

Algoritmos e Estrutura de Dados I

Explorando os Ponteiros em C

Profa. Dra. Rosana Rego

Sumário

Introdução aos Ponteiros

Usando Ponteiros

Declarando Ponteiros

Operadores de Ponteiros

Usando Funções com Ponteiros

Conclusão

Introdução aos Ponteiros

Os ponteiros são uma característica importante da linguagem C, permitindo ao programador manipular valores e endereços de memória.

Eles permitem a transferência de dados entre funções, permitindo ao programador acessar e alterar dados de outras partes do programa.

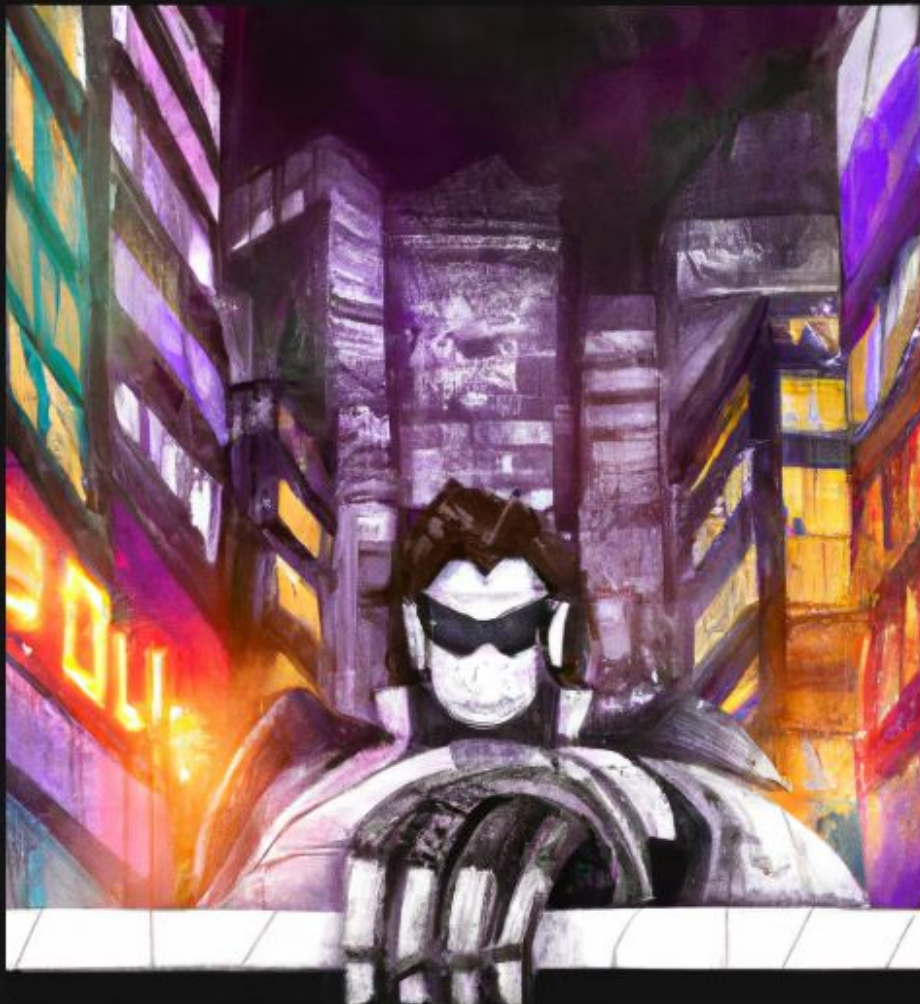


Endereço de uma variável

Local onde a mesma é armazenada em memória.

Os ponteiros são usados para acessar endereços de memória, permitindo a transferência de dados entre funções e diferentes partes do programa.

Esses endereços de memória são usados para acessar o conteúdo de variáveis, permitindo ao programador manipular os dados associados a elas.



Operador de endereço &

O operador de endereço & é usado para acessar o endereço de memória de uma variável, permitindo ao programador manipular os dados associados a ela.

