3º PERÍODO DE ENGENHARIA DE SOFTWARE ANO 2020

PROGRAMAÇÃO DE COMPUTADORES

PROFESSOR: FÁBIO GARCEZ BETTIO

ESTUDANTE: CLÍSTENES GRIZAFIS BENTO

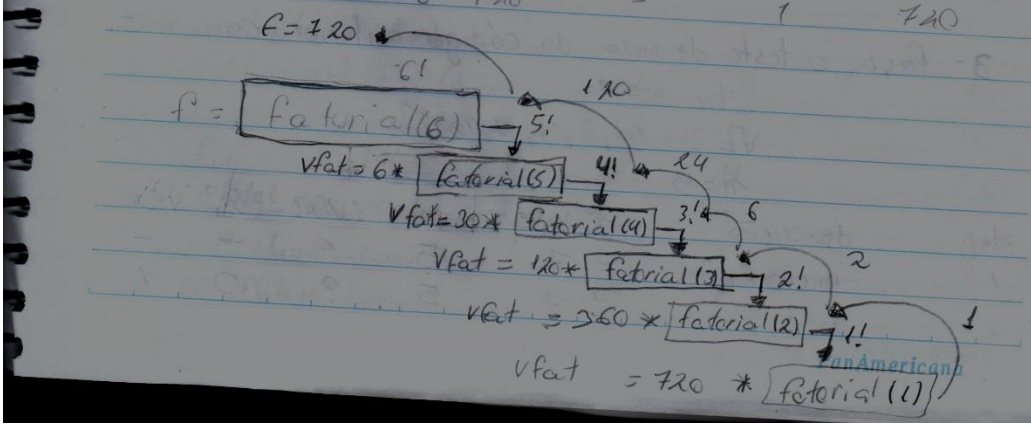
APS 2 - EXERCÍCIOS DE PESQUISA

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);
    }
}
```

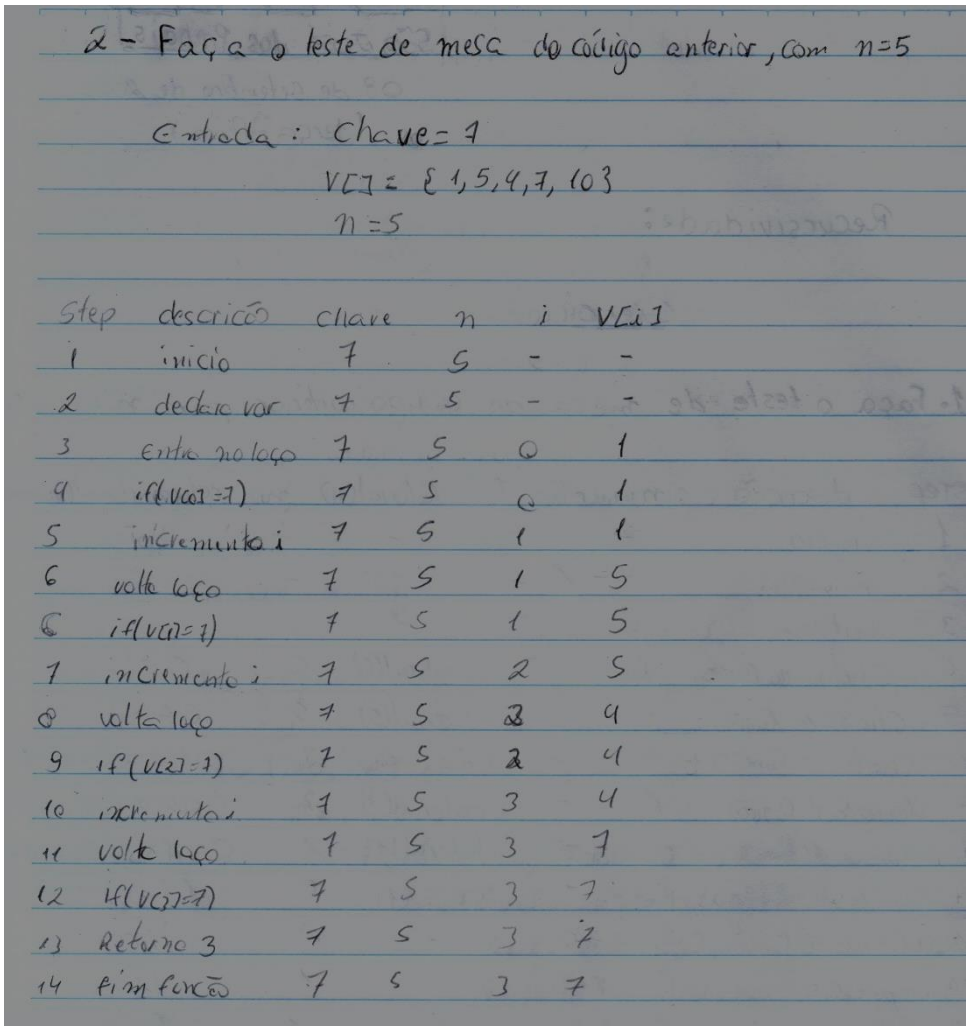
1. Faça o teste de mesa do código anterior, para número = 6:

Step	descrição	numero	f	fatorial(n)	n	retorno fatorial(numero)
1	inicio	-	-	-	-	-
2	mensagem	-	-	-	-	-
3	Leitura	6	-	-	-	-
4	Chamada função	6	-	fatorial(6)	6	6! fatorial(5)
5	Chamada função	6	-	fatorial(5)	5	6.5. fatorial(4)
