| program ejercicio2; const  clientes = 57;  sucMin = 101;  sucMax = 109; type  cliente = record  nombreapellido: string;  codcliente: integer;  gastoanual: real;  sucu2:integer;  end;     matriz = array[1..clientes, sucMin..sucMax] of cliente;  vector = array[sucMin..sucMax] of cliente;   //-------------------------- MODULOS ----------------------------------------------------- { 1) Haga un módulo que cargue en una matriz la información de todos los clientes de cada sucursal. En cada fila se deben almacenar los clientes de cada sucursal. Los clientes (generados al azar) inicialmente no tienen orden alguno }  PROCEDURE leercliente(var a: cliente); var   vnombre: array[1..10] of string = ('Juan', 'Pedro', 'Rodrigo', 'Mateo', 'Ana', 'Sandra', 'Carmen', 'Pilar', 'Lucio', 'Victoria');  vapellido: array[1..10] of string = ('Lopez', 'Salazar', 'Benitez', 'Gutierrez', 'Gonzalez', 'Amato', 'Grimaldi', 'Messi', 'Aguero', 'Lizarraga'); begin  a.nombreapellido := vnombre[random(10)+1] + ' ' + vapellido[random(10) + 1];    a.codcliente := random(1000) + 1;  a.gastoanual := random(90000 - 1000 + 1) + 1000; end;  //--------------------------------------------------- PROCEDURE cargarmatriz(var m: matriz); var  i, j: integer;  a: cliente; begin  for i := 1 to clientes do  for j := 101 to 109 do  begin  leercliente(a);  m[i, j] := a;  end; end;  PROCEDURE imprimirMatriz(m: matriz); var  i, j: integer; begin  for j := sucMin to sucMax do  begin  writeln('----------------------------------');  writeln('Clientes de la sucursal ', j, ':');  writeln('----------------------------------');   for i := 1 to clientes do  begin  writeln('Cliente ', i, ': ', m[i, j].nombreapellido);  writeln('Código: ', m[i, j].codcliente);  writeln('Gasto anual: ', m[i, j].gastoanual:6:2);  writeln();  writeln('---------');   end;  end; end;   {2) Haga un módulo recursivo que reciba la matriz y un código de sucursal y que devuelva el mejor cliente (el que haya gastado más dinero) para el código de sucursal recibido.}  PROCEDURE maximo(m: matriz; i: integer; sucursal: integer; var posmax: integer; var max: real); begin  if i>= 1 then  begin  if m[i, sucursal].gastoanual > max then  begin  max := m[i, sucursal].gastoanual;  posmax := i;  end;  maximo(m, i - 1, sucursal, posmax, max);  end; end;   {3) Haga un módulo que reciba la matriz y retorne en un vector los mejores clientes de cada una de las sucursales. Nota: este módulo debe invocar el módulo implementado en el punto 2.}   PROCEDURE mejores(m: matriz; var v: vector); var  posmax: integer;  sucursal: integer;  max: real; begin  for sucursal := sucMin to sucMax do  begin  posmax := -1;  max := -999;  maximo(m, clientes, sucursal, posmax, max);  v[sucursal]:= m[posmax, sucursal];  v[sucursal].sucu2:=sucursal;  end; end;  PROCEDURE imprimovector(v:vector); var i:integer; begin  for i:=sucmin to sucmax do  begin write('sucursal N ' ,i,' : '); write('| Cliente ',v[i].nombreapellido:20); write('| TOTAL ',v[i].gastoanual:15:2); writeln('| Sucursal ',v[i].sucu2:8);  end; end;   {4) Haga un módulo que reciba el vector y lo retorne ordenado alfabéticamente. Nota: preste atención que pasa con los códigos de sucursal al ordenar el vector.}  procedure ordenarxsel(var v: vector); var   i, j, p: integer;  temp: cliente; begin  for i := sucmin to sucmax - 1 do   begin  p := i;  for j := (i + 1) to sucmax do   begin  if v[j].nombreapellido < v[p].nombreapellido then   begin  p := j;  end;  end;  // Intercambio de v[i] con v[p] si es necesario  if p <> i then   begin  temp := v[i];  v[i] := v[p];  v[p] := temp;  end;  end; end;   {5) Haga un módulo recursivo que reciba el vector ordenado y el nombre y apellido de un cliente y devuelva el código de sucursal en el cual es mejor cliente, en caso de existir. Si no se encuentra el cliente se debe retornar el valor 0. La búsqueda debe ser eficiente en tiempo de ejecución.}   FUNCTION buscarClienteRecursivo(v: vector; inicio, fin: integer; nombreapellido: string): integer; var  medio: integer; begin  if inicio > fin then  buscarClienteRecursivo := 0 { No se encontró el cliente }  else  begin  medio := (inicio + fin) div 2;  if v[medio].nombreapellido = nombreapellido then  buscarClienteRecursivo := v[medio].sucu2 { Se encontró el cliente }  else if v[medio].nombreapellido > nombreapellido then  buscarClienteRecursivo := buscarClienteRecursivo(v, inicio, medio - 1, nombreapellido) { Buscar en la mitad izquierda }  else  buscarClienteRecursivo := buscarClienteRecursivo(v, medio + 1, fin, nombreapellido); { Buscar en la mitad derecha }  end; end;     {6) Escriba un programa que invoque a los cinco módulos implementados y compruebe el correcto funcionamiento del mismo.}  var  m: matriz;  v: vector;  codSucursal: integer;  clienteBuscado: string; begin  randomize;  cargarmatriz(m);    WRITELN('IMPRIMO MATRIZ');  imprimirMatriz(m);   WRITELN('IMPRIMO VECTOR');   mejores(m, v);  imprimovector(v);   WRITELN('IMPRIMO VECTOR ORDENADO');  ordenarxsel(v);  imprimovector(v);   readln(clienteBuscado); codSucursal := buscarClienteRecursivo(v, sucMin, sucMax, clienteBuscado);  if codSucursal = 0 then  writeln('Cliente no encontrado.')  else  writeln('El cliente ', clienteBuscado, ' es el mejor cliente de la sucursal ', codSucursal); end. |
| --- |