Hecho por: Christofer Fabián Chávez Carazas

1. Problema

Implementar todos los recorridos de un arbol binario, con sus respectivos inversos.

2. Código

2.1. ArbolBinario.h

```
#ifndef ARBOLBINARIO_H
#define ARBOLBINARIO_H
#include "iostream"
#include "list"

using namespace std;

class ArbolBinario
{
    public:
        class Nodo{
            public:
            Nodo();
            Nodo(int);
            int valor;
            Nodo * hijos[2];
        };
        ArbolBinario();
        void amplitud();
        void amplitudInverso();
        void insert(int);
        void postorden();
        void preorden();
        void inorden();
        void inordenInverso();
        void postordenInverso();
        void preordenInverso();
        virtual ~ArbolBinario();
    protected:
    private:
        void _postorden(Nodo *&);
        void _inorden(Nodo *&);
        void _preorden(Nodo *&);
        void _inordenInverso(Nodo *&);
        void _postordenInverso(Nodo *&);
        void _preordenInverso(Nodo *&);
        Nodo * root;

};

void ArbolBinario::amplitudInverso(){
    if(!root)return;
    list<Nodo *> result;
    result.push_back(root);
    for(auto iter = result.begin(); iter != result.end(); ++iter){
        if((*iter)->hijos[0]){
            result.push_back((*iter)->hijos[0]);
        }
        if((*iter)->hijos[1]){
            result.push_back((*iter)->hijos[1]);
        }
    }
    for(auto iter = result.begin(); iter != result.end(); ++iter){
        cout<<(*iter)->valor<<endl;
    }
};

void ArbolBinario::preordenInverso(){
    _preordenInverso(root);
}

void ArbolBinario::postordenInverso(){
    _postordenInverso(root);
}
void ArbolBinario::inordenInverso(){
```

```
    _inordenInverso(root);
}

void ArbolBinario::_preordenInverso(Nodo *&nodo){
    if(!nodo)return;
    cout<<nodo->valor<<endl;
    _preordenInverso(nodo->hijos[1]);
    _preordenInverso(nodo->hijos[0]);
}

void ArbolBinario::_postordenInverso(Nodo *&nodo){
    if(!nodo)return;
    _postordenInverso(nodo->hijos[1]);
    _postordenInverso(nodo->hijos[0]);
    cout<<nodo->valor<<endl;
}

void ArbolBinario::_inordenInverso(Nodo *&nodo){
    if(!nodo)return;
    _inordenInverso(nodo->hijos[1]);
    cout<<nodo->valor<<endl;
    _inordenInverso(nodo->hijos[0]);
}
void ArbolBinario::amplitud(){
    if(!root)return;
    list<Nodo *> result;
    result.push_back(root);
    for(auto iter = result.begin(); iter != result.end(); ++iter){
        if((*iter)->hijos[0]){
            result.push_back((*iter)->hijos[0]);
        }
        if((*iter)->hijos[1]){
            result.push_back((*iter)->hijos[1]);
        }
    }
    for(auto iter = result.begin(); iter != result.end(); ++iter){
        cout<<(*iter)->valor<<endl;
    }
}

void ArbolBinario::preorden(){
    _preorden(root);
}

void ArbolBinario::inorden(){
    _inorden(root);
}

void ArbolBinario::postorden(){
    _postorden(root);
}

void ArbolBinario::_postorden(Nodo *& nodo){
    if(!nodo)return;
    _postorden(nodo->hijos[0]);
    _postorden(nodo->hijos[1]);
    cout<<nodo->valor<<endl;
}

void ArbolBinario::_inorden(Nodo *& nodo){
    if(!nodo)return;
    _inorden(nodo->hijos[0]);
    cout<<nodo->valor<<endl;
    _inorden(nodo->hijos[1]);
}

void ArbolBinario::_preorden(Nodo *&nodo){
    if(!nodo)return;
    cout<<nodo->valor<<endl;
    _preorden(nodo->hijos[0]);
    _preorden(nodo->hijos[1]);
}

void ArbolBinario::insert(int valor){
    Nodo ** iter = &(root);
    while(*iter){
        iter = &((*iter)->hijos[(*iter)->valor <= valor]);
    }
    *iter = new Nodo(valor);
}

ArbolBinario::Nodo::Nodo(){
    valor = 0;
    hijos[0] = nullptr;
    hijos[1] = nullptr;
}

ArbolBinario::Nodo::Nodo(int valor){
    this->valor = valor;
    hijos[0] = nullptr;
    hijos[1] = nullptr;
}

ArbolBinario::ArbolBinario(){
```

```
    root = nullptr;
}

ArbolBinario::~ArbolBinario(){}


#endif // ARBOLBINARIO_H
```

