**Trabajo Práctico N° 4.2:**

**Vectores (Parte 2).**

**Ejercicio 1.**

**(a)** *Dado un vector de enteros de, a lo sumo, 500 valores, realizar un módulo que reciba dicho vector y un valor n y retorne si n se encuentra en el vector o no.*

**(b)** *Modificar el módulo del inciso (a) considerando, ahora, que el vector se encuentra ordenado de manera ascendente.*

program TP4\_E1;

*{$codepage UTF8}*

uses crt;

const

  num\_total=500;

type

  t\_numero=1..num\_total;

  t\_vector\_numeros=array[t\_numero] of int16;

procedure cargar\_vector\_numeros(var vector\_numeros: t\_vector\_numeros);

var

  i: t\_numero;

begin

  for i:= 1 to num\_total do

    vector\_numeros[i]:=random(1000);

end;

function buscar\_desordenado\_vector\_numeros(vector\_numeros: t\_vector\_numeros; num: int16): boolean;

var

  pos: int16;

begin

  pos:=1;

  while ((pos<=num\_total) and (vector\_numeros[pos]<>num)) do

    pos:=pos+1;

  buscar\_desordenado\_vector\_numeros:=(pos<=num\_total);

end;

procedure ordenar\_vector\_numeros(var vector\_numeros: t\_vector\_numeros);

var

  i, j, k: t\_numero;

  item: int16;

begin

  for i:= 1 to (num\_total-1) do

  begin

    k:=i;

    for j:= (i+1) to num\_total do

      if (vector\_numeros[j]<vector\_numeros[k]) then

        k:=j;

    if (k<>i) then

    begin

      item:=vector\_numeros[k];

      vector\_numeros[k]:=vector\_numeros[i];

      vector\_numeros[i]:=item;

    end;

  end;

end;

function buscar\_ordenado\_vector\_numeros(vector\_numeros: t\_vector\_numeros; num: int16): boolean;

var

  pos: int16;

begin

  pos:=1;

  while ((pos<=num\_total) and (vector\_numeros[pos]<num)) do

    pos:=pos+1;

  buscar\_ordenado\_vector\_numeros:=((pos<=num\_total) and (vector\_numeros[pos]=num));

end;

var

  vector\_numeros: t\_vector\_numeros;

  num: int16;

begin

  randomize;

  cargar\_vector\_numeros(vector\_numeros);

  num:=random(1000);

  textcolor(green); write('¿El número '); textcolor(yellow); write(num); textcolor(green); write(' se encontró en el vector (desordenado)?: '); textcolor(red); writeln(buscar\_desordenado\_vector\_numeros(vector\_numeros,num));

  ordenar\_vector\_numeros(vector\_numeros);

  textcolor(green); write('¿El número '); textcolor(yellow); write(num); textcolor(green); write(' se encontró en el vector (ordenado)?: '); textcolor(red); write(buscar\_ordenado\_vector\_numeros(vector\_numeros,num));

end.

**Ejercicio 2.**

*Realizar un programa que resuelva los siguientes incisos:*

**(a)** *Lea nombres de alumnos y los almacene en un vector de, a lo sumo, 500 elementos. La lectura finaliza cuando se lee el nombre “ZZZ”, que no debe procesarse.*

**(b)** *Lea un nombre y elimine la primera ocurrencia de dicho nombre en el vector.*

**(c)** *Lea un nombre y lo inserte en la posición 4 del vector.*

**(d)** *Lea un nombre y lo agregue al vector.*

*Nota: Realizar todas las validaciones necesarias.*

program TP4\_E2;

*{$codepage UTF8}*

uses crt;

const

  nombres\_total=500;

  nombre\_salida='ZZZ';

  pos\_corte=4;

type

  t\_nombre=1..nombres\_total;

  t\_vector\_nombres=array[t\_nombre] of string;

procedure inicializar\_vector\_nombres(var vector\_nombres: t\_vector\_nombres);

var

  i: t\_nombre;

begin

  for i:= 1 to nombres\_total do

    vector\_nombres[i]:='';

end;

function random\_string(length: int8): string;

var

  i: int8;

  string\_aux: string;

begin

  string\_aux:='';

  for i:= 1 to length do

    string\_aux:=string\_aux+chr(ord('A')+random(26));

  random\_string:=string\_aux;

end;

procedure leer\_nombre(var nombre: string);

var

  i: int16;

begin

  i:=random(100);

  if (i=0) then

    nombre:=nombre\_salida

  else

    nombre:=random\_string(5+random(6));

end;

procedure cargar\_vector\_nombres(var vector\_nombres: t\_vector\_nombres; var nombres: int16);

var

  nombre: string;

begin

  leer\_nombre(nombre);

  while ((nombre<>nombre\_salida) and (nombres<nombres\_total)) do

  begin

    nombres:=nombres+1;

    vector\_nombres[nombres]:=nombre;

    leer\_nombre(nombre);

  end;

end;

function buscar\_desordenado\_vector\_nombres(vector\_nombres: t\_vector\_nombres; nombres: int16; nombre: string): int16;

var

  pos: int16;

begin

  pos:=1;

  while ((pos<=nombres) and (vector\_nombres[pos]<>nombre)) do

    pos:=pos+1;

  if (pos<=nombres) then

    buscar\_desordenado\_vector\_nombres:=pos

  else

    buscar\_desordenado\_vector\_nombres:=-1;

end;

procedure eliminar\_vector\_nombres(var vector\_nombres: t\_vector\_nombres; var nombres: int16; nombre: string; pos: int16);

var

  i: t\_nombre;

begin

  if ((pos>=1) and (pos<=nombres)) then

  begin

    for i:= pos to (nombres-1) do

      vector\_nombres[i]:=vector\_nombres[i+1];

    nombres:=nombres-1;

  end;

end;

procedure insertar\_vector\_nombres(var vector\_nombres: t\_vector\_nombres; var nombres: int16; nombre: string; pos: int16);

var

  i: t\_nombre;

begin

  if ((nombres<nombres\_total) and ((pos>=1) and (pos<=nombres))) then

  begin

    for i:= nombres downto pos do

      vector\_nombres[i+1]:=vector\_nombres[i];

    vector\_nombres[pos\_corte]:=nombre;

    nombres:=nombres+1;

  end;

end;

procedure agregar\_vector\_nombres(var vector\_nombres: t\_vector\_nombres; var nombres: int16; nombre: string);

begin

  if (nombres<nombres\_total) then

  begin

    nombres:=nombres+1;

    vector\_nombres[nombres]:=nombre;

  end;

end;

procedure imprimir\_vector\_nombres(vector\_nombres: t\_vector\_nombres; nombres: int16);

var

  i: int16;

begin

  for i:= 1 to nombres do

  begin

    textcolor(green); write('Elemento ',i,' del vector: '); textcolor(red); writeln(vector\_nombres[i]);

  end;

end;

var

  vector\_nombres: t\_vector\_nombres;

  nombre: string;

  nombres: int16;

begin

  randomize;

  nombres:=0;

  inicializar\_vector\_nombres(vector\_nombres);

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_vector\_nombres(vector\_nombres,nombres);

  if (nombres>0) then

  begin

    imprimir\_vector\_nombres(vector\_nombres,nombres);

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    nombre:=vector\_nombres[1+random(nombres)];

    eliminar\_vector\_nombres(vector\_nombres,nombres,nombre,buscar\_desordenado\_vector\_nombres(vector\_nombres,nombres,nombre));

    imprimir\_vector\_nombres(vector\_nombres,nombres);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    nombre:=random\_string(5+random(6));

    insertar\_vector\_nombres(vector\_nombres,nombres,nombre,pos\_corte);

    imprimir\_vector\_nombres(vector\_nombres,nombres);

    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

    nombre:=random\_string(5+random(6));

    agregar\_vector\_nombres(vector\_nombres,nombres,nombre);

    imprimir\_vector\_nombres(vector\_nombres,nombres);

  end;

end.

