

Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías



IL365 - Estructura de Datos - Do1

Actividad de Aprendizaje #12

El Árbol Binario de Búsqueda, Implementación Dinámica

Alumna: Cervantes Araujo María Dolores

Código: 217782452

Fecha de Elaboración: 25 abril de 2023



Autoevaluación			
Concepto	Si	No	Acumulación
Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial)	-100 pts	0 pts	0
Incluí el código fuente en formato de texto (sólo si funciona cumpliendo todos los requerimientos)	+25pts	0 pts	25
Incluí las impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)	+25pts	0 pts	25
Incluí una portada que identifica mi trabajo (nombre, código, materia, fecha, título)	+25 pts	0 pts	25
Incluí una descripción y conclusiones de mi trabajo	+25 pts	0 pts	25
Suma:		100	

Introducción:

Un árbol es una estructura jerárquica aplicada sobre una colección de elementos u objetos llamados nodos; a raíz de esto podemos decir que el árbol binario es aquel en el cual cada nodo tiene no más de dos descendientes, es decir, que puede tener descendiente izquierdo o derecho o cero descendientes.

La búsqueda en árboles binarios es un método de búsqueda simple, dinámico y eficiente; el valor de cada nodo es mayor que los valores de su sub-árbol izquierdo y es menor que los valores de sub-árbol derecho y, además, ambos subárboles son árboles binarios de búsqueda.

Una vez comprendidas estas definiciones y el modelo correspondiente de los árboles binarios de búsqueda, esta semana trabajamos con un algoritmo que permite insertar elementos aleatorios a un árbol binario, hacer un recorrido en PreOrder, InOrder y PostOrder, así como también poder conocer la altura total del árbol, además de la altura del subárbol en lado izquierdo y subárbol del lado derecho.



Código Fuente:

arbolBinario.hpp

```
#ifndef ARBOLBINARIO_HPP_INCLUDED
#define ARBOLBINARIO_HPP_INCLUDED
#include <iostream>
#include <string>
#include <exception>

template <class T>
class ArbolB {
private:
    class Node {
private:
    T* dataPtr;
    Node* left;
    Node* right;

public:
    class Exception: public std::exception {
private:
    std::string msg;
public:
    explicit Exception(const char* message) : msg(message) {}
    explicit Exception(const std::string& message) :
msg(message) {}

    virtual ~Exception() throw() {}
    virtual const char* what() const throw() {
        return msg.c_str();
    }
};

Node();
Node(const T&);
~Node();

T* getDataPtr() const;
T getData() const;
Node*& getLeft();
Node*& getRight();

void setDataPtr(T* );
void setData(const T& );
void setLeft(Node*& );
void setRight(Node*& );
};

public:
    typedef Node* Position;

private:
```



```
Position root;

void copyAll(const ArbolB<T>&);
void insertData(Position&, const T&);
void deleteData(Position&, const T&);

Position& findData(Position&, const T&);
Position& getLowest(Position&);
Position& getHighest(Position&);

void preOrder(Position&);
void inOrder(Position&);
void posOrder(Position&);

public:
    class Exception: public std::exception {
        private:
            std::string msg;
        public:
            explicit Exception(const char* message): msg(message) {}
            explicit Exception(const std::string& message): msg(message) {}
            virtual ~Exception() throw() {}
            virtual const char* what() const throw() {
                return msg.c_str();
            }
    };
ArbolB();
ArbolB(const ArbolB<T>&);
~ArbolB();

bool isEmpty() const;
bool isLeaf(Position&) const;

void insertData(const T&);
void deleteData(const T&);
void deleteAll();
T retrieve(Position&) const;

int getHeight(Position&);
int getHeight();
int getHeightRight();
int getHeightLeft();
Position& findData(const T&);
Position& getLowest();
Position& getHighest();

void preOrder();
void inOrder();
void posOrder();
```



```
ArbolB& operator = (const ArbolB&);
};

#endif // ARBOLBINARIO_HPP_INCLUDED

///IMPLEMENTACION NODO

using namespace std;
template <class T>
ArbolB<T>::Node::Node() : dataPtr(nullptr), left(nullptr), right(nullptr) {}

template <class T>
ArbolB<T>::Node::Node(const T& m) : dataPtr(new T(m)), left(nullptr),
right(nullptr) {
    if(dataPtr == nullptr) {
        throw Exception("Memoria insuficiente, se creara un nodo");
    }
}

template <class T>
ArbolB<T>::Node::~Node() {
    delete dataPtr;
}

template <class T>
T* ArbolB<T>::Node::getDataPtr() const {
    return dataPtr;
}

template <class T>
T ArbolB<T>::Node::getData() const {
    if(dataPtr == nullptr) {
        throw Exception("Dato inexistente, getData");
    }
    return *dataPtr;
}

template <class T>
typename ArbolB<T>::Position& ArbolB<T>::Node::getLeft() {
    return left;
}

template <class T>
typename ArbolB<T>::Position& ArbolB<T>::Node::getRight() {
    return right;
}

template <class T>
void ArbolB<T>::Node::setDataPtr(T* pos) {
    dataPtr = pos;
}
```



```
template <class T>
void ArbolB<T>::Node::setData(const T& e) {
    if(dataPtr == nullptr) {
        if((dataPtr = new T(e)) == nullptr) {
            throw Exception("Memoria no disponible, setData");
        }
    } else {
        *dataPtr = e;
    }
}

template <class T>
void ArbolB<T>::Node::setLeft(Position& pos) {
    left = pos;
}

template <class T>
void ArbolB<T>::Node::setRight(Position& pos) {
    right = pos;
}

///IMPLEMENTACION ARBOL

template <class T>
ArbolB<T>::ArbolB() : root(nullptr) {}

template <class T>
ArbolB<T>::ArbolB(const ArbolB& t) : root(nullptr) {
    copyAll(t);
}

template <class T>
ArbolB<T>::~ArbolB() {
    deleteAll();
}

template <class T>
void ArbolB<T>::copyAll(const ArbolB<T>& l) {
    Position newNode;

    try {
        if((newNode = new Node(l.aux->getData())) == nullptr) {
            throw Exception("Memoria no disponible, copyAll");
        }
    }
    catch(Exception ex) {
        throw Exception(ex.what());
    }
    root = new Node(l.root->getData());
    root->getLeft() = l.root->getLeft();
    root->getRight() = l.root->getRight();
}
```



```
template <class T>
bool ArbolB<T>::isEmpty() const {
    return root == nullptr;
}

template <class T>
void ArbolB<T>::insertData(const T& d) {
    insertData(root, d);
}

template <class T>
void ArbolB<T>::insertData(Position& r, const T& d) {
    if(r == nullptr) {
        try {
            if((r= new Node(d)) == nullptr)
                throw Exception("Memoria no disponible, insertData");
        }
        catch(typename Node::Exception ex) {
            throw Exception(ex.what());
        }
    }
    else {
        if(d < r->getData())
            insertData(r->getLeft(), d);
        else
            insertData(r->getRight(), d);
    }
}

template <class T>
void ArbolB<T>::deleteData(const T& d) {
    deleteData(root, d);
}

template <class T>
void ArbolB<T>::deleteData(Position& e, const T& d) {
    if(e == nullptr)
        return;

    if(e->getData() == d) {
        if(isLeaf(e)) { //si es hoja
            delete e;
            e = nullptr;
            return;
        }

        //caso de dos hojas
        if(e->getLeft() != nullptr && e->getRight() != nullptr) {
```



```
T myData (getHighest (e->getLeft ()) ->getData ()) ;  
  
    deleteData (e->getLeft () , myData) ;  
    e->setData (myData) ;  
    return ;  
}  
  
///Casi una sola hoja  
Position aux (e) ;  
e = e->getLeft () == nullptr ? e->getRight () : e->getLeft () ;  
delete aux ;  
return ;  
}  
  
if (d < e->getData ()) {  
    deleteData (e->getLeft () , d) ;  
}  
else {  
    deleteData (e->getRight () , d) ;  
}  
}  
  
template <class T>  
T ArbolB<T>::retrieve (Position& r) const {  
    return r->getData () ;  
}  
  
template <class T>  
typename ArbolB<T>::Position& ArbolB<T>::findData (const T& e) {  
    return findData (root , e) ;  
}  
  
template <class T>  
typename ArbolB<T>::Position& ArbolB<T>::findData (Position& r , const T& e) {  
    if (r == nullptr || r->getData () == e) {  
        return r ;  
    }  
  
    if (e < r->getData ()) {  
        return findData (r->getLeft () , e) ;  
    }  
    else {  
        return findData (r->getRight () , e) ;  
    }  
}  
  
template <class T>  
typename ArbolB<T>::Position& ArbolB<T>::getLowest () {  
    return getLowest (root) ;  
}  
  
template <class T>  
typename ArbolB<T>::Position& ArbolB<T>::getLowest (Position& r) {
```



```
if (r == nullptr || r->getLeft() == nullptr) {
    return r;
}
return getLowest(r->getLeft());
}

template <class T>
typename ArbolB<T>::Position& ArbolB<T>::getHighest() {
    return getHighest(root);
}

template <class T>
typename ArbolB<T>::Position& ArbolB<T>::getHighest(Position& r) {
    if (r == nullptr || r->getRight() == nullptr) {
        return r;
    }
    return getHighest(r->getRight());
}

template <class T>
bool ArbolB<T>::isLeaf(Position& r) const {
    return r != nullptr && r->getLeft() == r->getRight();
}

template <class T>
int ArbolB<T>::getHeight() {
    return getHeight(root);
}

template <class T>
int ArbolB<T>::getHeight(Position& r) {
    if (r == nullptr) {
        return 0;
    }

    int lH(getHeight(r->getLeft()));
    int rH(getHeight(r->getRight()));

    return (lH > rH ? lH : rH) + 1;
}

template <class T>
int ArbolB<T>::getHeightRight() {
    return getHeight(root->getRight());
}

template <class T>
int ArbolB<T>::getHeightLeft() {
    return getHeight(root->getLeft());
}

template <class T>
void ArbolB<T>::preOrder() {
```



```
    preOrder(root);
}

template <class T>
void ArbolB<T>::preOrder(Position& r) {
    if(r == nullptr) {
        return;
    }
    std::cout << r->getData() << ", ";
    preOrder(r->getLeft());
    preOrder(r->getRight());
}

template <class T>
void ArbolB<T>::inOrder() {
    inOrder(root);
}

template <class T>
void ArbolB<T>::inOrder(Position& r) {
    if(r == nullptr) {
        return;
    }
    inOrder(r->getLeft());
    std::cout << r->getData() << ", ";
    inOrder(r->getRight());
}

template <class T>
void ArbolB<T>::posOrder() {
    posOrder(root);
}

template <class T>
void ArbolB<T>::posOrder(Position& r) {
    if(r == nullptr) {
        return;
    }
    posOrder(r->getLeft());
    posOrder(r->getRight());
    std::cout << r->getData() << ", ";
}

template <class T>
void ArbolB<T>::deleteAll() {
    if(root == nullptr) {
        return;
    }
    deleteData(root->getLeft(), root->getData());
    deleteData(root->getRight(), root->getData());
    delete root;
    root = nullptr;
}
```



```
template <class T>
ArbolB<T>& ArbolB<T>::operator = (const ArbolB& e) {
    deleteAll();
    copyAll(e);
    return *this;
}
```

