



**INSTITUTO POLITÉCNICO NACIONAL**

---

**ESCUELA SUPERIOR DE CÓMPUTO**

**ESTRUCTURAS DE DATOS**

**PRÁCTICA 7**

**HERNÁNDEZ CASTELLANOS CÉSAR URIEL**

**MARTÍNEZ ISLAS MAURICIO JOEL**

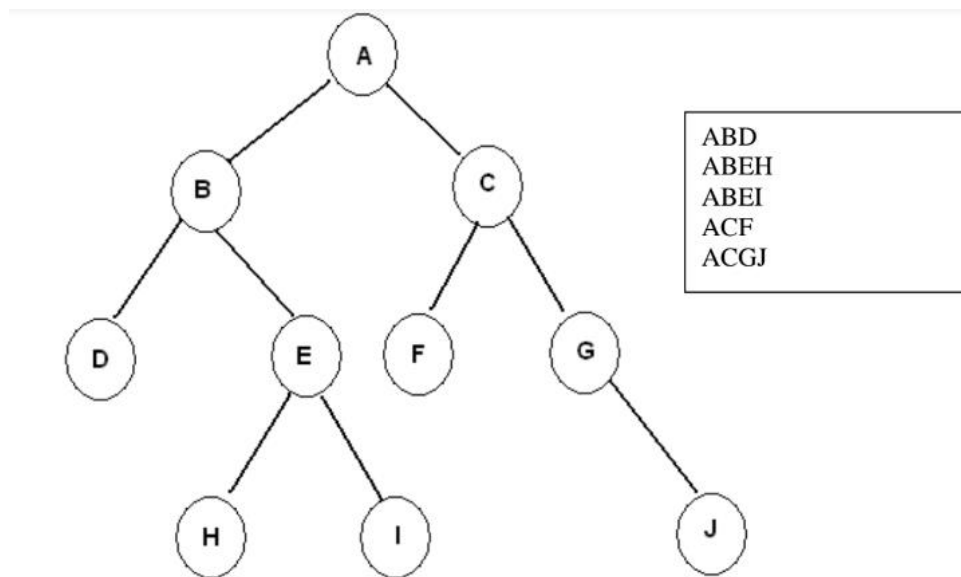
**1CV7**



## Resumen.

En el presente reporte se muestra la documentación de la práctica, cuya función principal es la siguiente:

Dado un árbol de caracteres, mostrar todos los caminos posibles desde la raíz hasta las hojas.



2. Implementar un árbol de expresión.

Entrada: Una expresión validada con paréntesis.

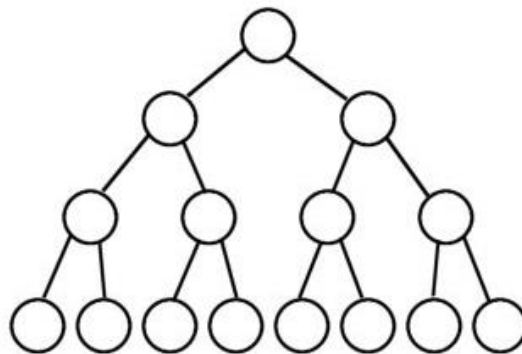
Salida: Un árbol de expresión. Mostrar el recorrido preorden, enorden y postorden.

Introducción.

## Árboles binarios.

Los árboles a diferencia de las listas son una estructura de datos de no lineal, atendiendo más a una estructura de tipo jerárquico. Los árboles son, sin duda, una de las estructuras de datos no lineales, empleadas en informática, tanto para resolver problemas de hardware como de software. Los árboles de directorios son organizaciones bastante empleadas por cualquier usuario o programador de una computadora. De igual manera cumplen un buen papel en la toma de decisiones, valido como árbol de decisiones.

Los árboles genealógicos y los organigramas son ejemplos comunes. Entre otras aplicaciones, los árboles se emplean para analizar circuitos eléctricos y para representar la estructura de fórmulas matemáticas, así como para organizar la información de bases de datos, para representar la estructura sintáctica de un programa fuente en compiladores y para la toma de decisiones.



