Parciales Taller

{Una cadena de gimnasios que tiene 5 sucursales necesita procesar las asistencias de los clientes.

Implementar un programa en Pascal con:

a) Un módulo que lea la información de las asistencias realizadas en cada sucursal y que genere

un árbol ordenado por dni de cliente, donde cada nodo contenga dni de cliente, còdigo y la

cantidad total de minutos asistidos en todas las sucursales. De cada asistencia se lee: código

de sucursal (1..5), dni del cliente, código del cliente, fecha y cantidad de minutos que asistió. La

lectura finaliza con el dni de cliente -1, el cual no se procesa.

b) Un módulo que reciba el árbol generado en a) y un dni de cliente y devuelva una lista con los

dni de los clientes cuyo dni es mayor al dni ingresado y el total de minutos es impar}

program parcialG;

const sucursales = 5;

type

rango = 1..sucursales;

asistencias = record

dni:integer;

codigo:integer;

totalMinutos:integer;

end;

arbol = ^nodo;

nodo = record

HI:arbol;

HD:arbol;

dato:asistencias;

end;

vector = array [rango] of arbol;

lista = ^nodoL;

nodoL = record

sig:lista;

dato:integer;

end;

procedure generarV(var v:vector);

procedure leerR(var r:asistencias);

begin

r.dni:=random(9) - 1;

if r.dni <> -1 then begin

r.codigo:=random(15);

r.totalMinutos:=random(60);

writeln('dni: ', r.dni, ' codigo: ', r.codigo, ' total minutos: ', r.totalMinutos);

end;

end;

procedure initV(var v:vector);

var i:rango;

begin

for i:=1 to sucursales do

v[i]:=nil;

end;

procedure agregarHoja(var a:arbol; r:asistencias; var yahabia:boolean);

begin

if (a = nil) then begin

new(a);

a^.HI:=nil;

a^.HD:=nil;

a^.dato:=r;

end

else

if (r.dni < a^.dato.dni) then agregarHoja(a^.HI, r, yahabia)

else

if (r.dni > a^.dato.dni) then agregarHoja(a^.HD, r, yahabia)

else begin

a^.dato.totalMinutos:= a^.dato.totalMinutos + r.totalMinutos;

yahabia:=true;

writeln('qsys ', a^.dato.totalMinutos);

end;

end;

procedure buscar(a:arbol; r:asistencias; var minutosFaltantes:integer);

begin

if a <> nil then

if a^.dato.dni > r.dni then buscar(a^.HI, r, minutosFaltantes)

else

if a^.dato.dni < r.dni then buscar(a^.HD, r, minutosFaltantes)

else

begin

minutosFaltantes:= a^.dato.totalMinutos;

writeln('al nodo que contiene ', a^.dato.totalMinutos, ' le sumamos: ', r.totalMinutos);

a^.dato.totalMinutos:= a^.dato.totalMinutos + r.totalMinutos;

end;

end;

procedure actualizarMinutos (v:vector; r:asistencias; sucur:rango; yahabia:boolean);

var i:rango; minutosFaltantes:integer;

begin

minutosFaltantes:=0;

for i:=1 to sucursales do

if i <> sucur then

buscar(v[i], r, minutosFaltantes);

if (minutosFaltantes > 0) and (yahabia = false) then begin

writeln('debug ', minutosFaltantes);

r.totalMinutos:=minutosFaltantes;

buscar(v[sucur], r, minutosFaltantes);

end;

end;

//esta solucion me parece mala pero no se me ocurre una mejor

var r:asistencias; {fecha:integer;} sucur:rango; yahabia:boolean;

begin

initV(v);

Randomize;

leerR(r);

while (r.dni <> -1) do begin

yahabia:=false;

