**UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE**

**CARRERA DE INGENIERÍA DE SOFTWARE SEDE LATACUNGA**

**ESTRUCTURA DE DATOS**

**COMP[A0J09]-NRC[6473]-SECCION[902]-PARALELO[3SOA]**

TAREA No.10

TEMA: **ORDENACIÓN, BÚSQUEDA Y ESTRUCTURAS DE DATOS AVANZADAS:** ÁRBOLES- CLASIFICACIÓN DE ÁRBOLES (BINARIOS, AVL, B, B+)- OPERACIONES BÁSICAS CON ÁRBOLES (CREACIÓN, INSERCIÓN, ELIMINACIÓN, RECORRIDOS SISTEMÁTICOS, BALANCEO)

SEMANA 10: DEL 04 DE JULIO 2022 AL 08 DE JULIO 2022

FECHA DE ENTREGA: 04-JULIO-2022

GRUPO No. 1

ALUMNO:

Chancusig Diego

Castro Ariel

Kenneth Acurio

ACTIVIDADES:

1. CONCEPTUALIZACIÓN EXPLÍCITA: ÁRBOLES BINARIOS

2. DISEÑE E IMPLEMENTE EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS; ADEMÁS, IMPLEMENTE LA INSTRUCCIÓN CORRESPONDIENTE QUE LE PERMITA GESTIONAR UNA PRESENTACIÓN PROFESIONAL: ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: SI/NO, SI SELECCIONA “SI” EL ALGORITMO DEBE PERMITIRLE INGRESAR POR TECLADO LOS ELEMENTOS Y REALIZAR LAS OPERACIONES SOBRE ÁRBOLES BINARIOS, CASO CONTRARIO, SI SELECCIONA “NO”, EL ALGORITMO DEBE PRESENTAR, “EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS HA TERMINADO”.

3. DISEÑE E IMPLEMENTE EL ALGORITMO EN BASE A UN PROYECTO SELECCIONADO POR EL GRUPO CORRESPONDIENTE A LAS OPERACIONES SOBRE ÁRBOLES BINARIOS; ADEMÁS, IMPLEMENTE LA INSTRUCCIÓN CORRESPONDIENTE QUE LE PERMITA GESTIONAR UNA PRESENTACIÓN PROFESIONAL: ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: SI/NO, SI SELECCIONA “SI” EL ALGORITMO DEBE PERMITIRLE INGRESAR POR TECLADO LOS ELEMENTOS Y REALIZAR LAS OPERACIONES SOBRE ÁRBOLES BINARIOS, CASO CONTRARIO, SI SELECCIONA “NO”, EL AGORITMO DEBE PRESENTAR, “EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS HA TERMINADO”

4. Nombrar el presente documento de acuerdo al formato indicado a continuación:

TAREA No.10\_MOLINA AGUAYO ERIK DANIEL

DESARROLLO:

1. CONCEPTUALIZACIÓN EXPLÍCITA: ÁRBOLES BINARIOS

**Concepto de Árboles Binarios:**

Los árboles imponen una estructura jerárquica sobre una colección de objetos, como ejemplos comunes se pueden mencionar los árboles genealógicos y los organigramas, y entre otras cosas, los árboles son útiles para analizar circuitos eléctricos y para representar la estructura de fórmulas matemáticas.

La terminología fundamental del árbol dice que éste es una colección de elementos llamados nodos, uno de los cuales se distingue como raíz, junto con una relación de paternidad, que impone una estructura jerárquica sobre todos los nodos. Los nodos, como elementos de la lista, pueden ser de cualquier tipo.

Un árbol binario puede ser relleno cuando todos sus nodos tienen dos hijos o bien son una hoja, sin permitir nodos con un único hijo. Puede ser completo cuando todas sus hojas tienen la misma profundidad.

**Operaciones básicas con Árboles:**

**Creación:**

El método consiste en ir dividiendo los recorridos del árbol en pequeños subárboles, se va encontrando la raíz con el preorden o postorden y se divide en dos subárboles basándonos en el recorrido en orden. En el caso de que los nodos se repitan es conveniente tener los 3 recorridos para identificar más fácilmente cuál de los nodos es la raíz actual

**Inserción:**

La Inserción consta de 3 pasos

1. Creación del objeto de datos compuesto conteniendo el dato y su clave.

2. Determinación del punto de inserción en el árbol binario de búsqueda, después de realizar la búsqueda, si la clave no está en el árbol, la búsqueda regresará un puntero nulo, en el cual deberá ser insertado el nuevo objeto.