2.2. main.h

```
#include <iostream>
#include "ArbolBinario.h"

using namespace std;

int main()
{
    ArbolBinario arbolito;
    arbolito.insert(17);
    arbolito.insert(25);
    arbolito.insert(32);
    arbolito.insert(43);
    arbolito.insert(44);
    arbolito.insert(72);
    arbolito.insert(106);
    arbolito.insert(27);
    arbolito.insert(12);
    arbolito.insert(8);
    arbolito.insert(1);
    cout<<"Preorden->"<<endl;
    arbolito.preorden();
    cout<<endl<<"Inorden->"<<endl;
    arbolito.inorden();
    cout<<endl<<"Posorden->"<<endl;
    arbolito.postorden();
    cout<<endl<<"Amplitud->"<<endl;
    arbolito.amplitud();
    cout<<endl<<"Preorden Inverso->"<<endl;
    arbolito.preordenInverso();
    cout<<endl<<"Inorden Inverso->"<<endl;
    arbolito.inordenInverso();
    cout<<endl<<"Posorden Inverso->"<<endl;
    arbolito.postordenInverso();
    cout<<endl<<"Amplitud Inversa->"<<endl;
    arbolito.amplitudInverso();
}
```

3. Ejemplo

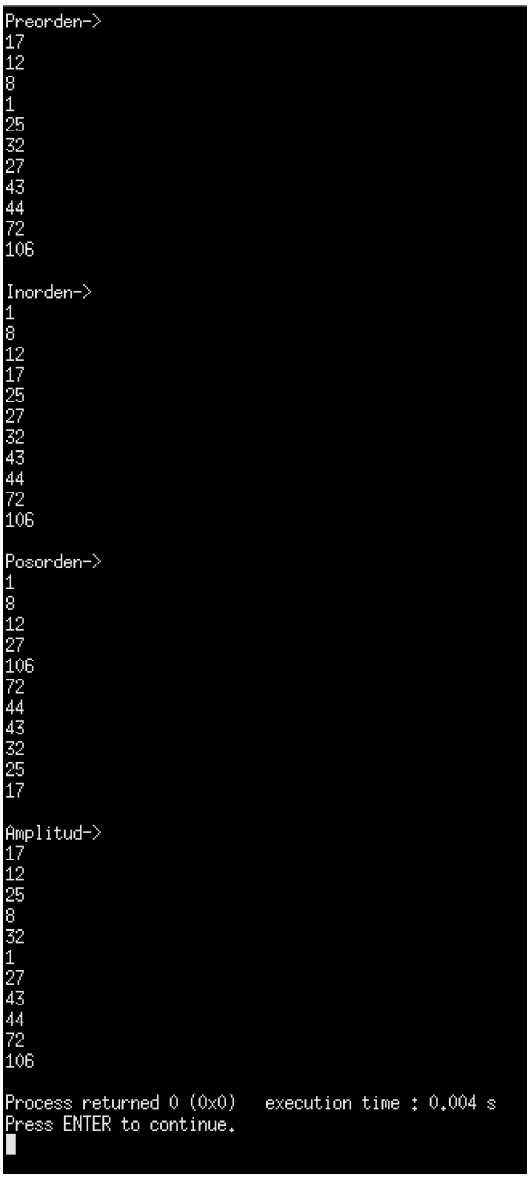

Figura 1: Pruebas part. 1

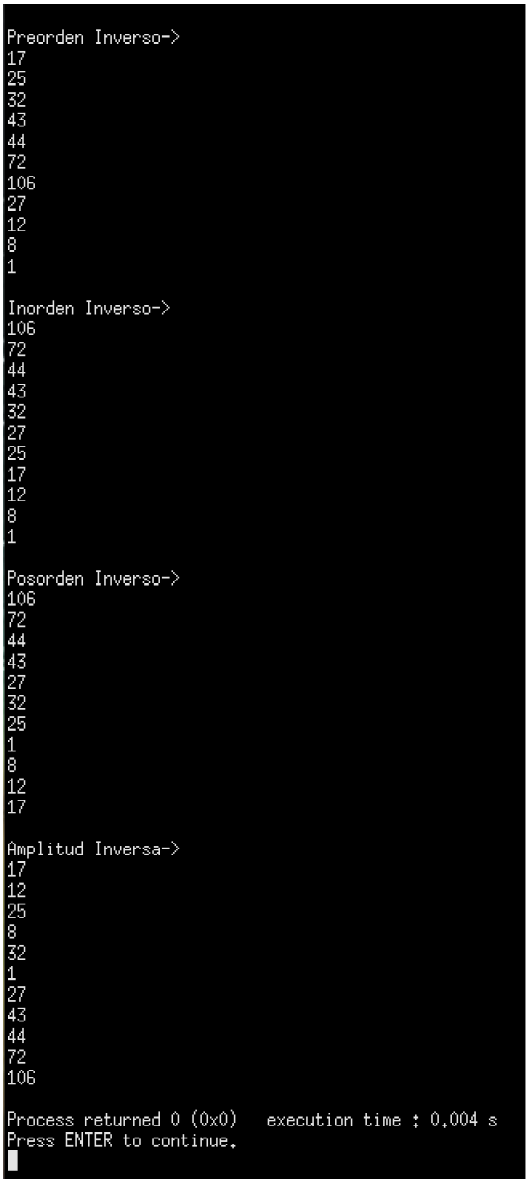

Figura 2: Pruebas part. 2