Práctico 1: Archivos

NOTA: Para trabajar con archivo en el entorno: se debe guardar cada archivo en /work/, y con un nombre que los distinga de los demás(iniciales del nombre y apellido), dado que el repositorio es compartido por todos, por ejemplo:

```
assign (archivo, '/work/lgomez_enteros.dat');
```

En este ejemplo el archivo conceptualmente se llama **enteros.dat** pero antes se agregó **/work/** porque en ese lugar están todos los archivos y luego **lgomez_** que es la inicial y el apellido del alumno para que no se "pise" con archivos de otros alumnos.

1. En cada caso realizar un procedimentos/función según corresponda:

a) Crear un archivo "Enteros.dat" y cargar tres valores pedidos al usuario.

b) Leer todos los datos que posee el archivo "Enteros.dat" e imprimirlos por pantalla. El archivo existe y posee valores.

c) Agregar un valor entero pedido por pantalla al final del archivo "Enteros.dat" . El archivo existe y posee valores.

d) Devolver el promedio de los valores de "Enteros.dat" . El archivo existe y posee valores.

e) Devolver el mayor de los valores de "Enteros.dat" . El archivo existe y posee valores.

2. Dadas las siguientes versiones del programa SumarArchivo:

```
program SumarArchivo;
type tarch = File of Integer;
procedure abrirArch(var arch : tarch; nombre : String; var error : Boolean);
begin
   error := false;
   assign(arch, nombre);
   {$I-} { desactiva la verificación de errores de entrada/salida (en tiempo de ejecución) }
   reset(arch); { al no estar activada la detección de errores, se puede intentar abrir
              archivos inexistentes y no se recibirán mensajes de error }
   {$I+} { activa la verificación de errores entrada/salida }
   if ioresult <> 0 then { ioresult devuelve un 0 si la última operación de entrada/salida
                     se realizó con éxito y <> 0 si hubo algún error }
      error := true;
end;
function sumarArch(var arch : tarch) : Integer;
var
   val, suma : Integer;
begin
   suma := 0;
   seek(arch, 0);
   while not eof(arch) do
   begin
      read(arch, val);
      suma := suma + val;
   end;
   sumarArch := suma;
end;
procedure escribirFinal(var arch : tarch; val : Integer);
begin
   seek(arch, fileSize(arch));
   write(arch, val);
end;
```

Versión 1

```
var
   arch : tarch;
   nombre : String;
   error : Boolean;
begin
   writeln('Ingresar nombre del archivo:');
   readln(nombre);
   abrirArch(arch, nombre, error);
   if not error then
   begin
      escribirFinal(arch, sumarArch(arch));
      close(arch);
   end
   else
      writeln('El archivo no existe');
end.
```

Versión 2

```
var
   orig, sum : tarch;
   nombre : String;
   error : Boolean;
begin
   writeln('Ingresar nombre del archivo a sumar:');
   readln(nombre);
   abrirArch(orig, nombre, error);
   if not error then
   begin
      abrirArch(sum, 'sumas.dat', error);
      if error then
         rewrite(sum);
      escribirFinal(sum, sumarArch(orig));
      close(sum);
      close(orig);
   end else
      writeln('El archivo no existe');
end.
```

a. Explique brevemente qué hace cada versión del programa.
b. Analice y compare el comportamiento de cada versión del programa al ejecutarla tres veces consecutivas. Suponga que el archivo utilizado, en todas las ejecuciones, es "origen.dat". El contenido inicial de origen.dat es: "4, 6, 10". El archivo "sumas.dat", al que hace referencia la versión 2, inicialmente no existe.

3. Teniendo en cuenta que los siguientes archivos son de chars, realice programas que resuelvan a los siguientes incisos:

a. Dado un nombre de archivo, guardar en el archivo todos los caracteres vocales ingresados por teclado.
b. Dado un nombre de archivo, generar un arreglo con los elementos ordenados ascendentemente. Suponer que el archivo contiene como máximo 100 elementos.
c. Dado un nombre de archivo de origen clonarlo con otro nombre; es decir, crear otro que tenga exactamente el mismo contenido pero con un nombre distinto.

4. Teniendo en cuenta las suposiciones de los incisos a) y b) respectivamente, codifique una función que dado un número lo busque en un archivo de números y retorne la posición en la que lo encontró. Si no lo encuentra debe retornar -1.

a. Suponga que los elementos del archivo se encuentran desordenados.
b. Suponga que los elementos del archivo se encuentran ordenados. Tenga en cuenta que, si el archivo está ordenado, no es necesario recorrerlo completamente. Explique por qué.

5. Suponga que tiene un archivo donde cada registro es un arreglo de 5 enteros. Cada entero corresponde a la nota en la materia 1, 2, 3, 4 y 5 de los alumnos de primer año de una carrera. Cada registro del archivo corresponde a las notas de un alumno y el archivo contiene las notas de todos los alumnos de primer año de la carrera. Codifique un procedimiento (¿por qué no función?) que devuelva un arreglo con la nota promedio en cada una de las materias.