**Ejercicio 3.**

*Una empresa de transporte de caudales desea optimizar el servicio que brinda a sus clientes. Para ello, cuenta con información sobre todos los viajes realizados durante el mes de marzo. De cada viaje, se cuenta con la siguiente información: día del mes (de 1 a 31), monto de dinero transportado y distancia recorrida por el camión (medida en kilómetros).*

**(a)** *Realizar un programa que lea y almacene la información de los viajes (a lo sumo, 200). La lectura finaliza cuando se ingresa una distancia recorrida igual a 0 km, que no debe procesarse.*

**(b)** *Realizar un módulo que reciba el vector generado en (a) e informe:*

* *El monto promedio transportado de los viajes realizados.*
* *La distancia recorrida y el día del mes en que se realizó el viaje que transportó menos dinero.*
* *La cantidad de viajes realizados cada día del mes.*

**(c)** *Realizar un módulo que reciba el vector generado en (a) y elimine todos los viajes cuya distancia recorrida sea igual a 100 km.*

*Nota: Para realizar el inciso (b), el vector debe recorrerse una única vez.*

program TP4\_E3;

*{$codepage UTF8}*

uses crt;

const

  dia\_ini=1; dia\_fin=31;

  viajes\_total=200;

  distancia\_salida=0;

  distancia\_corte=100;

type

  t\_viaje=1..viajes\_total;

  t\_dia=dia\_ini..dia\_fin;

  t\_registro\_viaje=record

    dia: t\_dia;

    monto: real;

    distancia: real;

  end;

  t\_vector\_viajes=array[t\_viaje] of t\_registro\_viaje;

  t\_vector\_dias=array[t\_dia] of int16;

procedure inicializar\_vector\_dias(var vector\_dias: t\_vector\_dias);

var

  i: t\_dia;

begin

  for i:= dia\_ini to dia\_fin do

    vector\_dias[i]:=0;

end;

procedure leer\_viaje(var registro\_viaje: t\_registro\_viaje);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_viaje.distancia:=distancia\_salida

  else if (i<=50) then

    registro\_viaje.distancia:=distancia\_corte

  else

    registro\_viaje.distancia:=1+random(1000);

  if (registro\_viaje.distancia<>distancia\_salida) then

  begin

    registro\_viaje.dia:=dia\_ini+random(dia\_fin);

    registro\_viaje.monto:=1+random(100);

  end;

end;

procedure cargar\_vector\_viajes(var vector\_viajes: t\_vector\_viajes; var viajes: int16);

var

  registro\_viaje: t\_registro\_viaje;

begin

  leer\_viaje(registro\_viaje);

  while ((registro\_viaje.distancia<>distancia\_salida) and (viajes<viajes\_total)) do

  begin

    viajes:=viajes+1;

    vector\_viajes[viajes]:=registro\_viaje;

    leer\_viaje(registro\_viaje);

  end;

end;

procedure actualizar\_minimo(monto: real; dia: t\_dia; distancia: real; var monto\_min: real; var dia\_min: int8; var distancia\_min: real);

begin

  if (monto<monto\_min) then

  begin

    monto\_min:=monto;

    dia\_min:=dia;

    distancia\_min:=distancia;

  end;

end;

procedure calcular\_informar\_vector\_viajes(vector\_viajes: t\_vector\_viajes; viajes: int16);

var

  vector\_dias: t\_vector\_dias;

  i: t\_viaje;

  j: t\_dia;

  dia\_min: int8;

  monto\_total, monto\_prom, monto\_min, distancia\_min: real;

begin

  monto\_total:=0; monto\_prom:=0;

  monto\_min:=9999999; distancia\_min:=0; dia\_min:=0;

  inicializar\_vector\_dias(vector\_dias);

  for i:= 1 to viajes do

  begin

    monto\_total:=monto\_total+vector\_viajes[i].monto;

    actualizar\_minimo(vector\_viajes[i].monto,vector\_viajes[i].dia,vector\_viajes[i].distancia,monto\_min,dia\_min,distancia\_min);

    vector\_dias[vector\_viajes[i].dia]:=vector\_dias[vector\_viajes[i].dia]+1;

  end;

  monto\_prom:=monto\_total/viajes;

  textcolor(green); write('El monto promedio de los viajes realizados es $'); textcolor(red); writeln(monto\_prom:0:2);

  textcolor(green); write('La distancia recorrida y el día del mes en que se realizó el viaje que transportó menos dinero son '); textcolor(red); write(distancia\_min:0:2); textcolor(green); write(' y '); textcolor(red); write(dia\_min); textcolor(green); writeln(', respectivamente');

  for j:= dia\_ini to dia\_fin do

  begin

    textcolor(green); write('La cantidad de viajes realizados el día ',j,' del mes de marzo es '); textcolor(red); writeln(vector\_dias[j]);

  end;

end;

procedure buscar\_desordenado\_vector\_viajes(vector\_viajes: t\_vector\_viajes; viajes: int16; var pos: int16);

begin

  while ((pos<=viajes) and (vector\_viajes[pos].distancia<>distancia\_corte)) do

    pos:=pos+1;

  if (pos>viajes) then

    pos:=-1;

end;

procedure eliminar\_vector\_viajes(var vector\_viajes: t\_vector\_viajes; var viajes: int16; pos: int16);

var

  i: t\_viaje;

begin

  if ((pos>=1) and (pos<=viajes)) then

  begin

    for i:= pos to (viajes-1) do

      vector\_viajes[i]:=vector\_viajes[i+1];

    viajes:=viajes-1;

  end;

end;

procedure buscar\_eliminar\_vector\_viajes(var vector\_viajes: t\_vector\_viajes; var viajes: int16);

var

  pos: int16;

begin

  pos:=0;

  buscar\_desordenado\_vector\_viajes(vector\_viajes,viajes,pos);

  while ((pos>=1) and (pos<=viajes)) do

  begin

    eliminar\_vector\_viajes(vector\_viajes,viajes,pos);

    buscar\_desordenado\_vector\_viajes(vector\_viajes,viajes,pos);

  end;

end;

procedure imprimir\_vector\_viajes(vector\_viajes: t\_vector\_viajes; viajes: int16);

var

  i: int16;

begin

  for i:= 1 to viajes do

  begin

    textcolor(green); write('Elemento ',i,' del vector (elemento distancia): '); textcolor(red); writeln(vector\_viajes[i].distancia:0:2);

  end;

end;

var

  vector\_viajes: t\_vector\_viajes;

  viajes: int16;

begin

  randomize;

  viajes:=0;

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_vector\_viajes(vector\_viajes,viajes);

  if (viajes>0) then

  begin

    imprimir\_vector\_viajes(vector\_viajes,viajes);

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    calcular\_informar\_vector\_viajes(vector\_viajes,viajes);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    buscar\_eliminar\_vector\_viajes(vector\_viajes,viajes);

    imprimir\_vector\_viajes(vector\_viajes,viajes);

  end;

end.

**Ejercicio 4.**

*Una cátedra dispone de información de sus alumnos (a lo sumo, 1000). De cada alumno, se conoce número de alumno, apellido y nombre y cantidad de asistencias a clase. Dicha información se encuentra ordenada por número de alumno de manera ascendente. Se pide:*

**(a)** *Un módulo que retorne la posición del alumno con un número de alumno recibido por parámetro. El alumno seguro existe.*

**(b)** *Un módulo que reciba un alumno y lo inserte en el vector.*

**(c)** *Un módulo que reciba la posición de un alumno dentro del vector y lo elimine.*

**(d)** *Un módulo que reciba un número de alumno y elimine dicho alumno del vector.*

**(e)** *Un módulo que elimine del vector todos los alumnos con cantidad de asistencias en 0.*

*Nota: Realizar el programa principal que invoque los módulos desarrollados en los incisos previos con datos leídos de teclado.*

program TP4\_E4;

