19-5-2022

Practica 6 árbol

Materia: Seminario de estructura de datos 1

Sección: D13.

Código: 216584703

Carrera: Ingeniería en computación.

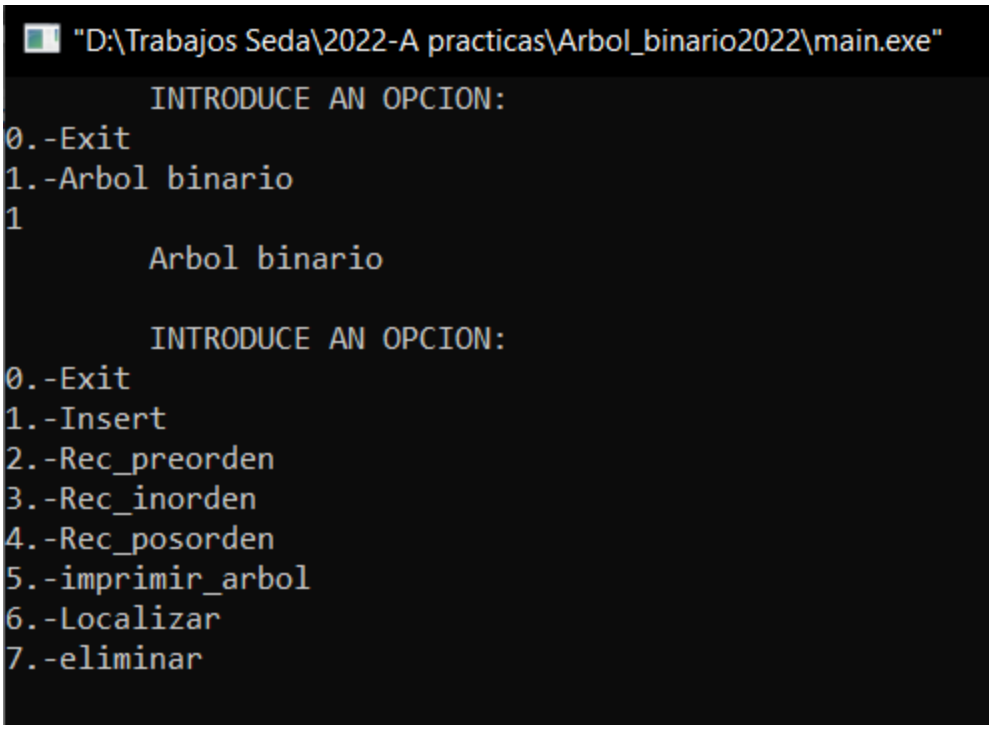
Nombre alumno: Padilla Pérez Jorge Daray