3. Asignación de la dirección del nuevo objeto de datos a la variable que contiene el puntero nulo mencionado anteriormente. Es importante mencionar que si la inserción se está haciendo sobre un árbol vacío, el nuevo nodo se convierte en la raíz del árbol.

**Eliminación:**

Para la eliminación, el primer paso es una búsqueda, en caso de ser encontrado el nodo, se

seguirán diferentes pasos para eliminarlo, de acuerdo al número de hijos que existan.

Nodo sin hijos. Este es el caso más sencillo, sólo se establece el puntero apropiado del nodo

padre al valor nulo.

Nodo con un hijo. Se asigna el valor del puntero del nodo padre, para que en lugar de apuntar

al nodo que está siendo borrado, apunte al hijo.

Nodo con dos hijos. Se reemplaza el nodo que se quiere borrar con su predecesor en recorrido

inorden, de esta manera se garantiza el mantenimiento de la propiedad de los árboles binarios de búsqueda.

**Recorridos Sistemáticos:**

* **Preorden:** (raíz, izquierdo, derecho). Para recorrer un árbol binario no vacío en preorden, se deben realizar las siguientes operaciones recursivamente en cada nodo, comenzando con el nodo de raíz:

1. Visite la raíz
2. Atraviese el sub-árbol izquierdo
3. Atraviese el sub-árbol derecho

* **Inorden:** (izquierdo, raíz, derecho). Para recorrer un árbol binario no vacío en inorden (simétrico), se deben realizar las siguientes operaciones recursivamente en cada nodo:

1. Atraviese el sub-árbol izquierdo
2. Visite la raíz
3. Atraviese el sub-árbol derecho

* **Postorden:** (izquierdo, derecho, raíz). Para recorrer un árbol binario no vacío en postorden, se deben realizar las siguientes operaciones recursivamente en cada nodo:

1. Atraviese el sub-árbol izquierdo
2. Atraviese el sub-árbol derecho
3. Visite la raíz

En general, la diferencia entre preorden, inorden y postorden es cuándo se recorre la raíz. En los tres, se recorre primero el sub-árbol izquierdo y luego el derecho.

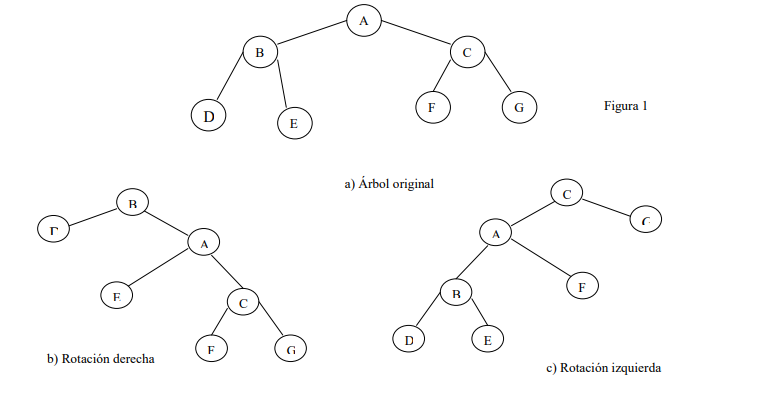
Preorden (antes), inorden (en medio), postorden (después).

**Balanceo:**

Un árbol binario balanceado es un árbol en el cual las alturas de los dos subárboles de todo nodo difiere a lo sumo en 1. El balance de un nodo en un árbol binario se define como la altura de su subárbol izquierdo menos la altura de su subárbol derecho. Cada nodo en un árbol binario balanceado tiene balance igual a 1,-1, o 0 dependiendo si la altura de su subárbol izquierdo es mayor que, menor que o igual a la altura de su subárbol derecho.

Para que el árbol se mantenga balanceado es necesario realizar una transformación en el mismo de manera que:

1. El recorrido en inorden del árbol transformado sea el mismo para que el árbol original(es decir, que el árbol transformado siga siendo un árbol de búsqueda binaria).
2. El árbol transformado esté balanceado

****

El árbol de la figura b) es una rotación del árbol con raíz en A de manera similar, el árbol de la figura c) se dice que es una rotación del árbol con raíz en A.