7	6	5	7	8
3	9	8	5	4
6	8	9	5	3
5	6	5	7	8
3	9	2	5	4
6	8	2	4	3

Archivo

5	8	5	6	5

Promedio de cada materia

6. Invierta el orden de los elementos de un archivo de caracteres.

7. Teniendo en cuenta que los siguientes archivos son de números enteros, realice procedimientos que resuelvan a los siguientes incisos:

a. Recorra (una vez sola) un archivo de forma tal que si un número es mayor que el anterior los intercambie.
b. Utilizando el procedimiento anterior, ordene un archivo por el método de burbujeo sobre el mismo archivo.

8. Suponga que tiene un archivo donde cada registro es un arreglo de 6 reales. Considere las cinco primeras casillas de cada arreglo son notas de materias y la última casilla el promedio. Cada promedio está inicialmente en cero. Codifique un procedimiento para actualizar el archivo de manera de completar cada promedio

7,50	6	5	7,50	8	0
3	9,50	8	5	4	0
6	8,50	9	5	3	0
5	6	5	7	8,50	0
3	9,50	2	9,50	4	0
8,50	8	2	4	3	0

9. Codifique un procedimiento para realizar una búsqueda binaria en un archivo (ordenado) de enteros. Compare la cantidad de accesos promedio de esta estrategia respecto a la del inciso 4.b.

10. Se tienen dos archivos con los números (enteros) de legajo de alumnos, uno posee los inscriptos en Intro II y el otro en Ciencias de la Computación I. Ambos archivos están ordenados por el nº de legajo. Confeccione un programa que genere un nuevo archivo con los nº de legajo de los alumnos que están inscriptos en ambas materias.

Práctico 2: Nuevos Tipos de Datos

1. Defina un registro para representar una fecha(DIA/MES/AÑO) y codifique procedimientos o funciones para manejo de fechas:

   a. para cargar una fecha desde teclado, validando que sea correcta.
   b. para sumar N días a una fecha.
   c. para calcular la cantidad de días entre dos fechas.
   d. para imprimir una fecha.
   e. para verificar si una fecha1 es menor a una fecha2

2. Se tiene un arreglo donde cada elemento es un registro. Cada registro contiene los datos de un alumno: nro. de libreta, nombre del alumno, código de Facultad (1..9), Código de carrera (10..99) y año que cursa. El arreglo se halla ordenado alfabéticamente según el apellido del alumno. Codificar un procedimiento que, dados el arreglo y un registro con los datos de un nuevo alumno, inserte dicho alumno en el arreglo, manteniendo el orden.

3. Cuando una matriz tiene muchas de sus celdas vacías (más del 75 %) se la denomina "rala" y conviene representarla con un arreglo cuyos elementos sean registros con los campos fila, columna y contenido. Si tiene una matriz de números reales con las precipitaciones diarias del corriente año, cuyas filas representan meses y las columnas días (esto es la matriz es de 12 x 31) y se sabe que no llueve más de 50 veces al año, defina ambas formas de representación para las lluvias de un año (matriz y arreglo de registros) y codifique los dos procedimientos para copiar los datos de una representación a la otra y viceversa.

4. Crear un archivo desordenado de SOCIOS de un club con registros con la siguiente información: DNI, Nro. Socio, Nombre y Apellido, año ingreso. La carga termina cuando se ingresa un DNI=0 . b) Hacer un programa que dado un DNI imprima todos los datos del socio si existe en el Archivo SOCIOS.

5. Realice un sistema que permita mantener la información de los empleados de una empresa en el archivo "EMPLE.DAT". Cada registro de empleado contiene los siguientes datos y lo identifica su número de legajo:

   nro. de legajo. (de 1 a 1000), nombre y apellido, edad, fecha de ingreso al trabajo, categoría (administrativo, operario, gerente, maestranza) y sueldo del mes.

Se necesita un programa que permita actualizar los datos de estos empleados, entendiendo por actualización la posibilidad de agregar un nuevo empleado, borrar uno ya existente o modificar cualquiera de sus datos. Además se necesita obtener un listado de todos los empleados cargados ordenado por número de legajo.

Defina y declare los tipos y las variables y codifique el programa principal con los procedimientos/funciones necesarios.

6. Para la solución planteada en el ejercicio anterior, realice las siguientes verificaciones y explique qué hace el sistema y qué debería hacer en los siguientes casos.

   a. Al agregar un nuevo empleado, si su número de legajo ya existe.
   b. Al borrar un empleado, si el número de legajo no existe.
   c. Al borrar un empleado, si el archivo no tiene registros.
   d. Al listar todos los empleados, si el archivo no tiene registros.

7. Se tiene un archivo "Alumnos.Dat" con los datos de los alumnos de la universidad. Cada registro está definido según el ejercicio 2. El archivo se encuentra ordenado por facultad, por carrera, por año que cursa y alfabéticamente por alumno.

Codifique el procedimiento que, a partir del archivo, emite un listado de todos los alumnos de la universidad, agrupándolos por facultad, por carrera, y por año que cursa. Al comenzar cada "categoría" imprimir un título, y al finalizar emitir el total de alumnos, o sea, de cada año, cada carrera, cada facultad y la universidad.

