Programação de Computadores

Linguagem C

Tipo Ponteiro

- Um ponteiro em C é uma **variável que armazena o endereço de memória**.
- O ponteiro **"aponta"** (armazena o endereço de memória) para a localização na memória onde os dados estão armazenados.
- Isso **permite manipular e acessar eficientemente a memória**, o que é especialmente útil para trabalhar com:
  - **arrays**,
  - **alocação dinâmica de memória** e
  - **passagem de parâmetros por referência**.
- Ao passar um ponteiro como argumento para uma função, a função pode acessar e modificar o valor da variável original usando o operador de desreferência.

Programação de Computadores

Linguagem C

Tipo Ponteiro

Vamos explorar os principais conceitos e usos dos ponteiros:

1.Declaração de um ponteiro:

- Para declarar um ponteiro, utilizamos o **operador asterisco (*) após o tipo de dado**. Por exemplo, **int*** ponteiro; declara um ponteiro para um inteiro.
- É importante inicializar o ponteiro antes de utilizá-lo, atribuindo a ele o endereço de memória de uma variável válida. Caso contrário, o ponteiro conterá um valor indeterminado ou nulo. **Exemplo: int* ponteiro1;**

2.Operador de endereço **(&)**:

- O operador de endereço **(&) é usado para obter o endereço de memória de uma variável** existente. Por exemplo, **&variavel** retorna o endereço de memória da variável variavel.
- Ao atribuir o endereço de uma variável a um ponteiro, estamos dizendo ao ponteiro para apontar para essa variável. **Exemplo: ponteiro1 = №**

Programação de Computadores

Linguagem C

Tipo Ponteiro

Vamos explorar os principais conceitos e usos dos ponteiros:

3.Operador de desreferência (*):

- O operador de desreferência **(*) é usado para acessar o valor armazenado no endereço de memória apontado por um ponteiro**.
- Por exemplo, *ponteiro retorna o valor armazenado no endereço de memória apontado por ponteiro. **Exemplo: *ponteiro2 = 100;**

4.Manipulação de arrays:

- Os ponteiros são frequentemente utilizados para manipular arrays em C. Um ponteiro pode apontar para o primeiro elemento de um array.
- Ao incrementar ou decrementar um ponteiro, podemos navegar pelos elementos de um array.
- O operador de desreferência pode ser usado para acessar e modificar os valores dos elementos do array.

Programação de Computadores

Linguagem C

Tipo Ponteiro

Vamos explorar os principais conceitos e usos dos ponteiros:

5.Alocação dinâmica de memória:

- Os ponteiros são usados para alocar e liberar memória dinamicamente durante a execução do programa.
- As funções **malloc( )**, **calloc( )**, **realloc( )** e **free( )** são utilizadas para alocar e liberar blocos de memória dinamicamente, retornando e recebendo ponteiros.

6.Passagem de parâmetros por referência:

- Os ponteiros permitem passar parâmetros por referência para funções em C.
- Ao passar um ponteiro como argumento para uma função, a função pode acessar e modificar o valor da variável original usando o operador de desreferência.

Programação de Computadores

Linguagem C

Tipo Ponteiro

- É importante ter cuidado ao usar ponteiros para evitar erros:
  - como desreferenciar um ponteiro não inicializado,
  - acessar memória desalocada
  - ou ultrapassar os limites de um array.
- Esses erros podem levar a comportamentos indefinidos e falhas no programa.
- Os ponteiros são uma característica poderosa da linguagem C, permitindo o gerenciamento eficiente de memória e a manipulação de estruturas de dados complexas.
- São amplamente utilizados em tarefas como alocação dinâmica de memória, manipulação de strings, implementação de estruturas de dados e acesso direto a hardware.