*{$codepage UTF8}*

uses crt;

const

  alumnos\_total=1000;

  numero\_salida=-1;

  asistencias\_corte=0;

type

  t\_alumno=1..alumnos\_total;

  t\_registro\_alumno=record

    numero: int16;

    apellido: string;

    nombre: string;

    asistencias: int8;

  end;

  t\_vector\_alumnos=array[t\_alumno] of t\_registro\_alumno;

function random\_string(length: int8): string;

var

  i: int8;

  string\_aux: string;

begin

  string\_aux:='';

  for i:= 1 to length do

    string\_aux:=string\_aux+chr(ord('A')+random(26));

  random\_string:=string\_aux;

end;

procedure leer\_alumno(var registro\_alumno: t\_registro\_alumno);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_alumno.numero:=numero\_salida

  else

    registro\_alumno.numero:=1+random(high(int16));

  if (registro\_alumno.numero<>numero\_salida) then

  begin

    registro\_alumno.apellido:=random\_string(5+random(6));

    registro\_alumno.nombre:=random\_string(5+random(6));

    registro\_alumno.asistencias:=random(100);

  end;

end;

function buscar\_ordenado1\_vector\_alumnos(vector\_alumnos: t\_vector\_alumnos; alumnos, numero: int16): int16;

var

  pos: int16;

begin

  pos:=1;

  while ((pos<=alumnos) and (vector\_alumnos[pos].numero<numero)) do

    pos:=pos+1;

  buscar\_ordenado1\_vector\_alumnos:=pos;

end;

procedure insertar\_vector\_alumnos(var vector\_alumnos: t\_vector\_alumnos; var alumnos: int16; registro\_alumno: t\_registro\_alumno; pos: int16);

var

  i: t\_alumno;

begin

  if ((alumnos<alumnos\_total) and ((pos>=1) and (pos<=alumnos))) then

    for i:= alumnos downto pos do

      vector\_alumnos[i+1]:=vector\_alumnos[i];

  if ((alumnos<alumnos\_total) and ((pos>=1) and (pos<=alumnos+1))) then

  begin

    vector\_alumnos[pos]:=registro\_alumno;

    alumnos:=alumnos+1;

  end;

end;

procedure cargar\_vector\_alumnos(var vector\_alumnos: t\_vector\_alumnos; var alumnos: int16);

var

  registro\_alumno: t\_registro\_alumno;

  pos: int16;

begin

  pos:=0;

  leer\_alumno(registro\_alumno);

  while ((registro\_alumno.numero<>numero\_salida) and (alumnos<alumnos\_total)) do

  begin

    pos:=buscar\_ordenado1\_vector\_alumnos(vector\_alumnos,alumnos,registro\_alumno.numero);

    insertar\_vector\_alumnos(vector\_alumnos,alumnos,registro\_alumno,pos);

    leer\_alumno(registro\_alumno);

  end;

end;

function calcular\_a(vector\_alumnos: t\_vector\_alumnos; alumnos, numero: int16): int16;

begin

  calcular\_a:=buscar\_ordenado1\_vector\_alumnos(vector\_alumnos,alumnos,numero);

end;

procedure calcular\_b(var vector\_alumnos: t\_vector\_alumnos; var alumnos: int16; registro\_alumno: t\_registro\_alumno);

var

  pos: int16;

begin

  pos:=0;

  if (alumnos<alumnos\_total) then

  begin

    pos:=buscar\_ordenado1\_vector\_alumnos(vector\_alumnos,alumnos,registro\_alumno.numero);

    insertar\_vector\_alumnos(vector\_alumnos,alumnos,registro\_alumno,pos);

  end;

end;

procedure calcular\_c(var vector\_alumnos: t\_vector\_alumnos; var alumnos: int16; pos: int16);

var

  i: t\_alumno;

begin

  if ((pos>=1) and (pos<=alumnos)) then

  begin

    for i:= pos to (alumnos-1) do

      vector\_alumnos[i]:=vector\_alumnos[i+1];

    alumnos:=alumnos-1;

  end;

end;

function buscar\_ordenado2\_vector\_alumnos(vector\_alumnos: t\_vector\_alumnos; alumnos, numero: int16): int16;

var

  pos: int16;

begin

  pos:=1;

  while ((pos<=alumnos) and (vector\_alumnos[pos].numero<numero)) do

    pos:=pos+1;

  if ((pos<=alumnos) and (vector\_alumnos[pos].numero=numero)) then

    buscar\_ordenado2\_vector\_alumnos:=pos

  else

    buscar\_ordenado2\_vector\_alumnos:=-1;

end;

procedure calcular\_d(var vector\_alumnos: t\_vector\_alumnos; var alumnos: int16; numero: int16);

begin

  calcular\_c(vector\_alumnos,alumnos,buscar\_ordenado2\_vector\_alumnos(vector\_alumnos,alumnos,numero));

end;

procedure buscar\_desordenado\_vector\_alumnos(vector\_alumnos: t\_vector\_alumnos; alumnos: int16; var pos: int16);

begin

  while ((pos<=alumnos) and (vector\_alumnos[pos].asistencias<>asistencias\_corte)) do

    pos:=pos+1;

  if (pos>alumnos) then

    pos:=-1;

end;

procedure calcular\_e(var vector\_alumnos: t\_vector\_alumnos; var alumnos: int16);

var

  pos: int16;

begin

  pos:=1;

  buscar\_desordenado\_vector\_alumnos(vector\_alumnos,alumnos,pos);

  while ((pos>=1) and (pos<=alumnos)) do

  begin

    calcular\_c(vector\_alumnos,alumnos,pos);

    buscar\_desordenado\_vector\_alumnos(vector\_alumnos,alumnos,pos);

  end;

end;

var

  registro\_alumno: t\_registro\_alumno;

  vector\_alumnos: t\_vector\_alumnos;

  alumnos, pos, numero: int16;

begin

  randomize;

  alumnos:=0;

  cargar\_vector\_alumnos(vector\_alumnos,alumnos);

  if (alumnos>0) then

  begin

    writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

    numero:=1+random(high(int16));

    textcolor(green); write('La posición en el vector del alumno con número de alumno '); textcolor(yellow); write(numero); textcolor(green); write(' es '); textcolor(red); writeln(calcular\_a(vector\_alumnos,alumnos,numero));

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    leer\_alumno(registro\_alumno);

    calcular\_b(vector\_alumnos,alumnos,registro\_alumno);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    pos:=1+random(alumnos);

    calcular\_c(vector\_alumnos,alumnos,pos);

    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

    numero:=1+random(high(int16));

    calcular\_d(vector\_alumnos,alumnos,numero);

    writeln(); textcolor(red); writeln('INCISO (e):'); writeln();

    calcular\_e(vector\_alumnos,alumnos);

  end;

end.

**Ejercicio 5.**

*La empresa Amazon Web Services (AWS) dispone de la información de sus 500 clientes monotributistas más grandes del país. De cada cliente, conoce la fecha de firma del contrato con AWS, la categoría del monotributo (entre la A y la F), el código de la ciudad donde se encuentran las oficinas (entre 1 y 2400) y el monto mensual acordado en el contrato. La información se ingresa ordenada por fecha de firma de contrato (los más antiguos primero, los más recientes últimos). Realizar un programa que lea y almacene la información de los clientes en una estructura de tipo vector. Una vez almacenados los datos, procesar dicha estructura para obtener:*

* *Cantidad de contratos por cada mes y cada año, y año en que se firmó la mayor cantidad de contratos.*
* *Cantidad de clientes para cada categoría de monotributo.*
* *Código de las 10 ciudades con mayor cantidad de clientes.*
* *Cantidad de clientes que superan, mensualmente, el monto promedio entre todos los clientes.*

program TP4\_E5;