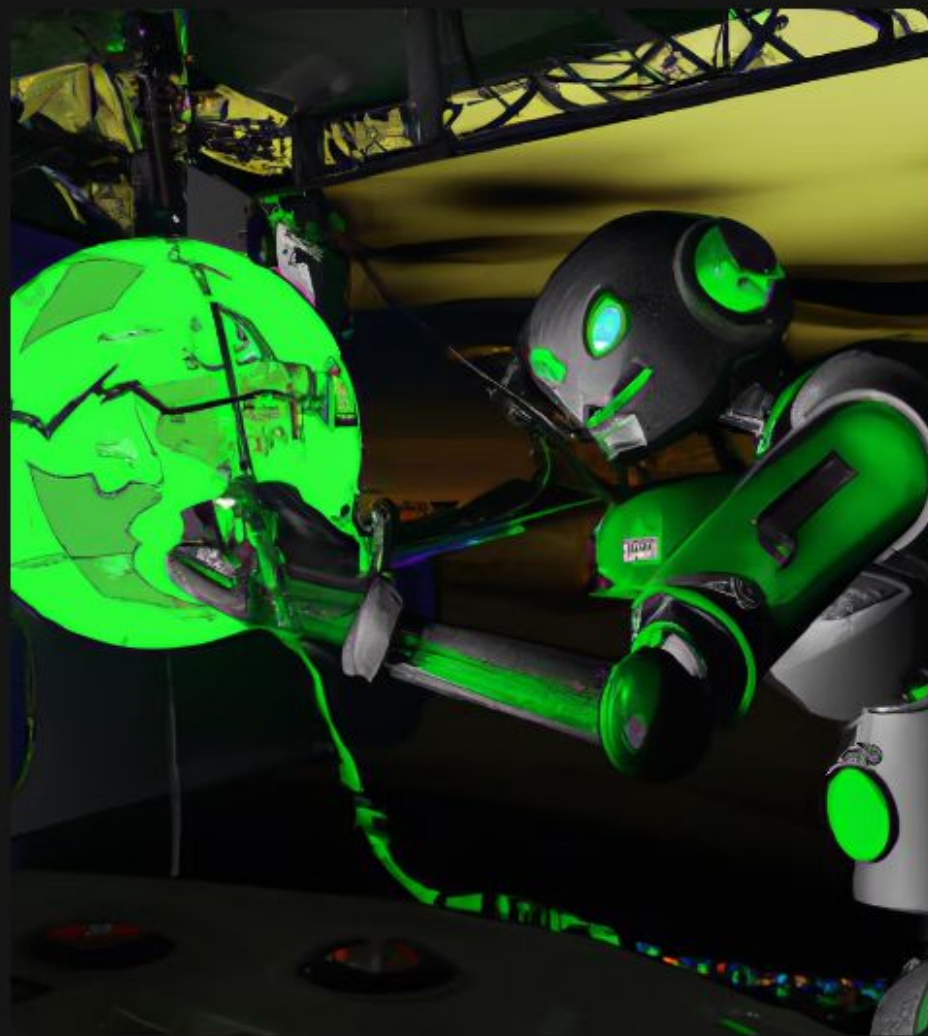
Não é permitido modificar o endereço de uma variável por meio de uma atribuição.



Usando Ponteiros

Os ponteiros são usados para acessar e manipular dados de uma maneira mais eficiente.

Eles são usados para passar informações entre funções e também para realizar operações sobre dados armazenados em memória.



Declarando Ponteiros

Os ponteiros são declarados usando o operador `*` seguido do tipo de dados específico.

Eles podem ser usados para acessar e alterar os dados armazenados em memória.



Implementação: Ponteiros em C

Declarando ponteiros

```
tipo_apontado * nome_da_variavel_ponteiro;
```

```
int * point;
```

Imprimindo o valor de um ponteiro

```
printf("O ponteiro: %p \n", point);
```



Acesso ao endereço de uma variável

```
int numero = 2;
```

```
int * ponteiro;
```

```
ponteiro = &numero;
```



Indireção de Ponteiros com *

Os ponteiros podem ser usados para acessar e manipular os dados armazenados em um endereço de memória usando o operador *.