menu.hpp

```
#ifndef MENU_HPP_INCLUDED
#define MENU_HPP_INCLUDED
#include <string>
#include <random>
#include <chrono>
#include <iostream>
#include <windows.h>
#include <functional>
#include "arbolBinario.hpp"

class Menu {
private:
    void view(ArbolB<int>& );
public:
    Menu (ArbolB<int>& );
};

#endif // MENU_HPP_INCLUDED
```

menu.cpp

```
#include "menu.hpp"

using namespace std;

Menu::Menu (ArbolB<int> &tree) {
    system("Color E5");
    view(tree);
}

void Menu::view (ArbolB<int> &tree) {
    int num, i=0, val;
    std::chrono::steady_clock::time_point begin, end;
    default_random_engine
generator (chrono::system_clock::now().time_since_epoch().count());
    uniform_int_distribution<int> distribution(0,100000);
    auto random = bind(distribution, generator);

    cout<<"Cuantos numeros desea insertar en el arbol?"<<endl;
    cin>>num;
    cin.ignore();
    cout<<endl<<endl;
```



```
while(i < num) {
    val = random();
    cout<<"Insertando: "<< val << endl;
    tree.insertData(val);
    i++;
}

cout << endl << endl << endl;

cout << "----- RECORRIDO PRE-ORDER -----" << endl << endl;
tree.preOrder();
cout << endl << endl;

cout << "----- RECORRIDO IN-ORDER -----" << endl << endl;
tree.inOrder();
cout << endl << endl;

cout << "----- RECORRIDO POST-ORDER -----" << endl << endl;
tree.postOrder();
cout << endl << endl;

cout << "----- ELEMENTO MENOR EN EL ARBOL -----" << endl << endl;
cout<< "- " << tree.retrieve(tree.getLowest());
cout << endl << endl;

cout << "----- ELEMENTO MAYOR EN EL ARBOL -----" << endl << endl;
cout<< "- " << tree.retrieve(tree.getHighest());
cout << endl << endl;

cout << "----- ALTURA DEL ARBOL -----" << endl << endl;
cout<< "- " << tree.getHeight();
cout << endl << endl;

cout << "----- ALTURA DEL SUB-ARBOL IZQUIERDO -----" << endl <<
endl;
cout<< "- " << tree.getHeightLeft();
cout << endl << endl;

cout << "----- ALTURA DEL SUB-ARBOL DERECHO -----" << endl <<
endl;
cout<< "- " << tree.getHeightRight();
cout << endl << endl;

cout << "Eliminando arbol..." << endl << endl;
tree.deleteAll();
cout << "Se elimino correctamente :)"<<endl;

}
```



main.cpp

```
#include "menu.hpp"
int main() {
    ArbolB<int> m;
    Menu start(m);
}
```

Impresiones de Pantalla:

Prueba 1:

```
C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe"
Cuantos numeros desea insertar en el arbol?
1000

Insertando: 3557
Insertando: 84797
Insertando: 36468
Insertando: 4236
Insertando: 98742
Insertando: 7390
Insertando: 8143
Insertando: 70673
Insertando: 62088
Insertando: 87755
Insertando: 46076
Insertando: 81825
Insertando: 80911
Insertando: 33275
Insertando: 39276
Insertando: 88328
Insertando: 87089
Insertando: 63524
Insertando: 17682
Insertando: 77278
Insertando: 73550
Insertando: 20890
Insertando: 90336
Insertando: 18692
Insertando: 54348
Insertando: 98274
Insertando: 39831
Insertando: 23468
Insertando: 22105
Insertando: 8654
Insertando: 43094
Insertando: 55655
Insertando: 65090
Insertando: 31150
Insertando: 22524
Insertando: 61155
Insertando: 1051
Insertando: 74713
Insertando: 63699
Insertando: 50681
Insertando: 68302
Insertando: 15125
Insertando: 10489
Insertando: 87800
Insertando: 13025
Insertando: 19448
Insertando: 53383
Insertando: 74910
Insertando: 37643
Insertando: 47063
Insertando: 57185
Insertando: 86660
```



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



```
C:\Users\cervra\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe
```

Insertando: 63489
Insertando: 31577
Insertando: 9628
Insertando: 24377
Insertando: 484
Insertando: 51075
Insertando: 93678
Insertando: 53687
Insertando: 37923
Insertando: 50943
Insertando: 74532
Insertando: 26775
Insertando: 95677
Insertando: 94773