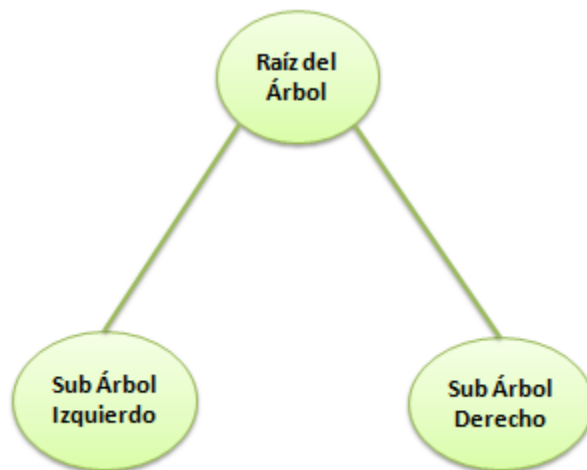
Definición de árboles

Los árboles binarios son estructuras de datos muy similares a las listas doblemente enlazadas, en el sentido que tienen dos punteros que apuntan a otros elementos, pero no tienen una estructura lógica de tipo lineal o secuencial como aquellas, sino ramificada. Tienen aspecto de árbol, de ahí su nombre.

Un árbol binario es una estructura de datos no lineal en la que cada nodo puede apuntar a uno o máximo a dos nodos. También se suele dar una definición recursiva que indica que es una estructura compuesta por un dato y dos árboles. Esto son definiciones simples. Este tipo de árbol se caracteriza porque tienen un vértice principal y de él se desprende dos ramas. La rama izquierda y la rama derecha a las que también se les conoce como subárboles.

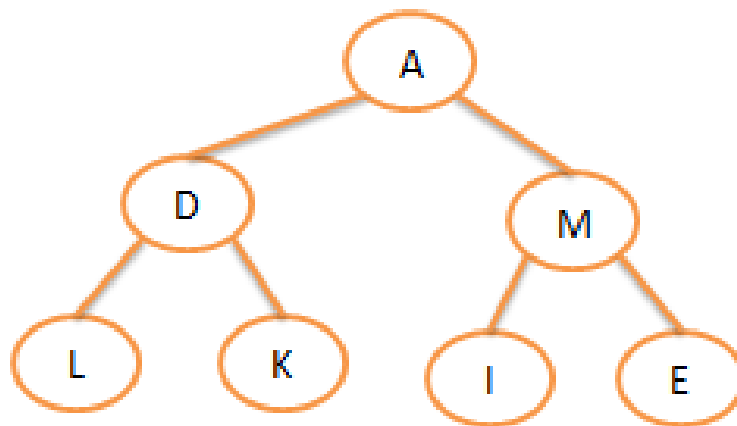
Una representación gráfica de la estructura general de un árbol binario se puede visualizar en la imagen1 que presente a continuación.

Imagen 1. Estructura general de un árbol binario



La rama izquierda y la derecha, también son dos árboles binarios. El Vértice principal se denomina raíz y cada una de las ramas se puede denominar como subárbol izquierdo y subárbol derecho.

Imagen 2. Representación gráfica de un árbol binario



Teniendo en cuenta la gráfica del árbol binario de la figura 2 podemos identificar algunas generalidades y partes del árbol.

**Nodo:** Un árbol binario es un conjunto de elementos cada uno de los cuales se denomina nodo. Un árbol Binario puede tener cero nodos y este caso se dice que está vacío. Puede tener un sólo nodo, y en este caso solamente existe la raíz del árbol o puede tener un número finito de nodos. Cada nodo puede estar ramificado por la izquierda o por la derecha o puede no tener ninguna ramificación.

Con relación al tipo de nodos que hacen parte de los árboles, se identifican algunos nodos:

Nodo hijo: cualquiera de los nodos apuntados por uno de los nodos del árbol. En la gráfica de la imagen 2, se tiene , 'D' y 'M' son hijos de 'A'.

Nodo padre: nodo que contiene un puntero al nodo actual. En el ejemplo, el nodo 'A' es padre de 'D' y 'M'.