Práctico 3: Punteros y Listas

1) Codifique un módulo que cargue por teclado N caracteres en una lista (uno por nodo). El fin de la carga se detecta cuando el usuario ingresa el carácter "*".

2) Realice todos los procedimientos y funciones necesarios para realizar las siguientes tareas:
   a) Dado como parámetro un arreglo de enteros de longitud N, crear una lista vinculada con todos los elementos del arreglo
   b) Retornar la suma de los elementos de la lista vinculada.
   c) Retornar la cantidad de elementos de la lista
   d) Retornar el promedio de los elementos de la lista vinculada
   e) Retornar el máximo elemento de la lista vinculada.

3) a)Codifique un módulo que inserte un número entero en una lista ordenada ascendentemente conservando el orden. b) Para la solución del ejercicio a), indique para cada uno de los casos especificados, el resultado final **esperado** de la ejecución (estado de la lista) y el resultado **real** (obtenido del seguimiento del código con el ejemplo de datos dado)

lista: 8 10 (nro. a insertar: 9)

lista: 8 10 (nro. a insertar: 16)

lista: 8 10 (nro. a insertar: 3)

lista: 8 10 (nro. a insertar: 8)

lista: nil (nro. a insertar: 8)

4) Dada una lista ordenada de enteros y un elemento, codifique un módulo que lo elimine si está en la lista. b) Dada una lista desordenada de enteros y un elemento, codifique un módulo que lo elimine si está en la lista.

5) Dada una lista de chars, realice un procedimiento que invierta el orden de sus elementos únicamente sus vínculos.

6) Ordene una lista de enteros según los métodos de: a) selección b) inserción.

7) En base a la siguientes definiciones

```
Type PuntJugador = ^ TipoJugador;
     TipoJugador = Record
                       Alias: String
                       Puntaje:0..100000;
                       Sig:PuntJugador ;
                   End;
```

Defina una lista Jugadores de registros TipoJugador y realice módulos independientes para:

a) Insertar un jugador en la lista Jugadores de manera ordenada según el alias. La lista puede estar o no vacía

b) Dado un Alias buscarlo en la lista Jugadores y eliminar el nodo si es que existe

c) Devolver el jugador con mayor puntaje de la lista

d) Crear otra lista apuntada por Jugador_Puntaje con los mismos datos que Jugadores pero ordenada ascendentemente por puntaje.

e) Pasar los jugadores de la lista Jugadores a un Archivo de jugadores conservando el orden según el alias

8) Se tiene una lista PACIENTES con los siguientes datos: nro_pac, apellido y nombre, edad, domicilio(calle, número, ciudad), teléfono, Obra Social(booleano que tiene un True si el paciente posee OS y False si no tiene). La lista está ordenada por nro_pac

Escribir un programa que permita realizar las siguientes operaciones:

a) Dado un nro_pac, y una nro. de telefóno, modificar el mismo, si es que existe el paciente en la lista
b) obtener el porcentaje de pacientes en las siguientes categorías:

   niños: hasta 13 años

   jóvenes: mayores a 13 y menores a 30

   adultos: mayores a 30

c) Calcular el porcentaje de pacientes que posee Obra social

d) Ingresar un paciente conservando el orden

Práctico 4: Tipos de Listas y Lista de listas

1. Defina un tipo de datos para una Fila, implementándola como una lista vinculada. Los elementos son números enteros. Luego codifique las funciones / procedimientos implementados en el "ESTRUCTU" para la fila:

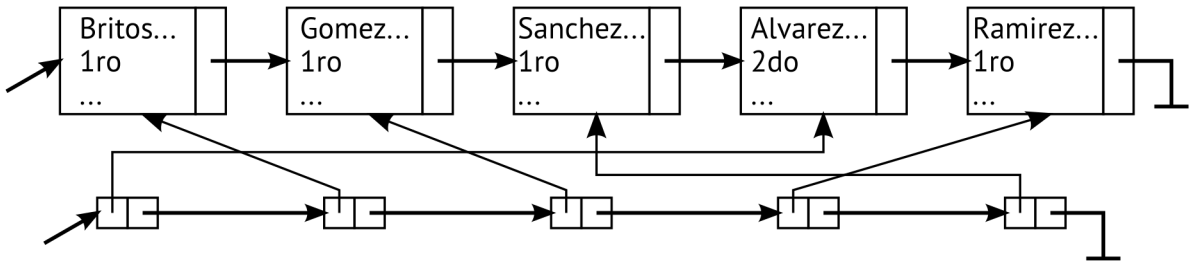
Inic_fila(no es necesario implementarlo)	Fila_vacia	Write_fila
Agregar	Primero	Read_fila (no es necesario implementarlo)
Extraer		

2. Codifique un módulo que dada una lista ordenada doblemente vinculada de reales y un número que representa una posición, elimine el elemento correspondiente a esa posición

3. Codifique un módulo que elimine, si existe, un número de una lista circular de enteros.

4. Se tiene una lista vinculada de alumnos en el que cada nodo posee los siguientes datos:
   - Apellido y Nombre
   - Curso (1ro, 2do, 3ro, 4to , 5to ó 6to)
   - DNI
   - Domicilio
   - Teléfono