----- RECORRIDO PRE-ORDER -----

□ "C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe"

----- RECORRIDO IN-ORDER -----

85, 105, 233, 341, 460, 484, 558, 655, 825, 837, 943, 1051, 1075, 1131, 1183, 1209, 1221, 1302, 1342, 1497, 1529, 1784, 1961, 2219, 2230, 2318, 2479, 2529, 2559, 2627, 2677, 2733, 2886, 3029, 31, 3428, 3480, 3509, 3557, 3737, 3911, 3960, 4057, 4236, 4252, 4426, 4636, 5002, 5063, 5104, 5155, 5191, 5358, 5323, 5351, 5361, 5371, 5388, 5402, 5412, 5475, 5581, 5705, 5735, 5771, 5966, 6030, 6047, 6249, 6283, 6558, 6695, 6699, 6771, 6851, 7054, 7181, 7130, 7191, 7381, 7390, 7445, 7577, 7709, 7855, 8003, 8143, 8145, 8148, 8197, 8217, 8230, 8425, 8553, 8584, 8593, 8654, 8844, 9025, 9148, 9273, 9289, 9339, 9459, 9577, 9621, 9682, 9651, 9875, 10189, 10446, 10851, 10610, 10779, 10939, 10966, 10993, 11044, 11161, 11164, 11223, 11788, 11428, 11523, 11549, 11766, 11871, 12031, 12461, 12614, 12688, 12744, 12957, 13013, 13025, 13054, 13175, 13192, 13367, 13595, 13739, 13760, 13815, 13879, 14003, 14051, 14143, 14194, 14230, 14341, 14376, 14696, 14851, 14968, 15065, 15096, 15125, 15163, 15846, 15899, 15921, 15997, 16048, 16059, 16189, 16195, 16273, 16277, 16451, 16791, 16824, 16844, 16907, 17000, 17235, 17666, 17682, 17687, 17695, 17889, 18000, 18062, 18059, 18301, 18556, 18650, 18692, 18705, 18801, 18974, 19000, 19038, 19066, 19123, 19157, 19189, 19255, 19448, 19659, 19759, 19867, 20265, 20324, 20412, 20536, 20896, 20992, 22922, 23084, 23161, 23229, 23454, 23466, 23526, 23643, 23685, 23781, 23821, 24095, 24284, 24478, 24730, 24835, 24877, 24446, 24458, 24759, 24935, 25134, 25146, 25273, 25340, 25397, 25454, 25491, 26216, 26009, 26617, 26746, 26261, 26335, 26344, 26638, 26694, 26739, 26763, 26775, 26814, 26991, 27062, 27837, 27131, 27638, 27811, 27859, 28042, 28084, 28268, 28343, 28386, 28666, 28936, 28945, 29348, 29418, 29448, 29424, 29424, 31, 3291, 34944, 35599, 36959, 39285, 39619, 39919, 2992, 30082, 30257, 30283, 30359, 30446, 30514, 30559, 30667, 31837, 31150, 31244, 31364, 31573, 31577, 31656, 31714, 31832, 31979, 32366, 32411, 32422, 32550, 32615, 32743, 32854, 32860, 33031, 33175, 33258, 33273, 33582, 33671, 33908, 33927, 34033, 34075, 34116, 34562, 34582, 34624, 34643, 34699, 34743, 34965, 35229, 35253, 3528, 35365, 35454, 35638, 35685, 35687, 35699, 35715, 35936, 35956, 36036, 36346, 36686, 36758, 36864, 36984, 37359, 37364, 37481, 37561, 37563, 37645, 37872, 37923, 38088, 3817, 38309, 38320, 38415, 38629, 38761, 38774, 39011, 39276, 39277, 39415, 39598, 39591, 39831, 39852, 39965, 40532, 40756, 40788, 40919, 41118, 41168, 41247, 41274, 41367, 41463, 41467, 41461, 41671, 41752, 41810, 41886, 41989, 42011, 42101, 42273, 42327, 42423, 42501, 42640, 42786, 42937, 42956, 43084, 43129, 43248, 43257, 43315, 43336, 43364, 43448, 43489, 43663, 43704, 43793, 44171, 44244, 44480, 44638, 44844, 45278, 45284, 45132, 45362, 45428, 45479, 45496, 45546, 45578, 45728, 45737, 45807, 45884, 46760, 46766, 46855, 46933, 47067, 47155, 47215, 47481, 47476, 47512, 47775, 47808, 47965, 47981, 48278, 48484, 48487, 48637, 48761, 48813, 48868, 48898, 49067, 49195, 49293, 49315, 49574, 49744, 49745, 49758, 49865, 50026, 50114, 50174, 50239, 50308, 50318, 50349, 50353, 50476, 50480, 50549, 50583, 50587, 50610, 50681, 50834, 51004, 51075, 51182, 51389, 51563, 51697, 51703, 51859, 51925, 52022, 52037, 52217, 52361, 52429, 52444, 52529, 52580, 52830, 52853, 53087, 53128, 53171, 53337, 53338, 53449, 53567, 53586, 53607, 53650, 53768, 53850, 53876, 54044, 54094, 54117, 54172, 54278, 54338, 54438, 54547, 54445, 54452, 54471, 54861, 54833, 54939, 54976, 55006, 55231, 55283, 55338, 55536, 55601, 55655, 55699, 55720, 55737, 55773, 55881, 56044, 56268, 56485, 56689, 56731, 56873, 56876, 58944, 58998, 59077, 59155, 59252, 5932, 5933, 56793, 56898, 57050, 57184, 57185, 57267, 57432, 57693, 57886, 57907, 57977, 57997, 58055, 58346, 58583, 58618, 58642, 58662, 60154, 60269, 60219, 60383, 60388, 60428, 60685, 60599, 60713, 60822, 60844, 61111, 61151, 61339, 61387, 61412, 61464, 61688, 61894, 61968, 61969, 62088, 62096, 62134, 62265, 62269, 62291, 62349, 62395, 63096, 63306, 63408, 63489, 63524, 63656, 63699, 63996, 64143, 64247, 64334, 64352, 64561, 64639, 64844, 64982, 65079, 65182, 65289, 65396, 65491, 65575, 65617, 65684, 65781, 65881, 65944, 66055, 66154, 66359, 66733, 66877, 66948, 67044, 67159, 67240, 67349, 67456, 67556, 67653, 67752, 67847, 67944, 68041, 68148, 68248, 68346, 68448, 68549, 68649, 68749, 68829, 68929, 69013, 69131, 69274, 69348, 69448, 69548, 69658, 69764, 69864, 69964, 70064, 70164, 70218, 70277, 70382, 70487, 70589, 70689, 70789, 70889, 70989, 71089, 71189, 71289, 71316, 71446, 71473, 71535, 71641, 71743, 71847, 71953, 72059, 72161, 72279, 72340, 72447, 72545, 72652, 72759, 72866, 72973, 73083, 73190, 73297, 73394, 73491, 73598, 73695, 73794, 73891, 73988, 74085, 74182, 74289, 74396, 74493, 74590, 74697, 74794, 74891, 74988, 75085, 75182, 75289, 75386, 75483, 75580, 75678, 75775, 75872, 75971, 76078, 76175, 77159, 77261, 77359, 77462, 77556, 77653, 77750, 77848, 77945, 78042, 78140, 78237, 78334, 78431, 78531, 78638, 78735, 78832, 78931, 79030, 79130, 79237, 79334, 79431, 79530, 79631, 79730, 79831, 79930, 80031, 80130, 80215, 80323, 80404, 80514, 80641, 80804, 80911, 81055, 81271, 81379, 81486, 81594, 81691, 81798, 81895, 81992, 82091, 82198, 82297, 82396, 82495, 82594, 82693, 82792, 82891, 82990, 83099, 83198, 83297, 83396, 83495, 83594, 83693, 83792, 83891, 83990, 84089, 84188, 84287, 84386, 84485, 84584, 84683, 84782, 84881, 84979, 84995, 85094, 85193, 85292, 85391, 85490, 85589, 85688, 85787, 85886, 85985, 86084, 86183, 86282, 86381, 86480, 86579, 86678, 86777, 86876, 86975, 87074, 87173, 87272, 87371, 87470, 87569, 87668, 87767, 87866, 87965, 88064, 88163, 88262, 88361, 88460, 88559, 88658, 88757, 88856, 88955, 89054, 89153, 89252, 89351, 89450, 89549, 89648, 89747, 89846, 89945, 90044, 90143, 90242, 90341, 90437, 90539, 90638, 90737, 90836, 90935, 91034, 91133, 91232, 91331, 91430, 91539, 91638, 91737, 91836, 91935, 92034, 92133, 92232, 92331, 92430, 92539, 92638, 92737, 92836, 92935, 93034, 93133, 93232, 93331, 93430, 93539, 93638, 93737, 93836, 93935, 94034, 94133, 94232, 94331, 94430, 94539, 94638, 94737, 94836, 94935, 95034, 95133, 95232, 95331, 95430, 95539, 95638, 95737, 95836, 95935, 96034, 96133, 96232, 96331, 96430, 96539, 96638, 96737, 96837, 96936, 97035, 97134, 97233, 97332, 97431, 97530, 97639, 97738, 97837, 97936, 98035, 98134, 98233, 98332, 98431, 98530, 98639, 98738, 98837, 98936, 99035, 99134, 99233, 99332, 99431, 99530, 99639, 99738, 99837, 99936, 99995

