da função
    return 0;
}
```