2. DISEÑE E IMPLEMENTE EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS; ADEMÁS, IMPLEMENTE LA INSTRUCCIÓN CORRESPONDIENTE QUE LE PERMITA GESTIONAR UNA PRESENTACIÓN PROFESIONAL: ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: SI/NO, SI SELECCIONA “SI” EL ALGORITMO DEBE PERMITIRLE INGRESAR POR TECLADO LOS ELEMENTOS Y REALIZAR LAS OPERACIONES SOBRE ÁRBOLES BINARIOS, CASO CONTRARIO, SI SELECCIONA “NO”, EL AGORITMO DEBE PRESENTAR, “EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS HA TERMINADO”

a. IMPLEMENTACIÓN: SELECCIONE EL ALGORITMO, COPIE Y PEGUE, NO COPIE LA PANTALLA. EXPLÍCITO Y CLARO AL 100%

#include <iostream>

#include <stdlib.h>

using namespace std;

typedef struct nodo{

int nro;

struct nodo \*Izquierda, \*Derecha;

}\*Arbol;

int numNodos = 0;

int raices = 0, k;

struct nodoCola{

Arbol ptr;

struct nodoCola \*sgte;

};

struct cola{

struct nodoCola \*delante;

struct nodoCola \*atras;

};

void InicioCola(struct cola &dato)

{

dato.delante = NULL;

dato.atras = NULL;

}

void Datos\_cola(struct cola &dato, Arbol n)

{

struct nodoCola \*dat2;

dat2 = new(struct nodoCola);

dat2->ptr = n;

dat2->sgte = NULL;

if(dato.delante==NULL)

dato.delante = dat2;

else

(dato.atras)->sgte = dat2;

dato.atras = dat2;

}

Arbol Datos\_cola2(struct cola &dato)

{

struct nodoCola \*dat2;

dat2 = dato.delante;

Arbol n = dat2->ptr;

dato.delante = (dato.delante)->sgte;

delete(dat2);

return n;

}

Arbol crearNodo(int numero\_ingresado)

{

Arbol nuevoNodo = new(struct nodo);

nuevoNodo->nro = numero\_ingresado;

nuevoNodo->Izquierda = NULL;

nuevoNodo->Derecha = NULL;

return nuevoNodo;

}

void insertar(Arbol &arbol, int numero\_ingresado)

{

if(arbol==NULL)

{

arbol = crearNodo(numero\_ingresado);

cout<<"\n Se ha ingresado un nuevo elemento"<<endl<<endl;

}

else if(numero\_ingresado < arbol->nro)

insertar(arbol->Izquierda, numero\_ingresado);

else if(numero\_ingresado > arbol->nro)

insertar(arbol->Derecha, numero\_ingresado);

}

void preOrden(Arbol arbol)

{

if(arbol!=NULL)

{

cout << arbol->nro <<" ";

preOrden(arbol->Izquierda);

preOrden(arbol->Derecha);

}

}

void enOrden(Arbol arbol)

{

if(arbol!=NULL)

{

enOrden(arbol->Izquierda);

cout << arbol->nro << " ";

enOrden(arbol->Derecha);

}

}

void postOrden(Arbol arbol)

{

if(arbol!=NULL)

{

enOrden(arbol->Izquierda);

enOrden(arbol->Derecha);

cout << arbol->nro << " ";

}

}

void verArbol(Arbol arbol, int n)

{

if(arbol==NULL)

return;

verArbol(arbol->Derecha, n+1);

for(int i=0; i<n; i++)

cout<<" ";

numNodos++;

cout<< arbol->nro <<endl;

verArbol(arbol->Izquierda, n+1);

}

bool busquedaRec(Arbol arbol, int dato)

{

int r=0;

if(arbol==NULL)

return r;

if(dato<arbol->nro)

r = busquedaRec(arbol->Izquierda, dato);

else if(dato> arbol->nro)

r = busquedaRec(arbol->Derecha, dato);

else

r = 1;

return r;

}

Arbol UnionArbol(Arbol Izquierda, Arbol Derecha)

{

if(Izquierda==NULL) return Derecha;

if(Derecha==NULL) return Izquierda;

Arbol centro = UnionArbol(Izquierda->Derecha, Derecha->Izquierda);

Izquierda->Derecha = centro;

Derecha->Izquierda = Izquierda;

return Derecha;

