Algoritmos y Estructuras de Datos

Práctico 4

Funciones y Procedimientos
Tratamiento de Listas.
Ejercicios Complementarios
Complemento Teórico

2021

TRABAJO PRÁCTICO N° 4

Tratamiento de Listas

1. Diseñar un algoritmo para acceder, eliminar o insertar el k-ésimo elemento de una lista (siendo k una posición dada). Si la lista está vacía o si el valor de k esta fuera del rango del índice de la lista, invocar al procedimiento de ERROR. En cualquier otro caso, efectuar el procedimiento solicitado.

2. Se dispone de una lista simplemente encadenada de números enteros, diseñar un algoritmo que a partir de ella genere otra lista conteniendo los nodos cuyos datos terminan en cero; dichos elementos, deberán ser eliminados de la lista original. Se asume que la lista está cargada, y que el algoritmo recibe como parámetro de entrada la dirección del primer elemento.

3. Dada una lista simplemente encadenada de números diseñar un algoritmo que calcule en forma independiente: La suma de los números impares. y la suma de los números pares.

4. Se dispone de una lista simplemente encadenada cuyos registros están ordenados en forma ascendente por una clave de tipo entero; diseñar un algoritmo que invierta el orden de la lista.

5. Dada una lista simplemente encadenada que contiene datos de todas las provincias de la República Argentina: nombre, capital, cantidad total de habitantes y cantidad de analfabetos, y está ordenada en forma decreciente por número de habitantes analfabetos, generar otras tres listas que contengan el nombre, la capital y el porcentaje de analfabetos de las Provincias que respondan a las siguientes restricciones.

L1: < = 10 % analfabetos
L2: 16 a 25 % analfabetos
L3: > = 26 % analfabetos

6. En el restaurante ÑOQUIS se está pensando en una solución informática para el soporte de datos del nuevo sistema de atención a clientes. Se han decidido por LISTAS por su dinamismo en cuanto a la cantidad de elementos. Diseñe un algoritmo que realice las siguientes funciones:

- Añadir cliente al ser atendido (lista simple ordenada por Nombre del Cliente).
- Registrar su consumo (Acumular el Total Consumido en valores de montos).
- Realizar el cobro (emitir ticket con Nombre, Fecha, Número de Mesa y Total).
- Eliminar del listado de atención.

La información almacenada debe mantenerse ordenada por Nombre del cliente.

7. Genere un algoritmo que recorra una secuencia texto y genere una lista simplemente encadenada con la frecuencia de utilización de cada letra. La lista debe mantenerse ordenada alfabéticamente y al final informar cual fue la frecuencia de cada letra y cuáles fueron la de mayor y menor frecuencia.

8. Escribir un algoritmo que permita buscar, insertar o borrar un elemento identificado con una clave determinada en una lista circular simplemente encadenada.

9. Dada una lista circular de 8 elementos que contienen, cada uno, un valor numérico entero para encriptación,

Encriptar (NODO)

Multiplicador (N5)	Prox (Puntero)

encriptar un texto ingresado por teclado (arreglo de caracteres de 255 elementos como máximo). El texto encriptado debe almacenarse en una lista de salida simplemente encadenada y, por último mostrar por pantalla.

La encriptación se realiza según las siguientes pautas:

1- Dada la posición del carácter dentro del texto, buscar en la lista circular de encriptación el multiplicador correspondiente, de acuerdo a su posición lógica en la lista. Por ejemplo: como solo hay 8

multiplicadores, si se ingresa un arreglo de 10 caracteres, para encriptar los ultimos dos se utilizarían los multiplicadores 1 y 2.

2- Se usa la función ASCII(Carácter) para convertir cada carácter del arreglo en un entero.
3- Luego se lo multiplican los valores obtenidos en los 2 pasos anteriores.
4- Por último se descompone el resultado obtenido en 3 en cada uno de sus dígitos, se los suma y se guarda en la lista de salida. Ej: si en el paso 3 me dio 123, 1+2+3=6.

10. Se dispone de una lista circular con un conjunto de números naturales. Este conjunto está dividido en 6 grupos. En cada grupo el primer elemento indica la cantidad de números subsiguientes a él que integran dicho grupo.
Se desea generar una nueva lista doblemente encadenada en la cual cada nodo contendrá el promedio de cada grupo. Dicha lista debe quedar ordenada en forma ascendente y, al final se debe informar cuál fue el máximo y el mínimo promedio de todo el conjunto de números.

11. En una empresa de servicios motorizados que NO posee una cantidad fija de integrantes, pero siempre es mayor que 1 (uno). Los datos de las personas está almacenados en un archivo (DNI, Nombre, antigüedad) ordenado por DNI. Realizar un algoritmo que permita generar una lista de asignación (usando una estructura de lista doblemente encadenada), ordenada al principio por antigüedad, y un proceso que realice dicha asignación, en cada solicitud, por turno según orden en la lista. Una vez asignado dicha persona se va al final de la lista para esperar su nueva asignación. Al final del día informar cuantas asignaciones tuvo cada integrante.