6	Chamada função	6	-	fatorial(4)	4	6.5.4. fatorial(3)]
7	Chamada da função	6	-	fatorial(3)	3	6.5.4(3. fatorial(2))]
8	Chamada da função	6	-	fatorial(2)	2	6.5.4(3.2. fatorial(1))))
9	Chamada da função	6	-	fatorial(1)	1	6.5.4(3.2.1)))
10	retorno da função	6	720	-	1	720
11	impressão	6	720	-	1	720
12	fim	6	720	-	1	720



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

```c
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
}
```



2 - Faça o teste de mesa do código anterior, com n=5

Entrada: Chave = 7

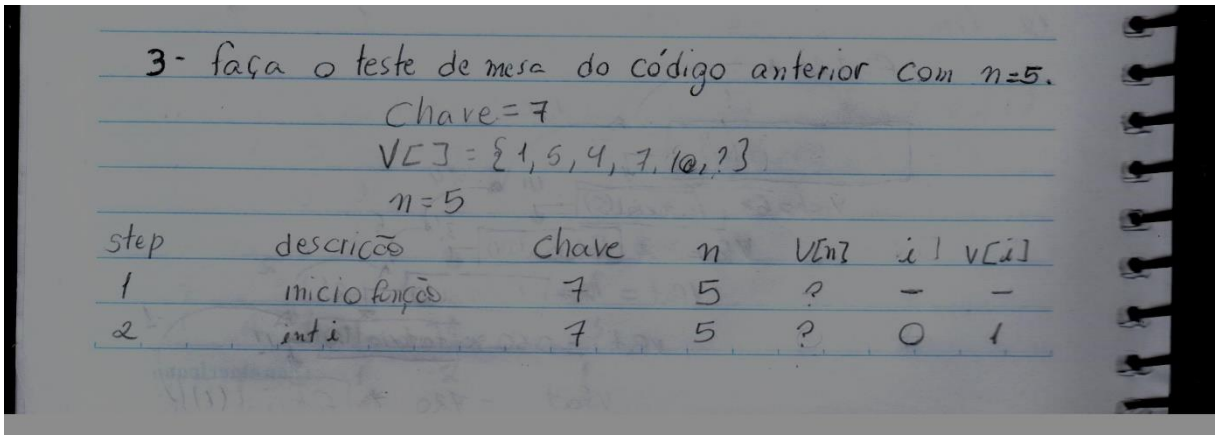