*{$codepage UTF8}*

uses crt;

const

  clientes\_total=500;

  ciudades\_total=2400;

  mes\_ini=1; mes\_fin=12;

  anio\_ini=2001; anio\_fin=2020;

  cat\_ini='A'; cat\_fin='F';

  ciudad\_ini=1; ciudad\_fin=10;

type

  t\_cliente=1..clientes\_total;

  t\_mes=mes\_ini..mes\_fin;

  t\_anio=anio\_ini..anio\_fin;

  t\_categoria=cat\_ini..cat\_fin;

  t\_ciudad1=1..ciudades\_total;

  t\_ciudad2=ciudad\_ini..ciudad\_fin;

  t\_registro\_cliente=record

    fecha: int16;

    categoria: t\_categoria;

    ciudad: t\_ciudad1;

    monto: real;

  end;

  t\_registro\_ciudad=record

    ciudad: int16;

    clientes: int16;

  end;

  t\_vector\_clientes=array[t\_cliente] of t\_registro\_cliente;

  t\_vector\_meses=array[t\_mes] of int16;

  t\_vector\_anios=array[t\_anio] of int16;

  t\_vector\_categorias=array[t\_categoria] of int16;

  t\_vector\_ciudades1=array[t\_ciudad1] of int16;

  t\_vector\_ciudades2=array[t\_ciudad2] of t\_registro\_ciudad;

procedure inicializar\_vectores(var vector\_meses1, vector\_meses2: t\_vector\_meses; var vector\_anios: t\_vector\_anios; var vector\_categorias: t\_vector\_categorias; var vector\_ciudades1: t\_vector\_ciudades1; var vector\_ciudades2: t\_vector\_ciudades2);

var

  i: t\_mes;

  j: t\_anio;

  k: t\_categoria;

  l: t\_ciudad1;

  m: t\_ciudad2;

begin

  for i:= mes\_ini to mes\_fin do

  begin

    vector\_meses1[i]:=0;

    vector\_meses2[i]:=0;

  end;

  for j:= anio\_ini to anio\_fin do

    vector\_anios[j]:=0;

  for k:= cat\_ini to cat\_fin do

    vector\_categorias[k]:=0;

  for l:= 1 to ciudades\_total do

    vector\_ciudades1[l]:=0;

  for m:= ciudad\_ini to ciudad\_fin do

  begin

    vector\_ciudades2[m].ciudad:=0;

    vector\_ciudades2[m].clientes:=0;

  end;

end;

procedure leer\_cliente(var registro\_cliente: t\_registro\_cliente);

begin

  registro\_cliente.fecha:=(anio\_ini\*12-1)+random((anio\_fin-anio\_ini+1)\*12);

  registro\_cliente.categoria:=chr(ord(cat\_ini)+random(6));

  registro\_cliente.ciudad:=1+random(ciudades\_total);

  registro\_cliente.monto:=1+random(100);

end;

function buscar\_ordenado\_vector\_clientes(vector\_clientes: t\_vector\_clientes; clientes, fecha: int16): int16;

var

  pos: int16;

begin

  pos:=1;

  while ((pos<=clientes) and (vector\_clientes[pos].fecha<fecha)) do

    pos:=pos+1;

  buscar\_ordenado\_vector\_clientes:=pos;

end;

procedure insertar\_vector\_clientes(var vector\_clientes: t\_vector\_clientes; var clientes: int16; registro\_cliente: t\_registro\_cliente; pos: int16);

var

  i: t\_cliente;

begin

  if ((clientes<clientes\_total) and ((pos>=1) and (pos<=clientes))) then

    for i:= clientes downto pos do

      vector\_clientes[i+1]:=vector\_clientes[i];

  if ((clientes<clientes\_total) and ((pos>=1) and (pos<=clientes+1))) then

  begin

    vector\_clientes[pos]:=registro\_cliente;

    clientes:=clientes+1;

  end;

end;

procedure cargar\_vector\_clientes(var vector\_clientes: t\_vector\_clientes; var monto\_prom: real);

var

  registro\_cliente: t\_registro\_cliente;

  i: t\_cliente;

  clientes, pos: int16;

  monto\_total: real;

begin

  clientes:=0; pos:=0;

  monto\_total:=0;

  for i:= 1 to clientes\_total do

  begin

    leer\_cliente(registro\_cliente);

    pos:=buscar\_ordenado\_vector\_clientes(vector\_clientes,clientes,registro\_cliente.fecha);

    insertar\_vector\_clientes(vector\_clientes,clientes,registro\_cliente,pos);

    monto\_total:=monto\_total+vector\_clientes[i].monto;

  end;

  monto\_prom:=monto\_total/clientes\_total;

end;

procedure agregar\_vector\_meses1(fecha: int16; var vector\_meses1: t\_vector\_meses);

begin

  vector\_meses1[(fecha mod 12)+1]:=vector\_meses1[(fecha mod 12)+1]+1;

end;

procedure agregar\_vector\_anios(fecha: int16; var vector\_anios: t\_vector\_anios);

begin

  vector\_anios[fecha div 12]:=vector\_anios[fecha div 12]+1;

end;

procedure agregar\_vector\_categorias(categoria: t\_categoria; var vector\_categorias: t\_vector\_categorias);

begin

  vector\_categorias[categoria]:=vector\_categorias[categoria]+1;

end;

procedure agregar\_vector\_ciudades1(ciudad: t\_ciudad1; var vector\_ciudades1: t\_vector\_ciudades1);

begin

  vector\_ciudades1[ciudad]:=vector\_ciudades1[ciudad]+1;

end;

procedure agregar\_vector\_meses2(fecha: int16; monto, monto\_prom: real; var vector\_meses2: t\_vector\_meses);

begin

  if (monto>monto\_prom) then

    vector\_meses2[(fecha mod 12)+1]:=vector\_meses2[(fecha mod 12)+1]+1;

end;

procedure actualizar\_maximo(vector\_anios: t\_vector\_anios; var anio\_max: int16);

var

  i: t\_anio;

  num\_max: int16;

begin

  num\_max:=low(int16);

  for i:= anio\_ini to anio\_fin do

    if (vector\_anios[i]>num\_max) then

    begin

      num\_max:=vector\_anios[i];

      anio\_max:=i;

    end;

end;

function buscar\_ordenado\_vector\_ciudades2(vector\_ciudades2: t\_vector\_ciudades2; ciudades, clientes: int16): int16;

var

  pos: int16;

begin

  pos:=1;

  while ((pos<=ciudades) and (vector\_ciudades2[pos].clientes>clientes)) do

    pos:=pos+1;

  buscar\_ordenado\_vector\_ciudades2:=pos;

end;

procedure insertar\_vector\_ciudades2(var vector\_ciudades2: t\_vector\_ciudades2; var ciudades: int16; registro\_ciudad: t\_registro\_ciudad; pos: int16);

var

  i: t\_ciudad2;

begin

  if ((ciudades<ciudad\_fin) and ((pos>1) and (pos<=ciudades))) then

    for i:= ciudades downto pos do

      vector\_ciudades2[i+1]:=vector\_ciudades2[i];

  if ((ciudades<ciudad\_fin) and ((pos>1) and (pos<=ciudades+1))) then

  begin

    vector\_ciudades2[pos]:=registro\_ciudad;

    ciudades:=ciudades+1;

  end;

end;

procedure actualizar\_maximos(vector\_ciudades1: t\_vector\_ciudades1; var vector\_ciudades2: t\_vector\_ciudades2);

var

  registro\_ciudad: t\_registro\_ciudad;

  i: t\_ciudad1;

  ciudades, pos: int16;

begin

  ciudades:=0; pos:=0;

  for i:= 1 to ciudades\_total do

  begin

    pos:=buscar\_ordenado\_vector\_ciudades2(vector\_ciudades2,ciudades,vector\_ciudades1[i]);

    if (pos<=ciudad\_fin) then

    begin

      if (ciudades=ciudad\_fin) then

        ciudades:=ciudades-1;

      registro\_ciudad.ciudad:=i;

      registro\_ciudad.clientes:=vector\_ciudades1[i];

      insertar\_vector\_ciudades2(vector\_ciudades2,ciudades,registro\_ciudad,pos);

    end;

  end;

end;

procedure procesar\_vector\_clientes(vector\_clientes: t\_vector\_clientes; monto\_prom: real; var vector\_meses1, vector\_meses2: t\_vector\_meses; var vector\_anios: t\_vector\_anios; var anio\_max: int16; var vector\_categorias: t\_vector\_categorias; var vector\_ciudades1: t\_vector\_ciudades1; var vector\_ciudades2: t\_vector\_ciudades2);

var

  i: t\_cliente;

begin

  for i:= 1 to clientes\_total do

  begin

    agregar\_vector\_meses1(vector\_clientes[i].fecha,vector\_meses1);

    agregar\_vector\_anios(vector\_clientes[i].fecha,vector\_anios);

    agregar\_vector\_categorias(vector\_clientes[i].categoria,vector\_categorias);