}

void elimina(Arbol &arbol, int numero\_ingresado)

{

if(arbol==NULL) return;

if(numero\_ingresado<arbol->nro)

elimina(arbol->Izquierda, numero\_ingresado);

else if(numero\_ingresado>arbol->nro)

elimina(arbol->Derecha, numero\_ingresado);

else

{

Arbol dat2 = arbol;

arbol = UnionArbol(arbol->Izquierda, arbol->Derecha);

delete dat2;

}

}

void menu()

{

system("cls");

cout<<" /////OPERACIONES CON ARBOLES BINARIOS////\n"<<endl;

cout<<"1. insertar un nuevo nodo"<<endl;

cout<<"2. Mostrar Arbol (completo)"<<endl;

cout<<"3. Buscar elemento en el Arbol"<<endl;

cout<<"4. Recorridos en fundidad(Pre, in, post - orden)"<<endl;

cout<<"5. Eliminar elemento del arbol"<<endl;

cout<<"6. Altura del arbol"<<endl;

cout<<"7. Salir"<<endl;

cout<<"\n ingrese la opcion: ";

}

int alturaAB(Arbol arbol)

{

int AltIzq, AltDer;

if(arbol==NULL)

return -1;

else

{

AltIzq = alturaAB(arbol->Izquierda);

AltDer = alturaAB(arbol->Derecha);

if(AltIzq>AltDer)

return AltIzq+1;

else

return AltDer+1;

}

}

void menu2()

{

system("cls");

cout << endl;

cout<<"\n RECORRIDOS EN PROFUNDIDAD\n"<<endl;

cout<<"1. Recorrido en profundidad Pre Orden"<<endl;

cout<<"2. Recorrido en profundidad inOrden"<<endl;

cout<<"3. Recorrido en profundidad PostOrden"<<endl;

cout<<"4. Mostrar todos los recorridos"<<endl;

cout<<" ingresar la opcion: ";

}

void continuar() {

int dec;

cout << "\nDesea continuar con EL ALGORITMO DE LAS OPERACIONES SOBRE ARBOLES BINARIOS? 1.SI/2.NO ";

cin >> dec;

if (dec == 1) {

system("cls");

}

else {

system("cls");

cout << "\nEl algoritmo de Arbol Binario ha finalizado" << endl;

exit(0);

}

}

int main()

{

Arbol arbol = NULL;

int numero\_ingresado,numero\_elementos;

int op, op2;

string op3;

do

{

menu(); cin>> op;

cout << endl;

switch(op)

{

case 1:

cout<<"cuantos elementos desea ingresar?: ";

cin>>numero\_elementos;

for(int i=0;i<numero\_elementos;i++){

cout << " Ingrese valor "<<i+1<<"/"<<numero\_elementos<<" : "; cin>> numero\_ingresado;

insertar( arbol, numero\_ingresado);

}

break;

case 2:

verArbol(arbol, 0);

break;

case 3:

bool band;

cout<<" Valor a buscar: "; cin>> numero\_ingresado;

band = busquedaRec(arbol,numero\_ingresado);

if(band==1)

cout << "\nElemento Encontrado";

else

cout << "\nElemento no encontrado";

break;

case 4:

menu2(); cin>> op2;

if(arbol!=NULL)

{

switch(op2)

{

case 1:

enOrden(arbol); break;

case 2:

preOrden(arbol); break;

case 3:

postOrden(arbol); break;

case 4:

cout<<"InOrden: ";

enOrden(arbol);

cout<<"\n"<<endl;

cout<<"preOrden: ";

preOrden(arbol);

cout<<"\n"<<endl;

cout<<"postOrden: ";

postOrden(arbol);

cout<<"\n"<<endl; break;

}

}

else

cout << "\n Arbol vacio";

break;

case 5:

cout<<" Valor a eliminar: "; cin>> numero\_ingresado;

elimina(arbol, numero\_ingresado);

cout << "\nElemento eliminado";

break;

case 6:

cout<<" la altura del arbol es: "<<alturaAB(arbol)<<endl;

break;

case 7:

break;

default:

cout<<"\n\n \*no es una opcion\*"<<endl;

break;

}

system("pause");

cout<<"DESEA CONTINUAR CON EL ALGORITMO DE ARBOLES BINARIOS?(SI/NO)"<<endl;