   La lista se encuentra ordenada por curso y dentro de cada curso por apellido. Para obtener un listado alfabético de todos los alumnos implementaremos una lista "invertida". Los nodos de una lista invertida apuntan a los nodos de la lista original pero se vinculan entre sí en otro orden, como en el siguiente gráfico:



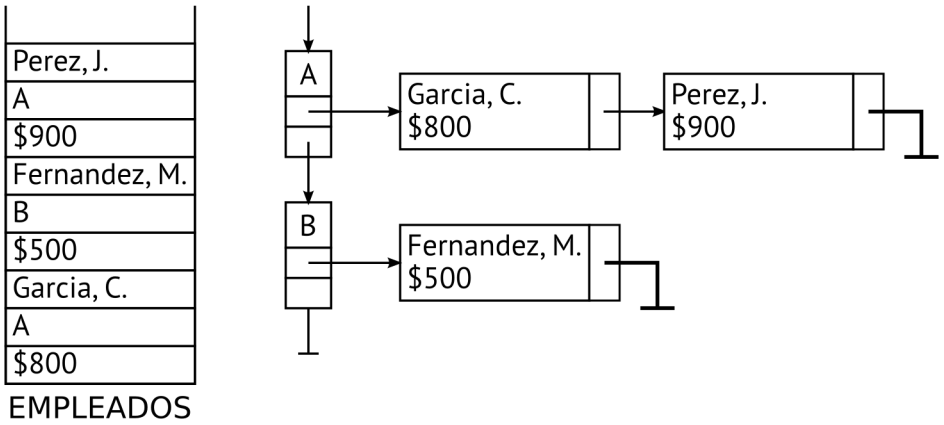
   Si recorremos la lista invertida, el orden de los nodos es alfabético (Alvarez, Britos, Gomez, Ramirez, Sanchez) en lugar de por curso y esto nos permite emitir el listado con una pasada.

   Codifique el procedimiento que, dada la lista vinculada original, construya y devuelva la lista invertida correspondiente.

   b) Codifique un procedimiento que dé de alta un nuevo alumno en la estructura anterior, actualizando las dos listas para que ambas mantengan el orden que corresponda. Para simplificar suponga que el procedimiento recibe como parámetro el nodo con los datos del nuevo alumno.

5. Se tiene un archivo EMPLEADOS cuyos elementos son registros con los datos de un operario de una fábrica. Los datos son los siguientes:
   - Apellido y nombre
   - Categoría
   - Sueldo

   a) Codifique un programa que lea del archivo EMPLEADOS para generar una lista ordenada de categorías, donde, para cada categoría, existe otra lista con los nombres y sueldos de los empleados de la misma ordenada por apellido.



b) Codifique un procedimiento que, dada la lista anterior, pida un número de categoría e imprima todos los empleados que la poseen.
c)
d) Codifique un procedimiento que, dada una categoría, si existe de la elimine de la lista anterior, junto con todos sus empleados.

6) Retome el ejercicio de la lista de Jugadores(el Nro 7 del práctico 3). Modifique el registro para que contenga otro puntero llamado Sig_puntaje (del mismo tipo) que permitirá recorrer la lista ordenada de manera ascendente por puntaje:

```
Type PuntJugador = ^ TipoJugador;

TipoJugador = Record

              Alias: String

              Puntaje:0..100000;

              Sig:PuntJugador ;

              Sig_Puntaje:PuntJugador;

              End;
```

Realizar un módulo que reciba como parámetro la Lista Jugadores y que deberá actualizar los punteros Sig_puntaje (inicialmente en nil) para que se pueda recorrer la lista ascendentemente por puntaje. El nuevo orden se recorrerá a partir del puntero Jugadores_Puntaje, inicialmente en nil y recibido también como parámetro.

Práctico 5: Recursión

1. Una función es recursiva si su código contiene invocaciones a sí misma. Teniendo en cuenta que

   $factorial(N)=N*factorial(N-1)$ para $N>0$ y

   $factorial(0)=1$,

   escriba el código de la función que dado un número retorna su factorial.

2. Calcular recursivamente el Máximo Común Divisor de dos enteros no negativos basándose en las siguientes fórmulas matemáticas.
   a. $MCD(X,X)=X \quad X<Y \Rightarrow MCD(X,Y)=MCD(Y,X) \quad X>Y \Rightarrow MCD(X,Y)=MCD(X-Y,Y)$
   b. $MCD(X,0)=X \quad X<Y \Rightarrow MCD(X,Y)=MCD(Y,X) \quad X \geq Y \Rightarrow MCD(X,Y)=MCD(Y,MOD(X,Y))$

3. Codifique una función que calcule el factorial del MCD de dos números dados, según el algoritmo 2.b. Hacer la secuencia de llamados y retornos de procedimientos con valores 28 y 20.