233, 105, 85, 484, 466, 341, 837, 943, 825, 655, 558, 1131, 1075, 1221, 1302, 1209, 1497, 1529, 1342, 1704, 1183, 2219, 1961, 2318, 2230, 2529, 2559, 2479, 2677, 2886, 2733, 3323, 3488, 3428, 3099, 3829, 2627, 1051, 3911, 4857, 3960, 3737, 4636, 5063, 5002, 5159, 1515, 5328, 5104, 4426, 5361, 5388, 5371, 5412, 5475, 5482, 5581, 5335, 4252, 5735, 5771, 6080, 5966, 5705, 6249, 6699, 6656, 6283, 6643, 7101, 7136, 7054, 6881, 6755, 7811, 7191, 7577, 7855, 8083, 7709, 7445, 8181, 8145, 8197, 8553, 8523, 8217, 8593, 8584, 9273, 9148, 9289, 9025, 8844, 9459, 9628, 9621, 9577, 10189, 9875, 10464, 9651, 9339, 10610, 10591, 10939, 10779, 10993, 11045, 11044, 11164, 11161, 11223, 10966, 11428, 11380, 11549, 11766, 11523, 11278, 12301, 12461, 11871, 12744, 12597, 1031, 12688, 12614, 13357, 13192, 13595, 13739, 13760, 13478, 13175, 13084, 13583, 14043, 14143, 14051, 14234, 14190, 14893, 14668, 10565, 14968, 14376, 15098, 14341, 13815, 18025, 18489, 15864, 15921, 16048, 15997, 16074, 16273, 16195, 16059, 16791, 16824, 16844, 17266, 17235, 17080, 16987, 16451, 16277, 15899, 15653, 15125, 1854, 18000, 17289, 17765, 17687, 18059, 18301, 18556, 18565, 18002, 18795, 18801, 18974, 19123, 19066, 19189, 19157, 19030, 19000, 19255, 19650, 20265, 20412, 20536, 20384, 19267, 19759, 19448, 18692, 20922, 21047, 20930, 21336, 21751, 21584, 21399, 212



