

29-11-2021



Examen árbol binario Búsqueda

Materia: Estructura de datos

Sección: D01.

Código: 216584703

Carrera: Ingeniería en computación.

Nombre alumno: Padilla Pérez Jorge
Daray

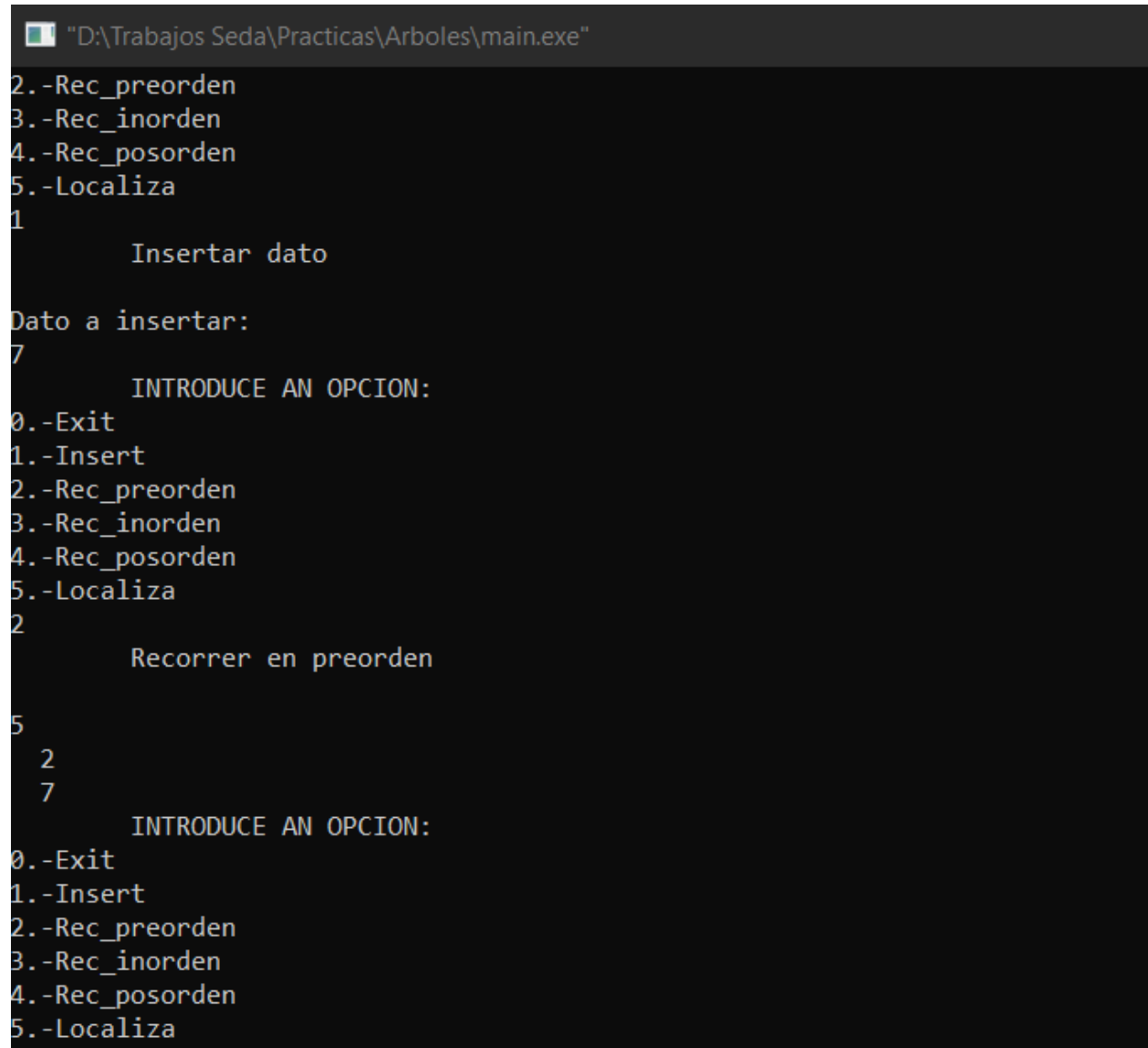
Nombre profesor: Julio Esteban
Valdes Lopez

Introducción

Mi práctica consiste en la implementación de un árbol binario de búsqueda en la cual se implementan lo típico de un TDA árbol lo que viene siendo insertar los nodos del árbol. En esta práctica no se logró hacer todo lo pedido para esta, además de no poder realizar la función de localizar posición por elemento la cual me costó más, y no se logró.