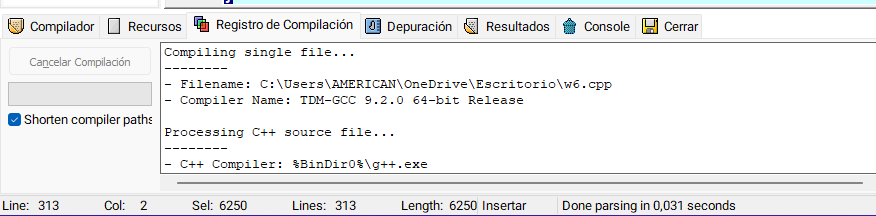
cin>>op3;

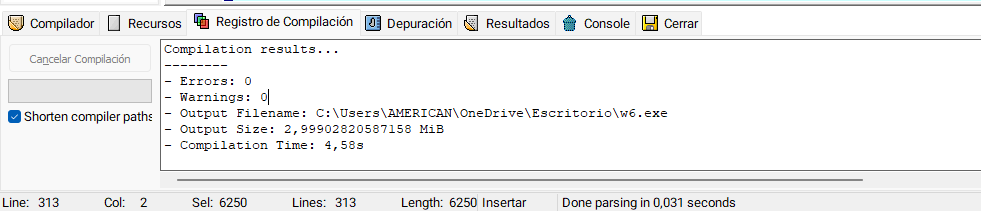
}while(op3=="si" || op3=="SI");

cout << "\nEl algoritmo de Arbol Binario ha finalizado" << endl;

}

b. COMPILACIÓN: COPIE Y PEGUE LA PANTALLA DE LA SECCIÓN DE WARNING Y ERROR. EXPLÍCITO Y CLARO AL 100%





c. EJECUCIÓN: REALICE DOS PRUEBAS CON LA OPCIÓN, ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: “SI”, EL ALGORITMO DEBE PERMITIRLE INGRESAR POR TECLADO LOS ELEMENTOS Y VISUALIZARLOS. LA OTRA PRUEBA CON “NO”, LA EJECUCIÓN DEL AGORITMO DEBE PRESENTAR, “EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS HA TERMINADO”. SELECCIONE, COPIE Y PEGUE LOS RESULTADOS DE CADA UNA DE LAS PRUEBAS. NO LAS PANTALLAS.

c.1. ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: “SI”, EL ALGORITMO DEBE PERMITIRLE INGRESAR POR TECLADO LOS ELEMENTOS Y REALIZAR LAS OPERACIONES SOBRE ÁRBOLES BINARIOS (DOS PRUEBAS)

desea continuar con el algoritmo?: si

\*\*\*MENU\*

1. insertar un nuevo nodo

2. Mostrar Arbol (completo)

3. Buscar elemento en el Arbol

4. Recorridos en profundidad(Pre, in, post - orden)

5. Eliminar elemento del arbol

ingrese la opcion: 2

::::::::ARBOL:::::::

9

8

7

6

1

FIN

Presione una tecla para continuar . . .

desea continuar con el algoritmo?: si

\*\*\*MENU\*

1. insertar un nuevo nodo

2. Mostrar Arbol (completo)

3. Buscar elemento en el Arbol

4. Recorridos en profundidad(Pre, in, post - orden)

5. Eliminar elemento del arbol

ingrese la opcion: 2

::::::::ARBOL:::::::

9

8

7

6

6

3

FIN

Presione una tecla para continuar . . .

c.2. ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: “NO”, “EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS HA TERMINADO”, LO CUAL DEBERÁ SER VISUALIZADO.

::::::::ARBOL:::::::

9

8

7

6

1

desea continuar con el algoritmo?: no

--------------------------------

Process exited after 21.91 seconds with return value 0

Presione una tecla para continuar . . .

::::::::ARBOL:::::::

9

8

7

6

6

3

FIN

desea continuar con el algoritmo?: no

--------------------------------

Process exited after 35.78 seconds with return value 0

Presione una tecla para continuar . . .