//fecha:=random(50) + 1973; no se porque dice que se lee si despues no se usa

sucur:=random(5) +1;

writeln('sucursal', sucur);

agregarHoja(v[sucur], r, yahabia);

writeln(yahabia);

actualizarMinutos(v, r, sucur, yahabia);

leerR(r);

end;

end;

procedure escribirV(v:vector);

procedure escribirA(a:arbol);

begin

if a <> nil then begin

writeln('dni: ', a^.dato.dni, ' codigo: ', a^.dato.codigo, ' total minutos: ', a^.dato.totalMinutos);

escribirA(a^.HI);

escribirA(a^.HD);

end;

end;

var i:rango;

begin

writeln('------------------');

for i:=1 to sucursales do begin

writeln('sucusal nro: ', i);

escribirA(v[i]);

end;

end;

procedure generoL(v:vector; var l:lista);

procedure agregarAdelante(var l:lista; r:integer);

var aux:lista;

begin

new(aux);

aux^.dato:=r;

aux^.sig:=l;

l:=aux;

end;

procedure requisitos(a:arbol; dni:integer; var l:lista);

begin

if (a <> nil) then

if a^.dato.dni <= dni then requisitos(a^.HD, dni, l)

else begin

if (a^.dato.dni mod 2 = 1) then agregarAdelante(l, a^.dato.dni);

requisitos(a^.HD, dni, l);

end;

end;

var i:rango; dni:integer;

begin

l:=nil;

readln(dni);

for i:=1 to 5 do

requisitos(v[i], dni, l)

end;

procedure escribirl(l:lista);

begin

writeln('--------------');

while (l <> nil) do begin

writeln('dni: ', l^.dato);

l:=l^.sig;

end;

end;

var v:vector; l:lista;

begin

generarV(v);

escribirV(v);

generoL(v, l);

escribirl(l);

end.

{Una aerolínea dispone de un árbol binario de búsqueda con la información de sus empleados. De cada

empleado se conoce: Número de legajo, Dni, Categoría (1..20) y año de ingreso a la empresa. El árbol se

encuentra ordenado por número de dni. Se solicita:

a. Implementar un módulo que reciba el árbol de empleados, número de DNI “A”, número de DNI “B” y un

número de categoría, y retorne un vector ordenado por número de DNI. El vector debe contener el

número de DNI y número de legajo de aquellos empleados cuyo número de DNI se encuentra

comprendido entre los números de DNI recibidos (“A” y “B”, siendo “A” menor que “B”) y la categoría se

corresponda con la recibida por parámetro. Por norma de la empresa, cada categoría puede contar con

a lo sumo 250 empleados.

b. Implementar un módulo recursivo que reciba la información generada en “a” y retorne el promedio de

los números de legajo}

program parcialAerolinea;

const categorias = 20;

type

rango = 1..categorias;

empleado = record

legajo:integer;

dni:integer;

categoria:rango;

anioIngreso:integer;

end;

arbol = ^nodo;

nodo = record

HI:arbol;

HD:arbol;

dato:empleado;

end;

incisoA = record

dni:integer;

legajo:integer;

end;

vector = array [1..250] of incisoA;

procedure generarA(var a:arbol);

procedure leerR(var r:empleado);

begin

r.dni:=random(30);

if r.dni <> 0 then begin

r.legajo:=random(15);

r.categoria:=random(20) +1;

r.anioIngreso:=random(50) +1950;

end;

end;

procedure agregarHoja(var a:arbol; r:empleado);

begin

if (a = nil) then begin

new(a);

a^.HI:=nil;

a^.HD:=nil;

a^.dato:=r;

end

else

if r.dni >=a^.dato.dni then agregarHoja(a^.HD, r)

else agregarHoja(a^.HI, r);

end;

var r:empleado;

begin

Randomize;

a:=nil;

leerR(r);

while r.dni <> 0 do begin

agregarHoja(a, r);

leerR(r);

end;

end;

procedure escribirA(a:arbol);

begin

if (a <> nil) then begin

writeln('dni: ', a^.dato.dni,' legajo: ', a^.dato.legajo, ' categoria: ', a^.dato.categoria, ' anio ingreso: ', a^.dato.anioIngreso);

escribirA(a^.HI);

escribirA(a^.HD);

end;

end;

procedure escribirV(v:vector; diml:integer);

var i:integer;

begin

for i:=1 to diml do

writeln('dni: ', v[i].dni,' legajo: ', v[i].legajo);

end;

procedure generarV(a:arbol; var v:vector; var diml:integer; dni1, dni2, numCategoria:integer);

function fueEncontrado(v:vector; diml, dniComparacion:integer):boolean;

var i:integer; aux:boolean;

begin

aux:=False;

i:=1;

while (i<= diml) and (aux = False) do begin

aux:= (v[i].dni = dniComparacion);

i:=i+1;

end;

fueEncontrado:=aux;