Usando o operador *, podemos acessar o conteúdo da variável referenciada pelo ponteiro, permitindo acesso direto ao conteúdo da variável.

Exemplo:

```
float var = 3.14;  
float *point = &var;  
float pi = *point;  
*point = 1.31456;
```



Operadores de Ponteiros

Os operadores de ponteiros permitem ao programador acessar e manipular dados armazenados em memória.

Eles permitem ao programador ler, escrever e alterar os dados armazenados em memória.



O Ponteiro Nulo NULL

Um ponteiro nulo é um ponteiro que não aponta para nenhum objeto.

É usado para indicar ausência de um valor válido.

```
#include <stdlib.h>
```

```
char * point;
```

```
point = NULL;
```



Compatibilidade e conversões entre ponteiros

Ponteiros são versáteis, permitindo aos programadores criar algoritmos complexos e modernos.

A conversão entre diferentes tipos de ponteiros pode ser usada para acessar memória e dados de forma segura.

```
float number;
```

```
int * pointer;
```

```
pointer = &number;
```

```
pointer = (int*) &number;
```



Aritmética de Ponteiros

A aritmética de ponteiros permite que os programadores manipulem memória e dados usando operações matemáticas simples.

O cálculo de deslocamento de ponteiros pode ser usado para acessar dados específicos em estruturas de dados complexas.



Operações aritméticas

Considerando que p é um ponteiro.

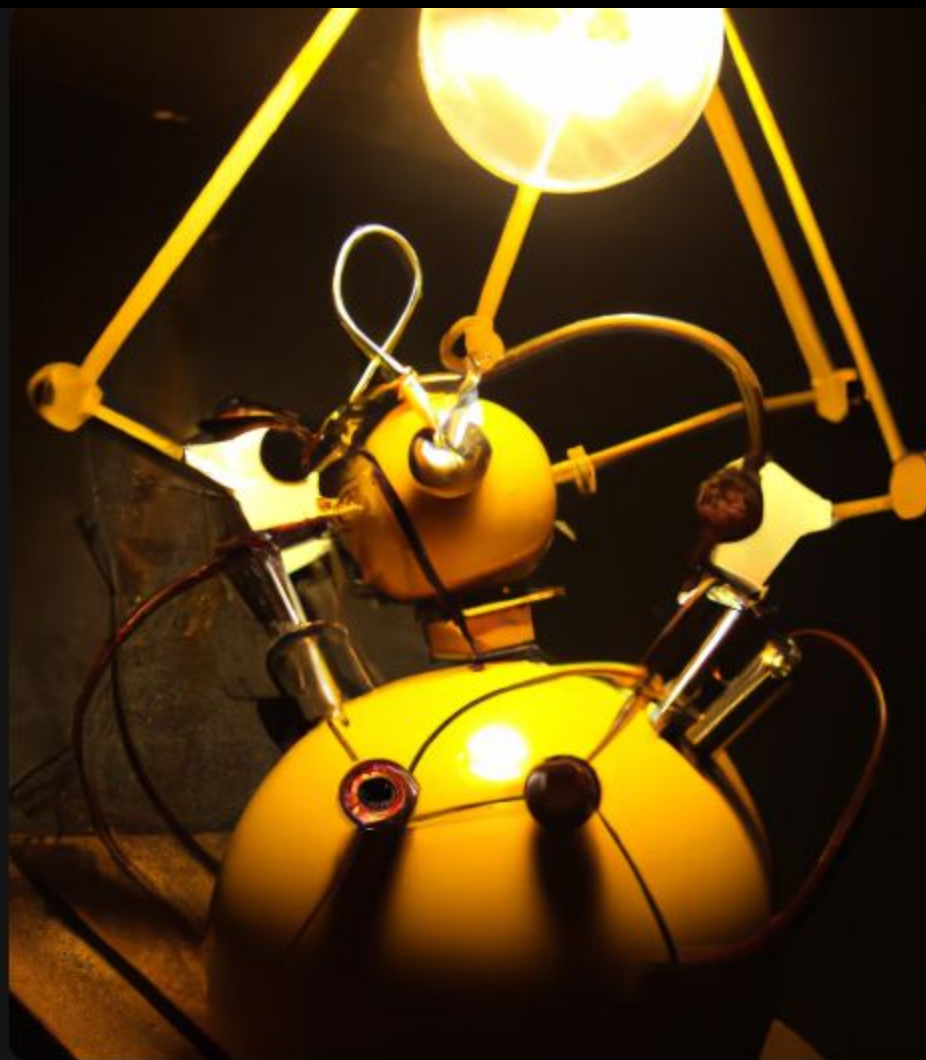
- Soma de um inteiro a uma ponteiro
 - $p+2$
- Subtração de um inteiro a um ponteiro
 - $p-3$
- Incremento de ponteiro
 - $++p$ ou $p++$
- Decremento de ponteiro
 - $--p$ ou $p--$
- Subtração entre dois ponteiros do mesmo tipo
 - $p - p2$