4. Para verificar que un arreglo es capicúa de manera recursiva se sigue la siguiente estrategia. Un arreglo es capicúa si el primer elemento es igual al último y si el "subarreglo" que queda entre estos dos elementos también es capicúa. Ejemplo:

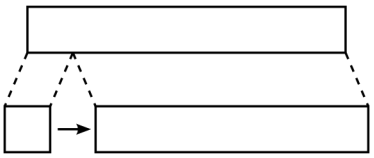
   Para poder plantearlo recursivamente, la clave está en poder aplicar la misma estrategia para verificar si el subarreglo interior es capicúa. Piense cómo solucionar este último problema, plantee cuándo "cortar" la recursión y codifique la solución.

5. Verificar recursivamente que una matriz cuadrada de 33 caracteres de lado es palíndroma (capicúa en todas sus filas y columnas). Compararlo con la Versión del ejercicio 5 del Práctico 8 de Introd. a la Prog. I.

6. Plantee y codifique una estrategia recursiva para grabar los datos de un archivo en otro de tal forma que los mismos queden en orden inverso.
   a) Piense los casos de verificación que podría incorporar para probar el ejercicio anterior incluyendo los resultados esperados. b) Incorporó los siguientes casos en el ejericicio anterior? i) archivo vacío. ii) archivo con sólo dos datos iguales.

   c) Pase el programa del ejercicio 6 a la computadora, y compare los resultados esperados con los obtenidos en la ejecución y los resultantes de una "ejecución manual".

7. Una forma interesante de conceptualizar una lista para pensar soluciones recursivas es pensarla como "cabeza-cola" donde la cabeza es el primer elemento de la lista y la cola el resto. Ej:

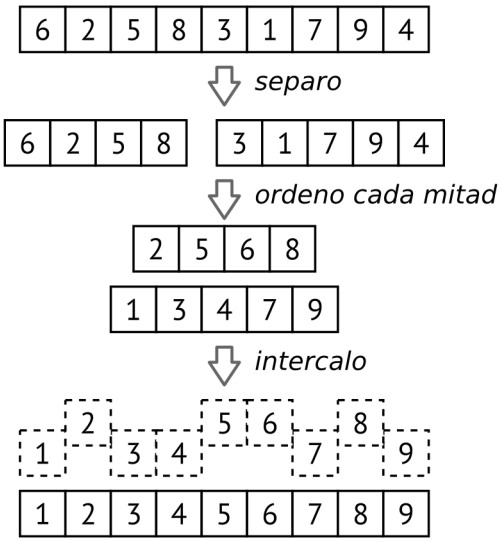


La ventaja es que la cola es una lista y por lo tanto podemos pensar recursivamente, es decir aplicar la misma estrategia para la cola que se aplica a toda la lista. Ejemplo: sumar los elementos de una lista vinculada. Solución: sumo el valor de la cabeza de la lista con el resultado de la función aplicada al resto (la cola) de la lista.

Codifique la solución.

8. Piense recursivamente y codifique una solución para devolver el menor de los elementos de una lista vinculada.

9. Merge sort: este método de ordenamiento trabaja de la siguiente manera: dada una lista se divide en dos mitades, se ordena cada mitad (para lo cual se puede aplicar recursión) y después se intercalan las mitades. (el procedimiento para intercalar es muy simple: se compara las cabezas de las listas y se toma la menor). Ej:

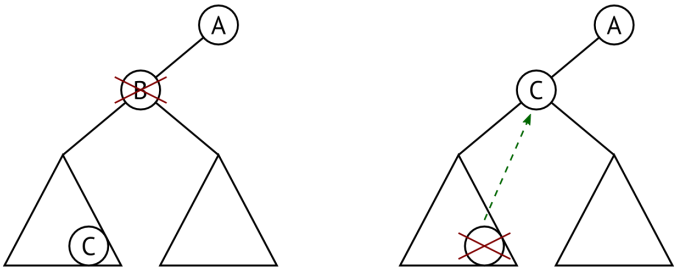


Implemente la solución recursivamente.

Práctico 6: Árboles binarios

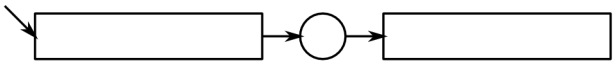
1) Se tiene un archivo desordenado con números enteros. Se pretende que realice un procedimiento que lea todos los números del archivo y genere un árbol ordenado en forma ascendente.
2) Hacer un procedimiento que imprima los números del árbol del ejercicio 1: a)en orden ascendente(in-order). b) en orden descendente(in-order) c) imprimir de tal forma que no se imprima un padre si no se han impreso todos sus hijos (post-order). d) imprimir los números del árbol anterior de tal forma que no se imprima un nodo si no se ha impreso su padre (pre-order).
3) Se tiene un árbol binario cuyos nodos contienen el Nro. Alumno y DNI de los alumnos de Intro II. El árbol está ordenado por DNI. Se pide: a) Hacer un procedimiento/función que dado un DNI retorne el Nro. de Alumno asociado b)Hacer un procedimiento/función que dado un Nro. de Alumno retorne el DNI asociado
4) Realice un procedimiento que dado un número lo busque en el árbol ordenado y lo borre si existe. Tenga en cuenta que dicho número se puede encontrar como una hoja del árbol (sin hijos que cuelguen de él) o como nodo interno con otros nodos colgando de él.