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos





Prueba 2:

```
[1] "C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe"
```

```
Cuantos numeros deseas insertar en el arbol?
```

```
1200
```

```
Insertando: 10945
Insertando: 58095
Insertando: 74600
Insertando: 60309
Insertando: 84376
Insertando: 67173
Insertando: 39877
Insertando: 99542
Insertando: 53870
Insertando: 63570
Insertando: 94692
Insertando: 40960
Insertando: 98863
Insertando: 27043
Insertando: 94603
Insertando: 39983
Insertando: 76778
Insertando: 62569
Insertando: 62511
Insertando: 88465
Insertando: 76812
Insertando: 29665
Insertando: 64332
Insertando: 98990
Insertando: 73641
Insertando: 37134
Insertando: 3179
Insertando: 42470
Insertando: 75252
Insertando: 18841
Insertando: 63483
Insertando: 26384
Insertando: 81689
Insertando: 1627
Insertando: 51697
Insertando: 45688
Insertando: 51757
Insertando: 51769
Insertando: 60731
Insertando: 67059
Insertando: 26918
Insertando: 1979
Insertando: 74114
Insertando: 96806
Insertando: 69240
Insertando: 85635
Insertando: 27951
Insertando: 54396
Insertando: 6462
Insertando: 15830
Insertando: 58187
Insertando: 19708
```



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



```
C:\Users\cerval\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe
```

----- RECORRIDO PRE-ORDER -----

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe"



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



----- RECORRIDO POST-ORDER -----

RECORRIDO POST-ORDER

C:\Users\cova\Escritorio\F Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe

Closes https://github.com/progress/structured-data#arbozip-exe

----- ELEMENTO MENOR EN EL ARBOL -----

- 57 -

----- ELEMENTO MAYOR EN EL ARBOL -----

- 99975

----- ALTURA DEL ARBOL -----

- 21

----- ALTURA DEL SUB-ARBOL IZQUIERDO -----

- 14

----- ALTURA

- 20

Eliminando arbol...

Se eliminó correctamente



Prueba 3:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe"

Cuantos numeros desea insertar en el arbol?

1500

```
Insertando: 70254
Insertando: 23501
Insertando: 77675
Insertando: 42160
Insertando: 71420
Insertando: 14973
Insertando: 53141
Insertando: 21222
Insertando: 64584
Insertando: 27151
Insertando: 11224
Insertando: 38988
Insertando: 10621
Insertando: 4858
Insertando: 56036
Insertando: 75166
Insertando: 75540
Insertando: 56513
Insertando: 93011
Insertando: 88286
Insertando: 65158
Insertando: 83872
Insertando: 89268
Insertando: 75564
Insertando: 65439
Insertando: 2977
Insertando: 39687
Insertando: 96958
Insertando: 12194
Insertando: 41064
Insertando: 42647
Insertando: 43235
Insertando: 36394
Insertando: 63093
Insertando: 65455
Insertando: 61811
Insertando: 31188
Insertando: 25049
Insertando: 97630
Insertando: 12585
Insertando: 17064
Insertando: 93376
Insertando: 14537
Insertando: 17965
Insertando: 41970
Insertando: 65818
Insertando: 65483
Insertando: 44784
Insertando: 59553
Insertando: 85090
Insertando: 55920
Insertando: 11573
```



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe"

----- RECORRIDO PRE-ORDER -----

```
C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe  
95316, 95355, 95481, 95635, 96612, 95928, 95813, 95766, 95759, 95  
, 97703, 97663, 97715, 98028, 97764, 97967, 98039, 99838, 99303,
```