3. DISEÑE E IMPLEMENTE EL ALGORITMO EN BASE A UN PROYECTO SELECCIONADO POR EL GRUPO CORRESPONDIENTE A LAS OPERACIONES SOBRE ÁRBOLES BINARIOS; ADEMÁS, IMPLEMENTE LA INSTRUCCIÓN CORRESPONDIENTE QUE LE PERMITA GESTIONAR UNA PRESENTACIÓN PROFESIONAL: ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: SI/NO, SI SELECCIONA “SI” EL ALGORITMO DEBE PERMITIRLE INGRESAR POR TECLADO LOS ELEMENTOS Y REALIZAR LAS OPERACIONES SOBRE ÁRBOLES BINARIOS, CASO CONTRARIO, SI SELECCIONA “NO”, EL AGORITMO DEBE PRESENTAR, “EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS HA TERMINADO”

**NOMBRE DEL PROYECTO:**

a. IMPLEMENTACIÓN: SELECCIONE EL ALGORITMO, COPIE Y PEGUE, NO COPIE LA PANTALLA. EXPLÍCITO Y CLARO AL 100%

#include <iostream>

#include <stdlib.h>

using namespace std;

typedef struct nodo{

string nro;

struct nodo \*Izquierda, \*Derecha;

}\*Arbol;

int numNodos = 0;

int raices = 0, k;

struct nodoCola{

Arbol ptr;

struct nodoCola \*sgte;

};

struct cola{

struct nodoCola \*delante;

struct nodoCola \*atras;

};

void InicioCola(struct cola &dato)

{

dato.delante = NULL;

dato.atras = NULL;

}

void Datos\_cola(struct cola &dato, Arbol n)

{

struct nodoCola \*dat2;

dat2 = new(struct nodoCola);

dat2->ptr = n;

dat2->sgte = NULL;

if(dato.delante==NULL)

dato.delante = dat2;

else

(dato.atras)->sgte = dat2;

dato.atras = dat2;

}

Arbol Datos\_cola2(struct cola &dato)

{

struct nodoCola \*dat2;

dat2 = dato.delante;

Arbol n = dat2->ptr;

dato.delante = (dato.delante)->sgte;

delete(dat2);

return n;

}

Arbol crearNodo(string numero\_ingresado)

{

Arbol nuevoNodo = new(struct nodo);

nuevoNodo->nro = numero\_ingresado;

nuevoNodo->Izquierda = NULL;

nuevoNodo->Derecha = NULL;

return nuevoNodo;

}

void insertar(Arbol &arbol, string numero\_ingresado)

{

if(arbol==NULL)

{

arbol = crearNodo(numero\_ingresado);

cout<<"\n Se ha ingresado un nuevo elemento"<<endl<<endl;

}

else if(numero\_ingresado < arbol->nro)

insertar(arbol->Izquierda, numero\_ingresado);

else if(numero\_ingresado > arbol->nro)

insertar(arbol->Derecha, numero\_ingresado);

}

void preOrden(Arbol arbol)

{

if(arbol!=NULL)

{

cout << arbol->nro <<" ";

preOrden(arbol->Izquierda);

preOrden(arbol->Derecha);

}

}

void enOrden(Arbol arbol)

{

if(arbol!=NULL)

{

enOrden(arbol->Izquierda);

cout << arbol->nro << " ";

enOrden(arbol->Derecha);

}

}

void postOrden(Arbol arbol)

{

if(arbol!=NULL)

{

enOrden(arbol->Izquierda);

enOrden(arbol->Derecha);

cout << arbol->nro << " ";

}

}

void verArbol(Arbol arbol, int n)

{

if(arbol==NULL)

return;

verArbol(arbol->Derecha, n+1);

for(int i=0; i<n; i++)

cout<<" ";

numNodos++;

cout<< arbol->nro <<endl;

verArbol(arbol->Izquierda, n+1);

}

bool busquedaRec(Arbol arbol, string dato)

{

int r=0;

if(arbol==NULL)

return r;

if(dato<arbol->nro)

r = busquedaRec(arbol->Izquierda, dato);

else if(dato> arbol->nro)

r = busquedaRec(arbol->Derecha, dato);

else

r = 1;

return r;

}

Arbol UnionArbol(Arbol Izquierda, Arbol Derecha)

{

if(Izquierda==NULL) return Derecha;

if(Derecha==NULL) return Izquierda;

Arbol centro = UnionArbol(Izquierda->Derecha, Derecha->Izquierda);

Izquierda->Derecha = centro;

Derecha->Izquierda = Izquierda;

return Derecha;

}

void elimina(Arbol &arbol, string numero\_ingresado)