   Si el nodo a eliminar (B) es interno y tiene las dos subramas no vacías se utiliza la estrategia de reemplazo por el Nodo más Derecho (C) del Subárbol Izquierdo.

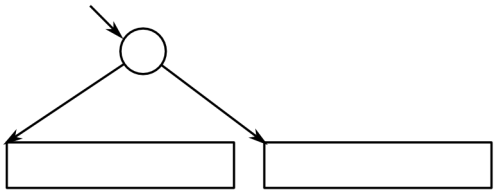


4.a) Para el ejercicio anterior indique todos los casos especiales que deberían testearse y el resultado esperado de la prueba. Plantee datos para cada uno de ellos y ejecute las pruebas en el entorno. Registre los resultados obtenidos. Si se detectan errores en el código, corríjalos y vuelva a realizar las mismas pruebas con los mismos datos.

5) Hacer un procedimiento/función que retorne el menor de los elementos de un árbol binario. a) árbol desordenado b) árbol ordenado
6) Hacer un procedimiento/función que retorne la longitud de la mayor rama de un árbol.
7) Dado un árbol transformarlo en lista recorriendolo en preorden.
8) Dado un árbol binario de números enteros, sumarle a cada nodo padre los valores de sus hijos de manera que la raíz pase a tener la suma de todos los elementos del árbol.
9) Dado un árbol binario ordenado de enteros a) hacer un procedimiento/función que retorne la cantidad de nodos del árbol b) hacer un procedimiento/función que retorne la cantidad de nodos del árbol que hay considerando sólo aquellos nodos cuyo valor de entero esté ente entre minValor y maxValor, dados como parámetros. No recorrer el árbol innecesariamente.
10) Dada una lista ordenada, armar un árbol binario balanceado. Para ello se toma el elemento central de la lista y se lo coloca como raíz de un árbol en el que cada sublista (transformada recursivamente en árbol) cuelga como hijo.



Lista original



Lista transformada en árbol. En realidad los hijos no son listas. Cada una de ellas se debe transformar en árbol.

Práctico 7: Ejercicios tipo Examen Parcial y Final

1) **PARCIAL /2019**

   Dado un árbol apuntado por ***arbol_nros*** cuyos nodos contienen como único dato ***nro*** y poseen, además de los dos punteros necesarios, un puntero adicional del mismo tipo llamado ***sig_suma*** inicialmente en nil. El árbol está ordenado por ***nro***. Se tiene además un puntero ***arbol_suma*** inicialmente en nil. Se pide implemente un módulo que dados **árbol-nros , arbol-suma** y un número que representa un **nivel** del árbol arme una lista simple, iniciando en ***arbol_suma*** que vincule a todos los nodos del árbol del **nivel** recibido como parámetro de tal forma que la lista quede ordenada ascendentemente (mediante el puntero ***sig_suma***) en base a la suma de los ***nro*** de todos los descendientes de cada nodo.

   Debe hacer el DE, escribir el programa principal con las constantes, tipos y variables necesarias. Asuma ya está implementado cargar_arbol (que lo crea con un conjunto de números) y sólo invóquelo. Modularice todo lo necesario.

2) **Recuperatorio /2019**

Se tiene un archivo con Empleados de la Empresa (nombre empleado [string], categoría [1..5]), un árbol con Empleados y un arreglo de Categorías. Se desea actualizar el árbol manteniéndolo ordenado por nombre de empleado donde cada nodo contiene nombre y categoría. Además de los punteros normales del árbol los nodos están vinculados formando una lista simple entre los que pertenecen a la misma categoría. Esta lista los vincula manteniéndolos también ordenados por nombre.

Además se tiene un arreglo de 5 categorías donde cada posición representa un número de categoría y el contenido es un puntero al nodo del árbol inicio de cada lista de esa categoría.

Se pide realizar un programa que lea los datos del archivo y que actualice el árbol y el arreglo de la siguiente forma:

- ➔ Si el árbol no contiene a ese empleado, insertarlo y actualizar la lista de acuerdo al número de categoría y el arreglo si es necesario.
- ➔ Si el empleado ya existe y la categoría del archivo es MAYOR al que posee en el árbol actualizarlo (cambiar la categoría dejando las listas actualizadas correctamente).

Todas las categorías que vienen en el archivo son posiciones válidas en el arreglo (valores de 1 a 5).
Realice el diagrama de estructura y el código pascal incluyendo definiciones de estructuras y programa principal. No debe utilizar estructuras auxiliares.