Nombre profesor: Julio Esteban Valdes Lopez

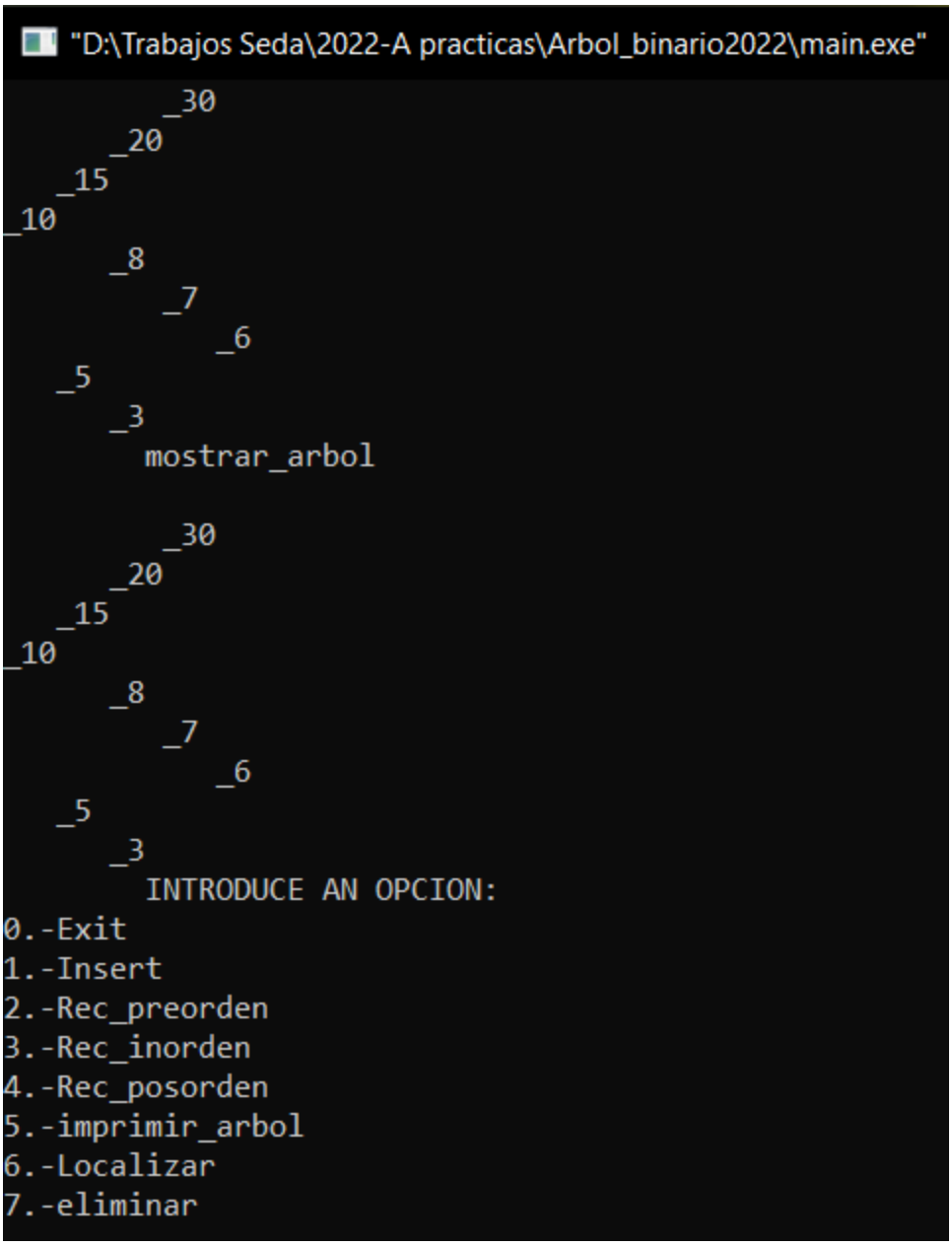
Introducción

En esta practica se realizó la implementación de un árbol binario, en el cual se implementa lo típico del TDA árbol el cual incluye el mostrar de las 3 formas el árbol, ya sea en orden, posorden y preorden se trato de imitar la impresión de los arboles del cmd y mas o menos se consiguió el parecido.

