Trabalho de Testes de Software

QeT 2017.1

Fazer uma função ou método em qualquer linguagem de programação estruturada que faça busca de um elemento em um array de números inteiros previamente ordenado sem duplicatas.

Os argumentos da função são:

- key: a chave de busca
- items: o array de números inteiros ordenado sem duplicatas
- size: o tamanho do array
- ascending: variável booleana indicando se o array está ordenado em ordem crescente
- type: o tipo de busca, conforme explicado abaixo.

A função ou método retorna um valor da classe abaixo, implementada em Java:

```
public class ResultType {

    public enum SearchResult {
        NotFound,
        FoundExact,
        FoundGreater,
        FoundLess
    };
    private SearchResult searchResult;
    private Integer resultIndex;
    private Integer resultValue;

    public ReturnType(SearchResult searchResult,
                      Integer resultIndex,
                      Integer resultValue) {
        this.searchResult = searchResult;
        this.resultIndex = resultIndex;
        this.resultValue = resultValue;
    }
}
```

A lógica de retorno será explicada a partir da lógica de cada tipo de busca, representado pela variável *type*. Para esta variável, utilizaremos um enumerador com o seguinte formato.

```
  Public enum SearchType {
      LessThan(0),
      LessThanEquals(1),
      Equals(2),
      GreaterThanEquals(3),
      GreaterThan(4)
  };
```

A lógica para cada um dos valores do tipo de busca é a seguinte:

LessThan: encontrar o maior elemento que é menor ou igual a chave de busca (*key*). Se este elemento existir, *FoundLess* é retornado; *NotFound*, caso contrário.

LessThanEquals: encontrar um elemento igual a chave de busca (*key*) ou o maior elemento que é menor ou igual a esta. Caso o elemento seja encontrado, *FoundExact* é retornado; caso contrário *FoundLess* para o caso de haver elemento menor ou igual, e *NotFound* em último caso.

Equals: encontrar um elemento igual a chave de busca (*key*). Caso o elemento seja encontrado, *FoundExact* é retornado; *NotFound*, caso contrário.

GreaterThan: encontrar o menor elemento que é maior ou igual a chave de busca (*key*). Se este elemento existir, *FoundGreater* é retornado; *NotFound*, caso contrário.

GreaterThanEquals: encontrar um elemento igual a chave de busca (*key*) ou o menor elemento que é maior ou igual a esta. Caso o elemento seja encontrado, *FoundExact* é retornado; caso contrário *FoundGreater* para o caso de haver elemento maior ou igual, e *NotFound* em último caso.

Em todos os casos, exceto *NotFound*, o valor encontrado é retornado em *resultValue* e o índice no array correspondente ao valor encontrado é retornado em *resultIndex*. Em caso de *NotFound*, *null* é retornado em ambos *resultValue* e *resultIndex*.

Exemplos:

1. Dado o array de entrada [0, 2, 4, 6, 8] em ordem crescente

key	*type*	Result		
		searchResult	*resultValue*	*resultIndex*
-1	*LessThanEquals*	*NotFound*	null	null
0	*LessThan*	*NotFound*	null	null
0	*Equals*	*FoundExact*	0	0
1	*Equals*	*NotFound*	null	null
2	*GreaterThanEquals*	*FoundExact*	2	1
3	*GreaterThanEquals*	*FoundGreater*	4	2
2	*GreaterThan*	*FoundGreater*	4	2

2. Dado o array [8, 6, 4, 2, 0] em ordem decrescente

key	*type*	Result		
		searchResult	*resultValue*	*resultIndex*
-1	*LessThan*	*NotFound*	null	null
4	*LessThanEquals*	*FoundExact*	4	2
8	*Equals*	*FoundExact*	8	0
5	*GreaterThanEquals*	*FoundGreater*	6	1
2	*GreaterThanEquals*	*FoundExact*	2	3
9	*GreaterThan*	*NotFound*	null	null

Fazer um programa em qualquer linguagem de programação estruturada que resolva este problema com a menor complexidade ciclomática possível. Lembre-se que não está sendo pedido o algoritmo mais eficiente, mas sim de menor complexidade ciclomática possível. Apresente o grafo do seu programa e o cálculo da complexidade ciclomática.

Monte um plano de testes utilizando as técnicas funcionais de particionamento por equivalência, análise do valor limite e error guessing; bem como técnicas estruturais de testes de instruções e testes de caminhos.

Não é necessário verificar as premissas dos dados de entrada nos testes, por exemplo, que o array está ordenado, sem duplicatas, e que a variável *size* corresponde corretamente ao tamanho do array.

Vocês podem modificar a forma das variáveis de entrada do problema como quiserem, desde que não desconfigurem o enunciado, por exemplo, a variável *ascending* pode ser do tipo inteira com valores +1 e -1, ao invés de ser do tipo *boolean*.

A entrega deve conter o código do programa que representa a função solicitada e também o código da execução do plano de testes, com comentários explicando os tipos de testes (funcionais e estruturais).

Os critérios de correção irão incluir:

- A corretude do programa
- A complexidade ciclomática do programa (deve ser mínima)
- A cobertura do plano de testes