{

if(arbol==NULL) return;

if(numero\_ingresado<arbol->nro)

elimina(arbol->Izquierda, numero\_ingresado);

else if(numero\_ingresado>arbol->nro)

elimina(arbol->Derecha, numero\_ingresado);

else

{

Arbol dat2 = arbol;

arbol = UnionArbol(arbol->Izquierda, arbol->Derecha);

delete dat2;

}

}

int alturaAB(Arbol arbol)

{

int AltIzq, AltDer;

if(arbol==NULL)

return -1;

else

{

AltIzq = alturaAB(arbol->Izquierda);

AltDer = alturaAB(arbol->Derecha);

if(AltIzq>AltDer)

return AltIzq+1;

else

return AltDer+1;

}

}

void menu()

{

system("cls");

cout<<" /////OPERACIONES CON ARBOLES BINARIOS////\n"<<endl;

cout<<"1. Ingrese nuevos nodos"<<endl;

cout<<"2. Mostrar Arbol (completo)"<<endl;

cout<<"3. Buscar elemento en el Arbol"<<endl;

cout<<"4. Recorridos en fundidad (Preorden,PostOrden, InOrden)"<<endl;

cout<<"5. Eliminar elemento del arbol"<<endl;

cout<<"6. Altura del arbol"<<endl;

cout<<"7. Salir"<<endl;

cout<<"\n ingrese la opcion: ";

}

void menu2()

{

system("cls");

cout << endl;

cout<<"\n RECORRIDOS EN PROFUNDIDAD\n"<<endl;

cout<<"1. Recorrido en profundidad Pre Orden"<<endl;

cout<<"2. Recorrido en profundidad InOrden"<<endl;

cout<<"3. Recorrido en profundidad PostOrden"<<endl;

cout<<"4. Mostrar todos los recorridos "<<endl;

cout<<" ingresar la opcion: ";

}

void continuar() {

int dec;

cout << "\nDesea continuar con EL ALGORITMO DE LAS OPERACIONES SOBRE ARBOLES BINARIOS? 1.SI/2.NO ";

cin >> dec;

if (dec == 1) {

system("cls");

}

else {

system("cls");

cout << "\nEl algoritmo de Arbol Binario ha finalizado" << endl;

exit(0);

}

}

int main()

{

Arbol arbol = NULL;

string numero\_ingresado;

int numero\_elementos;

int op, op2;

string op3;

do

{

menu(); cin>> op;

cout << endl;

switch(op)

{

case 1:

cout<<"cuantos elementos desea ingresar?: ";

cin>>numero\_elementos;

for(int i=0;i<numero\_elementos;i++){

cout << " Ingrese valor "<<i+1<<"/"<<numero\_elementos<<" : "; cin>> numero\_ingresado;

insertar( arbol, numero\_ingresado);

}

break;

case 2:

verArbol(arbol, 0);

break;

case 3:

bool band;

cout<<" Valor a buscar: "; cin>> numero\_ingresado;

band = busquedaRec(arbol,numero\_ingresado);

if(band==1)

cout << "\nElemento Encontrado";

else

cout << "\nElemento no encontrado";

break;

case 4:

menu2();

cin>> op2;

if(arbol!=NULL)

{

switch(op2)

{

case 1:

enOrden(arbol); break;

case 2:

preOrden(arbol); break;

case 3:

postOrden(arbol); break;

case 4:

cout<<"InOrden: ";

enOrden(arbol);

cout<<"\n"<<endl;

cout<<"preOrden: ";

preOrden(arbol);

cout<<"\n"<<endl;

cout<<"postOrden: ";

postOrden(arbol);

cout<<"\n"<<endl; break;

}

}

else

cout << "\n Arbol vacio";

break;

case 5:

cout<<" Valor a eliminar: "; cin>> numero\_ingresado;

elimina(arbol, numero\_ingresado);

cout << "\nElemento eliminado";

break;

case 6:

cout<<" la altura del arbol es: "<<alturaAB(arbol)<<endl;

break;

case 7:

break;

default:

cout<<"\n\n \*no es una opcion\*"<<endl;

break;

}

system("pause");

cout<<"DESEA CONTINUAR CON EL ALGORITMO DE ARBOLES BINARIOS?(SI/NO)"<<endl;

