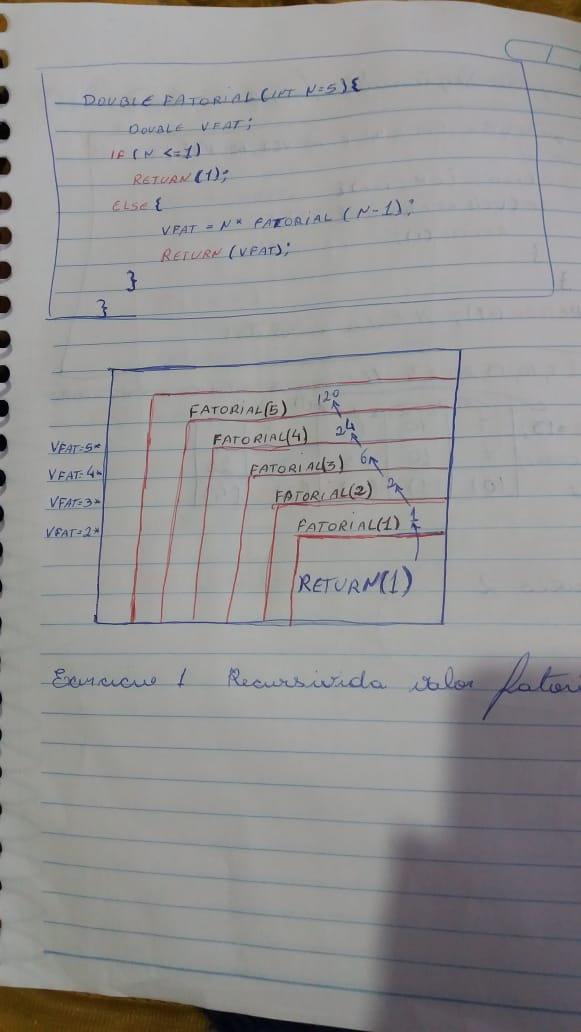
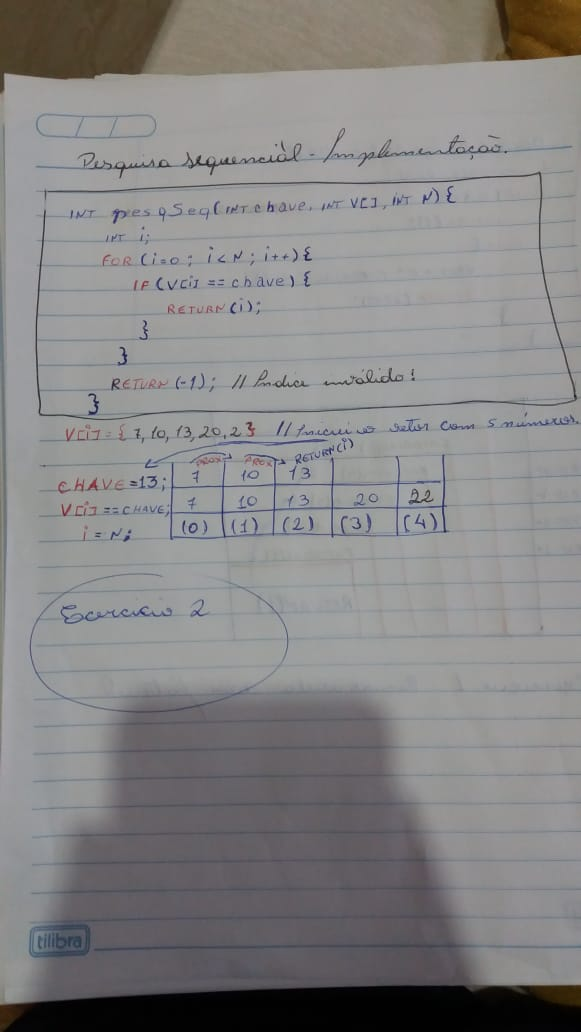
Nome: Amilton Gomes de Lara Filho AG: 21903

Engenharia de Software 3° período, noturno.