    agregar\_vector\_ciudades1(vector\_clientes[i].ciudad,vector\_ciudades1);

    agregar\_vector\_meses2(vector\_clientes[i].fecha,vector\_clientes[i].monto,monto\_prom,vector\_meses2);

  end;

  actualizar\_maximo(vector\_anios,anio\_max);

  actualizar\_maximos(vector\_ciudades1,vector\_ciudades2);

end;

procedure imprimir\_vector\_meses(vector\_meses: t\_vector\_meses);

var

  i: t\_mes;

begin

  for i:= mes\_ini to mes\_fin do

  begin

    textcolor(green); write('Mes ',i,': '); textcolor(red); writeln(vector\_meses[i]);

  end;

end;

procedure imprimir\_vector\_anios(vector\_anios: t\_vector\_anios);

var

  i: t\_anio;

begin

  for i:= anio\_ini to anio\_fin do

  begin

    textcolor(green); write('Año ',i,': '); textcolor(red); writeln(vector\_anios[i]);

  end;

end;

procedure imprimir\_vector\_categorias(vector\_categorias: t\_vector\_categorias);

var

  i: t\_categoria;

begin

  for i:= cat\_ini to cat\_fin do

  begin

    textcolor(green); write('Categoría ',i,': '); textcolor(red); writeln(vector\_categorias[i]);

  end;

end;

procedure imprimir\_vector\_ciudades(vector\_ciudades: t\_vector\_ciudades2);

var

  i: t\_ciudad2;

begin

  for i:= ciudad\_ini to ciudad\_fin do

  begin

    textcolor(green); write('Ciudad ',i,': '); textcolor(red); writeln(vector\_ciudades[i].ciudad);

  end;

end;

var

  vector\_clientes: t\_vector\_clientes;

  vector\_meses1, vector\_meses2: t\_vector\_meses;

  vector\_anios: t\_vector\_anios;

  vector\_categorias: t\_vector\_categorias;

  vector\_ciudades1: t\_vector\_ciudades1;

  vector\_ciudades2: t\_vector\_ciudades2;

  anio\_max: int16;

  monto\_prom: real;

begin

  randomize;

  anio\_max:=0;

  monto\_prom:=0;

  inicializar\_vectores(vector\_meses1,vector\_meses2,vector\_anios,vector\_categorias,vector\_ciudades1,vector\_ciudades2);

  cargar\_vector\_clientes(vector\_clientes,monto\_prom);

  procesar\_vector\_clientes(vector\_clientes,monto\_prom,vector\_meses1,vector\_meses2,vector\_anios,anio\_max,vector\_categorias,vector\_ciudades1,vector\_ciudades2);

  writeln(); textcolor(red); writeln('La cantidad de contratos para cada mes es '); writeln();

  imprimir\_vector\_meses(vector\_meses1);

  writeln(); textcolor(red); writeln('La cantidad de contratos para cada año es '); writeln();

  imprimir\_vector\_anios(vector\_anios);

  writeln(); textcolor(red); write('El año en que se firmó la mayor cantidad de contratos es '); textcolor(red); writeln(anio\_max); writeln();

  writeln(); textcolor(red); writeln('La cantidad de clientes para cada categoría de monotributo es '); writeln();

  imprimir\_vector\_categorias(vector\_categorias);

  writeln(); textcolor(red); writeln('El código de las 10 ciudades con mayor cantidad de clientes es '); writeln();

  imprimir\_vector\_ciudades(vector\_ciudades2);

  writeln(); textcolor(red); writeln('La cantidad de clientes que superan, mensualmente, el monto promedio entre todos los clientes es '); writeln();

  imprimir\_vector\_meses(vector\_meses2);

end.

**Ejercicio 6.**

*La compañía Canonical Llt. desea obtener estadísticas acerca del uso de Ubuntu Linux en La Plata. Para ello, debe realizar un programa que lea y almacene información sobre las computadoras con este sistema operativo (a lo sumo, 10000). De cada computadora se conoce: código de computadora, la versión de Ubuntu que utiliza (18.04, 17.10, 17.04, etc.), la cantidad de paquetes instalados y la cantidad de cuentas de usuario que posee. La información debe almacenarse ordenada por código de computadora de manera ascendente. La lectura finaliza al ingresar el código de computadora -1, que no debe procesarse. Una vez almacenados todos los datos, se pide:*

* *Informar la cantidad de computadoras que utilizan las versiones 18.04 o 16.04.*
* *Informar el promedio de cuentas de usuario por computadora.*
* *Informar la versión de Ubuntu de la computadora con mayor cantidad de paquetes instalados.*
* *Eliminar la información de las computadoras con código entre 0 y 500.*

program TP4\_E6;

*{$codepage UTF8}*

uses crt;

const

  computadoras\_total=10000;

  computadora\_salida=-1;

  version\_corte1='18.04'; version\_corte2='16.04';

  computadora\_corte1=0; computadora\_corte2=500;

type

  t\_computadora=1..computadoras\_total;

  t\_registro\_computadora=record

    computadora: int16;

    version: string;

    paquetes: int16;

    cuentas: int16;

  end;

  t\_vector\_computadoras=array[t\_computadora] of t\_registro\_computadora;

procedure leer\_computadora(var registro\_computadora: t\_registro\_computadora);

var

  vector\_versiones: array[1..4] of string=('18.04', '17.10', '17.04', '16.04');

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_computadora.computadora:=computadora\_salida

  else

    registro\_computadora.computadora:=1+random(high(int16));

  if (registro\_computadora.computadora<>computadora\_salida) then

  begin

    registro\_computadora.version:=vector\_versiones[1+random(4)];

    registro\_computadora.paquetes:=1+random(100);

    registro\_computadora.cuentas:=1+random(100);

  end;

end;

function buscar\_ordenado\_vector\_computadoras(vector\_computadoras: t\_vector\_computadoras; computadoras, computadora: int16): int16;

var

  pos: int16;

begin

  pos:=1;

  while ((pos<=computadoras) and (vector\_computadoras[pos].computadora<computadora)) do

    pos:=pos+1;

  buscar\_ordenado\_vector\_computadoras:=pos;

end;

procedure insertar\_vector\_computadoras(var vector\_computadoras: t\_vector\_computadoras; var computadoras: int16; registro\_computadora: t\_registro\_computadora; pos: int16);

var

  i: t\_computadora;

begin

  if ((computadoras<computadoras\_total) and ((pos>=1) and (pos<=computadoras))) then

    for i:= computadoras downto pos do

      vector\_computadoras[i+1]:=vector\_computadoras[i];

  if ((computadoras<computadoras\_total) and ((pos>=1) and (pos<=computadoras+1))) then

  begin

    vector\_computadoras[pos]:=registro\_computadora;

    computadoras:=computadoras+1;

  end;

end;

procedure cargar\_vector\_computadoras(var vector\_computadoras: t\_vector\_computadoras; var computadoras: int16);

var

  registro\_computadora: t\_registro\_computadora;

  pos: int16;

begin

  pos:=0;

  leer\_computadora(registro\_computadora);

  while ((registro\_computadora.computadora<>computadora\_salida) and (computadoras<computadoras\_total)) do

  begin

    pos:=buscar\_ordenado\_vector\_computadoras(vector\_computadoras,computadoras,registro\_computadora.computadora);

    insertar\_vector\_computadoras(vector\_computadoras,computadoras,registro\_computadora,pos);

    leer\_computadora(registro\_computadora);

  end;

end;

procedure actualizar\_maximo(paquetes: int16; version: string; var paquetes\_max: int16; var version\_max: string);

begin

  if (paquetes>paquetes\_max) then

  begin

    paquetes\_max:=paquetes;

    version\_max:=version;

  end;

end;

procedure eliminar\_vector\_computadoras(var vector\_computadoras: t\_vector\_computadoras; var computadoras: int16; pos: int16);

var

  i: t\_computadora;

begin

  if ((pos>=1) and (pos<=computadoras)) then

  begin

    for i:= pos to (computadoras-1) do

      vector\_computadoras[i]:=vector\_computadoras[i+1];

    computadoras:=computadoras-1;

  end;

end;

