```
#include <stdio.h>
#include <stdlib.h>
typedef struct arvore
{
    int info;
    struct arvore *dir;
    struct arvore *esq;

} arv;

arv *insere(arv *arvo, arv *n);
arv *lerArq(arv *arvore,FILE *arq);
arv *novo(int val);
arv *remo(arv *arvo, int val);
void impni(arv *arvo,int a,int b);
void implar(arv *arvo,int nivel);
void sair(arv *arvore);
void imparq(arv *arvo);
void maior(arv *arvo, int maio);
void menor(arv *arvo, int meno);
void impnm(arv *arvo,int meno,int maio);
void impfolha(arv *arvore,int num);
int entre(arv *arvo, int max,int min);
int altarv(arv *arvore);
int verarv(arv *arvo,int num);

int main()
{
    FILE *arq;
    char nomeArq[80];
    arv *arvore, *novon;
    int a;
    do
    {
        printf("Digite (1) para ler um arquivo\n");
        printf("Digite (2) para inserir um novo valor\n");
        printf("Digite (3) para remover da arvore\n");
        printf("Digite (4) para imprimir em largura\n");
        printf("Digite (5) para imprimir igual ao arquivo\n");
        printf("Digite (6) para imprimir os valores menores que x e maiores que
y (x deve ser menor que y)\n");
        printf("Digite (7) para contar numero de elementos entre x e y\n");
        printf("Digite (8) para imprimir as folhas de valor maior ou igual a
x\n");
        printf("Digite (9) para verificar se no existe na arvore\n");
        printf("Digite (0) para sair\n");
        scanf("%d",&a);
        system("cls");
        if(a==1)
        {
            system("cls");
            printf("Digite o nome do arquivo:\n");
```

```
    setbuf(stdin,NULL);
    scanf("%80[^\n]",nomeArq);
    arq = fopen(nomeArq,"r");
    if (!arq)
    {
        printf("Erro ao abrir o arquivo");
        exit(EXIT_FAILURE);
    }
    arvore = lerArq(arvore,arq);
}
if(a==2)
{
    int b;
    printf("Digite o valor a ser inserido:\n");
    scanf("%d",&b);
    novon = novo(b);
    arvore = insere(arvore,novon);
    system("cls");
    printf("\nValor inserido com sucesso:\n");
    system("pause");
}
if(a==3)
{
    int b=0;

    printf("Digite o valor a ser removido\n");
    scanf("%d",&b);
    arvore = remo(arvore,b);
    system("cls");
    printf("\nValor removido com sucesso\n");
    system("pause");
}
if(a==4)
{
    int b=0;
    b = altarv(arvore);
    implar(arvore,b);
    system("pause");
}
if(a==5)
{
    imparq(arvore);
    printf("\n");
    system("pause");
}
if(a==6)
{
    int menor,maior;
    printf("Digite o menor:\n");
    scanf("%d",&menor);
    printf("Digite o maior:\n");
    scanf("%d",&maior);
```

```
            impnm(arvore,menor,maior);
            system("pause");
        }
        if(a==7)
        {
            int menor,maior,c;
            printf("Digite o menor:\n");
            scanf("%d",&menor);
            printf("Digite o maior:\n");
            scanf("%d",&maior);
            c = entre(arvore, maior,menor);
            printf("Existem %d numeros entre %d e %d:\n",c,maior,menor);
            system("pause");
        }
        if(a==8)
        {
            int no;
            printf("digite o valor de x:\n");
            scanf("%d",&no);
            impfolha(arvore,no);
            system("pause");
        }
        if(a==9)
        {
            int c,no;
            printf("digite o numero a ser checado");
            scanf("%d",&no);
            c = verarv(arvore,no);
            if(c>0)
                printf("O no existe\n");
            else
                printf("No inexistente\n");
            system("pause");

        }
        if(a==0)
        {
            sair(arvore);
            printf("Desalocamento realizado com sucesso\n");
            system("pause");
        }
        system("cls");
    }
    while(a!=0);
    printf("Ate a proxima");
    return 0;
}
arv *lerArq(arv *arvore,FILE *arq)
{
    int val;
    fscanf(arq," %c",&val);
    fscanf(arq," %d",&val);
```

```
    if (val!=-1)
    {
        arvore = (arv*)malloc(sizeof(arv));
        if (!arq)
        {
            printf("Erro ao alocar memoria");
            exit(EXIT_FAILURE);
        }
        arvore->info = val;
        arvore->esq =lerArq(arvore->esq,arq);
        arvore->dir =lerArq(arvore->dir,arq);
        fscanf(arq," %c",&val);
        return arvore;
    }
    else
    {
        fscanf(arq," %c",&val);
        return NULL;
    }
}
arv *insere(arv *arvo, arv *n)
{
    if (arvo == NULL)
    {
        return n;
    }
    else
    {
        if(n->info <= arvo->info)
            arvo->esq = insere(arvo->esq,n);