Los árboles con los que trabajará tienen otra característica importante: cada nodo sólo puede ser apuntado por otro nodo, es decir, cada nodo sólo tendrá un padre. Esto hace que estos árboles estén fuertemente jerarquizados, y es lo que en realidad les da la apariencia de árboles.

En cuanto a la posición dentro del árbol se tiene:

Nodo raíz: nodo que no tiene padre. Este es el nodo que usaremos para referirnos al árbol. En el ejemplo anterior, es el nodo 'A'.

Nodo hoja: nodo que no tiene hijos. En el ejemplo hay varios: 'L', 'K', 'I', 'E'.

Existen otros conceptos que definen las características del árbol, en relación a su tamaño:

Orden: es el número potencial de hijos que puede tener cada elemento de árbol. De este modo, se dice que un árbol en el que cada nodo puede apuntar a otros dos es de orden dos, si puede apuntar a tres será de orden tres y así sucesivamente.

Grado: el número de hijos que tiene el elemento con más hijos dentro del árbol. En el árbol del ejemplo en la imagen 2, el grado es dos, ya que tanto 'A' como 'D' y 'M' tienen dos hijos, y no existen elementos con más de dos hijos.

Nivel: se define para cada elemento del árbol como la distancia a la raíz, medida en nodos. El nivel de la raíz siempre será cero y el de sus hijos uno. Así sucesivamente. En el ejemplo de la imagen 2, el nodo 'D' tiene nivel 1, el nodo 'L' tiene nivel 2.

Altura: la altura de un árbol se define como el nivel del nodo de mayor nivel. Como cada nodo de un árbol

puede considerarse a su vez como la raíz de un árbol, también se puede hablar de altura de ramas.

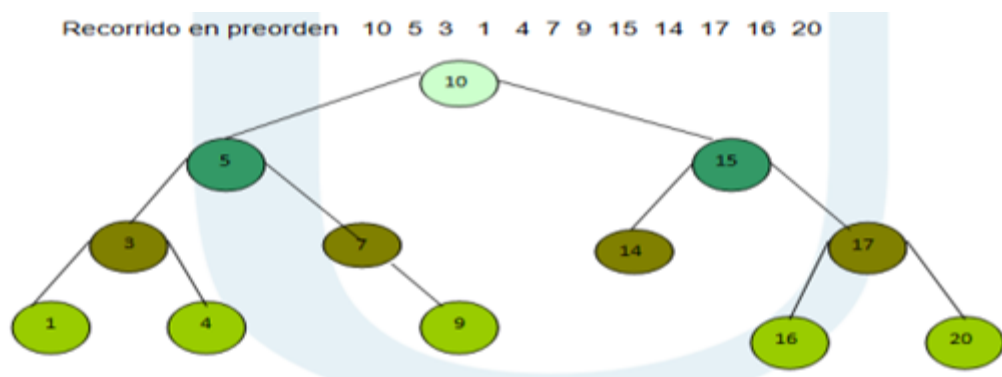
### Formas de recorrer un árbol binario.

Los árboles binarios, son estructuras de datos no lineales, son considerados como estructuras jerárquicas y como tal su forma de recorrerlos difiere sustancialmente en comparación con las listas enlazadas que son estructuras de datos de tipo lineal. En ese orden de ideas, el recorrido de un árbol binario se lleva a cabo en tres sentidos: preorden, inorden y postorden. A continuación se detalla cada caso.

Recorrido en preorden:

Recorrer un árbol en preorden consiste en primer lugar, examinar el dato del nodo raíz, posteriormente se recorrer el subárbol izquierdo en preorden y finalmente se recorre el subárbol derecho en preorden. Esto significa que para cada subárbol se debe conservar el recorrido en preorden, primero la raíz, luego la parte izquierda y posteriormente la parte derecha.

Imagen 3. Representación gráfica del árbol binario y su recorrido en preorden



El recorrido inicia con el subárbol izquierdo, el primer nodo a visitar es la raíz que es el nodo 10, luego se visita el subárbol izquierdo con el nodo 5, posteriormente el 3, luego el nodo 1, sigue con el nodo 4, pasamos al nodo 7 y luego el 9.