end;

procedure buscar(a:arbol; dni1, dni2, numCategoria:integer; var dni:integer; var v:vector);

begin

if a<> nil then

if dni1 > a^.dato.dni then buscar(a^.HD, dni1, dni2, numCategoria, dni,v)

else

if dni2 < a^.dato.dni then buscar(a^.HI, dni1, dni2, numCategoria, dni,v)

else

if (numCategoria = a^.dato.categoria) then begin

if fueEncontrado(v,diml,a^.dato.dni) then begin

buscar(a^.HI, dni1, dni2, numCategoria, dni, v);

if dni = -1 then

buscar(a^.HD, dni1, dni2, numCategoria, dni, v);

end

else begin

diml:=diml+1;

v[diml].dni:=a^.dato.dni;

v[diml].legajo:=a^.dato.legajo;

dni:=a^.dato.dni;

end;

end

else

if (numCategoria <> a^.dato.categoria) then begin

buscar(a^.HI, dni1, dni2, numCategoria, dni, v);

if dni = -1 then

buscar(a^.HD , dni1, dni2, numCategoria, dni, v);

end;

end;

var dni:integer;

begin

diml:=0;

dni:=-1;

buscar(a, dni1, dni2, numCategoria, dni, v);

//creo que no es necesario hacer el diml <=250 porque ya el arbol no deberia traer mas de 250 empleados por categoria

while (diml <250) and (dni <> -1) do begin

dni:=-1;

buscar(a, dni1, dni2, numCategoria, dni, v);

end;

end;

function promedioLegajo(v:vector; diml:integer):real;

function sumatoriaLegajo (v:vector; diml:integer):integer;

begin

if diml > 0 then

sumatoriaLegajo:= v[diml].legajo + sumatoriaLegajo(v,diml-1)

else

sumatoriaLegajo:=0;

end;

begin

if diml = 0 then promedioLegajo:=0

else

promedioLegajo:=sumatoriaLegajo (v,diml)/diml;

end;

var a:arbol; v:vector; diml:integer;

begin

generarA(a);

escribirA(a);

writeln('---------------------');

generarV(a, v,diml, 10, 40, 5);

writeln('diml ', diml);

escribirV(v, diml);

writeln('------------');

writeln(promedioLegajo(v,diml));

end.

La Organización Mundial de la Salud (OMS) necesita un sistema para obtener estadísticas

de las dosis de vacunas aplicadas contra el COVID-19 en cada país. Cada país notifica

diariamente a la OMS la cantidad de dosis aplicadas por tipo de vacuna. De cada

notificación se conoce: fecha, nombre del país, tipo de vacuna (1:Sinopharm, 2:CanSino,

3:AstraZeneca, 4:Sputnik) y cantidad de dosis aplicadas.

Se dispone una estructura que almacena las notificaciones agrupadas por tipo de vacuna.

Para cada tipo, las notificaciones realizadas se encuentran ordenadas por nombre de país.

Implemente un programa que:

a) Reciba la estructura que almacena las notificaciones y, usando la técnica de merge

     acumulador, genere un ABB (ordenado por cantidad de dosis) que contenga para

     cada país, la cantidad total de dosis aplicadas.

b) Implementar un módulo recursivo que reciba la estructura generada en a) y retorne

    el nombre del país que aplicó la mayor cantidad de dosis.

NOTA: Cada país puede aplicar más de un tipo de vacuna por día.

No es necesario implementar la carga de la estructura que se dispone. Alcanza con declarar

el encabezado del procedimiento e invocarlo para que el programa compile

satisfactoriamente.

Una perfumería dispone en papel de la información de las ventas de sus productos.

De cada venta se conoce código de producto, tipo de producto, fecha, cantidad vendida,

monto total de la venta.

Se pide realizar un programa que:

a) A partir de la información de ventas ingresadas por teclado, genere un ABB que almacene

    para cada código de producto, su tipo y la información de todas sus ventas (fecha,

    cantidad vendida, monto de la venta). El ABB debe estar ordenado por código de producto.

    El ingreso de información finaliza cuando se lee cantidad vendida -1.

b) Luego de generada la estructura, informar (debe implementar un módulo para cada inciso):

    i - El tipo de los productos cuyo código está comprendido entre los valores 1500 y 2200.

    ii – La cantidad de ventas cuyo monto total resultó superior a 100 p

{Se lee información acerca de las ventas de productos realizadas en las 5 sucursales de una empresa.

