Juan Menduiña

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(ve
ctor_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_cantidades=array[t_dia] of int16;
procedure inicializar_vector_cantidades(var vector_cantidades: t_vector_cantidades);
var
  i: t_dia;
begin
  for i:= dia_ini to dia_fin do
    vector_cantidades[i]:=0;
end;
procedure leer_viaje(var registro_viaje: t_registro_viaje);
var
  i: int8;
begin
  i:=random(101);
  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_cantidades: t_vector_cantidades;
  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_cantidades(vector_cantidades);
  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,m
onto_min,dia_min,distancia_min);
    vector_cantidades[vector_viajes[i].dia]:=vector_cantidades[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_cantidades[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,num
ero));
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_ciuda
des1,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('Los códigos de las 10 ciudades con mayor cantidad de
clientes son '); 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_computa
dora.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,paque
tes_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.mont
o_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_m
ax);
      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,ingre
santes_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_die
z_max2,legajo_diez_max1,legajo_diez_max2);
    calcular_g(vector_alumnos[i].notas_cero,vector_alumnos[i].legajo,notas_cero_max1,notas_cer
o_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,recur
santes_parcial,recursantes_total,alumnos_autoeva,alumnos_corte,alumnos_cero,legajo_diez_max1,l
egajo_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.alt
o;
      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_tot
al_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,v
olumen_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.
```