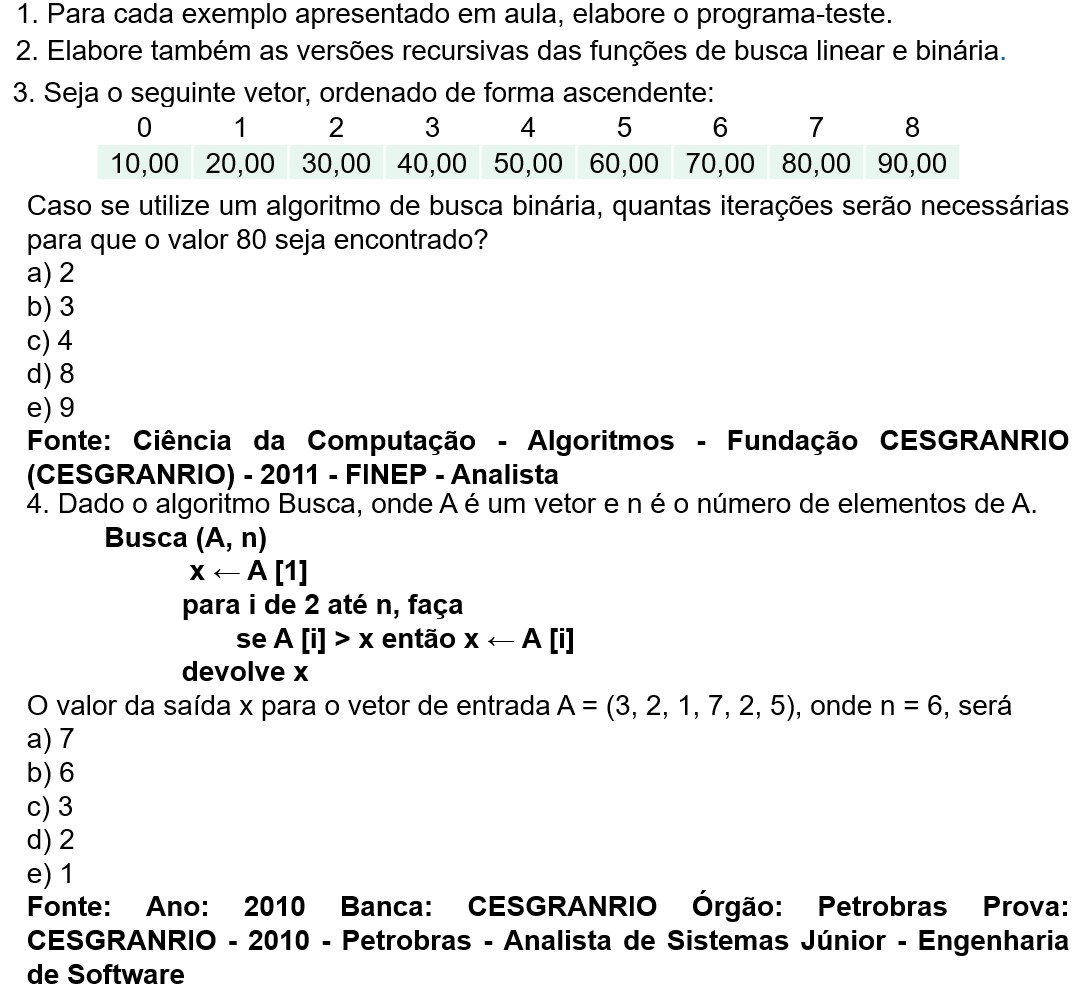
**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E**

**TECNOLOGIA SÃO PAULO - IFSP**

|  |  |  |  |
| --- | --- | --- | --- |
| **Disciplina** | EDDA2 | **Semestre** | 2º |
| **Professores** | Eurides Balbino | **Atividade-lista 4** | 11/09/2020 |
| **Aluno** | Vinicius Gomes Moreira | **Prontuário** | SP3039587 |

# LISTA 4





1)

