Estrutura de Dados Avançada

Daniel de Sousa Moraes
danielmoraes14@gmail.com

Revisão

- Endereços
- Ponteiros
- Pilha
- Fila
- Listas

Endereços

- A memória é uma sequência de ***bytes***.
- Cada byte armazena um de 256 possíveis valores
- Os bytes são numerados sequencialmente e o número de um byte é o seu endereço (= address).
- Cada objeto ocupa um certo número de bytes consecutivos na memória.
  - Char = 1 byte
  - Int = 4 bytes
  - Double = 8 bytes

Endereços

- Cada objeto possui um endereço na memória
- Em geral, o endereço de um objeto é o endereço do seu primeiro byte.

```
char c;
int i;
struct {
   int x, y;
} ponto;
int v[4];
```

```
c       89421
i       89422
ponto   89426
v[0]    89434
v[1]    89438
v[2]    89442
```

Endereços

- O endereço de um objeto (como uma variável, por exemplo) é dado pelo operador &. Se i é uma variável então &i é o seu endereço.

- Um exemplo: O segundo argumento da função de biblioteca scanf é o endereço da variável onde deve ser depositado o objeto lido do dispositivo padrão de entrada:

```
int i;
scanf ("%d", &i);
```

Ponteiros

- é um tipo especial de variável que armazena endereços
- C permite que o programador referencie a posição de objetos bem como os próprios objetos
- se x for declarado como um inteiro, &x se referirá à posição reservada para conter x. &x é chamado ponteiro.
- É possível declarar uma variável cujo tipo de dado seja um ponteiro e cujos possíveis valores sejam posições de memória.

Ponteiros

```
int *pi;
float *pf;
char *pc;
```

- um ponteiro é como qualquer outro tipo de dado em C.

- O valor de um ponteiro é uma posição de memória da mesma forma que o valor de um inteiro é um número.

- Os valores dos ponteiros podem ser atribuídos como quaisquer outros valores. Por exemplo, a declaração pi = &x atribui um ponteiro para o inteiro x à variável ponteiro pi.

Ponteiros

- A notação *pi em C refere-se ao inteiro na posição referenciada pelo ponteiro pi.

- A declaração x = *pi atribui o valor deste inteiro à variável inteira x

- Se pi é um ponteiro para um inteiro, então pi + 1 é o ponteiro para o inteiro imediatamente seguinte ao inteiro *pi em memória, pi - 1 é o ponteiro para o inteiro imediatamente anterior a *pi

Ponteiros

- Por exemplo, suponha que o valor de pi seja 100 (isto é, pi aponta para o inteiro *pi na posição 100)
- Sendo assim, o valor de pi - 1 é 96, o valor de pi +1 é 104
- O valor de *(pi - 1) é o conteúdo dos quatro bytes, 96, 97, 98 e 99, interpretado como um inteiro;
- o valor de *(pi + 1) é o conteúdo dos bytes 104, 105, 106 e 107, interpretado como um inteiro;

Ponteiros

- Observe também a diferença entre *pi + 1, que se refere a 1 somado ao inteiro *pi, e *(pi + 1), que se refere ao inteiro posterior ao inteiro na posição pi.

Uso de Ponteiros

- os ponteiros de C desempenham um notável papel é na passagem de parâmetros para funções.

- Normalmente, os parâmetros são passados para uma função em C por valor

- Se o valor de um parâmetro for alterado dentro da função, o valor no programa de chamada não será modificado

Uso de Ponteiros

- os ponteiros de C desempenham um notável papel é na passagem de parâmetros para funções.

- Normalmente, os parâmetros são passados para uma função em C por valor

- Se o valor de um parâmetro for alterado dentro da função, o valor no programa de chamada não será modificado

Referências

- AUGENSTEIN, Moshe J.; LANGSAM, Yedidyah; TENENBAUM, Aaron M. Estruturas de dados usando C. São Paulo, 1995.

- https://www.ime.usp.br/~pf/algoritmos/aulas/pont.html

Uso de Ponteiros

- os ponteiros de C desempenham um notável papel é na passagem de parâmetros para funções.

- Normalmente, os parâmetros são passados para uma função em C por valor

- Se o valor de um parâmetro for alterado dentro da função, o valor no programa de chamada não será modificado

Uso de Ponteiros

- os ponteiros de C desempenham um notável papel é na passagem de parâmetros para funções.

- Normalmente, os parâmetros são passados para uma função em C por valor

- Se o valor de um parâmetro for alterado dentro da função, o valor no programa de chamada não será modificado