Personal

NroPersonal (N5)	Nombre (AN50)	Asignaciones (N5)	Antigüedad (N2)

12. En el nuevo DELIVERY VAMOS-RAPIDO se está diseñando una solución informática para la atención de clientes. Se ha pensado en una LISTA por su dinamismo en cuanto a la cantidad de elementos. Diseñe un algoritmo que realice las siguientes funciones:

- Registrar su pedido en una lista Doble (se agrega en Estado P (Pendiente), y se mantiene ordenado por Nombre del Cliente).
- Realizar el envío (cambia estado del pedido de P (pendiente) a E (Enviado)).
- Eliminar del listado de atención una vez recibido el cobro (al atender el cobro del cadete que hizo el trabajo).
- Datos a almacenar por pedido: Nombre, Dirección, Teléfono y Total.

La información almacenada debe mantenerse ordenada por Nombre del cliente.

13. Dada una lista doblemente encadenada de enteros, ordenada en forma creciente, escriba un algoritmo que elimine las entradas repetidas. El algoritmo debe verificar al inicio que la lista que se pasa como argumento está ordenada; en caso afirmativo realizar la depuración y emitir un mensaje de "ÉXITO"; en caso contrario, emitir un mensaje de "ERROR".

14. Se desea ingresar una serie de números, e imprimirlos en el orden contrario al de ingreso. Diseñar un algoritmo que satisfaga tal requerimiento, utilizando la estructura de datos más apropiada.

15. Supóngase que, utilizando una lista encadenada, se implementa una cola Q (estructura "FIFO", es decir, "primero en entrar, primero en salir"). Diseñar un algoritmo que permita insertar o extraer un elemento, comprobando siempre si la cola está vacía o no.

16. Supóngase que, utilizando una lista encadenada, se implementa una pila P (estructura "LIFO", es decir, "último en entrar, primero en salir"). Diseñar un algoritmo que permita insertar o extraer un elemento, comprobando siempre si la pila está vacía o no.

17. Teniendo en cuenta el ejercicio 1.20 del TP2: "Se posee 2 secuencias (S1 y S2) con las cuales se desea generar una nueva secuencia (SAL) donde se intercalen las palabras de las secuencias de entrada, de la siguiente manera: copiar de S1 aquellas palabras que empiezan y terminan con la misma letra y de S2 aquellas palabras que posean al menos un digito numérico y además estén en posición par.", escribir un algoritmo que lo resuelva, teniendo en cuenta que conoce la estructura de listas.

Complementos Teóricos

Escribir un algoritmo que permita buscar, insertar o borrar un elemento identificado con una clave determinada en una **lista circular simplemente encadenada.**

```
Acción LISTA_CIRCULAR es
AMBIENTE
     elto= Registro
              valor: entero;
              sig: puntero a elto;
     Freg;
     PRIMERO: puntero a elto;
     DATO: entero;
     OP: 1 .. 5;
     P, T : puntero a elto;

ALGORITMO
     PRIMERO := NIL;
     Repetir
       Escribir ('* LISTA CIRCULAR *');
       Escribir ('     1- BUSCAR  ');
       Escribir ('     2- INSERTAR');
       Escribir ('     3- ELIMINAR');
       Escribir ('     4- LISTAR  ');
       Escribir ('     5- TERMINAR');
       Escribir ('* INGRESE OPCION:');
       Repetir Leer (OP) Hasta que OP En [1 .. 5];
       Si OP En [1 .. 3]
            Ent. Escribir ('*** DATO ? ');
                 Leer (DATO);
       Fsi;

       Según OP
            1: Si PRIMERO = NIL          { Buscar }
                 Ent. Escribir ('ERROR, Lista Vacía')
                 Sino T := PRIMERO;
                        Mientras (*T.SIG <> PRIMERO) y (*T.VALOR <> DATO) hacer
                                T := *T.SIG;
                        Fmientras;
                        Si (*T.VALOR <> DATO)
                          Ent. Escribir ('ERROR, ', DATO, ' no está  en la Lista')
                          Sino Escribir ('HALLADO VALOR: ', *T.VALOR);
                        Fsi;
              Fsi;
            2: NUEVO(P);                 { Insertar }
              *P.VALOR:=DATO;
              Si PRIMERO = NIL
                Ent.                     {  es el primer elemento de la lista }
                        *P.SIG:=P;
                        PRIMERO:=P;
                Sino                     {  la lista ya contenía eltos., inserta en el 2º lugar, para no
                                            modificar el puntero del último elto. }
                        *P.SIG := *PRIMERO.SIG;
                        *PRIMERO.SIG := P;
              Fsi;
            3: Si PRIMERO = NIL          { Eliminar }
```

```
            Ent. Escribir ('ERROR, Lista Vacía')
            Sino T := PRIMERO;
                  Mientras (*T.SIG <> PRIMERO) y (*T.VALOR <> DATO) hacer
                        ANT := T;    T := *T.SIG;
                  Fmientras;
                  Si (*T.VALOR <> DATO)
                    Ent. Escribir ('ERROR, ', DATO, ' no está  en la Lista')
                    Sino Si *T.SIG = T
                              Ent.                  { Es el único elemento de la lista }
                                    PRIMERO:=NIL
                              Sino                  { Lista con más de un elemento }
                                    Si PRIMERO = T
                                      Ent.                  { Hay que eliminar el primer elto. Ojo!
                                          P:= PRIMERO;      Actualizar el puntero del último elto.}
                                          Mientras *P.SIG <> PRIMERO  hacer
                                                P := *P.SIG;
                                          Fmientras;
                                          *P.SIG:= *T.SIG; PRIMERO:=*T.SIG;
                                      Sino
                                          *ANT.SIG := *T.SIG;
                                    Fsi;
                              Fsi;
                              DISPONER(T);
                  Fsi;
          Fsi;
        4: Si PRIMERO = NIL
            Ent. Escribir ('ERROR, Lista Vacía')
            Sino T := PRIMERO;
                  Mientras (*T.SIG <> PRIMERO) hacer
                        Escribir (*T.VALOR);   T := *T.SIG;
                  Fmientras;
                  Escribir (*T.VALOR);
          Fsi;
     Fsegún;
   Hasta que OP = 5;
Facción.
```