También aunque no se implementaron las funciones eliminar y anular creo que es una función difícil de comprender y de hacer espero que próximamente con más calma pueda realizar de mejor forma los trabajos que se piden.

Pantallazos



```
"D:\Trabajos Seda\Practicas\Arboles\main.exe"
2.-Rec_preorden
3.-Rec_inorden
4.-Rec_posorden
5.-Localiza
1
    Insertar dato
Dato a insertar:
7
    INTRODUCIR AN OPCION:
0.-Exit
1.-Insert
2.-Rec_preorden
3.-Rec_inorden
4.-Rec_posorden
5.-Localiza
2
    Recorrer en preorden
5
    2
    7
    INTRODUCIR AN OPCION:
0.-Exit
1.-Insert
2.-Rec_preorden
3.-Rec_inorden
4.-Rec_posorden
5.-Localiza
```

Aquí se muestra la inserción de 3 números que se muestran pre orden se insertó el 5 luego el 2 y por último el 7.

```
3
    Recorrer en orden

2
5
7
    INTRODUCIR UNA OPCION:
0.-Exit
1.-Insert
2.-Rec_preorden
3.-Rec_inorden
4.-Rec_posorden
5.-Localiza
4
    Recorrer en posorden

2
7
5
```

Aquí se muestra el recorrido en orden y el recorrido en posorden.

Conclusión

Respecto a la realización del código concluyo que no se logró el código de la forma esperada por motivos propios y más que nada por el tiempo ya que no me ajusto el tiempo para poder desarrollarlo en buenas condiciones además de no poder terminarlo se pudo aprender mucho de este ultimo que espero poder estudiar en las vacaciones con mas calma ya que se tranquilice mi vida.

También espero poder dearrollar mejores programas con los conocimientos que pude adquirir en esta materia me siento mejor preparado y comprendo mejor las técnicas con las que poder realizar programas mas difíciles es simplemente cuestión de pensar y como se enseñó en la clase poder realizar dibujos ayuda mucho para comprender el tema a resolver.

Código fuente

```
#include <iostream>

#include<stdlib.h>

#include <string.h>

#define TIPO_DATO int

using namespace std;

class Nodo{
    protected:
        Nodo * raiz; //ancla
        TIPO_DATO elem;
        Nodo * hijo_izq;
        Nodo * hijo_der;
    public:
        Nodo(){
            raiz = nullptr;
        };
        Nodo(TIPO_DATO elem);
        void insertar(TIPO_DATO dato);
        void rec_preorden(int nivel);
        void rec_inorden(int nivel);
        void rec_posorden(int nivel);
        void rec_izq(TIPO_DATO dato, int nivel);
        void rec_der(TIPO_DATO dato, int nivel);
};

Nodo::Nodo(TIPO_DATO elem)
{
```

```

        this->elem = elem;
        hijo_der = nullptr;
        hijo_izq = nullptr;
    }

```

```

void Nodo::insertar(TIPO_DATO dato)
{
    if( dato < elem){
        if(hijo_izq == nullptr){
            hijo_izq = new Nodo(dato);
        } else {
            hijo_izq->insertar(dato);
        }
    } else {
        if(hijo_der == nullptr){
            hijo_der = new Nodo(dato);
        } else {
            hijo_der->insertar(dato);
        }
    }
}

```

```

void Nodo::rec_preorden(int nivel)
{
    for (int i = 0 ; i < nivel ; i++) {
        std::cout << " ";
    }
    std::cout << elem << std::endl;
}

```

```

        if(hijo_izq !=nullptr){
            hijo_izq->rec_preorden(nivel+1);
        }

        if(hijo_der !=nullptr){
            hijo_der->rec_preorden(nivel+1);
        }
    }
}

```

```

void Nodo::rec_inorden(int nivel)
{
    for (int i = 0 ; i < nivel ; i++) {
        std::cout << " ";
    }
    if(hijo_izq !=nullptr){
        hijo_izq->rec_inorden(nivel+1);
    }

    std::cout << elem << std::endl;

    if(hijo_der !=nullptr){
        hijo_der->rec_inorden(nivel+1);
    }
}