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <locale.h>**

**#include <conio.h>**

**float abastece\_notas(float \* notas);**

**int busca\_linear (float n, float \* notas);**

**int busca\_binaria (float n, float \* notas);**

**void troca (float \*A, float \*B);**

**int TAMANHO\_VETOR;**

**int main(){**

**int opc, i, j;**

**float \*notas, n;**

**setlocale (LC\_ALL, "Portuguese");**

**printf("Informe quantas notas deseja alocar: \n");**

**scanf("%d", &TAMANHO\_VETOR);**

**notas = (float \*) malloc (TAMANHO\_VETOR \* sizeof(float));**

**abastece\_notas(notas);**

**for (i = 0; i < TAMANHO\_VETOR ; i++)**

**{**

**printf("\nPosição alocada %d:\t %.2f", i+1, notas[i]);**

**}**

**printf("\nInforme o valor a ser pesquisado: \n");**

**scanf ("%f", &n);**

**printf("\nPronto, iremos para a próxima etapa! Aperte qualquer tecla para continuar:");**

**getch();**

**system("cls");**

**printf("Informe qual método de busca deseja utilizar: \n");**

**printf("1 - Busca Linear\n");**

**printf("2 - Busca Binária\n");**

**scanf("%d", &opc);**

**switch(opc){**

**case 1:**

**j = busca\_linear(n, notas);**

**if (j < 0)**

**printf("\nNão existe o valor buscado dentro do vetor gerado.");**

**else**

**printf("\n\nO valor buscado se encontra na posição %d do vetor", j);**

**free(notas);**

**break;**

**case 2:**

**j = busca\_binaria(n, notas);**

**if (j < 0)**

**printf("\nNão existe o valor buscado dentro do vetor gerado.");**

**else**

**printf("\n\nO valor buscado se encontra na posição %d do vetor", j);**

**free(notas);**

**break;**

**default: printf("\nOpção inválida");**

**}**

**}**

**int busca\_linear (float n, float \* notas){**

**int i = 0;**

**while (i < TAMANHO\_VETOR){**

**printf("\nComparando posição %d do vetor, que contém o valor %.2f com a busca que é %.2f",i+1, notas[i], n);**

**if (notas[i++]==n)**

**return i;**

**}**

**return -1;**

**}**

**int busca\_binaria (float n, float \* notas){**

**void quick\_sort(float \* notas, int inicio, int fim);**

**int ini = 0, fim = TAMANHO\_VETOR, meio, i;**

**quick\_sort(notas, ini, fim);**

**for (i = 1; i <= TAMANHO\_VETOR ; i++)**

**{**

**printf("\nOrdenado por Quick Sort: Posição %i: \t %.2f\n", i, notas[i]);**

**}**

**while(ini <= fim)**

**{**

**meio = (ini+fim)/2;**

**if (notas[meio]==n)**

**return meio;**

**if (notas[meio]>n)**

**fim = meio-1;**

**else**

**ini = meio+1;**

**}**

**return -1;**

**}**

**void quick\_sort (float \* notas, int ini, int fim)**

**{**

**int particiona (float \* notas, int ini, int pivo);**

**int pivo;**

**if ( ini < fim ) /\* Caso base \*/**

**{**

**pivo = particiona ( notas, ini, fim ); /\* Particiona o vetor\*/**

**quick\_sort( notas, ini, pivo-1 ); /\* Ordena do início do vetor até antes do pivô \*/**

**quick\_sort( notas, pivo+1, fim); /\* Ordena desde após o pivô até o fim do vetor \*/**

**}**

**}**

**//Função auxiliar quick sort**

**int particiona (float \* vetor, int ini, int pivo)**

**{**

**int i;**

**int p\_maior = ini; /\* a posição do maior é a do início do vetor \*/**

**for ( i=ini; i<pivo; i++ ) /\* percorre o vetor da posição ini até a do pivô \*/**

**{**

**if ( vetor[i] < vetor[pivo] ) /\*se o elemento da posição atual for menor que o pivô...\*/**

**{**

**troca( &vetor[i], &vetor[p\_maior] ); /\*...troca o atual com o maior \*/**

**p\_maior++; /\* o maior avança uma posição \*/**

**}**

**}**

**/\* percorrido o vetor, o pivô troca de posição com a do maior elemento... \*/**

**troca( &vetor[p\_maior], &vetor[pivo] );**

**return p\_maior; /\* ... e o maior elemento passa a ser o novo pivô \*/**

**}**

**void troca (float \*A, float \*B)**

**{**

**float aux = \*A;**

**\*A = \*B;**

**\*B = aux;**

**}**

**float abastece\_notas (float \* notas)**

**{**

**int i;**

**for (i=0; i<TAMANHO\_VETOR; i++)**

**do**

**{**

**notas[i] = rand()%100\*1.13;**

**}**

**while (notas[i] > 100);**

**}**

2) **int busca\_linear (float n, float \* notas, int i){**

**if (i < TAMANHO\_VETOR){**

**printf("\nComparando posição %d do vetor, que contém o valor %.2f com a busca que é %.2f",i+1, notas[i], n);**

**if (notas[i++]==n)**

**return i;**

**else**

**busca\_linear(n, notas, i);**

**}**

**else return -1;**

**}**

**int busca\_binaria (float n, float \* notas, int ini, int fim){**

**int meio;**

**if (ini <= fim)**

**{**

**meio = (ini+fim)/2;**

**if (notas[meio]==n)**

**return meio;**

**if (notas[meio]>n)**

**busca\_binaria (n, notas, ini, meio-1);**

**else**

**busca\_binaria (n, notas, meio+1, fim);**

**}**

**else return -1;**

**}**

3) **3**

4) A (7)