----- RECORRIDO IN-ORDER -----

127	317	370	380	556	633	749	764	825	874	894	1167	1194	1212	1215	1344	1420	1439	1630	1751	1780	1818	1824	1874	1923	1939	1944	2112	2114	2199	2283	2225	2315	2408	2
425	2476	2637	2791	2932	2977	2999	3016	3057	3124	3138	3288	3296	3392	3398	3452	3468	3530	3536	3537	3541	3580	3605	3614	3642	3667	3748	3796	3864	4064	4070	4117	4165	418	
8	4186	4191	4211	4323	4327	4481	4484	4551	4554	4568	4586	4862	4867	4912	4999	5162	5229	5294	5363	5369	5448	5522	5619	5830	5876	6009	6122	6146	6486	6518	6669	6630		
84	3834	6857	6843	7048	7125	7128	7184	7221	7256	7288	7447	7491	7498	7498	7527	7567	7567	7613	7722	7823	7831	7958	8105	8139	8191	8239	8505	8544	8746	8781	8833	8		
914	3010	3066	3073	318	9193	9194	9226	9285	9443	9468	9474	9485	9495	9511	9521	9525	9549	9572	9603	9882	9914	10008	10059	10120	10168	10285	10346	10468	10500	10641				
1513	10604	10690	10623	10709	10827	10885	11026	11165	11206	11241	11271	11224	11308	11347	11351	11384	11528	11556	11573	11609	11645	11723	11759	11800	11864	11924	11961	12028	12091	12149	12247	12496	1435	
1483	12585	12624	12669	12711	13023	13075	13084	13084	13123	13257	13288	13347	13351	13384	13403	13518	13523	13565	13569	13572	13573	13745	13800	13827	14128	14177	14233	14263	14272	14296	14371			
1484	14413	14517	14733	14857	14933	14983	15113	15113	15149	15152	15158	15163	15169	15172	15173	15176	15177	15177	15178	15179	15180	15181	15182	15183	15184	15185	15186	15187	15188	15189	15190			
1693	16553	16592	16614	16663	16675	16738	16756	16823	16868	17046	17054	17064	17156	17223	17227	17231	17235	17256	17257	17282	17291	17335	17863	17864	17907	17941	1797	17987	18078	18159				
1817	1824	1828	1830	1838	1843	1850	1854	18564	18624	1879	1886	18963	19063	19243	19287	19364	19366	19510	19697	19723	19855	19900	20057	20130	20244	20301	20395	20407	20427					
20585	20698	20726	20731	20936	21176	2128	21256	21344	21361	21466	21722	21789	21851	21864	21949	22053	22082	22202	22349	22454	22469	22485	22545	22573	22578	22748	228							
248	22893	22936	22957	22988	23035	23083	23098	23159	23307	23427	23470	23501	23546	23649	23662	23869	23987	23997	24026	24048	24164	24200	24244	24457	24495	2452								
2592	24631	24644	24882	25035	25049	25088	25153	25278	25285	25314	25321	25410	25506	25586	25621	25630	25667	25762	25939	25941	25948	26099	26178	26266	26225	26254	26282	26324	26351	26384	26423			
27994	28106	28177	28220	28291	28383	28572	28590	28667	28706	28752	28754	28787	28808	28827	28847	28867	28902	29108	29118	29203	29214	29244	29275	29313	29432	29473	29534	2959	2974					
2811	29658	29831	29896	29914	29919	29985	29994	29998	30016	30026	30048	30063	30187	30211	30419	30199	30548	30577	30628	30797	30809	30836	30991	31082	31187	31191	31197	31199	31201	31203	31205			
31179	31309	31466	31671	3182	31846	31849	31888	31971	31978	32106	32227	32305	32313	32518	32674	32805	32841	32958	33053	33072	33167	33333	33646	33731	33779	33874	33974	34074	34174	34274	34374			
33810	33874	33890	34143	34324	34345	34362	34385	34419	34445	34548	34608	34612	34622	34844	34931	34991	35018	35080	35200	35206	35326	35351	35358	35422	35442	35452	35454	35456	35458	35459				
37458	35488	35583	35585	35586	35587	35588	35589	35590	35591	35592	35593	35594	35595	35596	35597	35598	35599	35600	35601	35602	35603	35604	35605	35606	35607	35608	35609	35610	35611	35612				
37482	36554	36707	36715	36933	37126	37365	37377	37428	37503	37510	37576	37667	37678	37697	37808	38017	38195	38232	38287	38381	38433	38460	38471	38510	38512	38514	38516	38518	38520	38522				
38842	38983	38985	38987	39017	39033	39065	39124	39149	39156	39287	39305	39347	39367	39368	39370	39371	39372	39373	39374	39375	39376	39377	39378	39379	39380	39381	39382	39383	39384	39385				
40854	40999	41064	41216	41261	41293	41385	41347	41435	41457	41474	41475	41476	41477	41478	41479	41480	41481	41482	41483	41484	41485	41486	41487	41488	41489	41490	41491	41492	41493	41494				
41577	41619	42124	42126	42127	42128	42129	42130	42131	42132	42133	42134	42135	42136	42137	42138	42139	42140	42141	42142	42143	42144	42145	42146	42147	42148	42149	42150	42151	42152	42153				
42498	42467	42468	42476	42479	42496	43016	43045	43048	43051	43054	43057	43060	43063	43066	43069	43072	43075	43078	43081	43084	43087	43090	43093	43096	43099	43102	43105	43108	43111	43114				
43457	43481	43484	44424	44524	44542	44589	44638	44643	44648	44651	44654	44657	44660	44663	44667	44670	44673	44676	44679	44682	44685	44688	44691	44694	44697	44698	44699	44700	44701					
46177	46195	46214	46331	46416	46481	46497	46563	46568	46572	46576	46580	46584	46587	46591	46595	46599	46615	46629	46632	46636	46640	46644	46648	46652	46656	46660	46664	46668	46672	46676	46680			
4835	48355	48357	48359	48361	48363	48365	48367	48369	48371	48373	48375	48377	48379	48381	48383	48385	48387	48389	48391	48393	48395	48397	48399	48401	48403	48405	48407	48409	48411					
4915	4956	49819	50041	50191	50461	50521	50524	50525	50526	50527	50528	50529	50530	50531	50532	50533	50534	50535	50536	50537	50538	50539	50540	50541	50542	50543	50544	50545	50546	50547				
52565	52667	52677	52781	52895	53016	53027	53121	53141	53149	53204	53234	53374	53408	53456	53500	53511	53513	53515	53517	53523	53526	53536	53691	53732	53736	53804	53848	53872	53903	53936				
5648	54481	54707	54711	54714	54746	54753	54761	54764	54765	54767	54768	54770	54771	54772	54773	54774	54775	54776	54777	54778	54779	54780	54781	54782	54783	54784	54785	54786	54787					
56858	56884	56891	56898	57029	57253	57388	57389	57393	57534	57599	57619	57641	57681	57746	57816	57846	57876	57916	57949	57983	58011	58018	58044	58061	58078	58095	58112	58129	58146	58163				
58942	58949	58950	58951	58952	58953	58954	58955	58956	58957	58958	58959	58960	58961	58962	58963	58964	58965	58966	58967	58968	58969	58970	58971	58972	58973	58974	58975	58976	58977					
59782	59835	59879	59897	59907	59919	59940	59944	59949	59954	59960	59964	59968	59970	59974	59978	59982	59986	59990	59994	59998	59999	59999	60000	60012	60151	60164	60176	60186	60236					
6219	92219	92260	92343	92346	92351	92592	92526	92619	92711	92731	92749	92946	92969	93011	93141	93194	93219	93359	93376	93408	93474	93501	93502	93526	93527	93528	93529	93530	93531	93532	93533	93534		