3) **Introducción a la Programación II PREFINAL 25/11/2019**

4) Dado un árbol apuntado por ***arbol_notas*** cuyos nodos contienen: ***nro_libreta_alumno***, ***nombre_materia*** y ***nota*** (0 a 10), el árbol está ordenado por ***nro_libreta_alumno*** *(un alumno aparece tantas veces como haya rendido materias, incluso puede rendir más de una vez una materia)*. Se pide que solicitando número mínimo de libreta y número máximo, recorra los nodos del árbol y con los que están en ese rango (incluyéndolos) genere una lista a partir del puntero ***lista_materias*** que inicialmente está en ***nil***. La lista contendrá en cada nodo, además del puntero al siguiente, ***nombre_materia****(aparece una sola vez)*, ***suma_total_notas****(va acumuladno las notas de una misma materia de los nodos del árbol)* y ***cantidad_notas****(la cantidad de veces que aparece esa materia en el árbol)***.** Con cada lectura de un nodo, debe mantener la lista siempre ordenada ascendentemente por el promedio de las notas (***suma_total_notas*** / ***cantidad_notas***). *Notas:* Realice el diagrama de estructura, codifique el programa principal con las constantes, tipos y variables necesarias, asumiendo que el árbol

está cargado. Modularice todo lo que considere necesario. Recorra una sola vez el árbol. Tenga en cuenta que los números repetidos de libreta de alumno están junto a los menores. No hacer recorridos innecesarios.

4)Parcial 2016

Se tiene un archivo de ***codigo*** (entero) en el que los registros están ordenados de menor a mayor y un árbol también ordenado ascendente con ***codigo*** (entero) que además tiene el atributo ***nivel***(0 para la raíz, 1 para los hijos, etc [indica la profundidad en el árbol]). Solicite al usuario el ingreso de ***codigo_buscado*** (un número entero) y realice lo siguiente:

1) Busque el ***codigo_buscado*** en el archivo y en el árbol, si está en el archivo y NO en el árbol agréguelo en el árbol en el lugar correspondiente (y con el valor de ***nivel*** que le corresponde según su profundidad) de lo contrario no haga nada.
2) Arme una lista simple de nodos con (***codigo*** y ***nivel***) ordenada de mayor a menor por ***nivel*** con todos los nodos del árbol que cumplen las siguientes condiciones:
   a) El ***codigo*** está entre ***codigo_buscado*** y ***codigo_buscado + 100***.
   b) El nodo posee cantidad impar de descendientes (hijos, nietos, etc).

   La lista debe estar en todo momento ordenada por ***nivel***. No importa el orden entre los nodos del mismo ***nivel***.

Notas: 1) No use bisección en el archivo. 2) Para la promoción: No recorrer de más archivo ni árbol. 3) El punto 2 es fundamental para aprobar. 4) Realice el DE y programa en Pascal.

5) Recuperatorio 2016

Se tiene una lista apuntada por ***CLIENTES*** cuyos nodos tiene la siguiente información: ***código_cliente***(entero), ***facturas***(puntero a ArbolFact), ***sig_cli*** (puntero a nodo), ***montoAdeudado***(integer, inicialmente en cero), ***sig_deudor***(puntero a nodo).
La lista ***CLIENTES*** está ordenada ascendente por ***código_cliente*** a través del puntero ***sig_cli***. Cada árbol apuntado por facturas corresponde a un árbol binario de facturas impagas de ese cliente, ordenado por nro factura y cuyo nodo contiene además, el ***monto_factura***.
Los punteros ***sig_deudor*** de cada nodo de la lista están ***nil***. A su vez existe otro puntero del mismo tipo ***INICIO_Deudas*** que también está en ***nil***.
Asimismo Se tiene un archivo ARC_CLIENTE cuyos registros contienen un solo campo, ***codigo_cliente***(entero).
Se pide: Se creará un un nuevo orden en la lista ***CLIENTES*** a través de los punteros ***sig_deudor*** para todos los nodos que cumplan la siguiente condición:
Leer el archivo ARC_CLIENTE y para cada cliente que está en el archivo se busca en la lista ***CLIENTES***. Si está, se debe actualizar el campo ***montoAdeudado*** con la suma de todas las facturas del árbol y se actualiza el puntero ***sig_deudor*** de este nodo para mantener a la lista ordenada ascendentemente por ***montoAdeudado***.
INICIO_Deudas es el puntero que permite acceder al primer nodo de la lista ***CLIENTE*** ordenada por ***montoAdeudado***.

6) PREFINAL 2016
Se tienen cargados en el archivo Equipos.dat los datos referidos a equipos participantes de un torneo deportivo. El archivo no se encuentra ordenado bajo ningún criterio. Para cada equipo se guarda: nombre (de tipo string), puntaje (integer).

Se pide que implemente una función / procedimiento en Pascal que genere un árbol ARB_EQUIPOS en memoria principal con todos los datos del archivo teniendo en cuenta lo siguiente:

- El árbol debe estar ordenado alfabéticamente por nombre del equipo.
- Si ingresa un equipo con un nombre que ya fue incorporado al árbol, no debe incorporarse.

· Los nodos del árbol deben tener un vínculo adicional de tal manera que los equipos puedan ser listados ascendentemente por puntaje mediante ese vínculo. Ese nuevo orden es accedido por un puntero inicial ORDEN_PUNTAJE . **NOTA:** El árbol debe estar vinculado y ordenado correctamente por puntaje en todo momento (ante cada nuevo equipo que se agrega). No puede usar estructuras auxiliares. Debe definir: diagrama de estructura, programa principal, procedimientos y funciones, constantes, tipos y variables.