Escribir un algoritmo que permita buscar, insertar o borrar un elemento determinado en una **lista doblemente encadenada.**

```
Acción LISTA_DOBLE es
AMBIENTE
   elto= Registro
           ant: punt;
           valor: entero;
           sig: puntero a elto;
   Freg;
   PRIMERO: puntero a elto;
   ULTIMO : puntero a elto;
   DATO: entero;
   OP: 1 .. 5;
   P, T: puntero a elto;
```

```
ALGORITMO
    PRIMERO := NIL; ULTIMO := NIL;
    Repetir
      Escribir ('* LISTA DOBLEMENTE ENCADENADA *');
      Escribir ('     1- BUSCAR  ');
      Escribir ('     2- INSERTAR');
      Escribir ('     3- ELIMINAR');
      Escribir ('     4- LISTAR  ');
      Escribir ('     5- TERMINAR');
      Escribir ('* INGRESE OPCION:');
      Repetir Leer (OP) Hasta que OP En [1 .. 5];
      Si OP En [1 .. 3]
            Ent. Escribir ('*** DATO ? ');
                Leer (DATO);
      Fsi;
      Según OP
            1: Si PRIMERO = NIL
                Ent. Escribir ('ERROR, Lista Vacía')
                Sino T := PRIMERO;
                        Mientras (*T.SIG <> NIL) y (*T.VALOR <> DATO)  hacer
                                T := *T.SIG;
                        Fmientras;
                        Si (*T.VALOR <> DATO)
                          Ent. Escribir ('ERROR, ', DATO, ' no está  en la Lista')
                          Sino Escribir ('HALLADO VALOR: ', *T.VALOR);
                        Fsi;
              Fsi;
            2: NUEVO(P);
              *P.VALOR:=DATO;
              Si PRIMERO = NIL
                Ent. *P.ANT:=NIL;
                        *P.SIG:=NIL;
                        PRIMERO:=P;
                        ULTIMO :=P;
                Sino *ULTIMO.SIG:=P;
                        *P.ANT := ULTIMO;
                        ULTIMO := P;
                        *P.SIG := NIL;
              Fsi;
            3: Si PRIMERO = NIL
                Ent. Escribir ('ERROR, Lista Vacía')
                Sino T := PRIMERO;
                        Mientras (*T.SIG <> NIL) y (*T.VALOR <> DATO)  hacer
                                T := *T.SIG;
                        Fmientras;
                        Si (*T.VALOR <> DATO)
                          Ent. Escribir ('ERROR, ', DATO, ' no está  en la Lista')
                          Sino Si PRIMERO = ULTIMO
                                  Ent. PRIMERO:=NIL;
                                       ULTIMO :=NIL;
                                  Sino Si PRIMERO = T
                                       Ent. *(*T.SIG).ANT:=NIL;
                                            PRIMERO:=*T.SIG;
                                       Sino Si ULTIMO = T
                                                Ent. *(*T.ANT).SIG:=NIL;
                                                     ULTIMO:=*T.ANT;
                                                Sino *(*T.ANT).SIG:=*T.SIG;
```

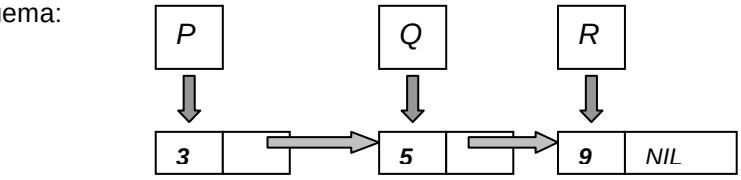
```
                                        *(*T.SIG).ANT:=*T.ANT;
                                    Fsi;
                                Fsi;
                            Fsi;
                            DISPONER(T);
                    Fsi;

        4: Si PRIMERO = NIL
            Ent. Escribir ('ERROR, Lista Vacía')
            Sino T := PRIMERO;
                    Mientras (T <> NIL) hacer
                            Escribir (*T.VALOR);
                            T := *T.SIG;
                    Fmientras;
          Fsi;
      Fsegún;
    Hasta que OP = 5;
Facción.
```

Ejercicios Complementarios T.P. Nro. 4

1. Dada la siguiente declaración y el siguiente esquema:

P, Q, R: puntero a NODO
NODO: registro
DATO: entero
PROX: puntero a nodo
Fin registro



P	Q	R	Comentario

¿Qué hacen las siguientes órdenes?
Considerar que se ejecutan secuencialmente.

1) P:= P ★. PROX
2) Q:= P
3) R:= P★.PROX
4) P★.DATO:= Q★. DATO
5) P★.DATO:= (Q★.PROX) ★. DATO
6) R★.PROX:= P

2. Dada una lista simplemente encadenada de números enteros, diseñar un algoritmo que a partir de ella genere otra lista conteniendo los nodos cuyos datos sean múltiplos de 3, dichos elementos deberán ser eliminados de la lista original. Se asume que la lista está cargada y que el algoritmo recibe como parámetro de entrada la dirección del primer elemento.