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



E:\Users\cerva\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe"

----- RECORRIDO POST-ORDER -----

■ Seleccionar "C:\Users\cerval\Escritorio\F.Prog\Estructura de datos\arbolBin\bin\Debug\arbolBin.exe"

, 85786, 85577, 85892, 85889, 86152, 85992, 86237, 86398, 86409, 86388, 86357, 86528, 86714, 86786, 86787, 86939, 86800, 86716, 86580, 86499, 86217, 85504, 87170, 87203, 87288, 87203, 87409, 87046, 87751, 87813, 87826, 88158, 88268, 88262, 87823, 87595, 87495, 85090, 88387, 88329, 88380, 88465, 88523, 88493, 88473, 88635, 88600, 88682, 88647, 88360, 88731, 88776, 88885, 89160, 88978, 89224, 88945, 88915, 88731, 89581, 89648, 89727, 89677, 89514, 89731, 89731, 89798, 89979, 8918, 89918, 89791, 90112, 90196, 90472, 90509, 90479, 90798, 90838, 90875, 90856, 91221, 91232, 90909, 91051, 91263, 90184, 91851, 91407, 91524, 91763, 91750, 91562, 91547, 92088, 92548, 92068, 92151, 92184, 91796, 91389, 90666, 92156, 92255, 92251, 92260, 92346, 92351, 92608, 92721, 92619, 92528, 92542, 92731, 92969, 92946, 92749, 92343, 92131, 92868, 88286, 93194, 93253, 93256, 93219, 93295, 93359, 93275, 93141, 93474, 93403, 93551, 93547, 93448, 93585, 9386, 93941, 93946, 93942, 93824, 93968, 93612, 94064, 93966, 94252, 94325, 94576, 94389, 94188, 94784, 94800, 94869, 94835, 94948, 95635, 95316, 95754, 95759, 95766, 95813, 95928, 96085, 96067, 96466, 96443, 96230, 96812, 95663, 95828, 94748, 96760, 96953, 96783, 96963, 97037, 97080, 97160, 97176, 97210, 97048, 97431, 97628, 97294, 97663, 97967, 97764, 98839, 98028, 97715, 97783, 98174, 98162, 98153, 98485, 98533, 98553, 9820, 98321, 98584, 98765, 98683, 98584, 99861, 99284, 98978, 98912, 98320, 99403, 99426, 99381, 99531, 99482, 99482, 99383, 99978, 99890, 99845, 99838, 98876, 97630, 96958, 93101, 77675, 78254,

----- ELEMENTO MENOR EN EL ARBOL -----

- 127

----- ELEMENTO MAYOR EN EL ARBOL -----

- 999/8

ALTOURA DEL ARBOL

----- ALTURA DEL SUB

- 19



Conclusión

En lo personal, desconocía el algoritmo de los árboles binarios, solo los había escuchado o teorizado, pero no me había tocado entrar en la parte de la codificación, es algo interesante como se forma el algoritmo para tomar los valores tanto del subárbol del lado izquierdo como el del lado derecho y que todo en conjunto sea el árbol completo.

Fue una práctica interesante, lo que más trabajo me costó fue crear la función para eliminar un dato del árbol y eliminar todo el árbol; para poder crear el algoritmo me guie del pseudocódigo proporcionado en las diapositivas del profesor y para eliminar todo el árbol, me guie del algoritmo de la Lista Dummy, a partir de ahí, ya solo cree unas modificaciones para el árbol y lo hice más simple y práctico; aunque no aparece la función de eliminar un dato del árbol en el reporte porque no es parte de los requerimientos del problema de la semana, si comprobé su funcionamiento al momento de depurar el código, dando resultados exitosos.