```

```

void Nodo::rec_posorden(int nivel)
{
    for (int i = 0 ; i < nivel ; i++) {
        std::cout << " ";
    }
}

```



```

        if(hijo_izq != nullptr){
            hijo_izq->rec_posorden(nivel+1);
        }

        if(hijo_der != nullptr){
            hijo_der->rec_posorden(nivel+1);
        }
        std::cout << elem << std::endl;
    }

void Nodo::rec_izq(TIPO_DATO dato, int nivel)
{
    if(hijo_izq != nullptr){
        hijo_izq->rec_izq(dato, nivel+1);
        for (int i = 0 ; i < nivel ; i++) {
            std::cout << " ";
        }
    }
}

void Nodo::rec_der(TIPO_DATO dato, int nivel)
{
    if(hijo_der != nullptr){
        hijo_der->rec_der(dato, nivel+1);
        for (int i = 0 ; i < nivel ; i++) {
            std::cout << " ";
        }
    }
}

```

```

class Arbol: Nodo{
    public:
        bool vacia();
        void insertar(TIPO_DATO dato);
        void rec_preorden();
        void rec_inorden();
        void rec_posorden();
        void Localiza_elemento(TIPO_DATO dato);
        void rec_izq();
        void rec_der();

};

```

```

bool Arbol::vacia()
{
    return raiz == nullptr;
}

```

```

void Arbol::insertar(TIPO_DATO dato)
{
    if(this->vacia()){
        raiz = new Nodo(dato);
    } else {
        raiz->insertar(dato);
    }
}

```

```

void Arbol::rec_preorden()

```

```

{
    if(not this->vacía()){
        raiz->rec_preorden(0);
    }
}

```

```

void Arbol::rec_inorden()
{
    if(not this->vacía()){
        raiz->rec_inorden(0);
    }
}

```

```

void Arbol::rec_posorden()
{
    if(not this->vacía()){
        raiz->rec_posorden(0);
    }
}

```

```

void Arbol::Localiza_elemento(TIPO_DATO dato)
{
    if(not this->vacía())
    {
        if(dato == this->elem)
        {
            return;
        }
        else if(dato > this->elem)
        {

```

```

        raiz->rec_izq(dato, 0);
    }
    else if (dato < this->elem)
    {
        raiz->rec_der(dato, 0);
    }
}
else
{
    printf("la lista esta vacia");
}
}

void menu()
{
    int opc;
    Arbol mi_arbolito;
    int dato;

    do{
        printf("\tINTRODUCE AN OPCION:\n");
        printf("0.-Exit\n1.-Arbol binario\n");
        scanf("%d", &opc);

        switch(opc){
            case 0: printf("\tGoodbye");
                    system("pause");
                    break;
            case 1: printf("\tArbol binario\n\n");

```

```

int opcion;
do{
    printf("\tINTRODUCE AN OPCION:\n");
    printf("0.-Exit\n1.-Insert\n2.-Rec_preorden\n3.-
Rec_inorden\n4.-Rec_posorden\n5.-Localiza\n");
    scanf("%d", &opcion);
    switch(opcion){
        case 0: printf("\t Adios\n");
                system("pause");
                break;
        case 1: printf("\tInsertar dato\n\n");
                printf("Dato a insertar:\n");
                scanf("%d", &dato);
                mi_arbolito.insertar(dato);
                break;
        case 2: printf("\tRecorrer en preorden\n\n");
                mi_arbolito.rec_preorden();
                break;
        case 3: printf("\tRecorrer en orden\n\n");
                mi_arbolito.rec_inorden();
                break;
        case 4: printf("\tRecorrer en posorden\n\n");
                mi_arbolito.rec_posorden();
                break;
        case 5: printf("\tLocalizar posicion\n\n");
                printf("Dato a buscar:\n");
                scanf("%d", &dato);
                mi_arbolito.Localiza_elemento(dato);
                break;
        /*case 6: printf("\tInsert first\n\n");

```

```

        mi_lista.getinsertlast();
        break;
    case 7: printf("\tInsert first\n\n");
        mi_lista.getLocaliza();
        break;
    case 8: printf("\tInsert first\n\n");
        mi_lista.getAnula();
        break;*/
    }
    }while(opcion!=0);
    break;
    default: printf("\tChoose a correct value");
    break;
}
system("cls");
}while(opc!=0);

}

int main()
{
    menu();
    return 0;
}

```