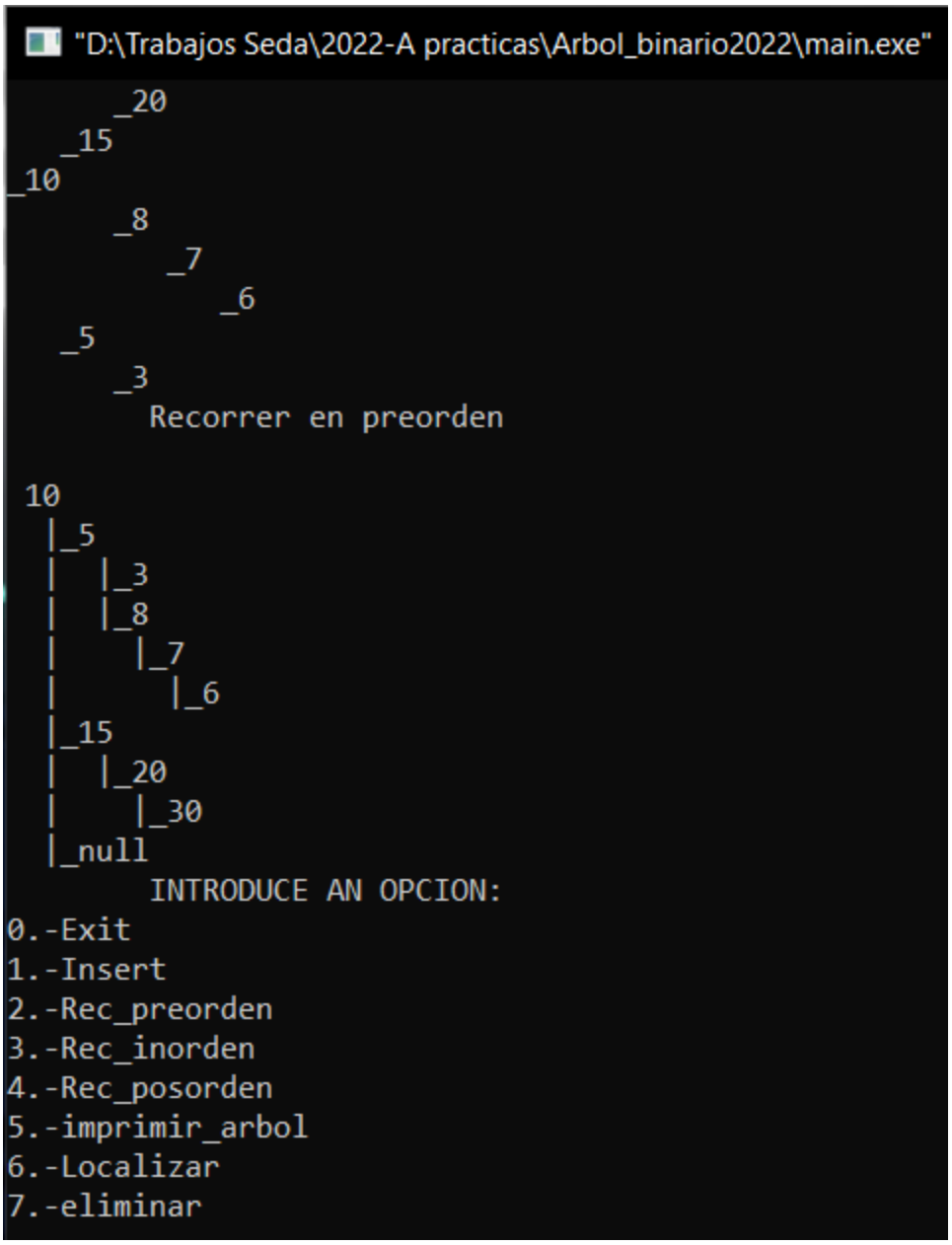
También se añadió la función de buscar el cual encuentra el nodo en el árbol y lo muestra en la pantalla lastimosamente no se pudo completar el árbol ya que falto la función de eliminar de las 3 formas que hay además de la función de anular el árbol.



Aquí se aprecia el menú del árbol.



Aquí insertamos varios nodos y se muestra de manera horizontal el arbol.



Aquí lo imprimimos en preorden.

```
  3  5  6  7  8  10  15  20  30 INTRODUCE AN OPCION:
0.-Exit
1.-Insert
2.-Rec_preorden
3.-Rec_inorden
4.-Rec_posorden
5.-imprimir_arbol
6.-Localizar
7.-eliminar
```

Inorden

```
  |  |_3
  |  |  |  |_6
  |  |  |_7
  |  |_8
  |_5
  |  |  |_30
  |  |_20
  |_15
 10
        INTRODUCE AN OPCION:
0.-Exit
1.-Insert
2.-Rec_preorden
3.-Rec_inorden
4.-Rec_posorden
5.-imprimir_arbol
6.-Localizar
7.-eliminar
```

Y la impresión en pos orden

Conclusión

Se pudo completar de manera correcta el programa, utilizando una clase nodo y otra llamda árbol, en la que se almacenan los nodos, en la cual pues se tienen los métodos principales de un arbol menos los de elimina que falla y el de anular el arbol.

También se desarrolló la impresión del árbol como lo hace el panel de comandos de Windows que se ve bien, no es idéntico pero le hace el parecido ya que esta basado en su impresión.