Exemplo:

```
int * pointer;
```

```
pointer+3 /* representa o endereço da  
posição de memória que está três objetos  
do tipo int adiante do endereço do  
objeto para o qual pointer aponta. */
```



Exercício

Determine se as expressões a seguir representam expressões aritméticas legais ou ilegais:

```
long *p1, *p2;  
int j;  
char *p3;
```

```
p2 = p1 + 4;  
j = p2 - p1;  
j = p1 - p2;  
p1 = p2 - 2;  
p3 = p1 - 1;  
j = p1 - p3;
```

```
long *p1, *p2;  
int j;  
char *p3;
```

```
p2 = p1 + 4;    /* Legal */  
j = p2 - p1;    /* Legal – resultado: j recebe 4 */  
j = p1 - p2;    /* Legal – resultado: j recebe -4 */  
p1 = p2 - 2;    /* Legal – os ponteiros são compatíveis */  
p3 = p1 - 1;    /* Legal, mas os ponteiros não são compatíveis */  
j = p1 - p3;    /* ILEGAL – os ponteiros não são compatíveis */
```

Usando Funções com Ponteiros

As funções em C podem trabalhar com ponteiros de várias maneiras.

Uma das maneiras é passar ponteiros como argumentos para uma função. Isso permite que a função acesse e modifique o valor de uma variável fora de sua própria escopo.



Usando Funções com Ponteiros

```
void incrementa(int *ponteiro) {  
    (*ponteiro)++;  
}  
  
int main() {  
    int variavel = 5;  
    incrementa(&variavel);  
    printf("Valor da variavel: %d",  
variavel);  
    return 0;  
}
```



Usando Funções com Ponteiros

Usar ponteiros como retorno de função. Isso permite que uma função retorne um endereço de memória para uma variável. Por exemplo:

```
int *cria_variavel() {  
    int variavel = 5;  
    return &variavel;  
}
```

```
int main() {  
    int *ponteiro = cria_variavel();  
    printf("Valor da variavel: %d",  
        *ponteiro);  
    return 0;  
}
```



Ponteiros de Funções

Os ponteiros de função são usados para apontar para uma função específica, permitindo que a função possa ser chamada diretamente.

Os ponteiros de função também permitem que a função seja passada como parâmetro para outras funções.

Para declarar um ponteiro de função, você deve especificar o tipo de retorno da função seguido do nome do ponteiro e o tipo de parâmetros entre parênteses. Por exemplo:

```
int (*ponteiro)(int, int);
```

```
/*Isso declara um ponteiro chamado "ponteiro" que é um ponteiro  
para uma função que tem um tipo de retorno inteiro e dois  
parâmetros inteiros.*/
```



Ponteiros de Funções

```
int soma(int a, int b) {  
    return a + b;  
}  
  
int calcula(int x, int y, int (*operacao)(int, int)) {  
    return (*operacao)(x, y);  
}  
  
int main() {  
    int resultado = calcula(5, 3, soma);  
    printf("Resultado: %d", resultado);  
    return 0;  
}
```



Conclusão

Os ponteiros são uma característica importante da linguagem C, permitindo ao programador manipular valores e endereços de memória.

Eles permitem a transferência de dados entre funções, permitindo ao programador acessar e alterar dados de outras partes do programa.

