INSTITUTO FEDERAL
Catarinense
Campus Blumenau

Bacharelado em
ENGENHARIA
ELÉTRICA

Algoritmos - Ponteiros

Tarefa - Algoritmo - Ponteiros

1 Ponteiros

O conceito de ponteiros é um recurso poderoso da linguagem C, basicamente todas as variáveis são "nomes" para posições na memória, é muito mais fácil guardar um nome do que um endereço de memória, ou seja, é mais fácil trabalhar com uma área de armazenamento de dados que é identificada por "diametro" do que um número de endereço do tipo `0xabcdabcd`.

Uma variável do tipo ponteiro aponta para uma posição de memória e permite recuperar ou armazenar os dados naquela posição.

Manipulações básicas com ponteiros envolvem três operações. A primeira é a declaração ou criação do ponteiro, a segunda é a obtenção de um endereço de uma variável e a terceira é o acesso a dado da região de memória da qual o ponteiro aponta. Entretanto essas três operações envolvem apenas dois símbolos, ou seja, um mesmo símbolo será utilizado para duas operações e o contexto da escrita da programação ditará o qual é a operação sendo realizada no momento.

Para criar um ponteiro deve se escolher um tipo que pode ser um tipo básico (algum tipo inteiro ou ponto flutuante), uma `struct`, uma `union`, uma função, um array ou até mesmo um ponteiro. O operador `*` será interpretado como declaração de um ponteiro quando for utilizado na declaração de uma variável do tipo ponteiro. Um exemplo de criação de ponteiro é `int *a = 0`, isso criará um ponteiro apontando para 0 ou também conhecido como ponteiro nulo.

Para obter o endereço de uma variável utiliza-se o operador `&`, toda vez que o operador `&` preceder o nome de uma variável ele retornará o endereço da variável. Também é possível manipular endereços de funções, porém isso não será tratado neste momento.

Para acessar o valor da memória apontada pelo ponteiro deve-se utilizar o operador `*` antes da variável do tipo ponteiro, ou seja, utilizar `*` antes do ponteiro. Repare que um ponteiro também é uma variável, porém ele armazena um endereço.

O próximo código ilustra um estudo sobre ponteiros.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     // declando um inteiro, ou seja, uma posição na memória para
```

```c
    // armazenar um valor inteiro.
    int n = 0;

    // imprimir o valor que está armazenado em n
    printf("o valor de n é: %d\n", n);

    // declarando um ponteiro para um inteiro, ou seja, uma posição
    // na memória que armazenará o endereço de um dado do tipo inteiro.
    // este ponteiro será inicialmente nulo.
    int *p_n = 0;

    // imprimir o endereço armazenado em p_n
    printf("o endereço armazenado por p_n é: %p\n", p_n);

    // obter o endereço da variável nomeada como n e armazenar
    // na variável nomeada de p_n. Em outras palavras, guardar
    // o endereço da variável n no ponteiro para n.
    p_n = &n;

    // imprimir o endereço de n, o endereço de p_n e o endereço armazenado
    // por p_n. Não confunda! p_n é uma variável como qualquer outra e
    // também possui um endereço, a diferença é que ela além de possuir
    // um endereço também armazenará um endereço. Uma variável comum
    // possui um endereço e guarda um valor.
    printf("o endereço de n é: %p\n", &n);
    printf("o endereço de p_n é: %p (cuidado, aqui é o endereço ", &p_n) ;
    printf("de p_n e não o endereço guardado por p_n)\n");

    // repare que o endereço armazenado em p_n é o endereço de n
    printf("o endereço armazenado em p_n é: %p (repare que aqui ", p_n);
    printf("é o endereço da variável n)\n");

    // armazenar um valor em p_n, ou seja, armazenar um valor
    // na região de memória apontada por p_n, ou seja, armazenar
    // um valor através de p_n

    *p_n = 10;

    // imprimir o novo valor de n
    printf("o valor de n é: %d\n", n);

    // imprmir o valor de n através de p_n

    printf("o valor de n através de p_n é: %d\n", *p_n);

    // ler o valor de n, multiplicar por 2 e guardar novamente em n, tudo
    // através de p_n
    *p_n = *p_n * 2;
    printf("o valor de n através de p_n é: %d\n", *p_n);
}
```

Programa 1: Ponteiro em C

Os livros [Paul Deitel, 2022] e [Brian W. Kernighan, 1988] podem ser utilizados para obter mais informações sobre a estrutura de repetição `for`.

2 Exercícios

1. Compile o programa do código acima, execute-o e compare as saída dos `printf`s com as linhas do código. Você compreendeu a diferença entre o endereço armazenado por um ponteiro e o endereço de um ponteiro?

2. Faça um programa para cálcular o valor de uma função de segundo grau do tipo ax^2+bx+c para um determinado x. As variáveis devem ser acessadas somente através de ponteiros.

Referências

[Brian W. Kernighan, 1988] Brian W. Kernighan, D. M. R. (1988). *C Programming Language*. Prentice Hall, 2 edition.

[Paul Deitel, 2022] Paul Deitel, H. D. (2022). *C How to Program*. Pearson, 9 edition.