**3º PERÍODO DE ENGENHARIA DE SOFTWARE ANO 2020**

**PROGRAMAÇÃO DE COMPUTADORES**

PROFESSOR: FÁBIO GARCEZ BETTIO

ESTUDANTE: CLÍSTENES GRIZAFIS BENTO

**APS 2 - EXERCÍCIOS DE PESQUISA**

1 – Faça o teste de mesa do exercício do código abaixo com n=6:

**double** fatorial(**int** n);

**int** main(**void**){

**int** numero;

**double** f;

printf("Digite o numero que deseja calcular o fatorial: ");

scanf("%d",&numero);

f = fatorial(numero);

printf("Fatorial de %d = %.0lf",numero,f);

getch();

**return** 0;

}

**double** fatorial(**int** n){

**double** vfat;

**if** ( n <= 1 )

**return** (1);

**else**{

vfat = n \* fatorial(n - 1);

**return** (vfat);

}

}

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

2 – Faça o teste de mesa do código abaixo, com n=5:

**int** pesqSeq(**int** chave, **int** v[], **int** n) {

**int** i;

**for** (i = 0; i < n; i++) {

**if** (v[i] == chave) {

**return** (i);

}

}

**return** (-1); *// índice inválido*

}

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

3 – Faça o teste de mesa do código abaixo, com n=5:

**int** pesqSeqSent(**int** chave, **int** v[], **int** n) {

**int** i = 0;

vet[n] = chave;

**while** (vet[i] != chave) {

i++;

}

**if**( i < n )

**return** i;

**return** (-1); *// índice inválido*

}

Texto preto sobre fundo branco

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

4 – Faça o teste de mesa do código abaixo, com n=10:

**int** pesqBin(**int** chave, **int** v[], **int** n) {

**int** inicio = 0;

**int** meio;

**int** fim = n - 1;

**while** (inicio <= fim) {

meio = (inicio + fim) / 2;

**if** (chave < v[meio]) {

fim = meio - 1;

}

**else** **if** (chave > v[meio]) {

inicio = meio + 1;

}

**else** {

**return** meio;

}

}

**return** -1; *// índice Impossível*

}

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

5 – Implemente uma versão recursiva da função pesquisa binária, teste com n=5.

* Protótipo da função: int pesqBinRec(int chave, int v[], int ini, int fim);

#include<stdio.h>

#include<locale.h>

/\*5. Implemente uma versão recursiva da função pesquisa binária, teste com n = 5.

Protótipo da função: int pesqBinRec(int chave, int v[], int ini, int fim);

\*/

**int** pesqBin(**int** chave, **int** v[],**int** ini, **int** fi);

**int** main(){

setlocale(LC\_ALL,"portuguese");

**int** vetor[10]={1,3,5,9,10,11,12,14,16,17};

**int** key = 5;

**int** inicio =0;

**int** size = 10;

**int** indiceDoArmazenamento;

indiceDoArmazenamento = pesqBin(key,vetor,inicio,size);

printf("O índice é %d",indiceDoArmazenamento);

**return** 0;

}

**int** pesqBin(**int** chave, **int** v[],**int** ini, **int** fi){

**int** inicio = ini;

**int** meio;

**int** fim = fi-1;

**if**(inicio<=fim){

meio = (inicio+fim)/2;

**if**(chave<v[meio]){

fim = meio;

**return** pesqBin(chave,v,inicio,fim);

}

**else** **if**(chave>v[meio]){

inicio = meio+1;

fim +=1;

**return** pesqBin(chave,v,inicio,fim);

}

**else**

**return** meio;

}

**else**

**return** -1;

}

6 – Faça o teste de mesa do código abaixo, com n=10:

**int** pesqInter(**int** chave, **int** v[], **int** n) {

**int** ini = 0, meio, fim = n - 1;

**while** (ini <= fim) {

meio = ini + ((fim-ini)\*(chave-v[ini])) / (v[fim]-v[ini]);

printf("\n O indice do meio foi: %i", meio);

**if** (chave < v[meio]) {

fim = meio - 1;

}

**else** **if** (chave > v[meio]) {

ini = meio + 1;

}

**else** {

**return** meio;

}

}

**return** -1; *// índice Impossível*

}

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

7- Crie um programa em C que preencha um vetor de inteiros de 1000 posições com números aleatórios e implemente o menu de opções abaixo:

* 1. Ordena Vetor (use qualquer método que aprendeu na aula passada)
* 2. Pesquisa Sequencial (Solicite um número inteiro e informe seu tempo para localizá-lo)
* 3. Pesquisa Sentinela (Solicite um número inteiro e informe seu tempo para localizá-lo)
* 4. Pesquisa Binária (Solicite um número inteiro e informe seu tempo para localizá-lo)
* 5. Pesquisa Interpolada (Solicite um número inteiro e informe seu tempo para localizá-lo)
* 6. Sair