V[] = {1, 5, 4, 7, 10}

n = 5

Step	descrição	chave	n	i	V[i]
1	início	7	5	-	-
2	declara var	7	5	-	-
3	entra no laço	7	5	0	1
4	if(v[0]=7)	7	5	0	1
5	incrementa i	7	5	1	1
6	volta laço	7	5	1	5
6	if(v[1]=7)	7	5	1	5
7	incrementa i	7	5	2	5
8	volta laço	7	5	2	4
9	if(v[2]=7)	7	5	2	4
10	incrementa i	7	5	3	4
11	volta laço	7	5	3	7
12	if(v[3]=7)	7	5	3	7
13	Retorno 3	7	5	3	7
14	Fim função	7	5	3	7

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

```c
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
}
```



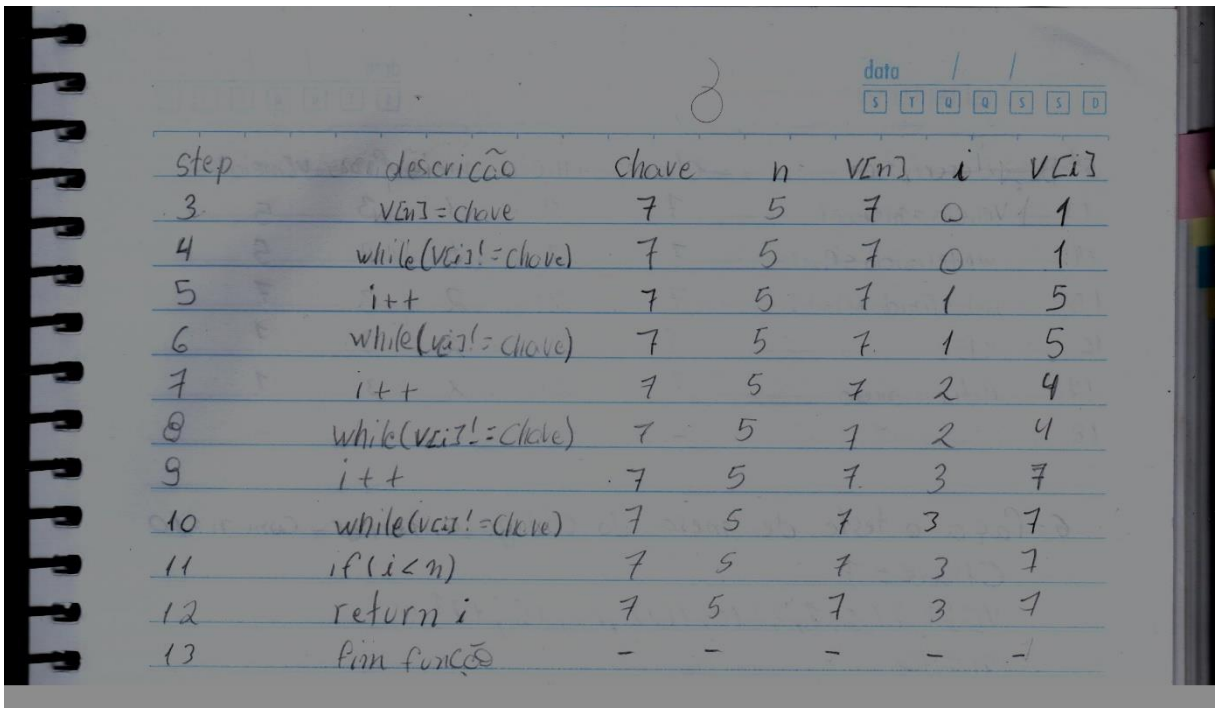
3- faça o teste de mesa do código anterior com n=5.

