

- Data de entrega: 29 de janeiro até 17h.

- Procedimento para a entrega:

1. Submissão: via **RunCodes**.
2. Os protótipos das funções devem ser mantidos como sugerido, o que inclui: nome, tipo de retorno, tipo e ordem dos parâmetros.
3. Os nomes dos arquivos devem ser mantidos como sugerido.
4. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
5. A solução deve ser devidamente modularizada e separar a especificação da implementação em arquivos `.h` e `.c` sempre que cabível.
6. Os arquivos a serem entregues, incluindo aquele que contém `main()`, devem ser compactados (`.zip`), sendo o arquivo resultante submetido via **RunCodes**.
7. Dentre os arquivos submetidos, deve existir um intitulado `compilcao.txt`, contendo os comandos especificados no `prompt/console` para compilar e executar seu programa.
8. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.

- Bom trabalho!

Objetivo Geral

O vetor é uma estrutura de dados indexada, que pode armazenar uma determinada quantidade de valores do mesmo tipo. Os dados armazenados em um vetor são chamados de itens do vetor. Para localizar a posição de um item em um vetor usamos um número inteiro denominado índice do vetor.¹

Nesta aula prática, vamos implementar um biblioteca com algumas funções utilitárias para a manipulação de vetores, tais como: buscar um elemento no vetor e comparar dois vetores.

- Os protótipos das funções serão especificados no arquivo `vetor_util.h`.
- As implementações das funções especificadas devem estar no arquivo `vetor_util.c`.

Questão 01

Busca Linear ou Busca Sequencial: A forma mais simples de fazer uma busca em um vetor consiste em percorrer o vetor, elemento a elemento, verificando se o elemento de interesse é igual a um dos elementos do vetor.

Implemente a busca linear em conformidade com o protótipo especificado a seguir:

```
1  /*retorno:
2  * índice do vetor, caso o elemento exista.
3  * -1, se elemento nao for encontrado
4  *parametros:
5  * int*v - vetor de inteiros
6  * int - tamanho do vetor
7  * int elemento: elemento a ser buscado
8  */
9
10
11 int buscaSequencial(int *vetor, int n, int elemento);
12
13
```

¹<http://linguagemc.com.br/vetores-ou-arrays-em-linguagem-c/>

Exemplos de Entradas e Saídas

Entrada	Saída
v etor = {2, 4, 5, 6, 7, 8, 10, 9} n = 8 elemento = 4 v etor = {-1, 0, 3, 5} n = 4 elemento = 6	1 -1

Questão 02

Busca Binária Se os elementos do vetor estiverem ordenados, podemos aplicar um algoritmo mais eficiente para realizar a busca. Trata-se do algoritmo de *busca binária*. A ideia do algoritmo é testar o elemento que buscamos com valor do elemento armazenado no meio do vetor. Se o elemento que buscamos for menor que o elemento do meio, sabemos que se o elemento estiver presente no vetor, ele estará na primeira parte do vetor; se for maior, estará na segunda parte do vetor; se for igual, achamos o elemento no vetor. Esse procedimento é continuamente repetido, subdividindo a parte de interesse, até encontrarmos o elemento ou chegarmos a uma parte do vetor com tamanho zero.

Implemente, de **forma iterativa**, a busca binária em conformidade com o protótipo especificado a seguir:

```
1  /*retorno:
2  * indice do vetor, caso o elemento exista.
3  * -1, se elemento nao for encontrado
4  *parametros:
5  * int*v - vetor de inteiros
6  * int - tamanho do vetor
7  * int elemento: elemento a ser buscado
8  */
9
10
11 int buscaBinaria(int *vetor, int n, int elemento);
12
13
```

Exemplos de Entradas e Saídas

Entrada	Saída
v etor = {2, 4, 5, 6, 7, 8, 9, 10} n = 8 elemento = 4 v etor = {} n = 0 elemento = -1	1 -1

Questão 03

Intercalação de Vetores: Uma operação muito comum em computação é a intercalação de dois vetores que já estão ordenados. Considere um vetor A de tamanho N e outro vetor B de tamanho M. O vetor resultante C é construído a partir dos dois originais e terá tamanho N + M. Há inclusive, um algoritmo de ordenação, o *MergeSort* ou ordenação por fusão, que utiliza a intercalação de vetores em seus procedimentos.

Implemente a intercalação de vetores ordenados conforme o protótipo especificado a seguir:

```
1  /*retorno:
2  * int* vetor de inteiros resultante
3  *parametros:
4  * int* nums1 - vetor 1 devidamente ordenados
5  * int nums1Tam - numero de elementos do vetor 1
6
```

```

7      * int* nums2 - vetor 2 devidamente ordenados
8      * int nums2Tam - numero de elementos do vetor 2
9      */
10
11      int* intercalaVetoresOrdenados(int* nums1, int nums1Tam, int* nums2, int nums2Tam);
12
13

```

Exemplos de Entradas e Saídas

Entrada	Saída
nums1 = {1,3,5,7} nums1Tam = 4 nums2 = {0,2,4,6} nums2Tam = 4 nums1 = {} nums1Tam = 0 nums2 = {1,2} nums2Tam = 2	{0,1,2,3, 4,5,6,7} {1,2}

|

Questão 04

Comparação de Vetores: Uma operação muito comum em computação é a comparação de dois vetores, que consiste em verificar se eles possuem os mesmos elementos.

Implemente, **de forma iterativa**, a comparação de vetores conforme o protótipo especificado a seguir:

```

1      /*retorno:
2      * 1, se vetores forem iguais
3      * 0 caso contrario
4      *parametros:
5      * int* nums1 - vetor 1
6      * int nums1Tam - numero de elementos do vetor 1
7      * int* nums2 - vetor 2
8      * int nums2Tam - numero de elementos do vetor 2
9      */
10
11      int comparaVetores(int* nums1, int* nums2, int nums1Tam, int nums2Tam);
12
13
14

```

Exemplos de Entradas e Saídas

Entrada	Saída
nums1 = {1,3,5,7} nums1Tam = 4 nums2 = {1,3,5} nums2Tam = 3 nums1 = {} nums1Tam = 0 nums2 = {} nums2Tam = 0	0 1

Referências

- [1] Celes, Waldemar and Cerqueira, Renato and Rangel, José *Introdução a Estruturas de Dados com Técnicas de Programação em C*. Elsevier Brasil, 2016. ISBN 978-85-352-8345-7.