3. Dada una lista doblemente encadenada que contiene datos de todas las provincias de la República Argentina: nombre, capital, cantidad total de habitantes y cantidad de analfabetos, y está ordenada alfabéticamente por nombre de provincia, se desea generar otra lista simplemente encadenada pero ordenada en orden decreciente por número de habitantes analfabetos.

4. Dada una lista circular con los datos de los socios de un club: número, nombre y condición (´A´= alta, ´B´= baja); hacer un algoritmo que cree otra lista simplemente encadenada con los socios múltiplos de 100, que no estén dados de baja y eliminar de la lista original los socios dados de baja (cond = ´B´).

5. Una empresa FARMACÉUTICA, posee una lista simplemente enlazada de pedidos realizados por sus clientes en el último mes. Se desea generar dos grupos con la siguiente información:

- Los pedidos de los clientes deudores.
- Los pedidos de los clientes regulares.

Para ello se cuenta con un archivo “CLIENTES” indexado por ***Nro_cliente***.

Deben tenerse en cuenta las siguientes consideraciones:

- La lista de entrada está ordenada por Nro_cliente y pueden existir varios pedidos de un mismo cliente.
- En el caso de recibir un pedido de un cliente que se encuentre dado de baja, debe ser dado de alta automáticamente.
- El total (Monto $) de Dinero_deuda debe ser actualizado con la suma de los pedidos en Cuenta Corriente. Modificar la condición a Deudor si esa suma es mayor a cero.

Formato lista de entrada

Nro_cliente	Nro pedido	Precio pedido	TipoPedido (C:Contado, R: Cuenta Corriente)

Formato lista salida

Nro_cliente	Total pedido

Formato archivo “Clientes”

Nro_cliente	DNI	Domicilio	Dinero_deuda	Deudor (si/no)	Baja

6. Usando las operaciones de Lista, escribir un procedimiento AGRUPA (PRIM, ULT: puntero a nodo; A:entero) que, dada una lista doblemente encadenada de enteros L agrupe (sume) elementos de tal manera que en L queden sólo elementos mayores o iguales que **A**. El algoritmo recorre la lista y cuando encuentra un elemento menor empieza a agrupar (sumar) el elemento con los siguientes hasta llegar al valor **A** o hasta que se acabe la lista (el elemento menor se debe eliminar de la lista.)
Por ejemplo si L = {3,4,2,4,1,4,4,3,2,2,4,1,4,1,4,4,1,4,4,2}, entonces AGRUPA(PRIM, ULT,10) da L = {13,12,13,10,10}. En la lista final NO deben quedar elementos menores que A, salvo, eventualmente, el último.

7. La empresa "Remises Yapú S.A." necesita solucionar el problema de asignación de vehículos para sus clientes. La empresa posee 105 autos, y maneja dos colas (estructuras FIFO) una llamada "No Asignados" y otra llamada "Asignados".
En la primera se encuentran todos los autos que no han sido asignados a algún cliente en lo que va del día y en la segunda los autos que ya han sido asignados. La idea de la cola “Asignados” es manejar en forma equitativa la asignación de turnos, es decir que cada vehículo tenga igual oportunidad de obtener un cliente.
Los vehículos se asignan de la siguiente forma:

- Si existen autos “No Asignados”, se toma uno de ellos y se lo mueve a la cola de “Asignados”.
- Si no existen autos “No Asignados”, se toma uno de la cola de “Asignados” , y se lo coloca al final de la cola.

Siempre que se asigna un vehículo se debe incrementar en uno un campo “cantidad_de_clientes", a fin de saber cuantos clientes tuvo.
Por final de proceso se requiere saber la cantidad total de clientes de ese día.
Se pide: Confeccionar una subacción que realice la asignación de vehículos y el mantenimiento de las colas, usando listas.

8. Se dispone de una lista doblemente enlazada con un conjunto de números naturales. Este conjunto está dividido en 10 grupos. En cada grupo el primer elemento indica la cantidad de números subsiguientes a él que integran dicho grupo.
Se desea generar una nueva lista circular, simplemente encadenada, en la cual cada nodo contendrá el valor máximo de cada grupo. Dicha lista debe quedar ordenada en forma ascendente y, al final se debe informar cuál es el promedio de todos los máximos.

9. El ciclo de desarrollo de software tradicional, incremental e iterativo cuenta con las siguientes etapas: Captura de Requisitos, Análisis, Diseño, Desarrollo, Implementación y Prueba.
De la última etapa se obtiene un informe que contiene las modificaciones que deben realizarse al software para que cumpla con las funcionalidades para las cuales fue creado inicialmente.
Suponga que cuenta con una lista de proyectos de software de los cuales se tiene la siguiente información:

Proy AN(20)	Resp AN(30)	Fch_Ini Fecha	C_Error N(3)

El dato Cant_Errores indica la cantidad de errores que deben solucionarse en cada proyecto.