7) Se tiene un árbol de números positivos ordenado por los mismos. Se pide que lo recorra in-order llevando en todo momento la suma acumulada de los nodos por los que se pasó. Debe imprimir todos los nodos que cumplan con la condición que la suma acumulada es igual a la suma del subárbol derecho (mayores) del mismo. La función que suma el subárbol derecho no debe seguir recorriendo nodos si detecta que la suma se superó. Realice todas las definiciones, DE y el programa principal. No puede utilizar estructuras auxiliares.

8) Se tiene un arbol CIUDADES donde cada nodo tiene el nombre de la ciudad, la provincia a la que pertenece y la cantidad de habitantes. El árbol está ordenado por nombre de ciudades.
Se pide que arme una lista POBLACION a partir de los nodos del árbol que se encuentran dentro de un rango ciudad1:ciudad2 que debe pedir por teclado. No recorrer el árbol innecesariamente.
La lista población estará formada por un nodo que contiene la provincia y un entero con la cantidad de población acumulada correspondiente a todas las ciudades de esa provincia que caen en el rango.
La lista población debe estar siempre ordenada ascendentemente por la cantidad de población de provincia. No puede usar estructuras auxiliares.

9) Se tiene un árbol binario con información de Exámenes de Alumnos, cada nodo posee número de alumno, materia, notaFinal. El árbol está ordenado por número de Alumno, cada Alumno puede tener varios nodos, uno por materia (las materias para un alumno no se repiten).
Se desea que cree una lista doblemente vinculada con todos los alumnos cuyos números estàn FUERA de un rango NumeroAlumnoMinimo:NumeroAlumnoMaximo, es decir, son menores estrictos a NumeroAlumnoMinimo o mayores estrictos a NumeroAlumnoMaximo.
Cada nodo de la lista poseerá el número de Alumno, y la nota màs alta de final obtenida para todas las materias que rindió. La lista debe estar en todo momento ordenada ascendentemente por la nota más alta.
Debe recorrer una única vez el árbol y no pasar por nodos innecesariamente.

10) Declare la siguiente estructura de datos:

Un árbol ordenado por fechas donde cada nodo tiene además de la fecha un puntero a una lista de los nodos-factura correspondientes a esa fecha (ordenada por numero de factura), y una lista de clientes (ordenadas por nro cliente) en el cual cada componente contiene el nro de cliente y un puntero a la lista de nodos-factura correspondientes a ese cliente (ordenada por fecha). Cada nodo-factura tiene entonces dos campos puntero y los siguientes campos de datos: nro de factura, fecha, nro de cliente, importe total, y un campo booleano que indique si la factura está paga.

Suponiendo que la estructura está cargada, diseñe e implemente los siguiente módulos

a) El procedimiento que permite el ingreso de los datos de una factura y actualice la estructura de datos;
b) La función que dado el nro de cliente devuelva el importe total de las facturas impagas;
c) El procedimiento que dada una fecha emita el listado con el detalle de las facturas correspondientes;

d) El programa principal que invoque los procedimientos / funciones anteriores.

VIEJOS

11) Declare la siguiente estructura de datos de un diccionario de sinónimos:

Un arreglo con un índice de las letras del alfabeto, en el cual cada componente es puntero a la lista ordenada de palabras que comienzan con esa letra, donde además las palabras que son sinónimos están vinculadas por medio de una lista circular.

Codifique:

e) El procedimiento que inicialice la estructura de datos;
f) El procedimiento que permite el ingreso de dos palabras que son sinónimos (puede ser que ninguna, una o las dos palabras ya existan en el diccionario);
g) El procedimiento que dada una letra del alfabeto emita el listado de todas las palabras que comiencen con esa letra con sus respectivos sinónimos;
h) Un programa principal que invoque los procedimientos anteriores.

Ejercicio para entregar programado

(Corresponde al ejercicio 8 del práctico 3)

En base a la siguientes definiciones

```
Type PuntAlumno = ^ TipoAlumno;
      TipoAlumno = Record
                              Nro.Alu: Integer
                              Nota:0..10;
                              Sig:PuntAlumno ;
                   End;
```

Defina una lista AlumnosProgII de registros TipoAlumno y realice módulos independientes para:

a) Insertar un alumno en la lista AlumnosProgII de manera ordenada según el Nro.Alu. La lista puede estar o no vacía

b) Dado un Nro.Alu, buscarlo en la lista AlumnosProgII y eliminar el nodo si es que existe

c) Devolver el alumno con mayor nota de la lista

d) Sacar el promedio de notas de la lista

e) Imprimir los datos del tercer alumno de la lista, si es que hay tres ó más nodos en la lista, sino, imprimir mensaje de error.

f) Crear otra lista apuntada por Alumnos_Nota con los mismos datos que AlumnosProgII pero ordenada ascendentemente por nota.

g) Realice el programa principal que implemente un menú desplegando todas las opciones de los módulos anteriores (del [a] al [f]) y que le permita al usuario elegir repetidamente cualquiera de ellas hasta que decida finalizar.