#include<stdio.h>

#include<stdlib.h>

#include<locale.h>

#include<time.h>

/\*7. Crie um programa em C que preencha um vetor de inteiros de 1000 posi絥s com nmeros aleat󲩯s e implemente o menu de op絥s abaixo:

1. Ordena Vetor (Use qualquer m鴯do que aprendeu na aula passada)

2. Pesquisa Sequencial (Solicite um numero inteiro e informe o tempo para localiz᭬o)

3. Pesquisa Sentinela (Solicite um numero inteiro e informe o tempo para localiz᭬o)

4. Pesquisa BinᲩa (Solicite um numero inteiro e informe o tempo para localiz᭬o)

5. Pesquisa Interpolada (Solicite um numero inteiro e informe o tempo para localiz᭬o)

6. Sair

\*/

**int** vetor[1001];

**void** gerarVetor();

**void** ordenaVetor();

**int** pesquisaSequencial(**int** key);

**int** pesquisaSentinela(**int** key);

**int** pesquisaBinaria(**int** key);

**int** pesquisaInterpolada(**int** key);

**void** bubbleSort();

**void** selectionSort();

**void** insertionSort();

**void** shelfSort();

**int** main(){

setlocale(LC\_ALL,"portuguese");

**int** opcao;

**int** chave;

gerarVetor();

**while**(1){

system("cls");

printf("SEJA BEM-VINDO");

printf("\n\nPor gentileza escolha uma opção:");

printf("\n1-Ordena vetor \n2-Pesquisa sequencial");

printf("\n3-Pesquisa sentinela \n4-Pesquisa binária");

printf("\n5-Pesquisa Interpolada \n6-Visualizar vetor");

printf("\n7-sair");

printf("\n\nEscolha: ");

scanf("%d",&opcao);

**switch**(opcao){

**case** 1:

ordenaVetor();

**break**;

**case** 2:

printf("\n\nDigite o valor que deseja buscar: ");

scanf("%d",&chave);

printf("\nO valor buscado encontra-se no índice %d\n\n\n",pesquisaSequencial(chave));

system("pause");

**break**;

**case** 3:

printf("\n\nDigite o valor que deseja buscar: ");

scanf("%d",&chave);

printf("\nO valor buscado encontra-se no índice %d\n\n\n",pesquisaSentinela(chave));

system("pause");

**break**;

**case** 4:

printf("\n\nDigite o valor que deseja buscar: ");

scanf("%d",&chave);

printf("\nO valor buscado encontra-se no índice %d\n\n\n",pesquisaBinaria(chave));

system("pause");

**break**;

**case** 5:

printf("\n\nDigite o valor que deseja buscar: ");

scanf("%d",&chave);

printf("\nO valor buscado encontra-se no índice %d\n\n\n",pesquisaInterpolada(chave));

system("pause");

**break**;

**case** 6:

printf("O vetor gerado foi: \n");

**for**(**int** i=0;i<**sizeof**(vetor)/4-1;i++){

printf(" %d,",vetor[i]);

}

printf("\n\n\n");

system("pause");

**break**;

**case** 7:

**return** 0;

**break**;

**default**:

system("cls");

printf("Opção inválida, digite novamente\n\n\n");

system("pause");

**break**;

}

}

**return** 0;

}

**void** gerarVetor(){

srand(time(**NULL**));

**for**(**int** i=0;i<**sizeof**(vetor)/4-1;i++){

vetor[i]=rand()%(10\*(**sizeof**(vetor)/4)-1);

**for**(**int** j=i-1;j>=0;j--){

**if**(vetor[i]==vetor[j]){

j=-1;

i--;

}

}

}

}

**void** ordenaVetor(){

**int** opcao;

printf("\n\nPor gentileza escolha uma opção");

printf("\n1-Ordena o vetor por Bubble Sort ");

printf("\n2-Ordena o Vetor por Selection Sort");

printf("\n3-Ordena o Vetor por Insertion Sort");

printf("\n4-Ordena o Vetor por Shelf Sort");

printf("\n5-Retorna ao menu principal");

printf("\n\nEscolha: ");

scanf("%d",&opcao);

**switch**(opcao){

**case** 1:

bubbleSort();

**break**;

**case** 2:

selectionSort();

**break**;

**case** 3:

insertionSort();

**break**;

**case** 4:

shelfSort();

**break**;

**case** 5:

**break**;

**default**:

system("cls");

printf("Opção inválida!!!\n\n\n");

system("pause");

**break**;

}

}

**int** pesquisaSequencial(**int** key){

**clock\_t** inicio=clock(),fim;

**int** size = **sizeof**(vetor)/4-1;

**for**(**int** i=0;i<size;i++){

*// printf("\n%d size: %d",vetor[i],size);*

**if**(vetor[i]==key){

*//system("pause");*

fim=clock();

printf("O tempo de processamento foi de %d milissegundos",fim-inicio);