La información respecto a cada Error se almacena en una lista de acuerdo al orden de los proyectos y teniendo en cuenta la cantidad que le corresponden a éste. Por ejemplo si en la Lista 1 el Proyecto A tiene 3 errores, los 3 primeros nodos de la Lista 2 corresponden al Proyecto A. La información que se almacena por Error es la siguiente:

Desc_E AN(30)	Est AN(1)

Est – M: En Modificación | O: En Observación | R: Resuelto

Durante el proceso de Control de Proyectos se analiza cada error de cada proyecto, y se le consulta al Responsable el estado del Error (en caso que no haya sido Resuelto) para actualizarlo.
El Responsable debe tener la opción de recorrer todos los proyectos, en el orden almacenados, las veces que considere necesario, luego de analizar el último en la lista.

Suponiendo que se necesita simular el control de los proyectos, creando dos listas para almacenar por un lado **Proyectos** y por otro **Errores**. Y que además se debe informar al final del proceso qué cantidad de proyectos pasaron el control con todos sus errores resueltos. Se le solicita a ud:

a) Dado el Algoritmo resuelto pero incompleto, analice la propuesta presentada e indique las sentencias incorrectas, encerrándolas con un círculo, e indique las modificaciones necesarias para que funcione, complete la líneas de puntos y agregue las acciones que falten.

b) Desarrolle la acción Control_Proyectos y complete el ambiente si fuera necesario

```
Accion Ej_1  (PRIM: Puntero a Nodo, PRIM2: Puntero a Nodo2)
Es
  Ambiente
   NODO = Registro
    Proy: N(30)
    Resp: AN(30)
    Fecha: Fecha
    C_error: N(3)
    Prox: Puntero a Nodo
   FinRegistro
   NODO1 = Registro
    Desc_E: AN(30)
    Est: AN(1)
    Prox: Puntero
   FinRegistro
P, K: Puntero a Nodo
Q : Puntero a Nodo1
Cont, cant_proy: N(3)
Opc: caracter

Subaccion Cargar_Proyectos es
Escribir("Ingrese 'S' para empezar o cualquier otro caracter para
salir")
Leer(opc)
Mientras opc = 'S' hacer

   ……
   Leer(*P.Proy, *P.Resp, *P.Fch_Ini, *P.C)

   Si ……. Entonces
            P := *P.prox;
             PRIM:=P;
   Sino
            K := PRIM
            Mientras (*K.prox <> PRIM) hacer
               K := *K.prox
            FinMientras
            *K.prox := P
            *P.prox := PRIM
   Fsi;
  Cont:= 0
   Cargar_Errores
   Escribir("Ingrese 'S' para seguir cargando proyectos o cualquier
    otro caracter para salir")
    Leer(opc)
Fin Mientras
Fin Subaccion
```

```
Subacción Cargar_Errores es
Mientras Cont <= *P.C_error hacer
    NUEVO(Q)
    Escribir(*Q.Desc_E, *P.Est)
    Si PRIM2 = NIL Entonces
            *Q.prox := Nil

            …… := Q
     Sino
            *Q.prox := PRIM2
            PRIM2 := Q
     FinSi

     ……
Fin Mientras
FinSubaccion


Algoritmo

   Cargar_Proyectos;
   Control_Proyectos;
   Escribir('La cantidad de proyectos con errores resueltos es
de', *P.C_error)

Fin Accion
```

Algoritmos y Estructuras de Datos

Práctico 5

Recursividad
Árboles
Ejercicios Complementarios

2021

TRABAJO PRACTICO Nro. 5

Recursividad

1. Crear una función recursiva para solucionar cada uno de los enunciados siguientes:

   1.01. Calcular el factorial de un número positivo n. Tener en cuenta la definición matemática de n!:

   $$n! = \begin{cases} 1 \text{ si } n = 0 \\ n^* (n-1)! \quad \text{si } n>0 \end{cases}$$

   1.02. Dado un número n como parámetro de entrada, calcular el n-esimo número de la serie de Fibonacci. Tener en cuenta la siguiente definición:

   $$Fib(n) = \begin{cases} 1 \text{ si } n = 1, n = 2 \\ Fib(n-1) + Fib(n-2) \quad \text{si } n > 2 \end{cases}$$

   1.03. Dados dos números: a y b. Calcule la potencia a^b, usando sólo multiplicaciones sucesivas.

   1.04. Construir un algoritmo recursivo que permita determinar si los dígitos de un número n dado son todos pares.

   1.05. Dados dos números enteros, divídalos (división entera) y muestre el resultado, usando sólo la operación resta.

   1.06. Determine recursivamente si un número dado es par o impar, usando sólo la operación resta.

   1.07. El algoritmo de Euclides para encontrar el MCD (máximo común divisor) de dos números enteros positivos (m y n) se puede definir recursivamente.
   Algoritmo de Euclides: el MCD de dos enteros es el entero mayor que divide a ambos.

Dividendo	Divisor	Cociente	Resto
m	n	q1	r1
n	r1	q2	r2
r1	r2	q3	r3

   Cuando el último resto es cero (por ej. r3 = 0), el MCD es el último divisor (en ese caso, r2).
   El algoritmo recursivo se puede definir con los siguientes pasos:
   - MCD (m,n) = n, si n<= m y n divide a m
   - MCD (m,n) = MCD (n,m) si m < n
   - MCD (m,n) es MCD (n, resto de m dividido por n)

   Para simplificar el algoritmo considerar que siempre m> n

   1.08. Dado un vector de 10 números enteros, calcular la suma de sus elementos

2. Crear un procedimiento recursivo para solucionar cada uno de los enunciados siguientes:

   2.01. Imprimir las cifras de un número n (siendo n >= 0) en orden inverso al original. Por ej.: el inverso de 254 es 452.