De cada venta se conoce el código de producto, cantidad vendida y monto total de la venta.

Las ventas de cada sucursal se leen de manera consecutiva y ordenadas por código de producto.

La lectura por cada sucursal finaliza cuando se lee el código de producto -1, el cual no se procesa.

Implementar un programa para que a partir de la información leida, resuelva los siguientes ítems:

a) Utilizando la técnica de merge o merge acumulador según corresponda, generar una lista que

contenga la cantidad total vendida para cada código de producto, ordenada por código de

producto.

b) Realizar un módulo recursivo que reciba la lista generada en el punto a y retorne la cantidad de

productos para los cuales la cantidad total vendida supera las 500 unidades.}

Un psicólogo necesita un sistema para administrar a sus pacientes. De cada paciente registra:

dni, cód. de paciente, obra social (1: ioma 2: pami 3: osde 4: galeno 5: no tiene) y costo abonado

por sesión.

Implemente un programa que:

a) Genere un ABB ordenado por dni. Para ello, genere información hasta el paciente con dni 0.

A partir del ABB, realice módulos (uno por inciso) para:

b) Generar una estructura con dni y cód de paciente de los pacientes de ioma, ordenados

     por dni descendente.

c) Dado un dni, modificar la obra social de dicho paciente a una recibida. Considere que el

     paciente puede no existir.

NOTA: Realice el programa principal que invoque a los módulos desarrollados.

Se lee información acerca de las ventas de productos realizadas en las sucursales de una empresa. De

cada venta se conoce fecha, código de producto, cantidad vendida y monto total de la venta. La lectura

finaliza cuando se lee el código de producto -1, el cual no se procesa. Implementar un programa para

que a partir de la información leida, resuelva los siguientes ítems:

a) Generar un árbol binario de búsqueda ordenado por código de producto, donde cada nodo

contenga el código del producto y la cantidad total vendida. El código de producto no puede

repetirse en el árbol.

b) Realizar un módulo que reciba el árbol generado en a. y dos códigos de producto y retorne la suma

de todas las cantidades vendidas para los códigos de productos comprendidos en el rango.

Nota: La información se lee en forma desordenada. Puede existir mas de una venta para un mismo

código de producto.}

[15:05, 22/3/2023] Juan Informatica 2020: Una ciudad necesita un sistema para administrar la limpieza de sus esquinas. Para ello, el

sistema necesita un módulo que lea la información de cada esquina que se limpia: número

de calle, número de avenida, cantidad de objetos recolectados y número identificatorio del

camión (1..10) que se encargó de la limpieza de la esquina. A partir de esa información,

debe generar una estructura que almacene las esquinas agrupadas por número de camión.

Para cada camión, las esquinas deben ordenarse por número de avenida. La lectura finaliza

con el número de camión 0, el cual no se procesa.

Se pide implementar un programa en Pascal que incluya:

a) Un módulo Cargar que (de manera aleatoria) genere datos de las esquinas y los

almacene en una estructura adecuada

.

b) Un módulo que reciba la estructura generada en a) y (usando merge acumulador)

obtenga para cada avenida el total de objetos recolectados (entre todos los

camiones).

Genere un ABB que contenga número de avenida y el total de objetos

recolectados en

la avenida, ordenado por total de objetos recolectados.

[15:08, 22/3/2023] Juan Informatica 2020: //Enunciado:

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

{Se lee información acerca de las ventas de productos realizadas en las sucursales de una empresa. De

cada venta se conoce fecha, código de producto, cantidad vendida y monto total de la venta. La lectura

finaliza cuando se lee el código de producto -1, el cual no se procesa. Implementar un programa para

que a partir de la información leida, resuelva los siguientes ítems:

a) Generar un árbol binario de búsqueda ordenado por código de producto, donde cada nodo

contenga el código del producto y la cantidad total vendida. El código de producto no puede

repetirse en el árbol.

b) Realizar un módulo que reciba el árbol generado en a. y dos códigos de producto y retorne la suma

de todas las cantidades vendidas para los códigos de productos comprendidos en el rango.

Nota: La información se lee en forma desordenada. Puede existir mas de una venta para un mismo

código de producto.}

--------