**return** i;

}

}

*//system("pause");*

fim=clock();

printf("O tempo de processamento foi de %d milissegundos",fim-inicio);

**return** -1;

}

**int** pesquisaSentinela(**int** key){

**clock\_t** inicio=clock(),fim;

**int** i=0,size=**sizeof**(vetor)/4-1;

vetor[size] = key;

**while**(vetor[i]!=key){

i++;

}

**if**(i<size){

*//system("pause");*

fim=clock();

printf("O tempo de processamento foi de %d milissegundos",fim-inicio);

**return** i;

}

*//system("pause");*

fim=clock();

printf("O tempo de processamento foi de %d milissegundos",fim-inicio);

**return** -1;

}

**int** pesquisaBinaria(**int** key){

*//shelfSort();*

**clock\_t** start=clock(), stop;

**int** inicio =0;

**int** meio;

**int** fim = **sizeof**(vetor)/4-2;

**while**(inicio<=fim){

meio=(inicio+fim)/2;

**if**(key<vetor[meio])

fim = meio-1;

**else** **if**(key>vetor[meio])

inicio = meio+1;

**else**{

*// system("pause");*

stop=clock();

printf("O tempo de processamento foi de %d milissegundos",stop-start);

**return** meio;

}

}

*//system("pause");*

fim=clock();

printf("O tempo de processamento foi de %d milissegundos",stop-start);

**return** -1;

}

**int** pesquisaInterpolada(**int** key){

*//shelfSort();*

**clock\_t** start=clock(), stop;

**int** inicio =0;

**int** meio;

**int** fim = **sizeof**(vetor)/4-2;

**while**(inicio<=fim){

meio=inicio+((fim-inicio)\*(key-vetor[inicio]))/(vetor[fim]-vetor[inicio]);

**if**(key<vetor[meio])

fim = meio-1;

**else** **if**(key>vetor[meio])

inicio = meio+1;

**else**{

*// system("pause");*

stop=clock();

printf("O tempo de processamento foi de %d milissegundos",stop-start);

**return** meio;

}

}

*//system("pause");*

fim=clock();

printf("O tempo de processamento foi de %d milissegundos",stop-start);

**return** -1;

}

**void** bubbleSort(){

**int** vetorAux[**sizeof**(vetor)/4-1];

**int** auxiliar,j, verificador = 1, soma =0,size=(**sizeof**(vetor)/4)-2;

**for**(**int** i=0;i<**sizeof**(vetor)/4-1;i++){

vetorAux[i]=0;

}

**while**(verificador !=0){

**for**(**int** i=0;i<size;i++){

auxiliar = vetor[i];

j = i+1;

**if**(vetor[i]>vetor[j]){

vetor[i]=vetor[j];

vetor[j]=auxiliar;

}

}

**for**(**int** i=0;i<**sizeof**(vetor)/4-1;i++){

soma+=abs(vetor[i]-vetorAux[i]);

}

**for**(**int** i=0;i<**sizeof**(vetor)/4-1;i++){

vetorAux[i]=vetor[i];

}

verificador = soma;

soma = 0;

size--;

}

printf("\n\nOperação concluída!\n\n");

printf("\n\n\n\n");

system("pause");

}

**void** selectionSort(){

**int** auxiliar,indice, size=**sizeof**(vetor)/4-1;

**for**(**int** i=0;i<size-1;i++){

auxiliar = vetor[i];

indice = i;

**for**(**int** j=i+1;j<size;j++){

**if**(auxiliar>vetor[j]){

auxiliar = vetor[j];

indice = j;

}

}

vetor[indice]=vetor[i];

vetor[i]=auxiliar;

}

printf("\n\nOperação concluída!!\n\n");

printf("\n\n\n\n");

system("pause");

}

**void** insertionSort(){

**int** auxiliar,j,k, size=**sizeof**(vetor)/4-1;

**for**(**int** i=0;i<size-1;i++){

j=i+1;

k=i;

**for**(j;j>0;j--,k--){

**if**(vetor[k]>vetor[j]){

auxiliar=vetor[j];

vetor[j]=vetor[k];

vetor[k]=auxiliar;

}

**else**

j=0;

}

}

printf("\n\nOperação concluída!!\n\n");

printf("\n\n\n\n");

system("pause");

}

**void** shelfSort(){

**int** i, j, value, h=1, size=**sizeof**(vetor)/4-1;

**while**(h<size){

h=3\*h+1;

}

**while**(h>0){

**for**(i=h;i<size;i++){

value = vetor[i];

j=i;

**while**(j>h-1&&value<=vetor[j-h]){

vetor[j]=vetor[j-h];

j=j-h;

vetor[j]=value;

}

}

h=h/3;

}

printf("\n\nOperação concluída!!\n\n");

printf("\n\n\n\n");

system("pause");

}