Arrays, Interação e Invariantes

Por:

- **Dr. Alfredo Covele**
- **Eng. Cristiliano Maculuve**

Agosto 2023

Menu

No.	Conteúdo
1	Arrays
2	Ciclos e Interação
3	Invariantes
4	Exercícios

Arrays

- Na ciência da computação, a maneira óbvia de armazenar uma coleção ordenada de itens é de forma de array;
- Os itens da array são normalmente armazenados em uma sequência de localizações da memória do computador, mas para discuti-los, precisamos de uma maneira conveniente de anotá-los no papel.

Arrays

- Podemos simplesmente escrever os itens em ordem, separados por vírgulas e entre parenteses rectos. Nesse caso

  [1, 4, 17, 3, 90, 79, 4, 6, 81]

é exemplo de um *array* de números inteiros. Se designarmos o *array* como sendo *array a,* iremos representar da seguinte forma:

a = [1, 4, 17, 3, 90, 79, 4, 6, 81]

Arrays

- Este *array a* tem 9 itens e, portanto, dizemos que seu tamanho é 9. Na vida cotidiana, geralmente começamos a contar a partir de 1.
- Quando trabalhamos com *arrays* em ciência da computação, no entanto, com mais frequência (embora nem sempre) começamos do zero.
- Assim, para o nosso *array a*, suas posições são 0, 1, 2,. . . , 7, 8. O elemento na 8ª posição é 81, e usamos a notação *a[8]* para denotar esse elemento.

Arrays

- No geral, para qualquer inteiro *i* denotando uma posição, escrevemos *a[i]* para denotar o elemento na i-ésima posição.
- Esta posição *i* é chamada de *índice* (e o plural são *índices*). Então, no exemplo anterior, *a[0] = 1, a[1] = 4, a[2] = 17* e assim por diante.
- Na maioria das linguagens de programação modernas, = denota atribuição, enquanto a igualdade é expressa por ==.

Arrays

- Normalmente usaremos = em seu significado matemático, a menos que seja escrito como parte do código ou pseudocódigo.
- Dizemos que os itens individuais *a[i]* no *array a* são acessados usando seu índice *i*, e pode-se mover sequencialmente através do *array* aumentando ou diminuindo esse índice, ou saltar direto para um item específico dado seu valor de índice.

Arrays

- Algoritmos que processam dados armazenados como *array* normalmente precisam visitar sistematicamente todos os itens no *array* e aplicar as operações apropriadas sobre eles.

Ciclos e Interações

- A abordagem padrão na maioria das linguagens de programação para repetir um processo certo número de vezes, como mover-se sequencialmente através de um *array* para realizar as mesmas operações em cada item, envolve um ciclo.
- Em pseudocódigo, isso normalmente tomaria a forma geral:

For i = 1,...,N, do something.

Ciclos e Interações

- e em linguagens de programação como C e Java isso seria escrito como ciclo *for*

```
for( i = 0; i < N; i++)
{
        //do something
 }
```

em que contador i mantem registos de "do something" enumeras vezes (N).

Ciclos e Interações

- Por exemplo, podemos calcular a soma de todos os 20 itens em um *array* usando:

```
for (i = 0, sum = 0; i < 20 ; i ++) {
        sum += a[i];
}
```

- Dizemos que há iteração sobre o índice *i*. A estrutura geral ciclo *for* é FOR (INICIALIZAÇÃO; CONDIÇÃO; ACTUALIZAÇÃO) {PROCESSO A REPITIR}

Invariantes

- Um invariante, como o nome sugere, é uma condição que não muda durante a execução de um determinado programa ou algoritmo;
- Invariantes são importantes para estruturas de dados e algoritmos porque permitem provas e verificação com exatidão;
- Em particular, um ciclo invariante é uma condição verdadeira no inicio e no fim em cada interação do ciclo dado.

Atividade

- **Estudar os seguintes conceitos:**
  - (1) Dar exemplo de um ciclo invariante;
  - (2) Escreva um método que recebe elementos inteiros num *array* de forma aleatória e:
    - Listar todos os elementos introduzidos;
    - Listar todos os elementos de forma crescente;
    - Listar todos os elementos de forma decrescente