Codigo fuente

```
#include <iostream>
#define TIPO_DATO int

using namespace std;

class Nodo{
   private:
   TIPO_DATO elem;
   Nodo * hijo_izq;
   Nodo * hijo_der;
   Nodo * padre;
   public:
   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 mostrar_arbol(int nivel);
   int localizar_arbol(int nivel);
   void eliminar_nodo(int nivel);
   void eliminar();
   int minimo();
};

Nodo::Nodo(TIPO_DATO elem)
```

```
{
    this->elem = elem;
    hijo_der = nullptr;
    hijo_izq = nullptr;
    this->padre = padre;
}


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++) {
      if (i == 0){
         std::cout << "  |";
      }
      else {
         std::cout << "  ";
      }
      }
      if (nivel == 0){
         std::cout <<" "<< elem << std::endl;
      }
      else if (nivel == 1){
         std::cout << "_";
         std::cout << elem << std::endl;
      }
      else{
         std::cout << "|";
         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)
{
    if(hijo_izq !=nullptr){
        hijo_izq->rec_inorden(nivel);
    }
    std::cout << "  ";
    std::cout << elem ;

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

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

    if(hijo_der !=nullptr){
        hijo_der->rec_posorden(nivel+1);
    }
    for (int i = 0 ; i < nivel ; i++) {
            std::cout << "  |";
```

```
        }


        if (nivel == 0){
            std::cout <<" "<< elem << std::endl;
        }
        else if (nivel == 1){
            std::cout << "_";
            std::cout << elem << std::endl;
        }
        else{
            std::cout << "_";
            std::cout << elem << std::endl;
        }
}

void Nodo::mostrar_arbol(int nivel)
{
    if(hijo_der !=nullptr){
        hijo_der->mostrar_arbol(nivel+1);
    }
    for (int i = 0; i < nivel ; i++){
        std::cout << "   ";
    }
    std::cout << "_";
    std::cout << elem <<endl;
    if(hijo_izq !=nullptr){
        hijo_izq->mostrar_arbol(nivel+1);
```

```
    }
}

int Nodo::localizar_arbol(int nivel)
{
    if (nivel == elem){
        std::cout <<elem<<endl;
        return elem;
    }
    else if(nivel < elem){
        return hijo_izq->localizar_arbol(nivel);
    }
    else{
        return hijo_der->localizar_arbol(nivel);
    }
}

void Nodo::eliminar_nodo(int nivel)
{
    if (nivel == elem){
        eliminar();
    }
    else if (nivel < elem){
        hijo_izq->eliminar_nodo(nivel);
    }
    else{
        hijo_der->localizar_arbol(nivel);
```

```
    }
}

int Nodo::minimo()
{
    if (hijo_izq !=nullptr)
        {
            return hijo_izq->minimo();
        }
        else{
            return elem;
        }
}

void Nodo::eliminar()
{
    if (hijo_izq && hijo_der)
    {
        int menor = hijo_der->minimo();
        int aux = menor;
        eliminar_nodo(menor);
        elem = aux;
    }
}

class Arbol{
    private:
```

```
    Nodo * raiz; //ancla
    public:
    Arbol();
    bool vacia();
    void insertar(TIPO_DATO dato);

    void rec_preorden();
    void rec_inorden();
    void rec_posorden();
    void mostrar_arbol();
    void localizar_arbol(TIPO_DATO dato);
    void eliminar_nodo(TIPO_DATO dato);
};



Arbol::Arbol()
{
    raiz = nullptr;
}



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->vacia()){
      raiz->rec_preorden(0);
   }
}

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

void Arbol::rec_posorden()
{
   if(not this->vacia()){
```

```
        raiz->rec_posorden(0);
    }
}

void Arbol::mostrar_arbol()
{
    if(not this->vacia()){
        raiz->mostrar_arbol(0);
    }
}

void Arbol::localizar_arbol(TIPO_DATO dato)
{
    if(not this->vacia()){
        raiz->localizar_arbol(dato);
    }
}

void Arbol::eliminar_nodo(TIPO_DATO dato)
{
    if(not this->vacia()){
        raiz->eliminar_nodo(dato);
    }
}

void menu()
{
```

```
   int opc;
   Arbol miArbolito;
   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.-
imprimir_arbol\n6.-Localizar\n7.-eliminar\n");
            scanf("%d", &opcion);
            miArbolito.mostrar_arbol();
               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);
    miArbolito.insertar(dato);
    break;
case 2: printf("\tRecorrer en preorden\n\n");
    miArbolito.rec_preorden();
       std::cout << "   |";
       std::cout << "_";
       std::cout << "null"<<endl;
    break;
case 3: printf("\tRecorrer en orden\n\n");
    miArbolito.rec_inorden();
    break;
case 4: printf("\tRecorrer en posorden\n\n");
    miArbolito.rec_posorden();
    break;
case 5: printf("\tmostrar_arbol\n\n");
    miArbolito.mostrar_arbol();
    break;
case 6: printf("\tInsert first\n\n");
    std::cout << "Ingrese el numero a buscar: "<<endl;
    std::cin >> dato;
    miArbolito.localizar_arbol(dato);
    break;
case 7: printf("\tInsert first\n\n");
    std::cout << "Ingrese el numero a eliminar: "<<endl;
    std::cin >> dato;
    miArbolito.eliminar_nodo(dato);
```

```
                    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;
}
```