procedure procesar\_vector\_computadoras(var vector\_computadoras: t\_vector\_computadoras; var computadoras, versiones\_corte: int16; var cuentas\_prom: real; var version\_max: string);

var

  pos: int16;

  computadoras\_aux, cuentas\_total, paquetes\_max: int16;

begin

  pos:=1;

  computadoras\_aux:=computadoras;

  cuentas\_total:=0;

  paquetes\_max:=low(int16);

  while ((pos>=1) and (pos<=computadoras)) do

  begin

    if ((vector\_computadoras[pos].version=version\_corte1) or (vector\_computadoras[pos].version=version\_corte2)) then

      versiones\_corte:=versiones\_corte+1;

    cuentas\_total:=cuentas\_total+vector\_computadoras[pos].cuentas;

    actualizar\_maximo(vector\_computadoras[pos].paquetes,vector\_computadoras[pos].version,paquetes\_max,version\_max);

    if ((vector\_computadoras[pos].computadora>computadora\_corte1) and (vector\_computadoras[pos].computadora<computadora\_corte2)) then

    begin

      eliminar\_vector\_computadoras(vector\_computadoras,computadoras,pos);

      pos:=pos-1;

    end;

    pos:=pos+1;

  end;

  cuentas\_prom:=cuentas\_total/computadoras\_aux;

end;

var

  vector\_computadoras: t\_vector\_computadoras;

  computadoras, versiones\_corte: int16;

  cuentas\_prom: real;

  version\_max: string;

begin

  randomize;

  computadoras:=0;

  versiones\_corte:=0;

  cuentas\_prom:=0;

  version\_max:='';

  cargar\_vector\_computadoras(vector\_computadoras,computadoras);

  if (computadoras>0) then

  begin

    procesar\_vector\_computadoras(vector\_computadoras,computadoras,versiones\_corte,cuentas\_prom,version\_max);

    textcolor(green); write('La cantidad de computadoras que utilizan las versiones '); textcolor(yellow); write(version\_corte1); textcolor(green); write(' o '); textcolor(yellow); write(version\_corte2); textcolor(green); write(' es '); textcolor(red); writeln(versiones\_corte);

    textcolor(green); write('El promedio de cuentas de usuario por computadora es '); textcolor(red); writeln(cuentas\_prom:0:2);

    textcolor(green); write('La versión de Ubuntu de la computadora con mayor cantidad de paquetes instalados es '); textcolor(red); write(version\_max);

  end;

end.

**Ejercicio 7.**

*Continuando con los 3 ejercicios adicionales de la Guía opcional de actividades adicionales, ahora, se utilizarán vectores para almacenar la información ingresada por teclado. Consideraciones importantes:*

* *Los datos ingresados por teclado se deberán almacenar en una estructura de tipo vector apropiada. Dado que, en ninguno de los ejercicios se indica la cantidad máxima de datos a leer, para poder utilizar un vector, asumir que, en todos los casos, se ingresarán, a lo sumo, 5000 datos (donde cada dato será, o bien, una inversión, un alumno o un tanque de agua, según lo indica cada ejercicio).*
* *Una vez leídos y almacenados los datos, deberán procesarse (recorrer el vector) para resolver cada inciso. Al hacerlo, deberán reutilizarse los módulos ya implementados en la práctica anterior. En la medida de lo posible, el vector deberá recorrerse una única vez para resolver todos los incisos.*

Ejercicio 1:

program TP4\_E7a;

*{$codepage UTF8}*

uses crt;

const

  empresa\_salida=100;

  monto\_corte=50000.0;

  empresas\_total=5000;

type

  t\_empresa=1..empresas\_total;

  t\_registro\_empresa=record

    empresa: int16;

    inversiones: int16;

    monto\_total: real;

  end;

  t\_vector\_empresas=array[t\_empresa] of t\_registro\_empresa;

procedure leer\_inversiones(empresa, inversiones: int16; var monto\_total: real);

var

  i: int16;

  monto: real;

begin

  monto\_total:=0;

  for i:= 1 to inversiones do

  begin

    monto:=1+random(1000);

    monto\_total:=monto\_total+monto;

  end;

end;

procedure leer\_empresa(var registro\_empresa: t\_registro\_empresa);

var

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_empresa.empresa:=empresa\_salida

  else

    registro\_empresa.empresa:=1+random(high(int16));

  registro\_empresa.inversiones:=1+random(1000);

  leer\_inversiones(registro\_empresa.empresa,registro\_empresa.inversiones,registro\_empresa.monto\_total);

end;

procedure cargar\_vector\_empresas(var vector\_empresas: t\_vector\_empresas; var empresas: int16);

var

  registro\_empresa: t\_registro\_empresa;

begin

  repeat

    leer\_empresa(registro\_empresa);

    empresas:=empresas+1;

    vector\_empresas[empresas]:=registro\_empresa;

  until (vector\_empresas[empresas].empresa=empresa\_salida);

end;

procedure calcular\_a(empresa, inversiones: int16; monto\_total: real);

begin

  textcolor(green); write('El monto promedio de las inversiones de la empresa '); textcolor(yellow); write(empresa); textcolor(green); write(' es '); textcolor(red); writeln(monto\_total/inversiones:0:2);

end;

procedure calcular\_b(monto\_total: real; empresa: int16; var monto\_max: real; var empresa\_max: int16);

begin

  if (monto\_total>monto\_max) then

  begin

    monto\_max:=monto\_total;

    empresa\_max:=empresa;

  end;

end;

procedure calcular\_c(monto\_total: real; var empresas\_corte: int16);

begin

  if (monto\_total>monto\_corte) then

    empresas\_corte:=empresas\_corte+1;

end;

procedure procesar\_vector\_empresas(vector\_empresas: t\_vector\_empresas; empresas: int16; var empresa\_max, empresas\_corte: int16);

var

  i: t\_empresa;

  monto\_max: real;

begin

  monto\_max:=-9999999;

  for i:= 1 to empresas do

    if (vector\_empresas[i].inversiones>0) then

    begin

      calcular\_a(vector\_empresas[i].empresa,vector\_empresas[i].inversiones,vector\_empresas[i].monto\_total);

      calcular\_b(vector\_empresas[i].monto\_total,vector\_empresas[i].empresa,monto\_max,empresa\_max);

      calcular\_c(vector\_empresas[i].monto\_total,empresas\_corte);

    end;

end;

var

  vector\_empresas: t\_vector\_empresas;

  empresas, empresa\_max, empresas\_corte: int16;

begin

  randomize;

  empresas:=0;

  empresa\_max:=0;

  empresas\_corte:=0;

  cargar\_vector\_empresas(vector\_empresas,empresas);

  procesar\_vector\_empresas(vector\_empresas,empresas,empresa\_max,empresas\_corte);

  textcolor(green); write('El código de la empresa con mayor monto total invertido es '); textcolor(red); writeln(empresa\_max);

  textcolor(green); write('La cantidad de empresas con inversiones de más de $'); textcolor(yellow); write(monto\_corte:0:2); textcolor(green); write(' es '); textcolor(red); write(empresas\_corte);

end.