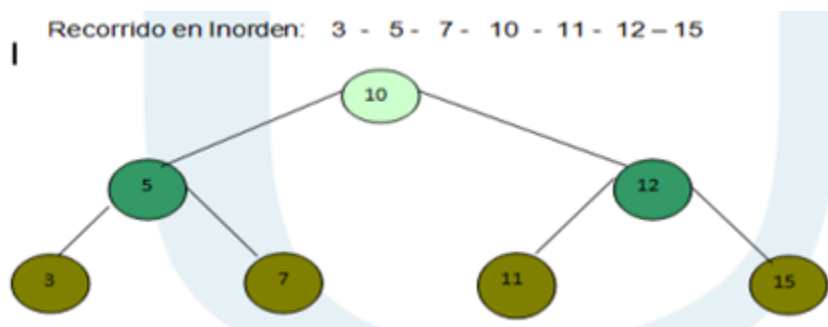
Continuamos con el recorrido del subárbol derecho en preorden, con la visita del nodo 15, luego el 14, se continúa con el 17, se visita el 16 y se finaliza con la visita del nodo 20.

El resultado completo del recorrido en preorden para el árbol de la imagen es: 10 - 5 - 3 - 1 - 4 - 7 - 9 - 15 - 14 - 17 - 16 - 20, Tal como se muestra en la imagen 3.

#### Recorrido en Inorden

Recorrer un árbol en Inorden consiste en primer lugar en recorrer el subárbol izquierdo en Inorden, luego se examina el dato del nodo raíz, y finalmente se recorre el subárbol derecho en Inorden. Esto significa que para cada subárbol se debe conservar el recorrido en Inorden, es decir, primero se visita la parte izquierda, luego la raíz y posteriormente la parte derecha.

Imagen 4. Representación grafica del árbol binario y su recorrido en Inorden





El recorrido inicia con el subárbol izquierdo, el primer nodo a visitar es el 3 luego se visita el 5 y posteriormente el 7, con esto se garantiza que el recorrido del subárbol izquierdo se hizo en Inorden.

Finalizado el recorrido del subárbol izquierdo se visita el nodo de la raíz, que para este caso es el numero 10.

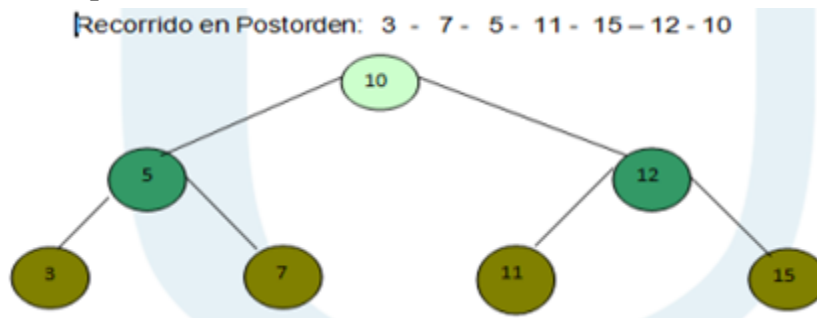
Solo queda recorrer el subárbol derecho en Inorden, es decir se visita el 11 luego el 12 y se finaliza con la visita del nodo 15

El resultado completo del recorrido en Inorden para el árbol de la imagen es: 3 - 5 - 7 - 10 - 11 - 12 - 15, Tal como se muestra en la imagen.

Recorrido en Postorden:

Recorrer un árbol en Postorden consiste en primer lugar en recorrer el subárbol izquierdo en Postorden, luego se recorre el subárbol derecho en Postorden y finalmente se visita el nodo raíz. Esto significa que para cada subárbol se debe conservar el recorrido en Postorden, es decir, primero se visita la parte izquierda, luego la parte derecha y por último la raíz.

Imagen 5. Representación grafica del árbol binario y su recorrido en postorden



El recorrido inicia con el subárbol izquierdo, el primer nodo a visitar es el 3 luego se visita el 7 y posteriormente el 5 que es la raíz, con esto se

garantiza que el recorrido del subárbol izquierdo se hizo en postorden.