        else
            arvo->dir = insere(arvo->dir,n);
    }
}
arv *novo(int val)
{
    arv *novo =(arv*)malloc(sizeof(arv));
    if (!novo)
    {
        printf("N foi possivel alocar memoria");
        exit(EXIT_FAILURE);
    }
    novo->info = val;
    novo->esq = NULL;
    novo->dir = NULL;
    return novo;
}
arv *remo(arv *arvo, int val)
{
    arv *temp=NULL;
    if(!arvo)
    {
```

```
        return arvo;
    }
    if(arvo->info == val)
    {
        if((arvo->dir!=NULL && arvo->esq==NULL)||(arvo->dir==NULL &&
arvo->esq!=NULL))
        {
            if(arvo->dir==NULL)
            {
                temp=arvo->esq;
                free(arvo);
                return temp;
            }
            else
            {
                temp=arvo->dir;
                free(arvo);
                return temp;
            }
        }
        else if (arvo->dir==NULL && arvo->esq==NULL)
        {
            free(arvo);
            return NULL;
        }
        else
        {
            temp = arvo;
            for(temp=temp->dir; temp->esq!=NULL; temp=temp->esq) {}
            arvo->info = temp->info;
            arvo->dir = remo(arvo->dir,temp->info);
            return arvo;
        }
    }
    else if(val<=arvo->info)
    {
        arvo->esq = remo(arvo->esq,val);
    }
    else
        arvo->dir = remo(arvo->dir,val);
}
int altarv(arv *arvore)
{
    int dir,esq;
    if(!arvore)
    {
        return 0;
    }
    else
    {
        dir = altarv(arvore->dir);
        esq = altarv(arvore->esq);
```

```
        if(dir>esq)
        {
            return dir +1;
        }
        else
        {
            return esq+1;
        }
    }
}
void impni(arv *arvo,int a,int b)
{
    if(arvo!=NULL)
    {
        if(a == b)
        {
            printf("%d\t", arvo->info);
        }
        else
        {
            impni(arvo->esq,a+1,b);
            impni(arvo->dir,a+1,b);

        }
    }
}
void implar(arv *arvo,int nivel)
{
    int x;
    if (arvo!=NULL)
    {
        for(x =1; x<=nivel; x++)
        {
            impni(arvo,1,x);
            printf("\n");
        }
    }
}
void sair(arv *arvore)
{
    arv *tempd,*tempe;
    if (arvore==NULL)
    {
        return;
    }
    else
    {
        tempd = arvore->dir;
        tempe = arvore->esq;
        free(arvore);
        sair(tempd);
        sair(tempe);
```

```
    }
}
void imparq(arv *arvo)
{
    if(arvo==NULL)
    {
        printf("(-1");
        printf(")");
    }
    else
    {
        printf("(");
        printf("%d",arvo->info);
        imparq(arvo->esq);
        imparq(arvo->dir);
        printf(")");
    }
}
void impnm(arv *arvo,int meno,int maio)
{
    printf("menores que %d:\n",meno);
    menor(arvo,meno);
    printf("maiores que %d:\n",maio);
    maior(arvo,maio);
}
void menor(arv *arvo, int meno)
{
    if(arvo==NULL)
    {
    }
    else
    {
        if (arvo->info<meno)
        {
            printf("%d\n",arvo->info);
            menor(arvo->esq, meno);
            menor(arvo->dir, meno);
        }
        else
        {

            menor(arvo->esq, meno);
        }
    }
}
void maior(arv *arvo, int maio)
{
    if(arvo==NULL)
    {
    }
    else
    {
```

```
        if (arvo->info>maio)
        {
            printf("%d\n",arvo->info);
            maior(arvo->esq, maio);
            maior(arvo->dir, maio);
        }
        else
        {
            maior(arvo->dir, maio);
        }
    }
}
int entre(arv *arvo, int max,int min)
{
    if(arvo==NULL)
    {
        return 0;
    }
    else if(arvo->info<max && arvo->info>min)
    {
        return 1+entre(arvo->dir,max,min)+entre(arvo->esq,max,min);
    }
    else
    {
        if(arvo->info>=max)
        {
            return entre(arvo->esq,max,min);

        }
        else
        {
            return entre(arvo->dir,max,min);
        }
    }
}
int verarv(arv *arvo,int num)
{
    if(!arvo)
    {
        return 0;
    }
    else
    {
        if(arvo->info == num)
            return 1;
        else
        {
            if(arvo->info>num)
                return verarv(arvo->esq,num);
            else
                return verarv(arvo->dir,num);
        }
```

```
    }
}
void impfolha(arv *arvore,int num)
{
    if(arvore==NULL)
    {
        return;
    }
    if (arvore->dir == NULL && arvore->esq == NULL && arvore->info>=num)
    {
        printf("%d\t",arvore->info);
    }
    else
    {
        impfolha(arvore->esq,num);
        impfolha(arvore->dir,num);
    }
}
```