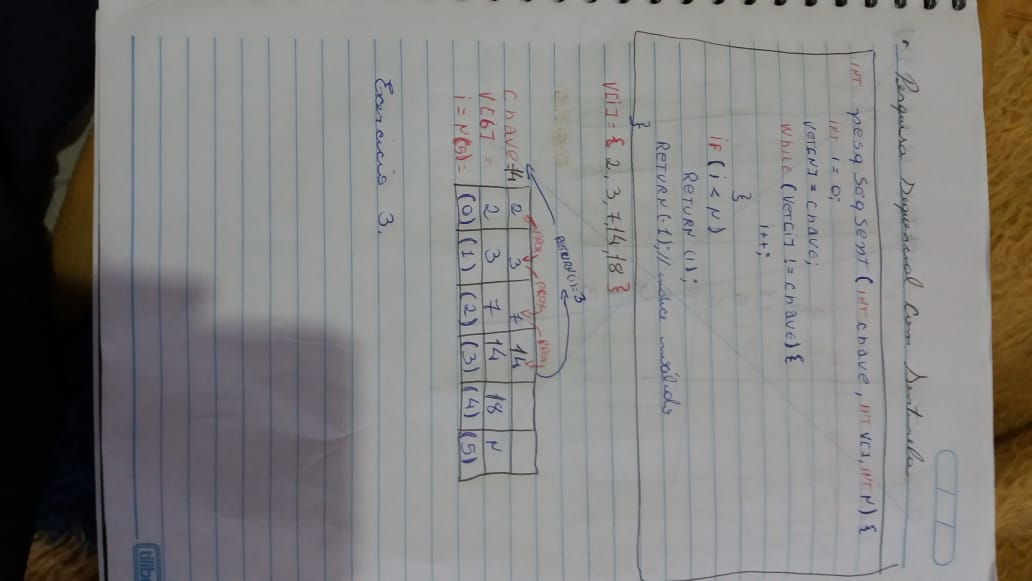
Exercício 1: Fatorial recursivo.



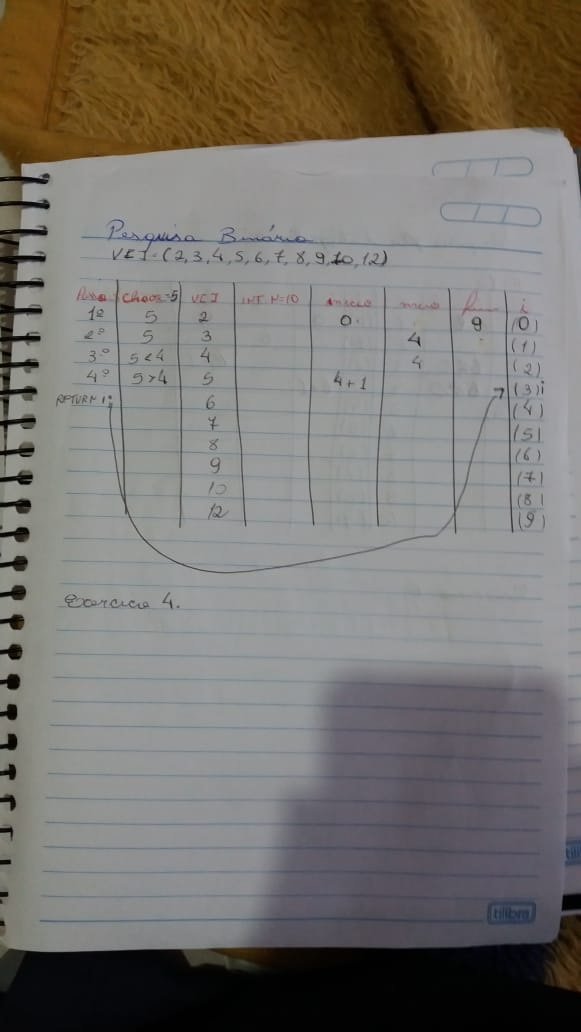
Exercício 2: Pesquisa Sequencial.



Exercício 3: Pesquisa Sequencial com Sentinela.



Exercício 4: Pesquisa Binaria.



Exercício 5: Código fonte pesquisa binária recursiva.

#include <stdio.h>

#include <stdlib.h>

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

int pesqBinRec(int chave, int \*vet, int ini, int fim){

int meio;

if(ini <= fim){

meio = (ini + fim)/2;

if(chave == vet[meio])

return meio;

else{

if(chave < vet[meio])

return pesqBinRec(chave, vet, ini, meio - 1);

else

return pesqBinRec(chave, vet, meio + 1, fim);

}

}

return -1; //Não esta encontrado.

}

int main(){

int vet[10] = {10,34,77,89,91,99,116,133,250,320};

int ini = 0;

int fim = 9;

int chave;

printf("\n Dentre os numeros = 10 - 34 - 77 - 89 - 91 - 99 - 116 - 133 - 250 - 320!\n Digite 00 pra Sair!\n");

do{

printf("\nPosição de qual número vc quer saber a posição: ");

scanf("%d",&chave);

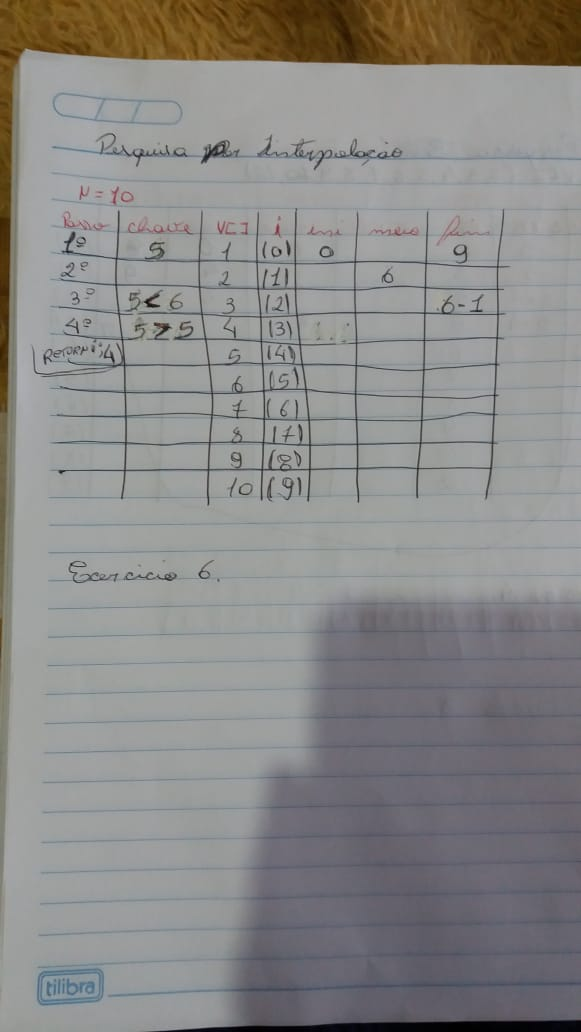
printf("\nO número %i está na %d º posição\n",chave, pesqBinRec(chave, vet, ini, fim));

}while(chave != 00);

return 0;

}

Exercício 6: Pesquisa Binária por interpolação.



Exercício 7:

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

#include <time.h>

/\*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 numero inteiro e informe o tempo para localizá-lo)

3. Pesquisa Sentinela (Solicite um numero inteiro e informe o tempo para localizá-lo)

4. Pesquisa Binária (Solicite um numero inteiro e informe o tempo para localizá-lo)

5. Pesquisa Interpolada (Solicite um numero inteiro e informe o tempo para localizá-lo)

6. Sair \*/

int shellSort(int \*vet); //Prototipo da função shell sort.

int pesqSeq(int chave, int \*vet, int fim); //Prototipo função pesquisa Sequencial

int pesqSenti(int chave, int \*vet, int fim);

int pesqBinRec(int chave, int \*vet, int inicio, int fim);

int pesqInter(int chave, int \*vet, int inicio, int fim);

int main(){

clock\_t sequencial = 0;

clock\_t sentinela = 0;

clock\_t binaria = 0;

clock\_t interpolacao = 0;

setlocale(LC\_ALL, "portuguese");

int vet[1000];

int chave;

int inicio = 0;

int fim = 1000;

int op;

for(int i = 0 ; i < vet[1000] ; i++) //Loop para pegar números aleatórios de 0 a 1000.

vet[i] = rand() % 5000 + 1;

printf("Olá bem vindo!\n");

printf("1. Ordena Vetor!\n");

printf("2. Pesquisa Sequencial (Solicite um numero inteiro e informe o tempo para localizá-lo)\n");

printf("3. Pesquisa Sentinela (Solicite um numero inteiro e informe o tempo para localizá-lo)\n");

printf("4. Pesquisa Binária (Solicite um numero inteiro e informe o tempo para localizá-lo)\n");

printf("5. Pesquisa Interpolada (Solicite um numero inteiro e informe o tempo para localizá-lo)\n");

printf("6. Sair\n");

do{

printf("\nDigite sua opção: ");

scanf("%d",&op);

fflush(stdin);

switch(op){

case 1:

shellSort(vet);

for(int i = 0 ; i < 1000 ; i++)

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

break;

case 2:

printf("Digite um numero para busca Sequencial: ");

scanf("%d",&chave);

sequencial = clock();

printf("Número %d está na %dº posição!",chave,pesqSeq(chave, vet, fim));

sequencial = clock() - sequencial;

printf("\nTempo de execução Busca Sequencial: %lf milissegundos\n\n",(sequencial) /(double)CLOCKS\_PER\_SEC/1000);

break;

case 3:

printf("Digite um numero para buscar por Pesquisa Sentinela: ");

scanf("%d",&chave);

sentinela = clock();

printf("Número %d está na %dº posição!",chave,pesqSenti(chave, vet, fim));

sentinela = clock() - sentinela;

printf("\nTempo de execução Busca Sentinela: %lf milissegundos\n\n",(sentinela) /(double)CLOCKS\_PER\_SEC/1000);

break;

case 4:

printf("Digite um número para buscar com Busca Binária: ");

scanf("%d",&chave);

binaria = clock();

printf("Número %d está na %d° posção!",chave,pesqBinRec(chave, vet, inicio, fim));

binaria = clock() - binaria;

printf("\nTempo de execução Busca Binária: %lf milissegundos\n\n",(binaria) /(double)CLOCKS\_PER\_SEC/1000);

break;

case 5:

printf("Digite um número para buscar com Busca Interpolação: ");

scanf("%d",&chave);

interpolacao = clock();

printf("Número %d está na %d° posição!",chave, pesqInter(chave, vet, inicio, fim));

interpolacao = clock() - interpolacao;

printf("\nTempo de execução Busca Interpolada: %lf milissegundos\n\n",(interpolacao) /(double)CLOCKS\_PER\_SEC/1000);

break;

case 6:

return 0;

default:

printf("Opção invalida! Tente novamente: ");

}

}while(op != 6);

}

int shellSort(int \*vet) //Função para ordenar por Shell Sort.

{

printf("\nOperação concluida por Shell Sort! ");

int i, j, troca;

int h = 1;

while(h < 1000) //Procura o valor da sequiencia Knuth {1,4,13,40...} para realizar os saltos no vetor.

{

h = 3\*h+1;

}

while(h > 0)

{

for(i = h ; i < 1000 ; i++) //Loop que faz as comparações de cada posição.

{

troca = vet[i];

j = i;

while(j > h-1 && troca <= vet[j-h]) //Testa as posições e faz a troca.

{

vet[j] = vet[j-h];

j = j - h;

}

vet[j] = troca;

}

h = h/3;

}

}

int pesqSeq(int chave, int \*vet, int fim){ // função para pesquisa sequencial.

for(int i = 0 ; i < fim ; i++){

if(vet[i] == chave)

return (i);

}

return (-1); //Indice invalido.

}

int pesqSenti(int chave, int \*vet, int fim) { //Função pesquisa sequencial sentinela.

int i = 0;

vet[fim] = chave;

while (vet[i] != chave) {

i++;

}

if( i < fim )

return i;

return (-1); // Índice inválido

}

int pesqBinRec(int chave, int \*vet, int inicio, int fim){ //Função para pesquisa Binaria.

int meio;

if(inicio <= fim){

meio = (inicio + fim)/2;

if(chave == vet[meio])

return meio;

else{

if(chave < vet[meio])

return pesqBinRec(chave, vet, inicio, meio - 1);

else

return pesqBinRec(chave, vet, meio + 1, fim);

}

}

return -1; //Não esta encontrado.

}

int pesqInter(int chave, int \*vet,int inicio, int fim){ //Função para pesquisa Interpolação.

int meio;

fim = fim - 1;

while(inicio <= fim){

meio = inicio +((fim - inicio)\*(chave - vet[inicio])) / (vet[fim] - vet[inicio]);

if(chave < vet[meio]){

fim = meio - 1;

}

else if(chave > vet[meio]){

inicio = meio + 1;

}

else{

return meio;

}

}

return -1; //Indice não encontrado.

}