   2.02. Leer una palabra (una cadena de caracteres) y la cantidad de caracteres y generar su palíndromo. El palíndromo de “Venezuela” es “aleuzeneV”.

   2.03. Dada una lista de nombres ordenada en forma ascendente, construir un procedimiento recursivo que imprima como salida la misma lista, pero en orden descendente, sin modificar la lista original.

2.04. Para convertir un número decimal a binario, simplemente debe dividirse sucesivas veces por dos (2) hasta quedarnos con el cociente cero (0). Todos los restos de las divisiones, tomados en orden inverso, forman el número binario objetivo. Escribir un procedimiento recursivo que, recibiendo como parámetro un número entero positivo, muestre en pantalla el mismo número codificado en binario.

Árboles

1.Responda las siguientes preguntas para el árbol de la figura:

a) ¿Qué nodos son hojas?

b) ¿Qué nodo es raíz?

c) ¿Cuál es el padre del nodo C?

d) ¿Qué nodos son los hijos de C?

e) ¿Qué nodos son los antecesores de C?

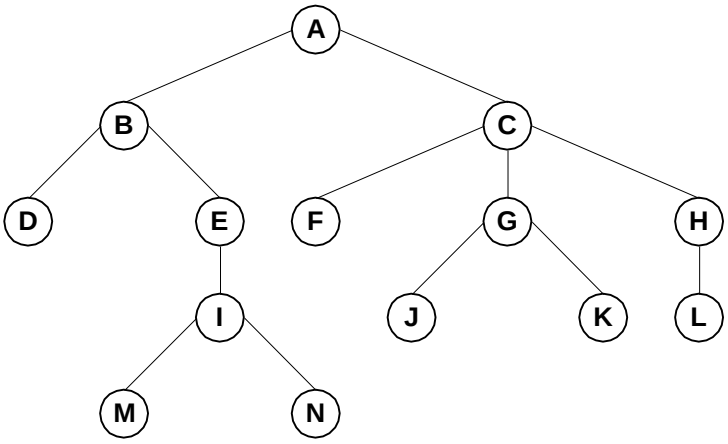
f) ¿Qué nodos son los descendientes de E?

g) ¿Cuáles son los hermanos derechos de D y E?

h) ¿Qué nodos están a izquierda y qué nodos a derecha de G?

i) ¿Cuál es la profundidad del árbol?

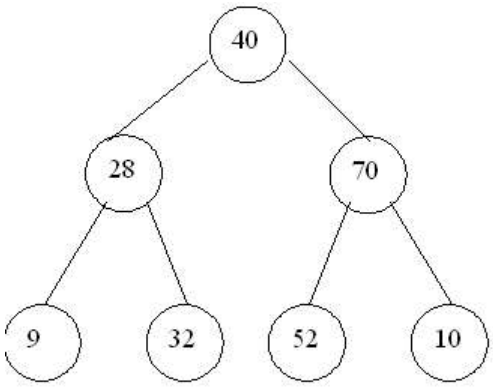
j) ¿Cuál es la altura del nodo C?



2. ¿Cuántos caminos de longitud 3 hay en el árbol representado en la figura anterior?

3. Dada la expresión de la siguiente línea, dibujar el árbol equivalente
[(x-y)*z] / (m+n**p)

4. Dado el siguiente árbol:



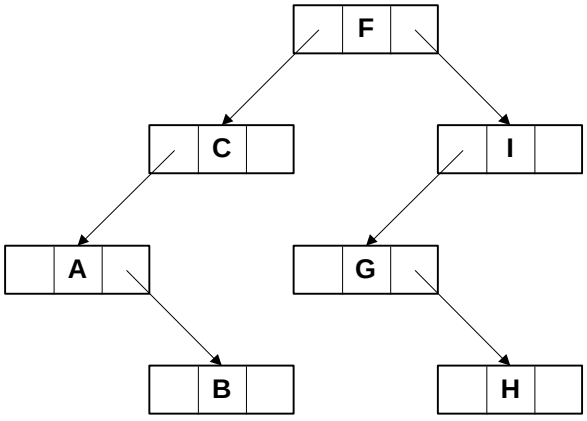
a) Indique el resultado del recorrido post-orden
b) Indique el resultado del recorrido en-orden
c) Elabore un algoritmo para el recorrido en-orden

5.Para las siguientes expresiones:

3.1. (5 + 7) / 8 – (6 * 7) **2
3.2. 3 – 6 + 6 * (8/3)
3.3. 4 / (8 – 6 * (8 + 3))

a) Grafique el árbol equivalente
b) Indique las expresiones resultantes de los recorridos EN-ORDEN, PRE-ORDEN y POST-ORDEN
c) Calcule los resultados numéricos que arroja la ejecución de cada uno de los recorridos.

6. Escriba un algoritmo que permita recorrer el siguiente árbol en los tres procedimientos, Realice la prueba de escritorio:



7. Escribir una función recursiva que encuentre el número de nodos de un árbol binario.

8. Escribir una función recursiva que encuentre la altura de un árbol binario.

9. Se dispone de un árbol binario de enteros. Escribir funciones que calculen:

a) La suma de sus elementos.
b) La suma de sus elementos que son múltiplos de 3.