Ejercicio 2:

program TP4\_E7b;

*{$codepage UTF8}*

uses crt;

const

  condicion\_i='I'; condicion\_r='R';

  autoeva\_total=5;

  nota\_incumple=-1;

  legajo\_salida=-1;

  nota\_corte=4;

  promedio\_corte=6.5;

  nota\_cero=0;

  nota\_diez=10;

  presente\_corte=0.75;

  alumnos\_total=5000;

type

  t\_alumno=1..alumnos\_total;

  t\_registro\_alumno=record

    legajo: int16;

    condicion: char;

    presente: int8;

    nota\_total: int8;

    notas\_cero: int8;

    notas\_diez: int8;

  end;

  t\_vector\_alumnos=array[t\_alumno] of t\_registro\_alumno;

procedure leer\_notas(var presente, nota\_total, notas\_cero, notas\_diez: int8);

var

  i, nota: int8;

begin

  presente:=0; nota\_total:=0; notas\_cero:=0; notas\_diez:=0;

  for i:= 1 to autoeva\_total do

  begin

    nota:=nota\_incumple+random(12);

    if ((nota<>nota\_incumple) and (nota>=nota\_corte)) then

      presente:=presente+1;

    if (nota<>nota\_incumple) then

      nota\_total:=nota\_total+nota;

    if (nota=nota\_cero) then

      notas\_cero:=notas\_cero+1;

    if (nota=nota\_diez) then

      notas\_diez:=notas\_diez+1;

  end;

end;

procedure leer\_alumno(var registro\_alumno: t\_registro\_alumno);

var

  vector\_condiciones: array[1..2] of char=(condicion\_i, condicion\_r);

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_alumno.legajo:=legajo\_salida

  else

    registro\_alumno.legajo:=1+random(high(int16));

  if (registro\_alumno.legajo<>legajo\_salida) then

  begin

    registro\_alumno.condicion:=vector\_condiciones[1+random(2)];

    leer\_notas(registro\_alumno.presente,registro\_alumno.nota\_total,registro\_alumno.notas\_cero,registro\_alumno.notas\_diez);

  end;

end;

procedure cargar\_vector\_alumnos(var vector\_alumnos: t\_vector\_alumnos; var alumnos: int16);

var

  registro\_alumno: t\_registro\_alumno;

begin

  leer\_alumno(registro\_alumno);

  while (registro\_alumno.legajo<>legajo\_salida) do

  begin

    alumnos:=alumnos+1;

    vector\_alumnos[alumnos]:=registro\_alumno;

    leer\_alumno(registro\_alumno);

  end;

end;

procedure calcular\_ab(condicion: char; presente: int8; var ingresantes\_total, ingresantes\_parcial, recursantes\_total, recursantes\_parcial: int16);

begin

  if (condicion=condicion\_i) then

  begin

    if (presente>=presente\_corte\*autoeva\_total) then

      ingresantes\_parcial:=ingresantes\_parcial+1;

    ingresantes\_total:=ingresantes\_total+1;

  end

  else

  begin

    if (presente>=presente\_corte\*autoeva\_total) then

      recursantes\_parcial:=recursantes\_parcial+1;

    recursantes\_total:=recursantes\_total+1;

  end;

end;

procedure calcular\_c(presente: int8; var alumnos\_autoeva: int16);

begin

  if (presente=autoeva\_total) then

    alumnos\_autoeva:=alumnos\_autoeva+1;

end;

procedure calcular\_d(nota\_total: int8; var alumnos\_corte: int16);

begin

  if (nota\_total/autoeva\_total>promedio\_corte) then

    alumnos\_corte:=alumnos\_corte+1;

end;

procedure calcular\_e(notas\_cero: int8; var alumnos\_cero: int16);

begin

  if (notas\_cero>=1) then

    alumnos\_cero:=alumnos\_cero+1;

end;

procedure calcular\_f(notas\_diez: int8; legajo: int16; var notas\_diez\_max1, notas\_diez\_max2: int8; var legajo\_diez\_max1, legajo\_diez\_max2: int16);

begin

  if (notas\_diez>notas\_diez\_max1) then

  begin

    notas\_diez\_max2:=notas\_diez\_max1;

    legajo\_diez\_max2:=legajo\_diez\_max1;

    notas\_diez\_max1:=notas\_diez;

    legajo\_diez\_max1:=legajo;

  end

  else

    if (notas\_diez>notas\_diez\_max2) then

    begin

      notas\_diez\_max2:=notas\_diez;

      legajo\_diez\_max2:=legajo;

    end;

end;

procedure calcular\_g(notas\_cero: int8; legajo: int16; var notas\_cero\_max1, notas\_cero\_max2: int8; var legajo\_cero\_max1, legajo\_cero\_max2: int16);

begin

  if (notas\_cero>notas\_cero\_max1) then

  begin

    notas\_cero\_max2:=notas\_cero\_max1;

    legajo\_cero\_max2:=legajo\_cero\_max1;

    notas\_cero\_max1:=notas\_cero;

    legajo\_cero\_max1:=legajo;

  end

  else

    if (notas\_cero>notas\_cero\_max2) then

    begin

      notas\_cero\_max2:=notas\_cero;

      legajo\_cero\_max2:=legajo;

    end;

end;

procedure procesar\_vector\_alumnos(vector\_alumnos: t\_vector\_alumnos; alumnos: int16; var ingresantes\_parcial, ingresantes\_total, recursantes\_parcial, recursantes\_total, alumnos\_autoeva, alumnos\_corte, alumnos\_cero, legajo\_diez\_max1, legajo\_diez\_max2, legajo\_cero\_max1, legajo\_cero\_max2: int16);

var

  i: t\_alumno;

  notas\_diez\_max1, notas\_diez\_max2, notas\_cero\_max1, notas\_cero\_max2: int8;

begin

  notas\_diez\_max1:=0; notas\_diez\_max2:=0;

  notas\_cero\_max1:=0; notas\_cero\_max2:=0;

  for i:= 1 to alumnos do

  begin

    calcular\_ab(vector\_alumnos[i].condicion,vector\_alumnos[i].presente,ingresantes\_total,ingresantes\_parcial,recursantes\_total,recursantes\_parcial);

    calcular\_c(vector\_alumnos[i].presente,alumnos\_autoeva);

    calcular\_d(vector\_alumnos[i].nota\_total,alumnos\_corte);

    calcular\_e(vector\_alumnos[i].notas\_cero,alumnos\_cero);

    calcular\_f(vector\_alumnos[i].notas\_diez,vector\_alumnos[i].legajo,notas\_diez\_max1,notas\_diez\_max2,legajo\_diez\_max1,legajo\_diez\_max2);

    calcular\_g(vector\_alumnos[i].notas\_cero,vector\_alumnos[i].legajo,notas\_cero\_max1,notas\_cero\_max2,legajo\_cero\_max1,legajo\_cero\_max2);

  end;

end;

var

  vector\_alumnos: t\_vector\_alumnos;

  alumnos, ingresantes\_parcial, ingresantes\_total, recursantes\_parcial, recursantes\_total, alumnos\_autoeva, alumnos\_corte, alumnos\_cero, legajo\_diez\_max1, legajo\_diez\_max2, legajo\_cero\_max1, legajo\_cero\_max2: int16;

begin

  randomize;

  alumnos:=0;

  ingresantes\_parcial:=0; ingresantes\_total:=0;

  recursantes\_parcial:=0; recursantes\_total:=0;

  alumnos\_autoeva:=0;

  alumnos\_corte:=0;

  alumnos\_cero:=0;

  legajo\_diez\_max1:=0; legajo\_diez\_max2:=0;

  legajo\_cero\_max1:=0; legajo\_cero\_max2:=0;

  cargar\_vector\_alumnos(vector\_alumnos,alumnos);

  if (alumnos>0) then

  begin

    procesar\_vector\_alumnos(vector\_alumnos,alumnos,ingresantes\_parcial,ingresantes\_total,recursantes\_parcial,recursantes\_total,alumnos\_autoeva,alumnos\_corte,alumnos\_cero,legajo\_diez\_max1,legajo\_diez\_max2,legajo\_cero\_max1,legajo\_cero\_max2);

    if (ingresantes\_total>0) then

    begin

      textcolor(green); write('La cantidad de alumnos INGRESANTES en condiciones de rendir el parcial y el porcentaje sobre el total de alumnos INGRESANTES son '); textcolor(red); write(ingresantes\_parcial); textcolor(green); write(' y '); textcolor(red); write(ingresantes\_parcial/ingresantes\_total\*100:0:2); textcolor(green); writeln('%, respectivamente');

    end

    else

    begin

      textcolor(red); writeln('No hay alumnos INGRESANTES (I)');

    end;

    if (recursantes\_total>0) then

    begin

      textcolor(green); write('La cantidad de alumnos RECURSANTES en condiciones de rendir el parcial y el porcentaje sobre el total de alumnos RECURSANTES son '); textcolor(red); write(recursantes\_parcial); textcolor(green); write(' y '); textcolor(red); write(recursantes\_parcial/recursantes\_total\*100:0:2); textcolor(green); writeln('%, respectivamente');

    end

    else

    begin

      textcolor(red); writeln('No hay alumnos RECURSANTES (R)');