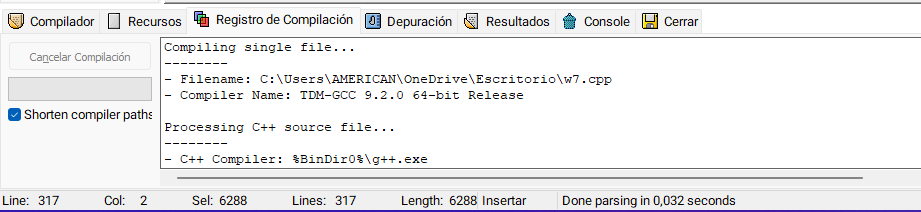
cin>>op3;

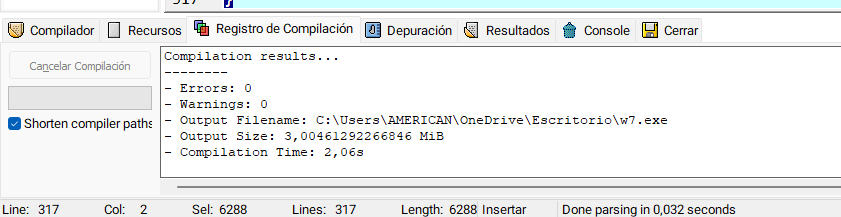
}while(op3=="si" || op3=="SI");

cout << "\nEl algoritmo de Arbol Binario ha finalizado" << endl;

}

b. COMPILACIÓN: COPIE Y PEGUE LA PANTALLA DE LA SECCIÓN DE WARNING Y ERROR. EXPLÍCITO Y CLARO AL 100%





c. EJECUCIÓN: REALICE DOS PRUEBAS CON LA OPCIÓN, ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: “SI”, EL ALGORITMO DEBE PERMITIRLE INGRESAR POR TECLADO LOS ELEMENTOS, REALIZAR LAS OPERACIONES SOBRE ÁRBOLES BINARIOS Y VISUALIZARLOS. LA OTRA PRUEBA CON “NO”, LA EJECUCIÓN DEL AGORITMO DEBE PRESENTAR, “EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS HA TERMINADO”. SELECCIONE, COPIE Y PEGUE LOS RESULTADOS DE CADA UNA DE LAS PRUEBAS. NO LAS PANTALLAS.

c.1. ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: “SI”, EL ALGORITMO DEBE PERMITIRLE INGRESAR POR TECLADO LOS ELEMENTOS Y REALIZAR LAS OPERACIONES SOBRE ÁRBOLES BINARIOS (DOS PRUEBAS)

desea continuar con el algoritmo?: si

\*\*MENU

1. insertar un nuevo nodo

2. Mostrar Arbol (completo)

3. Buscar elemento en el Arbol

4. Recorridos en profundidad(Pre, in, post - orden)

5. Eliminar elemento del arbol

ingrese la opcion: 2

::::::::ARBOL:::::::

Supervivencia

Sociales

Necesidades

Motivacion

Maslow

Fisiologicas

Crecimiento

FIN

Presione una tecla para continuar . . .

desea continuar con el algoritmo?: si

\*\*MENU

1. insertar un nuevo nodo

2. Mostrar Arbol (completo)

3. Buscar elemento en el Arbol

4. Recorridos en profundidad(Pre, in, post - orden)

5. Eliminar elemento del arbol

ingrese la opcion: 2

::::::::ARBOL:::::::

valor

Supervivencia

Sociales

Necesidades

Motivacion

Maslow

Fisiologicas

Empleo

Crecimiento

FIN

Presione una tecla para continuar . . .

c.2. ¿DESEA CONTINUAR CON LA PRESENTACIÓN DEL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS?: “NO”, “EL ALGORITMO SOBRE LAS OPERACIONES DE ÁRBOLES BINARIOS HA TERMINADO”, LO CUAL DEBERÁ SER VISUALIZADO.

::::::::ARBOL:::::::

Supervivencia

Sociales

Necesidades

Motivacion

Maslow

Fisiologicas

Crecimiento

FIN

desea continuar con el algoritmo?: no

--------------------------------

Process exited after 78.46 seconds with return value 0

Presione una tecla para continuar . . .

::::::::ARBOL:::::::

social

liderazgo

FIN

Presione una tecla para continuar . . .

desea continuar con el algoritmo?: no

--------------------------------

Process exited after 18.82 seconds with return value 0

Presione una tecla para continuar . . .