Programação de Computadores

Linguagem C

Tipo Ponteiro - Exemplo de uso dos operadores * e &

```c
#include <stdio.h>
int main() {
    int numero = 42;
    int *ponteiro1;
    int *ponteiro2;
    // Atribuição do endereço de memória de 'numero' aos ponteiros
    ponteiro1 = &numero;
    ponteiro2 = &numero;
    // Exemplo de uso do operador '*'
    printf("Valor apontado por ponteiro1: %d\n", *ponteiro1);
    // Exemplo de uso do operador '&'
    printf("Endereço de memória de 'numero': %p\n", &numero);
    // Modificando o valor através de um ponteiro
    *ponteiro2 = 100;
    // Impressão do novo valor de 'numero'
    printf("Novo valor de 'numero': %d\n", numero);
    return 0;
}
```

Programação de Computadores

Linguagem C

Tipo Ponteiro - Exemplo de manipulação de array com ponteiro

```c
#include <stdio.h>
int main() {
    int numeros[] = {10, 20, 30, 40, 50};
    int *ponteiro;
    // Atribuição do endereço do primeiro elemento do array ao ponteiro
    ponteiro = numeros;
    // Percorrendo o array e imprimindo seus elementos usando ponteiros
    for (int i = 0; i < 5; i++) {
        printf("Elemento %d: %d\n", i, *ponteiro);
        ponteiro++;
    }
    return 0;
}
```

Programação de Computadores

Linguagem C

Tipo Ponteiro - Exemplo de alocação dinâmica de memória.

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    int tamanho;
    int *numeros;
    printf("Digite o tamanho do array: ");
    scanf("%d", &tamanho);
    // Alocação de memória usando a função malloc()
    numeros = (int *)malloc(tamanho * sizeof(int));
    if (numeros == NULL) {
        printf("Falha na alocação de memória.\n");
        return 1;
    }
    // Preenchimento do array
    for (int i = 0; i < tamanho; i++) {
        numeros[i] = i + 1;
    }
    // Impressão do array
    printf("Array criado: ");
    for (int i = 0; i < tamanho; i++) {
        printf("%d ", numeros[i]);
    }
    printf("\n");
    // Redimensionamento do array usando a função realloc()
    int novoTamanho;
    printf("Digite o novo tamanho do array: ");
    scanf("%d", &novoTamanho);
    numeros = (int *)realloc(numeros, novoTamanho * sizeof(int));
    if (numeros == NULL) {
        printf("Falha na realocação de memória.\n");
        return 1;
    }
    // Impressão do array após o redimensionamento
    printf("Array redimensionado: ");
    for (int i = 0; i < novoTamanho; i++) {
        printf("%d ", numeros[i]);
    }
    printf("\n");
    // Liberação de memória usando a função free()
    free(numeros);
    return 0;
}
```

Programação de Computadores

Linguagem C

Tipo Ponteiro - Alocação dinâmica de memória

1.Função **malloc( ):**

- A função **malloc( )** é usada para alocar memória dinamicamente em C.
- Ela recebe um único argumento que representa o tamanho do bloco de memória a ser alocado, especificado em bytes.
- A função retorna um ponteiro para o início do bloco de memória alocado ou **NULL** em caso de falha na alocação.
- No exemplo, a função **malloc( )** é usada para alocar memória para um array de inteiros com o tamanho especificado pelo usuário.

2.Função **calloc( )**:

- A função **calloc( )** é usada para alocar memória e inicializar todos os bytes com zero.
- Ela recebe dois argumentos: o número de elementos a serem alocados e o tamanho de cada elemento em bytes.
- No exemplo, a função **calloc( )** é usada para alocar memória para um array de inteiros com o tamanho especificado pelo usuário. **numeros = (int *)calloc(tamanho, sizeof(int));**

Programação de Computadores

Linguagem C

Tipo Ponteiro - Alocação dinâmica de memória

3.Função **realloc( )**:

- A função **realloc( )** é usada para redimensionar um bloco de memória previamente alocado.
- Ela recebe dois argumentos: o ponteiro para o bloco de memória existente e o novo tamanho em bytes.
- No exemplo, a função **realloc( )** é usada para redimensionar o array de acordo com o novo tamanho especificado pelo usuário.

4.Função **free( )**:

- A função **free( )** é usada para liberar a memória alocada dinamicamente.
- Ela recebe um único argumento: o ponteiro para o bloco de memória a ser liberado.
- No exemplo, a função **free( )** é usada para liberar a memória alocada para o array quando não precisamos mais dela.