10. Suponiendo que un árbol está definido como la estructura recursiva de datos:

```
Arbol=registro
        X: entero;
        Izq, Dcha: puntero a arbol;
Freg;
```

Escribir un algoritmo que encuentre un elemento con una clave dada C, y realice una operación P con él.

11. Conviértase la expresión ((a + b) + c * (d + e) + f) * (g + h) en

a) expresión prefija

b) expresión postfija.

12. El recorrido en pre-orden de un determinado árbol binario es: **GEAIBMCLDFKJH** y en-orden **IABEGLDCFMKHJ**.

a) Dibujar el árbol binario.
b) Dar el recorrido en post-orden.
c) Diseñar un algoritmo para recorrer el árbol en post-orden.

13. Supongamos que tenemos una función valor tal que dado un valor de tipo char (una letra del alfabeto) devuelve un valor entero asociado a dicho identificador.
Supongamos también la existencia de un árbol de expresión T cuyos nodos hoja son letras del alfabeto y cuyos nodos interiores son los caracteres *, +, -, /.
Diseñar una función que tome como parámetros un nodo y un árbol binario y devuelva el resultado entero de la evaluación de la expresión representada.

EJERCICIOS DE ÁRBOLES BINARIOS DE BÚSQUEDA (ABB)

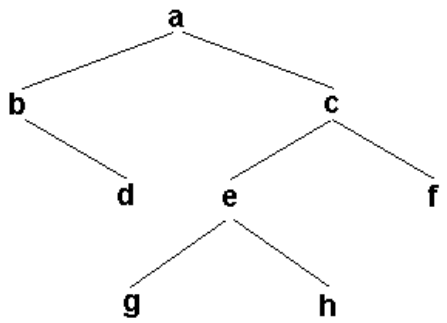
1) ¿Puede reconstruirse de forma única un ABB dado su inorden? ¿Y dados el preorden y el postorden?.

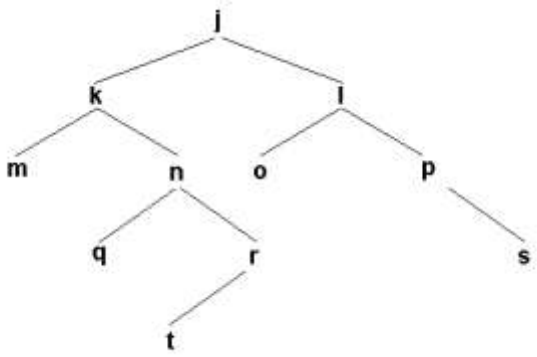
2) Construir un ABB con las claves 50, 25, 75, 10, 40, 60, 90, 35, 45, 70, 42.

3) ¿Bajo qué condiciones puede un árbol ser parcialmente ordenado y binario de búsqueda simultáneamente?. Razonar la respuesta.

EJERCICIOS DE ÁRBOLES EQUILIBRADOS AVL

Dados los siguientes árboles indicar cuál es AVL y cuál no.





Complementos Teóricos

ÁRBOLES EQUILIBRADOS AVL

Diremos que un árbol binario está equilibrado (en el sentido de Addelson-Velskii y Landis) si, para cada uno de sus nodos ocurre que las alturas de sus dos subárboles difieren como máximo en 1. Los árboles que cumplen esta condición son denominados a menudo árboles AVL.

Ejercicios Complementarios T.P. Nro. 5

1. Indique los estados iniciales, intermedios y finales para cada variable incluida en las siguientes acciones

1.01. Acción Calc_101 es
```
    C := TRUNC (ABSO (4/-2) + 3);
    B := REDOND (-4,3);
    B := B + C;
   Escribir (C, B);
Facción.
```

1.02. ALFA vale 2,5; JOTA vale 2;

```
Acción Calc_102 (ALFA, JOTA: entero): entero es
   BETA := ALFA**JOTA;
   ALFA := ALFA + BETA;
  Calc_105 := ALFA;
Faccion
```

1.03. H vale -10;

```
Acción Calc_103 (H: entero) es
   NUM := ABSO (H); A := 3;
   NUM2 := TRUNC (NUM/A) ** 2 /A;
   NUM := NUM2 - NUM * 5;
   B := REDOND (NUM/NUM2);
   A := (A + B) - 9 * 2 -1;
  Escribir ( NUM, A, NUM2, B);
Facción
```

2 Efectué los siguientes conjuntos de acciones, resolviendo las invocaciones en función de las consignas del punto 2. Realice la prueba de escritorio.

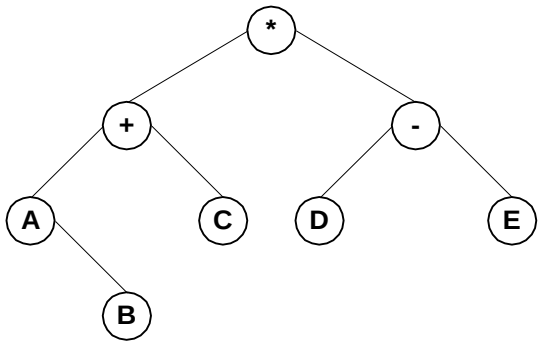
2.01 X:= 7; Y:= 10;
```
Z:=Calc_102 (X,Y);
```