    end;

    textcolor(green); write('La cantidad de alumnos que aprobaron todas las autoevaluaciones es '); textcolor(red); writeln(alumnos\_autoeva);

    textcolor(green); write('La cantidad de alumnos cuya nota promedio fue mayor a '); textcolor(yellow); write(promedio\_corte:0:2); textcolor(green); write(' puntos es '); textcolor(red); writeln(alumnos\_corte);

    textcolor(green); write('La cantidad de alumnos que obtuvieron cero puntos en, al menos, una autoevaluación es '); textcolor(red); writeln(alumnos\_cero);

    textcolor(green); write('Los legajos de los dos alumnos con mayor cantidad de autoevaluaciones con nota 10 (diez) son '); textcolor(red); write(legajo\_diez\_max1); textcolor(green); write(' y '); textcolor(red); writeln(legajo\_diez\_max2);

    textcolor(green); write('Los legajos de los dos alumnos con mayor cantidad de autoevaluaciones con nota 0 (cero) son '); textcolor(red); write(legajo\_cero\_max1); textcolor(green); write(' y '); textcolor(red); write(legajo\_cero\_max2);

  end

  else

  begin

    textcolor(red); write('No hay alumnos INGRESANTES (I) o RECURSANTES (R)');

  end;

end.

Ejercicio 3:

program TP4\_E7c;

*{$codepage UTF8}*

uses crt;

const

  tanque\_r='R'; tanque\_c='C';

  tanque\_salida='Z';

  alto\_corte=1.40;

  volumen\_corte=800.0;

  tanques\_total=5000;

type

  t\_tanque=1..tanques\_total;

  t\_registro\_tanque=record

    tanque: char;

    ancho: real;

    largo: real;

    alto: real;

    radio: real;

    volumen: real;

  end;

  t\_vector\_tanques=array[t\_tanque] of t\_registro\_tanque;

procedure leer\_tanque(var registro\_tanque: t\_registro\_tanque);

var

  vector\_tanques: array[1..2] of char=(tanque\_r, tanque\_c);

  i: int8;

begin

  i:=random(100);

  if (i=0) then

    registro\_tanque.tanque:=tanque\_salida

  else

    registro\_tanque.tanque:=vector\_tanques[1+random(2)];

  if (registro\_tanque.tanque<>tanque\_salida) then

  begin

    if (registro\_tanque.tanque=tanque\_r) then

    begin

      registro\_tanque.ancho:=1+random(391)/10;

      registro\_tanque.largo:=1+random(391)/10;

      registro\_tanque.alto:=1+random(21)/10;

      registro\_tanque.volumen:=registro\_tanque.ancho\*registro\_tanque.largo\*registro\_tanque.alto;

      registro\_tanque.radio:=-1;

    end

    else

    begin

      registro\_tanque.radio:=1+random(391)/10;

      registro\_tanque.alto:=1+random(21)/10;

      registro\_tanque.volumen:=pi\*registro\_tanque.radio\*registro\_tanque.radio\*registro\_tanque.alto;

      registro\_tanque.ancho:=-1;

      registro\_tanque.largo:=-1;

    end;

  end;

end;

procedure cargar\_vector\_tanques(var vector\_tanques: t\_vector\_tanques; var tanques: int16);

var

  registro\_tanque: t\_registro\_tanque;

begin

  leer\_tanque(registro\_tanque);

  while (registro\_tanque.tanque<>tanque\_salida) do

  begin

    tanques:=tanques+1;

    vector\_tanques[tanques]:=registro\_tanque;

    leer\_tanque(registro\_tanque);

  end;

end;

procedure calcular\_a(volumen: real; var volumen\_max1, volumen\_max2: real);

begin

  if (volumen>volumen\_max1) then

  begin

    volumen\_max2:=volumen\_max1;

    volumen\_max1:=volumen;

  end

  else

    if (volumen>volumen\_max2) then

      volumen\_max2:=volumen;

end;

procedure calcular\_bc(tanque: char; volumen: real; var volumen\_total\_c, volumen\_total\_r: real; var tanques\_c, tanques\_r: int16);

begin

  if (tanque=tanque\_c) then

  begin

    volumen\_total\_c:=volumen\_total\_c+volumen;

    tanques\_c:=tanques\_c+1;

  end

  else

  begin

    volumen\_total\_r:=volumen\_total\_r+volumen;

    tanques\_r:=tanques\_r+1;

  end;

end;

procedure calcular\_d(alto: real; var tanques\_corte\_alto: int16);

begin

  if (alto<alto\_corte) then

    tanques\_corte\_alto:=tanques\_corte\_alto+1;

end;

procedure calcular\_e(volumen: real; var tanques\_corte\_volumen: int16);

begin

  if (volumen<volumen\_corte) then

    tanques\_corte\_volumen:=tanques\_corte\_volumen+1;

end;

procedure procesar\_vector\_tanques(vector\_tanques: t\_vector\_tanques; tanques: int16; var volumen\_max1, volumen\_max2, volumen\_total\_c, volumen\_total\_r: real; var tanques\_c, tanques\_r, tanques\_corte\_alto, tanques\_corte\_volumen: int16);

var

  i: t\_tanque;

begin

  for i:= 1 to tanques do

  begin

    calcular\_a(vector\_tanques[i].volumen,volumen\_max1,volumen\_max2);

    calcular\_bc(vector\_tanques[i].tanque,vector\_tanques[i].volumen,volumen\_total\_c,volumen\_total\_r,tanques\_c,tanques\_r);

    calcular\_d(vector\_tanques[i].alto,tanques\_corte\_alto);

    calcular\_e(vector\_tanques[i].volumen,tanques\_corte\_volumen);

  end;

end;

var

  vector\_tanques: t\_vector\_tanques;

  tanques, tanques\_c, tanques\_r, tanques\_corte\_alto, tanques\_corte\_volumen: int16;

  volumen\_max1, volumen\_max2, volumen\_total\_c, volumen\_total\_r: real;

begin

  randomize;

  tanques:=0;

  volumen\_max1:=0; volumen\_max2:=0;

  tanques\_c:=0; volumen\_total\_c:=0;

  tanques\_r:=0; volumen\_total\_r:=0;

  tanques\_corte\_alto:=0;

  tanques\_corte\_volumen:=0;

  cargar\_vector\_tanques(vector\_tanques,tanques);

  if (tanques>0) then

  begin

    procesar\_vector\_tanques(vector\_tanques,tanques,volumen\_max1,volumen\_max2,volumen\_total\_c,volumen\_total\_r,tanques\_c,tanques\_r,tanques\_corte\_alto,tanques\_corte\_volumen);

    textcolor(green); write('El volumen de los mayores tanques vendidos es '); textcolor(red); write(volumen\_max1:0:2); textcolor(green); write(' y '); textcolor(red); writeln(volumen\_max2:0:2);

    if (tanques\_c>0) then

    begin

      textcolor(green); write('El volumen promedio de todos los tanques cilíndricos (C) vendidos es '); textcolor(red); writeln(volumen\_total\_c/tanques\_c:0:2);

    end

    else

    begin

      textcolor(red); writeln('No hay tanques cilíndricos (C) vendidos');

    end;

    if (tanques\_r>0) then

    begin

      textcolor(green); write('El volumen promedio de todos los tanques rectangulares (R) vendidos es '); textcolor(red); writeln(volumen\_total\_r/tanques\_r:0:2);

    end

    else

    begin

      textcolor(red); writeln('No hay tanques rectangulares (R) vendidos');

    end;

    textcolor(green); write('La cantidad de tanques cuyo alto es menor a '); textcolor(yellow); write(alto\_corte:0:2); textcolor(green); write(' metros es '); textcolor(red); writeln(tanques\_corte\_alto);

    textcolor(green); write('La cantidad de tanques cuyo volumen es menor a '); textcolor(yellow); write(volumen\_corte:0:2); textcolor(green); write(' metros cúbicos es '); textcolor(red); write(tanques\_corte\_volumen);

  end

  else

  begin

    textcolor(red); write('No hay tanques cilíndricos (C) o rectangulares (R) vendidos');

  end;

end.