Finalizado el recorrido del subárbol izquierdo se inicia la visita al subárbol derecho en postorden, es decir, se visita el 11 luego el 15 y se finaliza con la visita del nodo 12 que sería la raíz de este subárbol.

Solo queda recorrer la raíz del árbol que para este caso es el número 10.

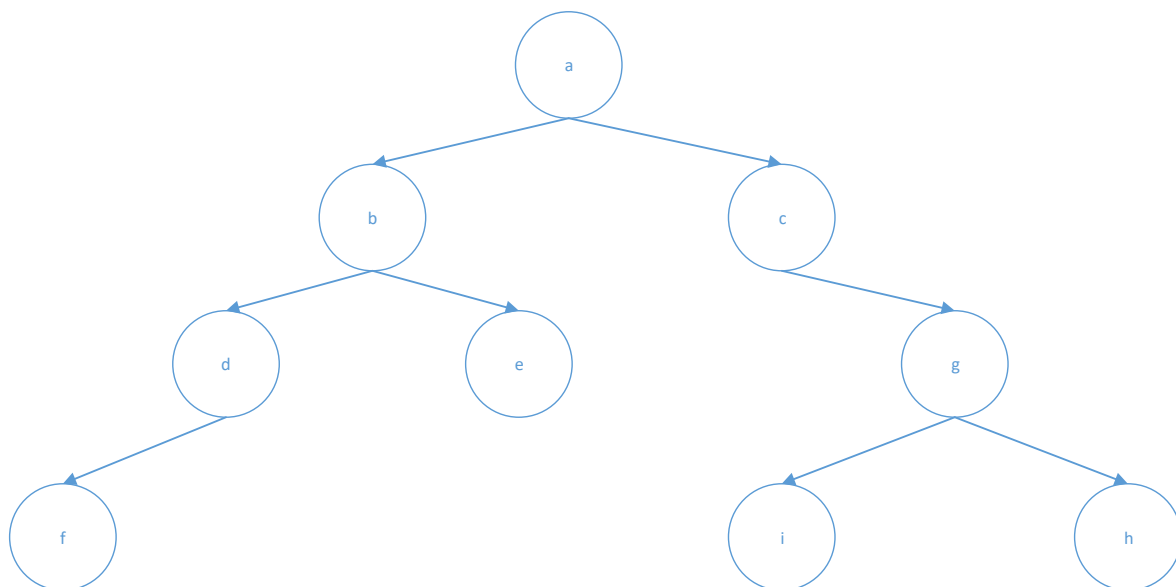
El resultado completo del recorrido en postorden para el árbol de la imagen es:

3 - 7 - 5 - 11 - 15 - 12 - 10

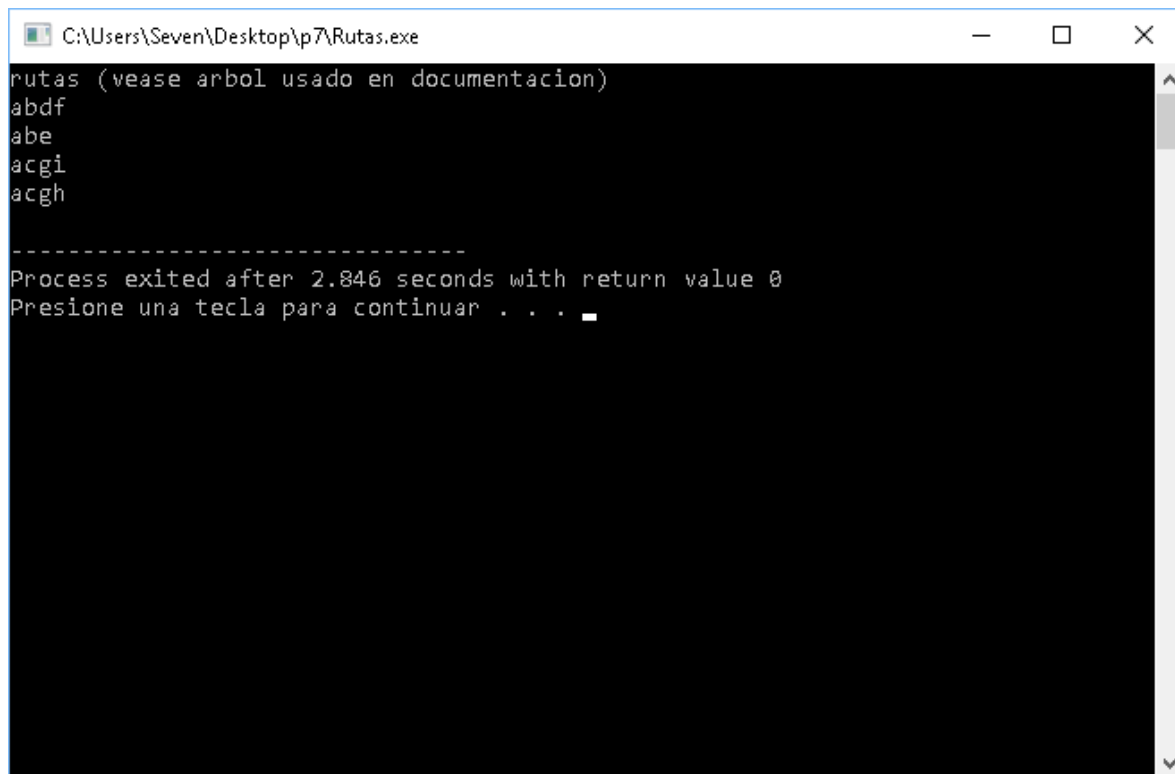
Experimentación y resultados.