2.02 X:= 4, Y:= 3;
```
K:= Calc_102 (X,Y) * 3 + ABSO(Y-10);
```

2.03 M:= -20;
```
Calc_103(M);
```

2.04. A:=0;
Calc_101;
A:= A + B;

3. Recorrer en los 3 modos el árbol siguiente:



4. El recorrido en orden de un árbol binario produce la secuencia de nodos: dfcgbea. Dibujar el árbol.

5. Escribir una función no recursiva para recorrer un árbol binario en el modo en-orden.

6. Escribir una función no recursiva para recorrer un árbol binario en Post-orden.

Algoritmos y Estructuras de Datos

Práctico 6

Complejidad Algorítmica

2021

Supongamos que disponemos de la siguiente definición de tipo: *vector = ARRAY [1..n] OF INTEGER;* *n = ...;*

Consideramos entonces los procedimientos y funciones siguientes:

a)SUBACCION Algoritmo1(a:vector);

```
 i,j:entero;
temp:entero;
PARA i:=1 HASTA n-1 HACER (* 1 *)
        PARA j:=n HASTA i+1 incremento -1 HACER (* 2 *)
                SI a[j-1]>a[j] ENTONCES (* 3 *)
                    temp:=a[j-1]; (* 4 *)
                    a[j-1]:=a[j]; (* 5 *)
                    a[j]:=temp (* 6 *)
                FIN (* 7 *)
        FIN (* 8 *)
FIN (* 9 *)
FIN Algoritmo1;
```

b)SUBACCION Algoritmo2(VAR a:vector;c:ENTERO):entero;

```
VAR inf,sup,i:entero;
  inf:=1; sup:=n; (* 1 *)
MIENTRAS (sup>=inf) HACER (* 2 *)
           i:=(inf+sup) DIV 2; (* 3 *)
           SI a[i]=c ENTONCES RETURN i (* 4 *)
           CONTRARIO SI c<a[i] ENTONCES sup:=i-1 (* 5 *)
                           CONTRARIO inf:=i+1 (* 6 *)
          FIN (* 7 *)
FIN; (* 8 *)
RETURN 0; (* 9 *)
FIN Algoritmo2;
```

c)SUBACCION Euclides(m,n:CARDINAL):CARDINAL;

```
VAR temp:CARDINAL;
 MIENTRAS m>0 HACER (* 1 *)
           temp:=m; (* 2 *)
           m:=n MOD m; (* 3 *)
           n:=temp (* 4 *)
END; (* 5 *)
RETURN n (* 6 *)
END Euclides;
```

d)SUBACCION Misterio(n:CARDINAL);

```
VAR i,j,k,s:ENTERO;

s:=0; (* 1 *)
PARA i:=1 HASTA n-1 HACER (* 2 *)
           PARA j:=i+1 HASTA n HACER (* 3 *)
                      PARA k:=1 HASTA j HACER (* 4 *)
                                 s:=s+2 (* 5 *)
                      FIN (* 6 *)
           FIN (* 7 *)
FIN (* 8 *)
FIN Misterio;
```

1) Calcular sus tiempos de ejecución en el mejor, peor, y caso medio.
2) Dar cotas asintóticas para las funciones anteriores.

EJERCICIOS COMPLEMENTARIOS GENERALES

Secuencias de Texto, Arreglos y Listas

1. Dada una secuencia TEXTO se desea un algoritmo que cuente cuántas veces aparece la primera palabra del texto dentro del mismo y dentro de cada oración.
2. Escribir un algoritmo que produzca una secuencia ORACION de salida, formada por palabras en posiciones impares y que finalicen con la letra “N” de una secuencia TEXTO de entrada; en la salida cambiar las vocales por su vocal inmediata siguiente. (Ej. Si es “a”, saldrá “e”, si es “u” saldrá “a”).
3. Escribir un algoritmo que permita desglosar una secuencia de caracteres en dos secuencias que contengan respectivamente los caracteres numéricos y los no numéricos contenidos en la secuencia original. Por final de proceso se deberá informar la cantidad de caracteres de cada una de las secuencias y las frecuencias de aparición de cada dígito ordenadas en forma creciente.
4. Dada una secuencia de texto (caracter, palabra, oración, texto), se desea generar otra secuencia que contenga la cantidad de palabra de longitud par de cada oración, y al final del proceso se emita un ranking decreciente por oración.
5. Dado un texto, efectuar un algoritmo que lo recorra y detecte la frecuencia de vocales por oración y luego liste dichos totales en orden decreciente.
6. Para un archivo texto confeccione un índice de palabras, indicando para cada una, en que renglón se encuentra y conque ocurrencia. Enunciar literalmente la filosofía seguida en la solución.

Ejemplo: ***EJERCICIOS DE EXAMENES FINALES DE ALGORITMOS Y ESTRUCTURAS DE DATOS***

1.Escribir un algoritmo que dadas dos secuencias oración de entrada, denominadas orac1 y orac2, genere una de salida, de manera tal que se intercalen las palabras de las secuencias de entrada, que cumplan con la siguiente condición:
De orac1 solo me interesan las palabras que comienzan y terminan con una vocal.
De orac2 solo me interesan las palabras de no más de 5 caracteres.
Al final del proceso informar cantidades de palabras de todas las secuencias.

3. Enuncie un trozo de algoritmo que produzca la inserción de un elemento en una lista doble.

4.Escriba un trozo de algoritmo para aplicar una búsqueda binaria sobre un arreglo unidimensional.

5.Clasifique los datos y sus estructuras, y describa en particular las listas lineales.