Chave = 7

V[] = {1, 5, 4, 7, 10, ?}

n = 5

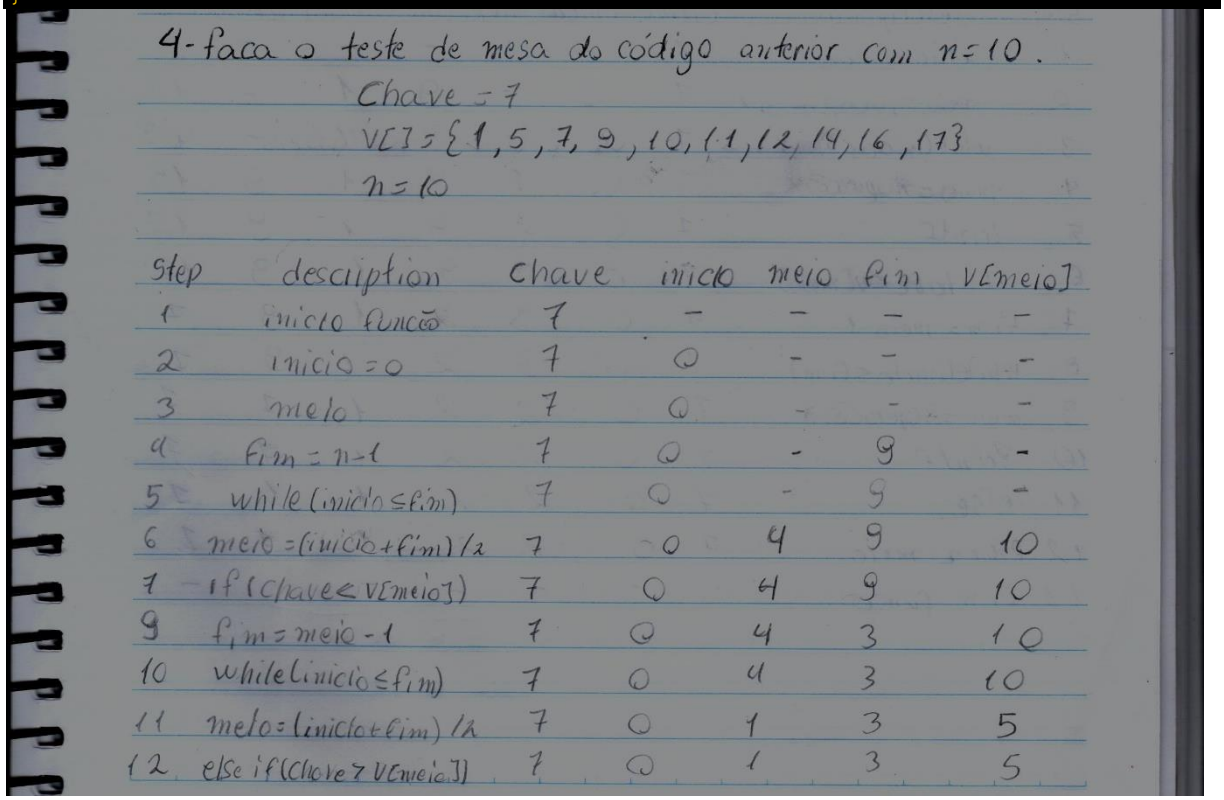
step	descrição	Chave	n	V[n]	i	V[i]
1	inicio função	7	5	?	–	–
2	int i	7	5	?	0	1



step	descrição	Chave	n	V[n]	i	V[i]
3	V[n] = chave	7	5	7	0	1
4	while(V[i] != chave)	7	5	7	0	1
5	i++	7	5	7	1	5
6	while(V[i] != chave)	7	5	7	1	5
7	i++	7	5	7	2	4
8	while(V[i] != chave)	7	5	7	2	4
9	i++	7	5	7	3	7
10	while(V[i] != chave)	7	5	7	3	7
11	if(i < n)	7	5	7	3	7
12	return i	7	5	7	3	7
13	fim função	–	–	–	–	–

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

```c
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
}
```



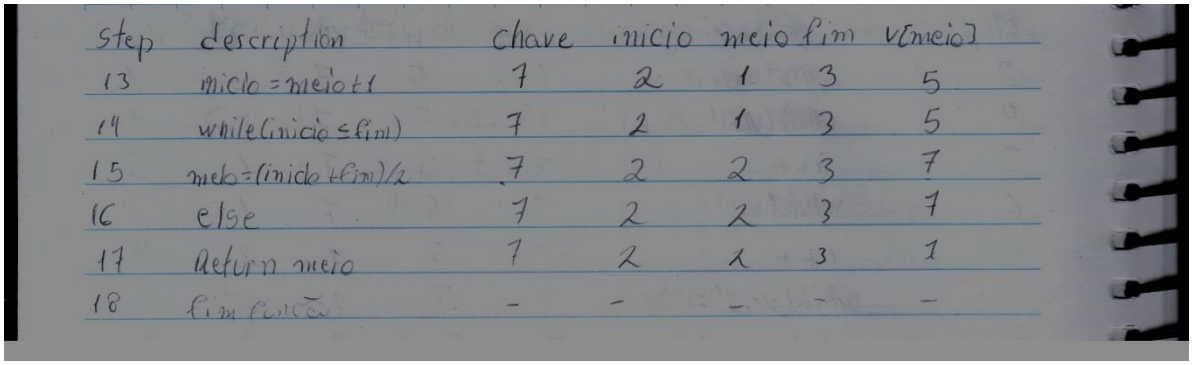
4- faça o teste de mesa do código anterior com n=10.

Chave = 7

V[] = {1, 5, 7, 9, 10, 11, 12, 14, 16, 17}

n = 10

Step	description	chave	inicio	meio	fim	V[meio]
1	inicio função	7	-	-	-	-
2	inicio = 0	7	0	-	-	-
3	meio	7	0	-	-	-
4	fim = n-1	7	0	-	9	-
5	while (inicio ≤ fim)	7	0	-	9	-
6	meio = (inicio+fim)/2	7	0	4	9	10
7	if (chave < V[meio])	7	0	4	9	10
9	fim = meio - 1	7	0	4	3	10
10	while (inicio ≤ fim)	7	0	4	3	10
11	meio = (inicio+fim)/2	7	0	1	3	5
12	else if (chave > V[meio])	7	0	1	3	5



Step	description	chave	inicio	meio	fim	V[meio]
13	inicio = meio+1	7	2	1	3	5
14	while (inicio ≤ fim)	7	2	1	3	5
15	meio = (inicio+fim)/2	7	2	2	3	7
16	else	7	2	2	3	7
17	Return meio	7	2	2	3	7
18	fim função	-	-	-	-	-

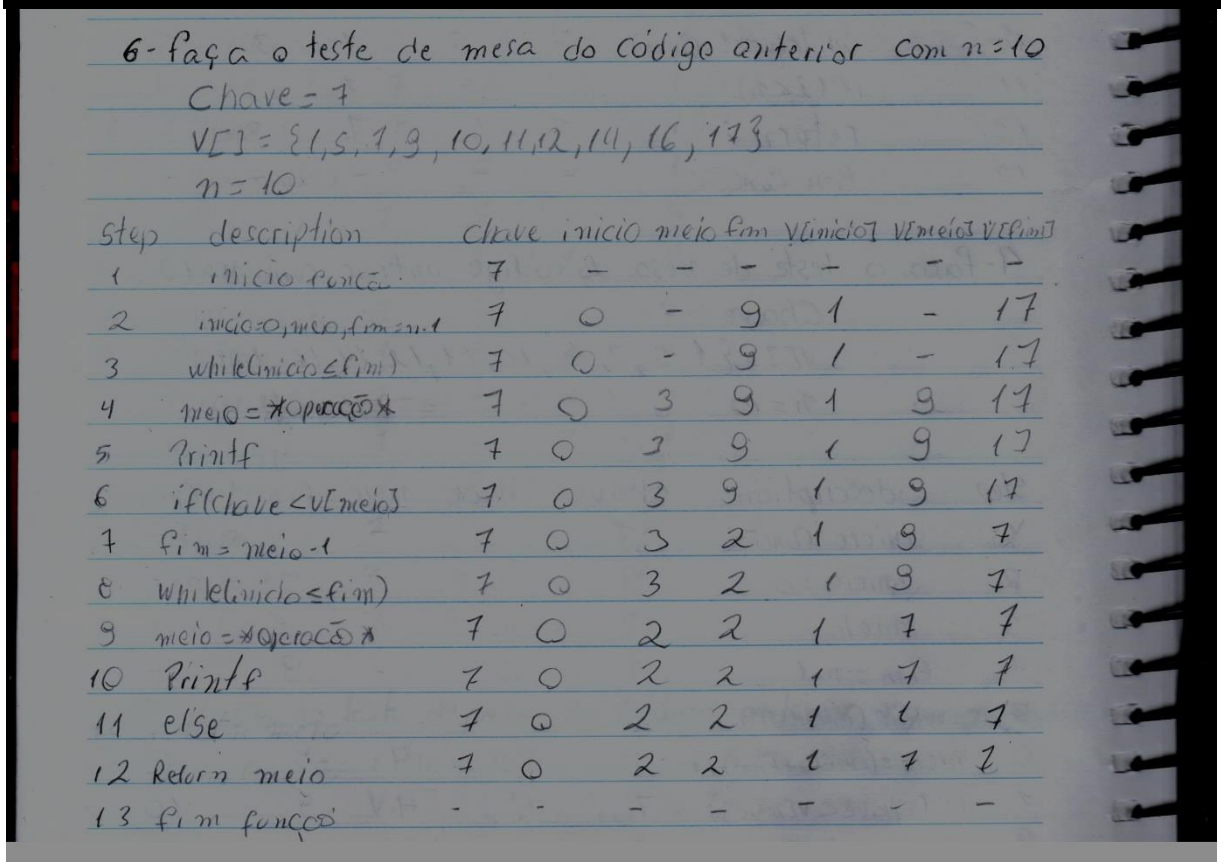
5 – Implemente uma versão recursiva da função pesquisa binária, teste com n=5.

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

```c
#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
}
```



6- faça o teste de mesa do código anterior com n=10

Chave = 7

V[] = {1,5,7,9,10,11,12,14,16,17}

n = 10

Step	description	chave	início	meio	fim	V[início]	V[meio]	V[fim]
1	início função	7	-	-	-	-	-	-
2	início=0, meio, fim=n-1	7	0	-	9	1	-	17
3	while(início≤fim)	7	0	-	9	1	-	17
4	meio = *operação*	7	0	3	9	1	9	17
5	Printf	7	0	3	9	1	9	17
6	if(chave<v[meio])	7	0	3	9	1	9	17
7	fim = meio-1	7	0	3	2	1	9	7
8	while(início≤fim)	7	0	3	2	1	9	7
9	meio = *operação*	7	0	2	2	1	7	7
10	Printf	7	0	2	2	1	7	7
11	else	7	0	2	2	1	1	7
12	Return meio	7	0	2	2	1	7	7
13	fim função	-	-	-	-	-	-	-

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

```c
#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紀s com nmeros aleat□s e implemente o menu de op紀s abaixo:
1. Ordena Vetor (Use qualquer m鴣do que aprendeu na aula passada)
2. Pesquisa Sequencial (Solicite um numero inteiro e informe o tempo
para localiz○□o)
3. Pesquisa Sentinela (Solicite um numero inteiro e informe o tempo
para localiz○□o)
4. Pesquisa Bin□a (Solicite um numero inteiro e informe o tempo para
localiz○□o)
5. Pesquisa Interpolada (Solicite um numero inteiro e informe o
tempo para localiz○□o)
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");
}
```