La práctica consiste en implementar un programa, el cual te muestre todos los caminos posibles hacia la hoja, el segundo programa trata de mostrar un el recorrido de un árbol de caracteres en enorden, postorden y preorden.

Árbol utilizado:

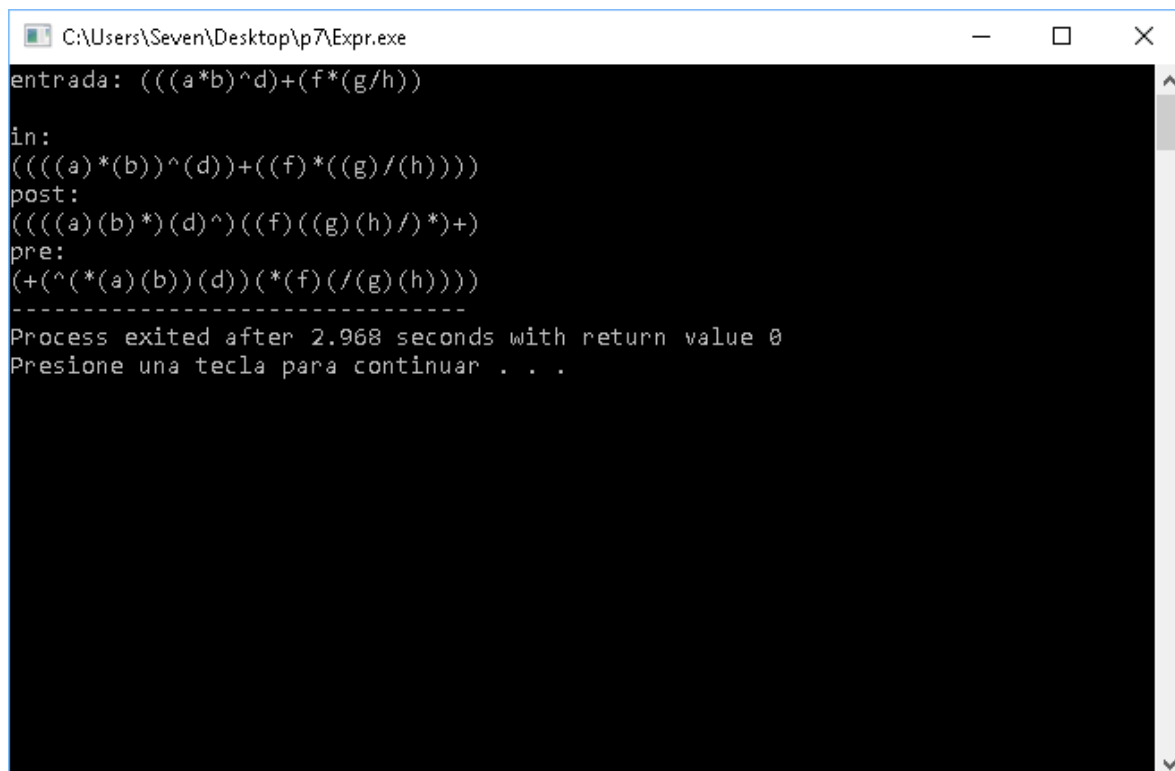


Salida de programa:



```
C:\Users\Seven\Desktop\p7\Rutas.exe
rutas (vease arbol usado en documentacion)
abdf
abe
acgi
acgh

-----
Process exited after 2.846 seconds with return value 0
Presione una tecla para continuar . . .
```



```
C:\Users\Seven\Desktop\p7\Expr.exe
entrada: (((a*b)^d)+(f*(g/h)))

in:
((((a)*(b))^(d))+((f)*((g)/(h))))
post:
((((a)(b)*(d)^((f)((g)(h)/)*+))
pre:
+(^(*a(b))(d))(*f(/g(h))))

-----
Process exited after 2.968 seconds with return value 0
Presione una tecla para continuar . . .
```

Pseudocódigo.

```
void rutasRaizHoja(Nodo* arbol, Pila p):
```

```
    Insertar arbol->valor en pila
```

```
    rutasRaizHoja(arbol->izq, p)
```

```
    Si arbol es hoja:
```

```
        imprimir pila
```

```
    rutasRaizHoja(arbol->der, p)
```

```
Arbol* entradaArbExpresion(char* inString):
```

```
    *arbol = *raiz = NULL
```

```
    Para i en inString:
```

```
        Si inString[i] == '(':
```

```
            Si !arbol:
```

```
                arbol = crear nodo nuevo vacío
```

```
                raiz = arbol
```

```
            Si no:
```

```
                Si !arbol->izq:
```

```
                    arbol = arbol->izq = crearNodo('
```

```
', arbol)
```

```
                Si no:
```

```
                    arbol = arbol->der = crearNodo('
```

```
', arbol)
```

```
        Si entonces inString[i] == ')':
```

```
            arbol = arbol->padre
```

```
Si entonces inString [i] == algun operador:
    arbol->valor = inString[i]
Si no:
    Si !arbol->izq
        arbol->izq = crearNodo(inString[i],
arbol)
    Si arbol->der
        arbol->der = crearNodo(inString[i],
arbol)
```

Conclusiones.

Martínez Islas Mauricio Joel.

Solamente hubo ciertos conflictos de en cuanto a la estructura conceptual de las funciones de arbol, pero como era una sola práctica, se decidió poner las dos funciones (árbol de expresión y rutas posibles) en el mismo archivo de árbol y no separar al árbol de expresión como un subtipo de árbol binario (esto era lo deseable). También surgieron dudas a hora de saber si la manera en la que se implementó la entrada de árbol binario fue eficiente o no, pero de ahí en fuera solamente son detalles.

Hernández Castellanos César Uriel

Un árbol como estructura nos da la posibilidad de almacenar una gran cantidad de datos, con una característica sobresaliente, el cual es que los datos se encuentran ordenados. Un árbol se representa con un conjunto de nodos entrelazados entre sí.

En cuanto a la práctica considero que tuvo un buen desarrollo, exceptuando la problemática que nos surgió por no codificar en el momento las funciones vistas en clase, lo que prolongo de forma considerable el desarrollo de la misma.

## Referencias

[1]Y. Langsam, M. Augenstein and A. Tenenbaum, Estructuras de datos con C y C++, 1st ed. Pearson Prentice Hall, 2000.

[2]S. Villalobos, a las Estructuras de Datos, 1st ed Pearson Educaci3n de M3xico, de

[3]"Data Structures - GeeksforGeeks", GeeksforGeeks, 2017. [Online]. Available: <http://www.geeksforgeeks.org/data-structures/>.

[4]"Data Structures - University of California, San Diego, Higher School of Economics | Coursera", Coursera, 2017. [Online]. Available: