Juan Menduiña

Trabajo Práctico N° 6:
Listas.

Ejercicio 1.

Dado el siguiente programa:

```
program TP6_E1;
{$codepage UTF8}
uses crt;
type
  lista=^nodo;
  nodo=record
    num: integer;
    sig: lista;
  end;
procedure armarNodo(var L: lista; v: integer);
var
  aux: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=L;
  L:=aux;
end;
var
  pri: lista;
  valor: integer;
begin
  pri:=nil;
  writeln('Ingrese un número');
  read(valor);
  while (valor<>0) do
  begin
    armarNodo(pri,valor);
    writeln('Ingrese un número');
    read(valor);
  end;
end.
```

(a) *Indicar qué hace el programa.*

El programa agrega números enteros a la lista *pri* hasta leer el número 0.

(b) *Indicar cómo queda conformada la lista si se lee la siguiente secuencia de números: 10 21 13 48 0.*

Si se lee la secuencia de números enteros 10, 21, 13, 48, 0, la lista queda conformada con 48, 13, 21, 10.

(c) *Implementar un módulo que imprima los números enteros guardados en la lista generada.*

(d) *Implementar un módulo que reciba la lista y un valor, e incremente con ese valor cada dato de la lista.*

```
program TP6_E1;
{$codepage UTF8}
uses crt;
type
  lista=^nodo;
  nodo=record
    num: integer;
    sig: lista;
  end;
procedure armarNodo(var L: lista; v: integer);
var
  aux: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=L;
  L:=aux;
end;
procedure imprimir_lista(L: lista);
var
  i: int16;
begin
  i:=0;
  while (L<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('Elemento ',i,' de la lista: '); textcolor(yellow);
writeln(L^.num);
    L:=L^.sig;
  end;
end;
procedure modificar_lista(var L: lista; valor: int16);
var
  aux: lista;
begin
  aux:=L;
  while (aux<>nil) do
  begin
    aux^.num:=aux^.num+valor;
    aux:=aux^.sig;
  end;
end;
var
  vector_numeros: array[1..5] of integer=(10, 21, 13, 48, 0);
  pri: lista;
  pos, valor: integer;
begin
  randomize;
  pri:=nil;
  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
  pos:=1;
  valor:=vector_numeros[pos];
  while (valor<>0) do
  begin
    armarNodo(pri,valor);
    pos:=pos+1;
    valor:=vector_numeros[pos];
  end;
  if (pri<>nil) then
  begin
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
```

```
    imprimir_lista(pri);
    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();
    valor:=1+random(100);
    modificar_lista(pri,valor);
    imprimir_lista(pri);
  end;
end.
```

Ejercicio 2.

Dado el siguiente código que lee información de personas hasta que se ingresa la persona con DNI 0 y, luego, imprime dicha información en el orden inverso al que fue leída, identificar los 9 errores.

Con errores:

```
program TP6_E2;
{$codepage UTF8}
uses crt;
type
  lista=^nodo;
  persona=record
    dni: integer;
    nombre: string;
    apellido: string;
  end;
  nodo=record
    dato: persona;
    sig: lista;
  end;
procedure leerPersona(p: persona);
begin
  read(p.dni);
  if (p.dni<>0) then
  begin
    read(p.nombre);
    read(p.apellido);
  end;
end;
{Agrega un nodo a la lista}
procedure agregarAdelante(l: lista; p: persona);
var
  aux: lista;
begin
  aux^.dato:=p;
  aux^.sig:=l;
  l:=aux;
end;
{Carga la lista hasta que llega el dni 0}
procedure generarLista(var l:lista);
var
  p: nodo;
begin
  leerPersona(p);
  while (p.dni<>0) do
  begin
    agregarAdelante(l,p);
  end;
end;
procedure imprimirInformacion(var l: lista);
begin
  while (l<>nil) do
  begin
    writeln('DNI: ',l^.dato.dni,'Nombre: ',l^.nombre,'Apellido: ',l^.apellido);
    l:=l^.sig;
  end;
end;
{Programa principal}
var
  l: lista;
begin
```

```
  generarLista(l);
  imprimirInformacion(l);
end.
```

Sin errores:

```
program TP6_E2;
{$codepage UTF8}
uses crt;
type
  lista=^nodo;
  persona=record
    dni: int32;
    nombre: string;
    apellido: string;
  end;
  nodo=record
    dato: persona;
    sig: lista;
  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 leerPersona(var p: persona);
var
  i: int8;
begin
  i:=random(100);
  if (i=0) then
    p.dni:=0
  else
    p.dni:=10000000+random(40000001);
  if (p.dni<>0) then
  begin
    p.nombre:=random_string(5+random(6));
    p.apellido:=random_string(5+random(6));
  end;
end;
procedure agregarAdelante(var l: lista; p: persona);
var
  aux: lista;
begin
  new(aux);
  aux^.dato:=p;
  aux^.sig:=l;
  l:=aux;
end;
procedure generarLista(var l: lista);
var
  p: persona;
begin
  leerPersona(p);
  while (p.dni<>0) do
  begin
    agregarAdelante(l,p);
    leerPersona(p);
  end;
```

```
end;
procedure imprimirInformacion(l: lista);
begin
  while (l<>nil) do
  begin
    writeln('DNI: ',l^.dato.dni,'; Nombre: ',l^.dato.nombre,'; Apellido: ',l^.dato.apellido);
    l:=l^.sig;
  end;
end;
var
  l: lista;
begin
  randomize;
  l:=nil;
  generarLista(l);
  if (l<>nil) then
    imprimirInformacion(l);
end.
```

Los 9 errores que existen en el programa son:

1. En el *procedure "leerPersona"*, el parámetro "*p*" debe ser por referencia.
2. En el *procedure "agregarAdelante"*, el parámetro *"l"* debe ser por referencia.
3. En el *procedure "agregarAdelante"*, falta el *new(aux)*;
4. En el *procedure "generarLista"*, la variable local al proceso *"p"* debe ser de tipo *persona*;
5. En el *procedure "generarLista"*, falta el *leerPersona(p)* al final del *while*;
6. En el *procedure "imprimirInformacion"*, el parámetro *"l"* debe ser por valor.
7. En el *procedure "imprimirInformacion"*, en el *write*, el acceso al elemento *nombre* del registro *persona* de la lista debe ser *l^.dato.nombre*.
8. En el *procedure "imprimirInformacion"*, en el *write*, el acceso al elemento *apellido* del registro *persona* de la lista debe ser *l^.dato.apellido*.
9. En el programa principal, falta inicializar la variable *"l"*.

Ejercicio 3.

Utilizando el programa del Ejercicio 1, realizar los siguientes cambios:

(a) *Modificar el módulo armarNodo para que los elementos se guarden en la lista en el orden en que fueron ingresados (agregar atrás).*

(b) *Modificar el módulo armarNodo para que los elementos se guarden en la lista en el orden en que fueron ingresados, manteniendo un puntero al último ingresado.*

```
program TP6_E3;
{$codepage UTF8}
uses crt;
type
  lista=^nodo;
  nodo=record
    num: integer;
    sig: lista;
  end;
procedure armarNodo1(var L: lista; v: integer);
var
  aux: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=L;
  L:=aux;
end;
procedure armarNodo2(var L: lista; v: integer);
var
  aux, ult: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=nil;
  if (L=nil) then
    L:=aux
  else
  begin
    ult:=L;
    while (ult^.sig<>nil) do
      ult:=ult^.sig;
    ult^.sig:=aux;
  end;
end;
procedure armarNodo3(var L, ult: lista; v: integer);
var
  aux: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=nil;
  if (L=nil) then
    L:=aux
  else
    ult^.sig:=aux;
  ult:=aux;
end;
procedure imprimir_lista(L: lista);
var
  i: int16;
begin
```

```
  i:=1;
  while (L<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('Elemento ',i,' de la lista: '); textcolor(yellow);
writeln(L^.num);
    L:=L^.sig;
  end;
end;
procedure modificar_lista(var L: lista; valor: int16);
var
  aux: lista;
begin
  aux:=L;
  while (aux<>nil) do
  begin
    aux^.num:=aux^.num+valor;
    aux:=aux^.sig;
  end;
end;
var
  vector_numeros: array[1..5] of integer=(10, 21, 13, 48, 0);
  pri, ult: lista;
  pos, valor: integer;
begin
  randomize;
  pri:=nil; ult:=nil;
  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();
  pos:=1;
  valor:=vector_numeros[pos];
  while (valor<>0) do
  begin
    //armarNodo1(pri,valor);
    armarNodo2(pri,valor);
    //armarNodo3(pri,ult,valor);
    pos:=pos+1;
    valor:=vector_numeros[pos];
  end;
  if (pri<>nil) then
  begin
    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();
    imprimir_lista(pri);
    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();
    valor:=1+random(100);
    modificar_lista(pri,valor);
    imprimir_lista(pri);
  end;
end.
```

Ejercicio 4.

Utilizando el programa del Ejercicio 1, realizar los siguientes módulos:

(a) *Máximo: recibe la lista como parámetro y retorna el elemento de valor máximo.*

(b) *Mínimo: recibe la lista como parámetro y retorna el elemento de valor mínimo.*

(c) *Múltiplos: recibe como parámetros la lista L y un valor entero A, y retorna la cantidad de elementos de la lista que son múltiplos de A.*

```
program TP6_E4;
{$codepage UTF8}
uses crt;
type
  lista=^nodo;
  nodo=record
    num: integer;
    sig: lista;
  end;
procedure armarNodo(var L: lista; v: integer);
var
  aux: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=L;
  L:=aux;
end;
procedure imprimir_lista(L: lista);
var
  i: int16;
begin
  i:=0;
  while (L<>nil) do
  begin
    i:=i+1;
    textcolor(green); write('Elemento ',i,' de la lista: '); textcolor(yellow);
writeln(L^.num);
    L:=L^.sig;
  end;
end;
procedure modificar_lista(var L: lista; valor: int16);
var
  aux: lista;
begin
  aux:=L;
  while (aux<>nil) do
  begin
    aux^.num:=aux^.num+valor;
    aux:=aux^.sig;
  end;
end;
function calcular_maximo(L: lista): integer;
var
  maximo: integer;
begin
  maximo:=low(integer);
  while (L<>nil) do
  begin
    if (L^.num>maximo) then
```

```
      maximo:=L^.num;
    L:=L^.sig;
  end;
  calcular_maximo:=maximo;
end;
function calcular_minimo(L: lista): integer;
var
  minimo: integer;
begin
  minimo:=high(integer);
  while (L<>nil) do
  begin
    if (L^.num<minimo) then
      minimo:=L^.num;
    L:=L^.sig;
  end;
  calcular_minimo:=minimo;
end;
function calcular_multiplos(L: lista; divisor: integer): integer;
var
  multiplos: integer;
begin
  multiplos:=0;
  while (L<>nil) do
  begin
    if (L^.num mod divisor=0) then
      multiplos:=multiplos+1;
    L:=L^.sig;
  end;
  calcular_multiplos:=multiplos;
end;
var
  vector_numeros: array[1..5] of integer=(10, 21, 13, 48, 0);
  pri: lista;
  pos, valor: integer;
begin
  randomize;
  pri:=nil;
  writeln(); textcolor(red); writeln('EJERCICIO 1. INCISO (b):'); writeln();
  pos:=1;
  valor:=vector_numeros[pos];
  while (valor<>0) do
  begin
    armarNodo(pri,valor);
    pos:=pos+1;
    valor:=vector_numeros[pos];
  end;
  if (pri<>nil) then
  begin
    writeln(); textcolor(red); writeln('EJERCICIO 1. INCISO (c):'); writeln();
    imprimir_lista(pri);
    writeln(); textcolor(red); writeln('EJERCICIO 1. INCISO (d):'); writeln();
    valor:=1+random(100);
    modificar_lista(pri,valor);
    imprimir_lista(pri);
    writeln(); textcolor(red); writeln('EJERCICIO 4. INCISO (a):'); writeln();
    textcolor(green); write('El elemento de valor máximo de la lista es '); textcolor(red);
writeln(calcular_maximo(pri));
    writeln(); textcolor(red); writeln('EJERCICIO 4. INCISO (b):'); writeln();
    textcolor(green); write('El elemento de valor mínimo de la lista es '); textcolor(red);
writeln(calcular_minimo(pri));
    writeln(); textcolor(red); writeln('EJERCICIO 4. INCISO (c):'); writeln();
    valor:=1+random(10);
    textcolor(green); write('La cantidad de elementos de la lista que son múltiplos de ');
textcolor(yellow); write(valor); textcolor(green); write(' es '); textcolor(red);
write(calcular_multiplos(pri,valor));
```

```
  end;
end.
```

Ejercicio 5.

Realizar un programa que lea y almacene la información de productos de un supermercado. De cada producto, se lee: código, descripción, stock actual, stock mínimo y precio. La lectura finaliza cuando se ingresa el código -1, que no debe procesarse. Una vez leída y almacenada toda la información, calcular e informar:

- *Porcentaje de productos con stock actual por debajo de su stock mínimo.*
- *Descripción de aquellos productos con código compuesto por, al menos, tres dígitos pares.*
- *Código de los dos productos más económicos.*

```
program TP6_E5;
{$codepage UTF8}
uses crt;
const
  producto_salida=-1;
  pares_corte=3;
type
  t_registro_producto=record
    producto: int16;
    descripcion: string;
    stock_actual: int16;
    stock_minimo: int16;
    precio: real;
  end;
  t_lista_productos=^t_nodo_productos;
  t_nodo_productos=record
    ele: t_registro_producto;
    sig: t_lista_productos;
  end;
procedure leer_producto(var registro_producto: t_registro_producto);
begin
  textcolor(green); write('Introducir código de producto del producto: ');
  textcolor(yellow); readln(registro_producto.producto);
  if (registro_producto.producto<>producto_salida) then
  begin
    textcolor(green); write('Introducir descripción del producto: ');
    textcolor(yellow); readln(registro_producto.descripcion);
    textcolor(green); write('Introducir stock actual del producto: ');
    textcolor(yellow); readln(registro_producto.stock_actual);
    textcolor(green); write('Introducir stock mínimo del producto: ');
    textcolor(yellow); readln(registro_producto.stock_minimo);
    textcolor(green); write('Introducir precio del producto: ');
    textcolor(yellow); readln(registro_producto.precio);
  end;
end;
procedure agregar_adelante_lista_productos(var lista_productos: t_lista_productos;
registro_producto: t_registro_producto);
var
  nuevo: t_lista_productos;
begin
  new(nuevo);
  nuevo^.ele:=registro_producto;
  nuevo^.sig:=lista_productos;
  lista_productos:=nuevo;
end;
procedure cargar_lista_productos(var lista_productos: t_lista_productos);
var
  registro_producto: t_registro_producto;
begin
```

```
  leer_producto(registro_producto);
  while (registro_producto.producto<>producto_salida) do
  begin
    agregar_adelante_lista_productos(lista_productos,registro_producto);
    leer_producto(registro_producto);
  end;
end;
function contar_pares(producto: int16): boolean;
var
  digito, pares: int8;
begin
  pares:=0;
  while ((producto<>0) and (pares<pares_corte)) do
  begin
    digito:=producto mod 10;
    if (digito mod 2=0) then
      pares:=pares+1;
    producto:=producto div 10;
  end;
  contar_pares:=(pares>=pares_corte);
end;
procedure actualizar_minimos(precio: real; producto: int16; var precio_min1, precio_min2:
real; var producto_min1, producto_min2: int16);
begin
  if (precio<precio_min1) then
  begin
    precio_min2:=precio_min1;
    producto_min2:=producto_min1;
    precio_min1:=precio;
    producto_min1:=producto;
  end
  else
    if (precio<precio_min2) then
    begin
      precio_min2:=precio;
      producto_min2:=producto;
    end;
end;
procedure procesar_lista_productos(lista_productos: t_lista_productos; var porcentaje_debajo:
real; var producto_min1, producto_min2: int16);
var
  productos_total, productos_debajo: int16;
  precio_min1, precio_min2: real;
begin
  productos_total:=0; productos_debajo:=0;
  precio_min1:=9999999; precio_min2:=9999999;
  while (lista_productos<>nil) do
  begin
    productos_total:=productos_total+1;
    if (lista_productos^.ele.stock_actual<lista_productos^.ele.stock_minimo) then
      productos_debajo:=productos_debajo+1;
    if (contar_pares(lista_productos^.ele.producto)=true) then
    begin
      textcolor(green); write('La descripción es este producto con código compuesto por, al
menos, tres dígitos pares es '); textcolor(red); writeln(lista_productos^.ele.descripcion);
    end;
    actualizar_minimos(lista_productos^.ele.precio,lista_productos^.ele.producto,precio_min1,p
recio_min2,producto_min1,producto_min2);
    lista_productos:=lista_productos^.sig;
  end;
  porcentaje_debajo:=productos_debajo/productos_total*100;
end;
var
  lista_productos: t_lista_productos;
  producto_min1, producto_min2: int16;
  porcentaje_debajo: real;
```

```
begin
  lista_productos:=nil;
  porcentaje_debajo:=0;
  producto_min1:=0; producto_min2:=0;
  cargar_lista_productos(lista_productos);
  if (lista_productos<>nil) then
  begin
    procesar_lista_productos(lista_productos,porcentaje_debajo,producto_min1,producto_min2);
    textcolor(green); write('El porcentaje de productos con stock actual por debajo de su
stock mínimo es '); textcolor(red); write(porcentaje_debajo:0:2); textcolor(green);
writeln('%');
    textcolor(green); write('Los códigos de los dos productos más econónomicos son ');
textcolor(red); write(producto_min1); textcolor(green); write(' y '); textcolor(red);
write(producto_min2); textcolor(green); write(', respectivamente');
  end;
end.
```

Ejercicio 6.

La Agencia Espacial Europea (ESA) está realizando un relevamiento de todas las sondas espaciales lanzadas al espacio en la última década. De cada sonda, se conoce su nombre, duración estimada de la misión (cantidad de meses que permanecerá activa), el costo de construcción, el costo de mantenimiento mensual y el rango del espectro electromagnético sobre el que realizará estudios. Dicho rango se divide en 7 categorías: 1. radio; 2. microondas; 3. infrarrojo; 4. luz visible; 5. ultravioleta; 6. rayos X; 7. rayos gamma. Realizar un programa que lea y almacene la información de todas las sondas espaciales. La lectura finaliza al ingresar la sonda llamada "GAIA", que debe procesarse. Una vez finalizada la lectura, informar:

- *El nombre de la sonda más costosa (considerando su costo de construcción y de mantenimiento).*
- *La cantidad de sondas que realizarán estudios en cada rango del espectro electromagnético.*
- *La cantidad de sondas cuya duración estimada supera la duración promedio de todas las sondas.*
- *El nombre de las sondas cuyo costo de construcción supera el costo promedio entre todas las sondas.*

Nota: Para resolver los incisos, la lista debe recorrerse la menor cantidad de veces posible.

```
program TP6_E6;
{$codepage UTF8}
uses crt;
const
  rango_ini=1; rango_fin=7;
  nombre_salida='GAIA';
type
  t_rango=rango_ini..rango_fin;
  t_registro_sonda=record
    nombre: string;
    duracion: int16;
    costo_construccion: real;
    costo_mantenimiento: real;
    rango: t_rango;
  end;
  t_lista_sondas=^t_nodo_sondas;
  t_nodo_sondas=record
    ele: t_registro_sonda;
    sig: t_lista_sondas;
  end;
  t_vector_rangos=array[t_rango] of int16;
procedure inicializar_vector_rangos(var vector_rangos: t_vector_rangos);
var
  i: t_rango;
begin
  for i:= rango_ini to rango_fin do
    vector_rangos[i]:=0;
end;
procedure leer_sonda(var registro_sonda: t_registro_sonda);
begin
  textcolor(green); write('Introducir nombre de la sonda: ');
  textcolor(yellow); readln(registro_sonda.nombre);
```

```
  textcolor(green); write('Introducir duración estimada de la misión de la sonda (cantidad de
meses que permanecerá activa): ');
  textcolor(yellow); readln(registro_sonda.duracion);
  textcolor(green); write('Introducir costo de construcción de la sonda: ');
  textcolor(yellow); readln(registro_sonda.costo_construccion);
  textcolor(green); write('Introducir costo de mantenimiento mensual de la sonda: ');
  textcolor(yellow); readln(registro_sonda.costo_mantenimiento);
  textcolor(green); write('Introducir rango del espectro electromagnético sobre el que
realizará estudios la sonda (1. radio; 2. microondas; 3. infrarrojo; 4. luz visible; 5.
ultravioleta; 6. rayos X; 7. rayos gamma): ');
  textcolor(yellow); readln(registro_sonda.rango);
end;
procedure agregar_adelante_lista_sondas(var lista_sondas: t_lista_sondas; registro_sonda:
t_registro_sonda);
var
  nuevo: t_lista_sondas;
begin
  new(nuevo);
  nuevo^.ele:=registro_sonda;
  nuevo^.sig:=lista_sondas;
  lista_sondas:=nuevo;
end;
procedure cargar_lista_sondas(var lista_sondas: t_lista_sondas; var duracion_prom, costo_prom:
real);
var
  registro_sonda: t_registro_sonda;
  sondas_total: int16;
  duracion_total, costo_total: real;
begin
  duracion_total:=0; sondas_total:=0;
  costo_total:=0;
  repeat
    leer_sonda(registro_sonda);
    agregar_adelante_lista_sondas(lista_sondas,registro_sonda);
    duracion_total:=duracion_total+lista_sondas^.ele.duracion;
    sondas_total:=sondas_total+1;
    costo_total:=costo_total+lista_sondas^.ele.costo_construccion;
  until (registro_sonda.nombre=nombre_salida);
  duracion_prom:=duracion_total/sondas_total;
  costo_prom:=costo_total/sondas_total;
end;
procedure actualizar_maximo(costo: real; nombre: string; var costo_max: real; var nombre_max:
string);
begin
  if (costo>costo_max) then
  begin
    costo_max:=costo;
    nombre_max:=nombre;
  end;
end;
procedure procesar_lista_sondas(lista_sondas: t_lista_sondas; duracion_prom, costo_prom: real;
var nombre_max: string; var vector_rangos: t_vector_rangos; var sondas_prom: int16);
var
  costo_sonda, costo_max: real;
begin
  costo_max:=-9999999;
  while (lista_sondas<>nil) do
  begin
    costo_sonda:=lista_sondas^.ele.costo_construccion+lista_sondas^.ele.costo_mantenimiento*li
sta_sondas^.ele.duracion;
    actualizar_maximo(costo_sonda,lista_sondas^.ele.nombre,costo_max,nombre_max);
    vector_rangos[lista_sondas^.ele.rango]:=vector_rangos[lista_sondas^.ele.rango]+1;
    if (lista_sondas^.ele.duracion>duracion_prom) then
      sondas_prom:=sondas_prom+1;
    if (lista_sondas^.ele.costo_construccion>costo_prom) then
    begin
```

```
      textcolor(green); write('El nombre de esta sonda cuyo costo de construcción supera el
costo promedio entre todas las sondas es '); textcolor(red);
writeln(lista_sondas^.ele.nombre);
    end;
    lista_sondas:=lista_sondas^.sig;
  end;
end;
procedure imprimir_vector_rangos(vector_rangos: t_vector_rangos);
var
  i: t_rango;
begin
  for i:= rango_ini to rango_fin do
  begin
    textcolor(green); write('La cantidad de sondas que realizarán estudios en el rango ',i,'
del espectro electromagnético es '); textcolor(red); writeln(vector_rangos[i]);
  end;
end;
var
  vector_rangos: t_vector_rangos;
  lista_sondas: t_lista_sondas;
  sondas_prom: int16;
  duracion_prom, costo_prom: real;
  nombre_max: string;
begin
  lista_sondas:=nil;
  nombre_max:='';
  inicializar_vector_rangos(vector_rangos);
  duracion_prom:=0; sondas_prom:=0;
  costo_prom:=0;
  cargar_lista_sondas(lista_sondas,duracion_prom,costo_prom);
  procesar_lista_sondas(lista_sondas,duracion_prom,costo_prom,nombre_max,vector_rangos,sondas_
prom);
  textcolor(green); write('El nombre de la sonda más costosa (considerando su costo de
construcción y de mantenimiento es '); textcolor(red); writeln(nombre_max);
  imprimir_vector_rangos(vector_rangos);
  textcolor(green); write('La cantidad de sondas cuya duración estimada supera la duración
promedio de todas las sondas es '); textcolor(red); write(sondas_prom);
end.
```

Ejercicio 7.

El Programa Horizonte 2020 (H2020) de la Unión Europea ha publicado los criterios para financiar proyectos de investigación avanzada. Para los proyectos de sondas espaciales vistos en el ejercicio anterior, se han determinado los siguientes criterios:

- *Sólo se financiarán proyectos cuyo costo de mantenimiento no supere el costo de construcción.*
- *No se financiarán proyectos espaciales que analicen ondas de radio, ya que esto puede realizarse desde la superficie terrestre con grandes antenas.*

A partir de la información disponible de las sondas espaciales (la lista generada en el Ejercicio 6), implementar un programa que:

(a) *Invoque un módulo que reciba la información de una sonda espacial y retorne si cumple o no con los nuevos criterios H2020.*

(b) *Utilizando el módulo desarrollado en (a), implemente un módulo que procese la lista de sondas de la ESA y retorne dos listados, uno con los proyectos que cumplen con los nuevos criterios y otro con aquellos que no los cumplen.*

(c) *Invoque a un módulo que reciba una lista de proyectos de sondas espaciales e informe la cantidad y el costo total (construcción y mantenimiento) de los proyectos que no serán financiados por H2020. Para ello, utilizar el módulo realizado en (b).*

```
program TP6_E7;
{$codepage UTF8}
uses crt;
const
  rango_ini=1; rango_fin=7;
  nombre_salida='GAIA';
  rango_corte=1;
type
  t_rango=rango_ini..rango_fin;
  t_registro_sonda=record
    nombre: string;
    duracion: int16;
    costo_construccion: real;
    costo_mantenimiento: real;
    rango: t_rango;
  end;
  t_vector_rangos=array[t_rango] of int16;
  t_lista_sondas=^t_nodo_sondas;
  t_nodo_sondas=record
    ele: t_registro_sonda;
    sig: t_lista_sondas;
  end;
procedure inicializar_vector_rangos(var vector_rangos: t_vector_rangos);
var
  i: t_rango;
begin
  for i:= rango_ini to rango_fin do
    vector_rangos[i]:=0;
end;
procedure leer_sonda(var registro_sonda: t_registro_sonda);
begin
  textcolor(green); write('Introducir nombre de la sonda: ');
```

```
  textcolor(yellow); readln(registro_sonda.nombre);
  textcolor(green); write('Introducir duración estimada de la misión de la sonda (cantidad de
meses que permanecerá activa): ');
  textcolor(yellow); readln(registro_sonda.duracion);
  textcolor(green); write('Introducir costo de construcción de la sonda: ');
  textcolor(yellow); readln(registro_sonda.costo_construccion);
  textcolor(green); write('Introducir costo de mantenimiento mensual de la sonda: ');
  textcolor(yellow); readln(registro_sonda.costo_mantenimiento);
  textcolor(green); write('Introducir rango del espectro electromagnético sobre el que
realizará estudios la sonda (1. radio; 2. microondas; 3. infrarrojo; 4. luz visible; 5.
ultravioleta; 6. rayos X; 7. rayos gamma): ');
  textcolor(yellow); readln(registro_sonda.rango);
end;
procedure agregar_adelante_lista_sondas(var lista_sondas: t_lista_sondas; registro_sonda:
t_registro_sonda);
var
  nuevo: t_lista_sondas;
begin
  new(nuevo);
  nuevo^.ele:=registro_sonda;
  nuevo^.sig:=lista_sondas;
  lista_sondas:=nuevo;
end;
procedure cargar_lista_sondas(var lista_sondas: t_lista_sondas; var duracion_prom, costo_prom:
real);
var
  registro_sonda: t_registro_sonda;
  sondas_total: int16;
  duracion_total, costo_total: real;
begin
  duracion_total:=0; sondas_total:=0;
  costo_total:=0;
  repeat
    leer_sonda(registro_sonda);
    agregar_adelante_lista_sondas(lista_sondas,registro_sonda);
    duracion_total:=duracion_total+lista_sondas^.ele.duracion;
    sondas_total:=sondas_total+1;
    costo_total:=costo_total+lista_sondas^.ele.costo_construccion;
  until (registro_sonda.nombre=nombre_salida);
  duracion_prom:=duracion_total/sondas_total;
  costo_prom:=costo_total/sondas_total;
end;
procedure actualizar_maximo(costo: real; nombre: string; var costo_max: real; var nombre_max:
string);
begin
  if (costo>costo_max) then
  begin
    costo_max:=costo;
    nombre_max:=nombre;
  end;
end;
procedure procesar1_lista_sondas(lista_sondas: t_lista_sondas; duracion_prom, costo_prom:
real; var nombre_max: string; var vector_rangos: t_vector_rangos; var sondas_prom: int16);
var
  costo_sonda, costo_max: real;
begin
  costo_max:=-9999999;
  while (lista_sondas<>nil) do
  begin
    costo_sonda:=lista_sondas^.ele.costo_construccion+lista_sondas^.ele.costo_mantenimiento*li
sta_sondas^.ele.duracion;
    actualizar_maximo(costo_sonda,lista_sondas^.ele.nombre,costo_max,nombre_max);
    vector_rangos[lista_sondas^.ele.rango]:=vector_rangos[lista_sondas^.ele.rango]+1;
    if (lista_sondas^.ele.duracion>duracion_prom) then
      sondas_prom:=sondas_prom+1;
    if (lista_sondas^.ele.costo_construccion>costo_prom) then
```

```
    begin
      textcolor(green); write('El nombre de esta sonda cuyo costo de construcción supera el
costo promedio entre todas las sondas es '); textcolor(red);
writeln(lista_sondas^.ele.nombre);
    end;
    lista_sondas:=lista_sondas^.sig;
  end;
end;
function cumple_criterios(registro_sonda: t_registro_sonda): boolean;
begin
  cumple_criterios:=((registro_sonda.costo_mantenimiento*registro_sonda.duracion<registro_sond
a.costo_construccion) and (registro_sonda.rango<>rango_corte));
end;
procedure cargar_listas_sondas(var lista_sondas_cumplen, lista_sondas_nocumplen:
t_lista_sondas; lista_sondas: t_lista_sondas);
begin
  while (lista_sondas<>nil) do
  begin
    if (cumple_criterios(lista_sondas^.ele)=true) then
      agregar_adelante_lista_sondas(lista_sondas_cumplen,lista_sondas^.ele)
    else
      agregar_adelante_lista_sondas(lista_sondas_nocumplen,lista_sondas^.ele);
    lista_sondas:=lista_sondas^.sig;
  end;
end;
procedure imprimir_vector_rangos(vector_rangos: t_vector_rangos);
var
  i: t_rango;
begin
  for i:= rango_ini to rango_fin do
  begin
    textcolor(green); write('La cantidad de sondas que realizarán estudios en el rango ',i,'
del espectro electromagnético es '); textcolor(red); writeln(vector_rangos[i]);
  end;
end;
procedure procesar2_lista_sondas(lista_sondas: t_lista_sondas);
var
  lista_sondas_cumplen, lista_sondas_nocumplen: t_lista_sondas;
  sondas_nocumplen: int16;
  costo_sondas_nocumplen: real;
begin
  lista_sondas_cumplen:=nil; lista_sondas_nocumplen:=nil;
  sondas_nocumplen:=0; costo_sondas_nocumplen:=0;
  cargar_listas_sondas(lista_sondas_cumplen,lista_sondas_nocumplen,lista_sondas);
  while (lista_sondas_nocumplen<>nil) do
  begin
    sondas_nocumplen:=sondas_nocumplen+1;
    costo_sondas_nocumplen:=costo_sondas_nocumplen+lista_sondas_nocumplen^.ele.costo_construcc
ion+lista_sondas_nocumplen^.ele.costo_mantenimiento*lista_sondas_nocumplen^.ele.duracion;
    lista_sondas_nocumplen:=lista_sondas_nocumplen^.sig;
  end;
  textcolor(green); write('La cantidad y el costo total (construcción y mantenimiento) de los
proyectos que no serán financiados por H2020 son '); textcolor(red); write(sondas_nocumplen);
textcolor(green); write(' y $'); textcolor(red); write(costo_sondas_nocumplen:0:2);
textcolor(green); writeln(', respectivamente');
end;
var
  vector_rangos: t_vector_rangos;
  lista_sondas: t_lista_sondas;
  sondas_prom: int16;
  duracion_prom, costo_prom: real;
  nombre_max: string;
begin
  lista_sondas:=nil;
  nombre_max:='';
  inicializar_vector_rangos(vector_rangos);
```

```
  duracion_prom:=0; sondas_prom:=0;
  costo_prom:=0;
  cargar_lista_sondas(lista_sondas,duracion_prom,costo_prom);
  procesar1_lista_sondas(lista_sondas,duracion_prom,costo_prom,nombre_max,vector_rangos,sondas
_prom);
  textcolor(green); write('El nombre de la sonda más costosa (considerando su costo de
construcción y de mantenimiento es '); textcolor(red); writeln(nombre_max);
  imprimir_vector_rangos(vector_rangos);
  textcolor(green); write('La cantidad de sondas cuya duración estimada supera la duración
promedio de todas las sondas es '); textcolor(red); writeln(sondas_prom);
  procesar2_lista_sondas(lista_sondas);
end.
```

Ejercicio 8.

Utilizando el programa del Ejercicio 1, modificar el módulo armarNodo para que los elementos de la lista queden ordenados de manera ascendente (insertar ordenado).

```
procedure armarNodo4(var L: lista; v: integer);
var
  anterior, actual, nuevo: lista;
begin
  new(nuevo);
  nuevo^.num:=v;
  anterior:=L; actual:=L;
  while ((actual<>nil) and (actual^.num<nuevo^.num)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=L) then
    L:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
```

```
program TP6_E8;
{$codepage UTF8}
uses crt;
type
  lista=^nodo;
  nodo=record
    num: integer;
    sig: lista;
  end;
procedure armarNodo1(var L: lista; v: integer);
var
  aux: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=L;
  L:=aux;
end;
procedure armarNodo2(var L: lista; v: integer);
var
  aux, ult: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=nil;
  if (L=nil) then
    L:=aux
  else
  begin
    ult:=L;
    while (ult^.sig<>nil) do
      ult:=ult^.sig;
    ult^.sig:=aux;
  end;
end;
procedure armarNodo3(var L, ult: lista; v: integer);
var
  aux: lista;
begin
```

```
  new(aux);
  aux^.num:=v;
  aux^.sig:=nil;
  if (L=nil) then
    L:=aux
  else
    ult^.sig:=aux;
  ult:=aux;
end;
procedure armarNodo4(var L: lista; v: integer);
var
  anterior, actual, nuevo: lista;
begin
  new(nuevo);
  nuevo^.num:=v;
  anterior:=L; actual:=L;
  while ((actual<>nil) and (actual^.num<nuevo^.num)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=L) then
    L:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure imprimir_lista(L: lista);
var
  i: int16;
begin
  i:=1;
  while (L<>nil) do
  begin
    textcolor(green); write('Elemento ',i,' de la lista: '); textcolor(yellow);
writeln(L^.num);
    L:=L^.sig;
    i:=i+1;
  end;
end;
procedure modificar_lista(var L: lista; valor: int16);
var
  aux: lista;
begin
  aux:=L;
  while (aux<>nil) do
  begin
    aux^.num:=aux^.num+valor;
    aux:=aux^.sig;
  end;
end;
function calcular_maximo(L: lista): integer;
var
  maximo: integer;
begin
  maximo:=low(integer);
  while (L<>nil) do
  begin
    if (L^.num>maximo) then
      maximo:=L^.num;
    L:=L^.sig;
  end;
  calcular_maximo:=maximo;
end;
function calcular_minimo(L: lista): integer;
var
```

```
  minimo: integer;
begin
  minimo:=high(integer);
  while (L<>nil) do
  begin
    if (L^.num<minimo) then
      minimo:=L^.num;
    L:=L^.sig;
  end;
  calcular_minimo:=minimo;
end;
function calcular_multiplos(L: lista; divisor: integer): integer;
var
  multiplos: integer;
begin
  multiplos:=0;
  while (L<>nil) do
  begin
    if (L^.num mod divisor=0) then
      multiplos:=multiplos+1;
    L:=L^.sig;
  end;
  calcular_multiplos:=multiplos;
end;
var
  pri, ult: lista;
  valor, divisor: integer;
begin
  pri:=nil; ult:=nil;
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(valor);
  while (valor<>0) do
  begin
    //armarNodo1(pri,valor);
    //armarNodo2(pri,valor);
    //armarNodo3(pri,ult,valor);
    armarNodo4(pri,valor);
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(valor);
  end;
  if (pri<>nil) then
  begin
    imprimir_lista(pri);
    textcolor(green); write('Introducir número entero con el cual se desea incrementar cada
dato de la lista: ');
    textcolor(yellow); readln(valor);
    modificar_lista(pri,valor);
    imprimir_lista(pri);
    textcolor(green); write('El elemento de valor máximo de la lista es '); textcolor(red);
writeln(calcular_maximo(pri));
    textcolor(green); write('El elemento de valor mínimo de la lista es '); textcolor(red);
writeln(calcular_minimo(pri));
    textcolor(green); write('Introducir número entero como divisor para calcular cuántos
elementos de la lista son múltiplos de él: ');
    textcolor(yellow); readln(divisor);
    textcolor(green); write('La cantidad de elementos de la lista que son múltiplos de ');
textcolor(yellow); write(divisor); textcolor(green); write(' es '); textcolor(red);
write(calcular_multiplos(pri,divisor));
  end;
end.
```

Ejercicio 9.

Utilizando el programa del Ejercicio 1, realizar los siguientes módulos:

(a) *EstaOrdenada: recibe la lista como parámetro y retorna true si la misma se encuentra ordenada o false en caso contrario.*

(b) *Eliminar: recibe como parámetros la lista y un valor entero, y elimina dicho valor de la lista (si existe). Nota: La lista podría no estar ordenada.*

(c) *Sublista: recibe como parámetros la lista y dos valores enteros A y B, y retorna una nueva lista con todos los elementos de la lista mayores que A y menores que B.*

(d) *Modificar el módulo Sublista del inciso anterior, suponiendo que la lista se encuentra ordenada de manera ascendente.*

(e) *Modificar el módulo Sublista del inciso anterior, suponiendo que la lista se encuentra ordenada de manera descendente.*

```
program TP6_E9;
{$codepage UTF8}
uses crt;
type
  lista=^nodo;
  nodo=record
    num: integer;
    sig: lista;
  end;
procedure armarNodo1(var L: lista; v: integer);
var
  aux: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=L;
  L:=aux;
end;
procedure armarNodo2(var L: lista; v: integer);
var
  aux, ult: lista;
begin
  new(aux);
  aux^.num:=v;
  aux^.sig:=nil;
  if (L=nil) then
    L:=aux
  else
  begin
    ult:=L;
    while (ult^.sig<>nil) do
      ult:=ult^.sig;
    ult^.sig:=aux;
  end;
end;
procedure armarNodo3(var L, ult: lista; v: integer);
var
  aux: lista;
begin
  new(aux);
```

```
  aux^.num:=v;
  aux^.sig:=nil;
  if (L=nil) then
    L:=aux
  else
    ult^.sig:=aux;
  ult:=aux;
end;
procedure armarNodo4(var L: lista; v: integer);
var
  anterior, actual, nuevo: lista;
begin
  new(nuevo);
  nuevo^.num:=v;
  anterior:=L; actual:=L;
  while ((actual<>nil) and (actual^.num<nuevo^.num)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=L) then
    L:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure imprimir_lista(L: lista);
var
  i: int16;
begin
  i:=1;
  while (L<>nil) do
  begin
    textcolor(green); write('Elemento ',i,' de la lista: '); textcolor(yellow);
writeln(L^.num);
    L:=L^.sig;
    i:=i+1;
  end;
end;
procedure modificar_lista(var L: lista; valor: int16);
var
  aux: lista;
begin
  aux:=L;
  while (aux<>nil) do
  begin
    aux^.num:=aux^.num+valor;
    aux:=aux^.sig;
  end;
end;
function calcular_maximo(L: lista): integer;
var
  maximo: integer;
begin
  maximo:=low(integer);
  while (L<>nil) do
  begin
    if (L^.num>maximo) then
      maximo:=L^.num;
    L:=L^.sig;
  end;
  calcular_maximo:=maximo;
end;
function calcular_minimo(L: lista): integer;
var
  minimo: integer;
```

```
begin
  minimo:=high(integer);
  while (L<>nil) do
  begin
    if (L^.num<minimo) then
      minimo:=L^.num;
    L:=L^.sig;
  end;
  calcular_minimo:=minimo;
end;
function calcular_multiplos(L: lista; divisor: integer): integer;
var
  multiplos: integer;
begin
  multiplos:=0;
  while (L<>nil) do
  begin
    if (L^.num mod divisor=0) then
      multiplos:=multiplos+1;
    L:=L^.sig;
  end;
  calcular_multiplos:=multiplos;
end;
function EstaOrdenadaAscendente(L: lista): boolean;
begin
  while ((L^.sig<>nil) and ((L^.num<L^.sig^.num))) do
    L:=L^.sig;
  EstaOrdenadaAscendente:=(L^.sig=nil);
end;
function EstaOrdenadaDescendente(L: lista): boolean;
begin
  while ((L^.sig<>nil) and ((L^.num>L^.sig^.num))) do
    L:=L^.sig;
  EstaOrdenadaDescendente:=(L^.sig=nil);
end;
procedure Eliminar(var L: lista; valor: integer);
var
  anterior, actual: lista;
begin
  anterior:=L; actual:=L;
  while (actual<>nil) do
  begin
    if (actual^.num<>valor) then
    begin
      anterior:=actual;
      actual:=actual^.sig;
    end
    else
    begin
      if (actual=L) then
        L:=L^.sig
      else
        anterior^.sig:=actual^.sig;
      dispose(actual);
      actual:=anterior;
    end;
  end;
end;
procedure Sublista1(L: lista; valorA, valorB: integer; var L2: lista);
begin
  while (L<>nil) do
  begin
    if ((L^.num>valorA) and (L^.num<valorB)) then
      armarNodo2(L2,L^.num);
    L:=L^.sig;
  end;
```

```
end;
procedure Sublista2(L: lista; valorA, valorB: integer; var L2: lista);
begin
  while ((L<>nil) and (L^.num<valorB)) do
  begin
    if (L^.num>valorA) then
      armarNodo2(L2,L^.num);
    L:=L^.sig;
  end;
end;
procedure Sublista3(L: lista; valorA, valorB: integer; var L2: lista);
begin
  while ((L<>nil) and (L^.num>valorA)) do
  begin
    if (L^.num<valorB) then
      armarNodo2(L2,L^.num);
    L:=L^.sig;
  end;
end;
var
  pri, ult, pri2: lista;
  valor, divisor, valorA, valorB: integer;
  ordenada_ascendente, ordenada_descendente: boolean;
begin
  pri:=nil; ult:=nil; pri2:=nil;
  ordenada_ascendente:=false; ordenada_descendente:=false;
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(valor);
  while (valor<>0) do
  begin
    armarNodo1(pri,valor);
    //armarNodo2(pri,valor);
    //armarNodo3(pri,ult,valor);
    //armarNodo4(pri,valor);
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(valor);
  end;
  if (pri<>nil) then
  begin
    imprimir_lista(pri);
    textcolor(green); write('Introducir número entero con el cual se desea incrementar cada
dato de la lista: ');
    textcolor(yellow); readln(valor);
    modificar_lista(pri,valor);
    imprimir_lista(pri);
    textcolor(green); write('El elemento de valor máximo de la lista es '); textcolor(red);
writeln(calcular_maximo(pri));
    textcolor(green); write('El elemento de valor mínimo de la lista es '); textcolor(red);
writeln(calcular_minimo(pri));
    textcolor(green); write('Introducir número entero como divisor para calcular cuántos
elementos de la lista son múltiplos de él: ');
    textcolor(yellow); readln(divisor);
    textcolor(green); write('La cantidad de elementos de la lista que son múltiplos de ');
textcolor(yellow); write(divisor); textcolor(green); write(' es '); textcolor(red);
writeln(calcular_multiplos(pri,divisor));
    ordenada_ascendente:=EstaOrdenadaAscendente(pri);
    textcolor(green); write('¿La lista está ordenada (ascendentemente)? '); textcolor(red);
writeln(ordenada_ascendente);
    if (ordenada_ascendente=false) then
    begin
      ordenada_descendente:=EstaOrdenadaDescendente(pri);
      textcolor(green); write('¿La lista está ordenada (descendentemente)? '); textcolor(red);
writeln(ordenada_descendente);
    end;
    textcolor(green); write('Introducir número entero que se desea eliminar de la lista (si
existe): ');
```

```
    textcolor(yellow); readln(valor);
    Eliminar(pri,valor);
    if (pri<>nil) then
    begin
      imprimir_lista(pri);
      textcolor(green); write('Introducir número entero A: ');
      textcolor(yellow); readln(valorA);
      textcolor(green); write('Introducir número entero B: ');
      textcolor(yellow); readln(valorB);
      if ((ordenada_ascendente=false) and (ordenada_descendente=false)) then
      begin
        textcolor(green); write('La lista pri está '); textcolor(red); write('desordenada ');
textcolor(green); write(', por lo que se genera la lista pri2 utilizando el procedure ');
textcolor(red); writeln('Sublista1');
        Sublista1(pri,valorA,valorB,pri2);
      end
      else
        if (ordenada_ascendente=true) then
        begin
          textcolor(green); write('La lista pri está '); textcolor(red); write('ordenada de
manera ascendente '); textcolor(green); write(', por lo que se genera la lista pri2 utilizando
el procedure '); textcolor(red); writeln('Sublista2');
          Sublista2(pri,valorA,valorB,pri2);
        end
        else
          if (ordenada_descendente=true) then
          begin
            textcolor(green); write('La lista pri está '); textcolor(red); write('ordenada de
manera descendente '); textcolor(green); write(', por lo que se genera la lista pri2
utilizando el procedure '); textcolor(red); writeln('Sublista3');
            Sublista3(pri,valorA,valorB,pri2);
          end;
      imprimir_lista(pri2);
    end;
  end;
end.
```

Ejercicio 10.

Una empresa de sistemas está desarrollando un software para organizar listas de espera de clientes. Su funcionamiento es muy sencillo: cuando un cliente ingresa al local, se registra su DNI y se le entrega un número (que es el siguiente al último número entregado). El cliente quedará esperando a ser llamado por su número, en cuyo caso sale de la lista de espera. Se pide:

(a) *Definir una estructura de datos apropiada para representar la lista de espera de clientes.*

(b) *Implementar el módulo RecibirCliente, que recibe como parámetro el DNI del cliente y la lista de clientes en espera, asigna un número al cliente y retorna la lista de espera actualizada.*

(c) *Implementar el módulo AtenderCliente, que recibe como parámetro la lista de clientes en espera y retorna el número y DNI del cliente a ser atendido y la lista actualizada. El cliente atendido debe eliminarse de la lista de espera.*

(d) *Implementar un programa que simule la atención de los clientes. En dicho programa, primero llegarán todos los clientes juntos, se les dará un número de espera a cada uno de ellos y, luego, se los atenderá de a uno por vez. El ingreso de clientes se realiza hasta que se lee el DNI 0, que no debe procesarse.*

```
program TP6_E10;
{$codepage UTF8}
uses crt;
const
  dni_salida=0;
type
  t_registro_cliente=record
    dni: int32;
    numero: int16;
  end;
  t_lista_clientes=^t_nodo_clientes;
  t_nodo_clientes=record
    ele: t_registro_cliente;
    sig: t_lista_clientes;
  end;
procedure RecibirCliente(dni: int32; var lista_clientes: t_lista_clientes);
var
  nuevo, ult: t_lista_clientes;
begin
  new(nuevo);
  nuevo^.ele.dni:=dni;
  nuevo^.sig:=nil;
  if (lista_clientes=nil) then
  begin
    nuevo^.ele.numero:=1;
    lista_clientes:=nuevo;
  end
  else
  begin
    ult:=lista_clientes;
    while (ult^.sig<>nil) do
      ult:=ult^.sig;
    nuevo^.ele.numero:=ult^.ele.numero+1;
```

```
    ult^.sig:=nuevo;
  end;
end;
procedure cargar_lista_clientes(var lista_clientes: t_lista_clientes);
var
  dni: int32;
begin
  textcolor(green); write('Introducir DNI del cliente: ');
  textcolor(yellow); readln(dni);
  while (dni<>dni_salida) do
  begin
    RecibirCliente(dni,lista_clientes);
    textcolor(green); write('Introducir DNI del cliente: ');
    textcolor(yellow); readln(dni);
  end;
end;
procedure AtenderCliente(var lista_clientes: t_lista_clientes; var numero: int16; var dni:
int32);
var
  lista_clientes_aux: t_lista_clientes;
begin
  if (lista_clientes<>nil) then
  begin
    lista_clientes_aux:=lista_clientes;
    dni:=lista_clientes_aux^.ele.dni;
    numero:=lista_clientes_aux^.ele.numero;
    lista_clientes:=lista_clientes^.sig;
    dispose(lista_clientes_aux);
  end;
end;
procedure vaciar_lista_clientes(var lista_clientes: t_lista_clientes);
var
  numero: int16;
  dni: int32;
begin
  numero:=0; dni:=0;
  while (lista_clientes<>nil) do
  begin
    AtenderCliente(lista_clientes,numero,dni);
    textcolor(green); write('El número y el DNI del cliente a ser atendido son ');
textcolor(red); write(numero); textcolor(green); write(' y '); textcolor(red); write(dni);
textcolor(green); writeln(', respectivamente');
  end;
end;
procedure imprimir_lista(lista_clientes: t_lista_clientes);
begin
  while (lista_clientes<>nil) do
  begin
    textcolor(green); write('El DNI del cliente es '); textcolor(red);
writeln(lista_clientes^.ele.dni);
    textcolor(green); write('El número del cliente es '); textcolor(red);
writeln(lista_clientes^.ele.numero);
    textcolor(green); writeln('-----------------------------');
    lista_clientes:=lista_clientes^.sig;
  end;
end;
var
  lista_clientes: t_lista_clientes;
begin
  lista_clientes:=nil;
  cargar_lista_clientes(lista_clientes);
  if (lista_clientes<>nil) then
  begin
    imprimir_lista(lista_clientes);
    vaciar_lista_clientes(lista_clientes);
    imprimir_lista(lista_clientes);
```

```
  end;
end.
```

Ejercicio 11.

La Facultad de Informática debe seleccionar los 10 egresados con mejor promedio a los que la UNLP les entregará el premio Joaquín V. González. De cada egresado, se conoce su número de alumno, apellido y el promedio obtenido durante toda su carrera. Implementar un programa que:

- *Lea la información de todos los egresados, hasta ingresar el código 0, el cual no debe procesarse.*
- *Una vez ingresada la información de los egresados, informe el apellido y número de alumno de los egresados que recibirán el premio. La información debe imprimirse ordenada según el promedio del egresado (de mayor a menor).*

```
program TP6_E11;
{$codepage UTF8}
uses crt;
const
  alumno_corte=10;
  alumno_salida=0;
type
  t_registro_alumno=record
    alumno: int16;
    apellido: string;
    promedio: real;
  end;
  t_lista_alumnos=^t_nodo_alumnos;
  t_nodo_alumnos=record
    ele: t_registro_alumno;
    sig: t_lista_alumnos;
  end;
procedure leer_alumno(var registro_alumno: t_registro_alumno);
begin
  textcolor(green); write('Introducir número de alumno del alumno: ');
  textcolor(yellow); readln(registro_alumno.alumno);
  if (registro_alumno.alumno<>alumno_salida) then
  begin
    textcolor(green); write('Introducir apellido del alumno: ');
    textcolor(yellow); readln(registro_alumno.apellido);
    textcolor(green); write('Introducir promedio obtenido durante toda la carrera del alumno:
');
    textcolor(yellow); readln(registro_alumno.promedio);
  end;
end;
procedure agregar_ordenado_lista_alumnos(var lista_alumnos: t_lista_alumnos; registro_alumno:
t_registro_alumno);
var
  anterior, actual, nuevo: t_lista_alumnos;
begin
  new(nuevo);
  nuevo^.ele:=registro_alumno;
  anterior:=lista_alumnos; actual:=lista_alumnos;
  while ((actual<>nil) and (actual^.ele.promedio>nuevo^.ele.promedio)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_alumnos) then
    lista_alumnos:=nuevo
  else
    anterior^.sig:=nuevo;
```

```
  nuevo^.sig:=actual;
end;
procedure cargar_lista_alumnos(var lista_alumnos: t_lista_alumnos);
var
  registro_alumno: t_registro_alumno;
begin
  leer_alumno(registro_alumno);
  while (registro_alumno.alumno<>alumno_salida) do
  begin
    agregar_ordenado_lista_alumnos(lista_alumnos,registro_alumno);
    leer_alumno(registro_alumno);
  end;
end;
procedure procesar_lista_alumnos(lista_alumnos: t_lista_alumnos);
var
  alumno: int16;
begin
  alumno:=0;
  while ((lista_alumnos<>nil) and (alumno<alumno_corte)) do
  begin
    alumno:=alumno+1;
    textcolor(green); write('El apellido y número de alumno del alumno ',alumno,' que recibirá
el premio son '); textcolor(red); write(lista_alumnos^.ele.apellido); textcolor(green);
write(' y '); textcolor(red); write(lista_alumnos^.ele.alumno); textcolor(green); writeln(',
respectivamente');
    lista_alumnos:=lista_alumnos^.sig;
  end;
end;
var
  lista_alumnos: t_lista_alumnos;
begin
  lista_alumnos:=nil;
  cargar_lista_alumnos(lista_alumnos);
  if (lista_alumnos<>nil) then
    procesar_lista_alumnos(lista_alumnos);
end.
```

Ejercicio 12.

Una empresa desarrolladora de juegos para teléfonos celulares con Android dispone de información de todos los dispositivos que poseen sus juegos instalados. De cada dispositivo, se conoce la versión de Android instalada, el tamaño de la pantalla (en pulgadas) y la cantidad de memoria RAM que posee (medida en GB). La información disponible se encuentra ordenada por versión de Android. Realizar un programa que procese la información disponible de todos los dispositivos e informe:

- *La cantidad de dispositivos para cada versión de Android.*
- *La cantidad de dispositivos con más de 3 GB de memoria y pantallas de, a lo sumo, 5 pulgadas.*
- *El tamaño promedio de las pantallas de todos los dispositivos.*

```
program TP6_E12;
{$codepage UTF8}
uses crt;
const
  version_salida=-1;
  ram_corte=3; tamanio_corte=5;
type
  t_registro_celular=record
    version: int16;
    tamanio: real;
    ram: real;
  end;
  t_lista_celulares=^t_nodo_celulares;
  t_nodo_celulares=record
    ele: t_registro_celular;
    sig: t_lista_celulares;
  end;
procedure leer_celular(var registro_celular: t_registro_celular);
begin
  registro_celular.version:=version_salida+random(100);
  if (registro_celular.version<>version_salida) then
  begin
    registro_celular.tamanio:=1+random(10);
    registro_celular.ram:=1+random(64);
  end;
end;
procedure agregar_ordenado_lista_celulares(var lista_celulares: t_lista_celulares;
registro_celular: t_registro_celular);
var
  anterior, actual, nuevo: t_lista_celulares;
begin
  new(nuevo);
  nuevo^.ele:=registro_celular;
  anterior:=lista_celulares; actual:=lista_celulares;
  while ((actual<>nil) and (actual^.ele.version<nuevo^.ele.version)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_celulares) then
    lista_celulares:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure cargar_lista_celulares(var lista_celulares: t_lista_celulares);
```

```
var
  registro_celular: t_registro_celular;
begin
  leer_celular(registro_celular);
  while (registro_celular.version<>version_salida) do
  begin
    agregar_ordenado_lista_celulares(lista_celulares,registro_celular);
    leer_celular(registro_celular);
  end;
end;
function cumple_criterios(registro_celular: t_registro_celular): boolean;
begin
  cumple_criterios:=((registro_celular.ram>ram_corte) and
(registro_celular.tamanio<=tamanio_corte));
end;
procedure procesar_lista_celulares(lista_celulares: t_lista_celulares; var celulares_corte:
int16; var tamanio_prom: real);
var
  version, celulares_version, celulares_total: int16;
  tamanio_total: real;
begin
  celulares_total:=0; tamanio_total:=0;
  while (lista_celulares<>nil) do
  begin
    version:=lista_celulares^.ele.version;
    celulares_version:=0;
    while ((lista_celulares<>nil) and (lista_celulares^.ele.version=version)) do
    begin
      celulares_version:=celulares_version+1;
      if (cumple_criterios(lista_celulares^.ele)=true) then
        celulares_corte:=celulares_corte+1;
      celulares_total:=celulares_total+1;
      tamanio_total:=tamanio_total+lista_celulares^.ele.tamanio;
      lista_celulares:=lista_celulares^.sig;
    end;
    textcolor(green); write('La cantidad de dispositivos para la versión de Android ');
textcolor(yellow); write(version); textcolor(green); write(' es '); textcolor(red);
writeln(celulares_version);
  end;
  tamanio_prom:=tamanio_total/celulares_total;
end;
var
  lista_celulares: t_lista_celulares;
  celulares_corte: int16;
  tamanio_prom: real;
begin
  randomize;
  lista_celulares:=nil;
  celulares_corte:=0;
  tamanio_prom:=0;
  cargar_lista_celulares(lista_celulares);
  if (lista_celulares<>nil) then
  begin
    procesar_lista_celulares(lista_celulares,celulares_corte,tamanio_prom);
    textcolor(green); write('La cantidad de dispositivos con más de '); textcolor(yellow);
write(ram_corte); textcolor(green); write(' GB de memoria y pantallas de, a lo sumo, ');
textcolor(yellow); write(tamanio_corte); textcolor(green); write(' pulgadas es ');
textcolor(red); writeln(celulares_corte);
    textcolor(green); write('El tamaño promedio de las pantallas de todos los dispositivos es
'); textcolor(red); write(tamanio_prom:0:2);
  end;
end.
```

Ejercicio 13.

El Portal de Revistas de la UNLP está estudiando el uso de sus sistemas de edición electrónica por parte de los usuarios. Para ello, se dispone de información sobre los 3600 usuarios que utilizan el portal. De cada usuario, se conoce su nombre, su email, su rol (1. Editor; 2. Autor; 3. Revisor; 4. Lector), revista en la que participa y cantidad de días desde el último acceso.

- *Imprimir el nombre de usuario y la cantidad de días desde el último acceso de todos los usuarios de la revista Económica. El listado debe ordenarse a partir de la cantidad de días desde el último acceso (orden ascendente).*
- *Informar la cantidad de usuarios por cada rol para todas las revistas del portal.*
- *Informar los emails de los dos usuarios que hace más tiempo que no ingresan al portal.*

```
program TP6_E13;
{$codepage UTF8}
uses crt;
const
  usuarios_total=3600;
  rol_ini=1; rol_fin=4;
  revista_corte='Economica';
type
  t_usuario=1..usuarios_total;
  t_rol=rol_ini..rol_fin;
  t_registro_usuario=record
    nombre: string;
    email: string;
    rol: t_rol;
    revista: string;
    dias: int16;
  end;
  t_vector_usuarios=array[t_usuario] of t_registro_usuario;
  t_vector_roles=array[t_rol] of int16;
  t_lista_usuarios=^t_nodo_usuarios;
  t_nodo_usuarios=record
    ele: t_registro_usuario;
    sig: t_lista_usuarios;
  end;
procedure inicializar_vector_roles(var vector_roles: t_vector_roles);
var
  i: t_rol;
begin
  for i:= rol_ini to rol_fin do
    vector_roles[i]:=0;
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_usuario(var registro_usuario: t_registro_usuario);
begin
  registro_usuario.nombre:=random_string(1+random(10));
  registro_usuario.email:=random_string(1+random(10));
```

```
  registro_usuario.rol:=rol_ini+random(rol_fin);
  registro_usuario.revista:=revista_corte+random_string(random(10));
  registro_usuario.dias:=random(high(int16));
end;
procedure cargar_vector_usuarios(var vector_usuarios: t_vector_usuarios);
var
  i: t_usuario;
  registro_usuario: t_registro_usuario;
begin
  for i:= 1 to usuarios_total do
  begin
    leer_usuario(registro_usuario);
    vector_usuarios[i]:=registro_usuario;
  end;
end;
procedure agregar_ordenado_lista_usuarios(var lista_usuarios: t_lista_usuarios;
registro_usuario: t_registro_usuario);
var
  anterior, actual, nuevo: t_lista_usuarios;
begin
  new(nuevo);
  nuevo^.ele:=registro_usuario;
  anterior:=lista_usuarios; actual:=lista_usuarios;
  while ((actual<>nil) and (actual^.ele.dias<nuevo^.ele.dias)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_usuarios) then
    lista_usuarios:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure actualizar_maximos(dias: int16; email: string; var dias_max1, dias_max2: int16; var
email_max1, email_max2: string);
begin
  if (dias>dias_max1) then
  begin
    dias_max2:=dias_max1;
    email_max2:=email_max1;
    dias_max1:=dias;
    email_max1:=email;
  end
  else
    if (dias>dias_max2) then
    begin
      dias_max2:=dias;
      email_max2:=email;
    end;
end;
procedure procesar_vector_usuarios(vector_usuarios: t_vector_usuarios; var lista_usuarios:
t_lista_usuarios; var vector_roles: t_vector_roles; var email_max1, email_max2: string);
var
  i: t_usuario;
  dias_max1, dias_max2: int16;
begin
  dias_max1:=low(int16); dias_max2:=low(int16);
  for i:= 1 to usuarios_total do
  begin
    if (vector_usuarios[i].revista=revista_corte) then
      agregar_ordenado_lista_usuarios(lista_usuarios,vector_usuarios[i]);
    vector_roles[vector_usuarios[i].rol]:=vector_roles[vector_usuarios[i].rol]+1;
    actualizar_maximos(vector_usuarios[i].dias,vector_usuarios[i].email,dias_max1,dias_max2,em
ail_max1,email_max2);
  end;
```

```
end;
procedure imprimir_lista_usuarios(lista_usuarios: t_lista_usuarios);
begin
  while (lista_usuarios<>nil) do
  begin
    textcolor(green); write('El nombre de usuario y la cantidad de días desde el último acceso
de este usuario de la revista '); textcolor(yellow); write(revista_corte); textcolor(green);
write(' son '); textcolor(red); write(lista_usuarios^.ele.nombre); textcolor(green); write(' y
'); textcolor(red); write(lista_usuarios^.ele.dias); textcolor(green); writeln(',
respectivamente');
    lista_usuarios:=lista_usuarios^.sig;
  end;
end;
procedure imprimir_vector_roles(vector_roles: t_vector_roles);
var
  i: t_rol;
begin
  for i:= rol_ini to rol_fin do
  begin
    textcolor(green); write('La cantidad de usuarios para el rol ',i,' para todas las revistas
del portal es '); textcolor(red); writeln(vector_roles[i]);
  end;
end;
var
  vector_usuarios: t_vector_usuarios;
  vector_roles: t_vector_roles;
  lista_usuarios: t_lista_usuarios;
  email_max1, email_max2: string;
begin
  randomize;
  lista_usuarios:=nil;
  inicializar_vector_roles(vector_roles);
  email_max1:=''; email_max2:='';
  cargar_vector_usuarios(vector_usuarios);
  procesar_vector_usuarios(vector_usuarios,lista_usuarios,vector_roles,email_max1,email_max2);
  if (lista_usuarios<>nil) then
    imprimir_lista_usuarios(lista_usuarios);
  imprimir_vector_roles(vector_roles);
  textcolor(green); write('Los emails de los dos usuarios que hace más tiempo que no ingresan
al portal son '); textcolor(red); write(email_max1); textcolor(green); write(' y ');
textcolor(red); write(email_max2); textcolor(green); write(', respectivamente');
end.
```

Ejercicio 14.

La oficina de becas y subsidios desea optimizar los distintos tipos de ayuda financiera que se brinda a alumnos de la UNLP. Para ello, esta oficina cuenta con un registro detallado de todos los viajes realizados por una muestra de 1300 alumnos durante el mes de marzo. De cada viaje, se conoce el código de alumno (entre 1 y 1300), día del mes, Facultad a la que pertenece y medio de transporte (1. colectivo urbano; 2. colectivo interurbano; 3. tren universitario; 4. tren Roca; 5. bicicleta). Tener en cuenta que un alumno puede utilizar más de un medio de transporte en un mismo día. Además, esta oficina cuenta con una tabla con información sobre el precio de cada tipo de viaje. Realizar un programa que lea la información de los viajes de los alumnos y los almacene en una estructura de datos apropiada. La lectura finaliza al ingresarse el código de alumno -1, que no debe procesarse. Una vez finalizada la lectura, informar:

- *La cantidad de alumnos que realizan más de 6 viajes por día.*
- *La cantidad de alumnos que gastan en transporte más de $80 por día.*
- *Los dos medios de transporte más utilizados.*
- *La cantidad de alumnos que combinan bicicleta con algún otro medio de transporte.*

```
program TP6_E14;
{$codepage UTF8}
uses crt;
const
  alumnos_total=1300;
  dia_ini=1; dia_fin=31;
  transporte_ini=1; transporte_fin=5;
  alumno_salida=-1;
  viajes_corte=6;
  gasto_corte=80;
  transporte_corte=5;
type
  t_alumno=1..alumnos_total;
  t_dia=dia_ini..dia_fin;
  t_transporte=transporte_ini..transporte_fin;
  t_registro_viaje=record
    alumno: int16;
    dia: t_dia;
    facultad: string;
    transporte: t_transporte;
  end;
  t_vector_precios=array[t_transporte] of real;
  t_vector_transportes=array[t_transporte] of int16;
  t_lista_viajes=^t_nodo_viajes;
  t_nodo_viajes=record
    ele: t_registro_viaje;
    sig: t_lista_viajes;
  end;
  t_vector_alumnos=array[t_alumno] of t_lista_viajes;
procedure cargar_vector_precios(var vector_precios: t_vector_precios);
var
  i: t_transporte;
begin
  for i:= transporte_ini to transporte_fin do
    vector_precios[i]:=10+random(90);
end;
procedure inicializar_vector_alumnos(var vector_alumnos: t_vector_alumnos);
var
  i: t_alumno;
```

```
begin
  for i:= 1 to alumnos_total do
    vector_alumnos[i]:=nil;
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_viaje(var registro_viaje: t_registro_viaje);
begin
  registro_viaje.alumno:=alumno_salida+random(100);
  if (registro_viaje.alumno<>alumno_salida) then
  begin
    registro_viaje.dia:=dia_ini+random(dia_fin);
    registro_viaje.facultad:=random_string(1+random(10));
    registro_viaje.transporte:=transporte_ini+random(transporte_fin);
  end;
end;
procedure agregar_ordenado_lista_viajes(var lista_viajes: t_lista_viajes; registro_viaje:
t_registro_viaje);
var
  anterior, actual, nuevo: t_lista_viajes;
begin
  new(nuevo);
  nuevo^.ele:=registro_viaje;
  anterior:=lista_viajes; actual:=lista_viajes;
  while ((actual<>nil) and (actual^.ele.dia<nuevo^.ele.dia)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_viajes) then
    lista_viajes:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure cargar_lista_viajes(var lista_viajes: t_lista_viajes; alumno: t_alumno);
var
  registro_viaje: t_registro_viaje;
begin
  leer_viaje(registro_viaje);
  while (registro_viaje.alumno<>alumno_salida) do
  begin
    registro_viaje.alumno:=alumno;
    agregar_ordenado_lista_viajes(lista_viajes,registro_viaje);
    leer_viaje(registro_viaje);
  end;
end;
procedure cargar_vector_alumnos(var vector_alumnos: t_vector_alumnos);
var
  i: t_alumno;
begin
  for i:= 1 to alumnos_total do
    cargar_lista_viajes(vector_alumnos[i],i);
end;
procedure inicializar_vector_transportes(var vector_transportes: t_vector_transportes);
var
  i: t_transporte;
begin
```

```
  for i:= transporte_ini to transporte_fin do
    vector_transportes[i]:=0;
end;
function cumple_criterio(vector_transportes2: t_vector_transportes): boolean;
var
  transporte: t_transporte;
  cumple: boolean;
begin
  transporte:=transporte_ini;
  cumple:=false;
  while ((transporte<transporte_fin) and (cumple<>true)) do
  begin
    if (vector_transportes2[transporte]>0) then
      cumple:=true;
    transporte:=transporte+1;
  end;
  cumple_criterio:=cumple;
end;
procedure procesar_vector_transportes1(vector_transportes1: t_vector_transportes; var
transporte_max1, transporte_max2: int8);
var
  i: t_transporte;
  viajes_max1, viajes_max2: int16;
begin
  viajes_max1:=low(int16); viajes_max2:=low(int16);
  for i:= transporte_ini to transporte_fin do
  begin
    if (vector_transportes1[i]>viajes_max1) then
    begin
      viajes_max2:=viajes_max1;
      transporte_max2:=transporte_max1;
      viajes_max1:=vector_transportes1[i];
      transporte_max1:=i;
    end
    else
      if (vector_transportes1[i]>viajes_max2) then
      begin
        viajes_max2:=vector_transportes1[i];
        transporte_max2:=i;
      end;
  end;
end;
procedure procesar_vector_alumnos(vector_alumnos: t_vector_alumnos; vector_precios:
t_vector_precios; var alumnos_corte_viajes, alumnos_corte_gasto, alumnos_transportes: int16;
var transporte_max1, transporte_max2: int8);
var
  vector_transportes1, vector_transportes2: t_vector_transportes;
  i: t_alumno;
  dia: t_dia;
  viajes_dia: int16;
  gasto_dia: real;
  cumple_viajes, cumple_gasto: boolean;
begin
  inicializar_vector_transportes(vector_transportes1);
  for i:= 1 to alumnos_total do
  begin
    cumple_viajes:=true; cumple_gasto:=true;
    inicializar_vector_transportes(vector_transportes2);
    while (vector_alumnos[i]<>nil) do
    begin
      dia:=vector_alumnos[i]^.ele.dia;
      viajes_dia:=0;
      gasto_dia:=0;
      while ((vector_alumnos[i]<>nil) and (vector_alumnos[i]^.ele.dia=dia)) do
      begin
        viajes_dia:=viajes_dia+1;
```

```
        gasto_dia:=gasto_dia+vector_precios[vector_alumnos[i]^.ele.transporte];
        vector_transportes1[vector_alumnos[i]^.ele.transporte]:=vector_transportes1[vector_alu
mnos[i]^.ele.transporte]+1;
        vector_transportes2[vector_alumnos[i]^.ele.transporte]:=vector_transportes2[vector_alu
mnos[i]^.ele.transporte]+1;
        vector_alumnos[i]:=vector_alumnos[i]^.sig;
      end;
      if ((cumple_viajes<>false) and (viajes_dia<=viajes_corte)) then
        cumple_viajes:=false;
      if ((cumple_gasto<>false) and (gasto_dia<=gasto_corte)) then
        cumple_gasto:=false;
    end;
    if (cumple_viajes=true) then
      alumnos_corte_viajes:=alumnos_corte_viajes+1;
    if (cumple_gasto=true) then
      alumnos_corte_gasto:=alumnos_corte_gasto+1;
    if ((vector_transportes2[transporte_corte]<>0) and
(cumple_criterio(vector_transportes2)=true)) then
      alumnos_transportes:=alumnos_transportes+1;
  end;
  procesar_vector_transportes1(vector_transportes1,transporte_max1,transporte_max2);
end;
var
  vector_precios: t_vector_precios;
  vector_alumnos: t_vector_alumnos;
  transporte_max1, transporte_max2: int8;
  alumnos_corte_viajes, alumnos_corte_gasto, alumnos_transportes: int16;
begin
  randomize;
  cargar_vector_precios(vector_precios);
  alumnos_corte_viajes:=0;
  alumnos_corte_gasto:=0;
  transporte_max1:=0; transporte_max2:=0;
  alumnos_transportes:=0;
  inicializar_vector_alumnos(vector_alumnos);
  cargar_vector_alumnos(vector_alumnos);
  procesar_vector_alumnos(vector_alumnos,vector_precios,alumnos_corte_viajes,alumnos_corte_gas
to,alumnos_transportes,transporte_max1,transporte_max2);
  textcolor(green); write('La cantidad de alumnos que realizan más de '); textcolor(yellow);
write(viajes_corte); textcolor(green); write(' viajes por día es '); textcolor(red);
writeln(alumnos_corte_viajes);
  textcolor(green); write('La cantidad de alumnos que gastan en transporte más de $');
textcolor(yellow); write(gasto_corte); textcolor(green); write(' por día es ');
textcolor(red); writeln(alumnos_corte_gasto);
  textcolor(green); write('Los dos medios de transporte más utilizados son '); textcolor(red);
write(transporte_max1); textcolor(green); write(' y '); textcolor(red);
write(transporte_max2); textcolor(green); writeln(', respectivamente');
  textcolor(green); write('La cantidad de alumnos que combinan bicicleta con algún otro medio
de transporte es '); textcolor(red); write(alumnos_transportes);
end.
```

Ejercicio 15.

La cátedra de CADP está organizando la cursada para el año 2019. Para ello, dispone de una lista con todos los alumnos que cursaron EPA. De cada alumno, se conoce su DNI, apellido, nombre y la nota obtenida. Escribir un programa que procese la información de alumnos disponible y los distribuya en turnos utilizando los siguientes criterios:

- *Los alumnos que obtuvieron, al menos, 8 en EPA deberán ir a los turnos 1 o 4.*
- *Los alumnos que obtuvieron entre 5 y 8 deberán ir a los turnos 2, 3 o 5.*
- *Los alumnos que no alcanzaron la nota 5 no se les asignará turno en CADP.*

Al finalizar, el programa debe imprimir en pantalla la lista de alumnos para cada turno.
Nota: La distribución de alumnos debe ser lo más equitativa posible.

```
program TP6_E15;
{$codepage UTF8}
uses crt;
const
  dni_salida=0;
  nota_ini=1; nota_fin=10;
  turno_ini=1; turno_fin=5;
  nota_corte1=8; nota_corte2=5;
type
  t_nota=nota_ini..nota_fin;
  t_turno=turno_ini..turno_fin;
  t_registro_alumno=record
    dni: int16;
    apellido: string;
    nota: t_nota;
  end;
  t_lista_alumnos=^t_nodo_alumnos;
  t_nodo_alumnos=record
    ele: t_registro_alumno;
    sig: t_lista_alumnos;
  end;
  t_vector_alumnos=array[t_turno] of t_lista_alumnos;
procedure inicializar_vector_alumnos(var vector_alumnos: t_vector_alumnos);
var
  i: t_turno;
begin
  for i:= turno_ini to turno_fin do
    vector_alumnos[i]:=nil;
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_alumno(var registro_alumno: t_registro_alumno);
begin
  registro_alumno.dni:=1+random(high(int16));
  if (registro_alumno.dni<>dni_salida) then
  begin
    registro_alumno.apellido:=random_string(1+random(10));
```

```
    registro_alumno.nota:=nota_ini+random(nota_fin);
  end;
end;
procedure agregar_adelante_lista_alumnos(var lista_alumnos: t_lista_alumnos; registro_alumno:
t_registro_alumno);
var
  nuevo: t_lista_alumnos;
begin
  new(nuevo);
  nuevo^.ele:=registro_alumno;
  nuevo^.sig:=lista_alumnos;
  lista_alumnos:=nuevo;
end;
procedure cargar_lista_alumnos(var lista_alumnos: t_lista_alumnos);
var
  registro_alumno: t_registro_alumno;
begin
  leer_alumno(registro_alumno);
  while (registro_alumno.dni<>dni_salida) do
  begin
    agregar_adelante_lista_alumnos(lista_alumnos,registro_alumno);
    leer_alumno(registro_alumno);
  end;
end;
function randomH(): int8;
var
  vectorH: array[1..2] of int8;
begin
  vectorH[1]:=1;
  vectorH[2]:=4;
  randomH:=vectorH[1+random(2)];
end;
function randomM(): int8;
var
  vectorM: array[1..3] of int8;
begin
  vectorM[1]:=2;
  vectorM[2]:=3;
  vectorM[3]:=5;
  randomM:=vectorM[1+random(3)];
end;
procedure procesar_lista_alumnos(lista_alumnos: t_lista_alumnos; var vector_alumnos:
t_vector_alumnos);
begin
  while (lista_alumnos<>nil) do
  begin
    if (lista_alumnos^.ele.nota>=nota_corte1) then
      agregar_adelante_lista_alumnos(vector_alumnos[randomH()],lista_alumnos^.ele)
    else
      if (lista_alumnos^.ele.nota>=nota_corte2) then
        agregar_adelante_lista_alumnos(vector_alumnos[randomM()],lista_alumnos^.ele);
    lista_alumnos:=lista_alumnos^.sig;
  end;
end;
procedure imprimir_vector_alumnos(vector_alumnos: t_vector_alumnos);
var
  i: t_turno;
begin
  for i:= turno_ini to turno_fin do
    while (vector_alumnos[i]<>nil) do
    begin
      textcolor(green); write('TURNO ',i,': '); textcolor(green); write('DNI - ');
textcolor(red); write(vector_alumnos[i]^.ele.dni); textcolor(green); write('; APELLIDO - ');
textcolor(red); write(vector_alumnos[i]^.ele.apellido); textcolor(green); write('; NOTA - ');
textcolor(red); writeln(vector_alumnos[i]^.ele.nota);
      vector_alumnos[i]:=vector_alumnos[i]^.sig;
```

```
    end;
end;
var
  vector_alumnos: t_vector_alumnos;
  lista_alumnos: t_lista_alumnos;
begin
  randomize;
  lista_alumnos:=nil;
  inicializar_vector_alumnos(vector_alumnos);
  cargar_lista_alumnos(lista_alumnos);
  if (lista_alumnos<>nil) then
  begin
    procesar_lista_alumnos(lista_alumnos,vector_alumnos);
    imprimir_vector_alumnos(vector_alumnos);
  end;
end.
```

Ejercicio 16.

La empresa distribuidora de una app móvil para corredores ha organizado una competencia mundial, en la que corredores de todos los países participantes salen a correr en el mismo momento en distintos puntos del planeta. La app registra, para cada corredor, el nombre y apellido, la distancia recorrida (en kilómetros), el tiempo (en minutos) durante el que corrió, el país y la ciudad desde donde partió y la ciudad donde finalizó su recorrido. Realizar un programa que permita leer y almacenar toda la información registrada durante la competencia. La lectura finaliza al ingresar la distancia -1. Una vez que se han almacenado todos los datos, informar:

- *La cantidad total de corredores, la distancia total recorrida y el tiempo total de carrera de todos los corredores.*
- *El nombre de la ciudad que convocó la mayor cantidad de corredores y la cantidad total de kilómetros recorridos por los corredores de esa ciudad.*
- *La distancia promedio recorrida por corredores de Brasil.*
- *La cantidad de corredores que partieron de una ciudad y finalizaron en otra ciudad.*
- *El paso (cantidad de minutos por km) promedio de los corredores de la ciudad de Boston.*

```
program TP6_E16;
{$codepage UTF8}
uses crt;
const
  distancia_salida=-1.0;
  pais_corte='Brasil';
  ciudad_corte='Boston';
type
  t_registro_corredor=record
    nombre: string;
    apellido: string;
    distancia: real;
    tiempo: real;
    pais: string;
    ciudad_ini: string;
    ciudad_fin: string;
  end;
  t_lista_corredores=^t_nodo_corredores;
  t_nodo_corredores=record
    ele: t_registro_corredor;
    sig: t_lista_corredores;
  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_corredor(var registro_corredor: t_registro_corredor);
begin
  repeat
    registro_corredor.distancia:=distancia_salida+random(1000);
  until (registro_corredor.distancia<>0);
  if (registro_corredor.distancia<>distancia_salida) then
```

```
  begin
    registro_corredor.nombre:=random_string(1+random(10));
    registro_corredor.apellido:=random_string(1+random(10));
    registro_corredor.tiempo:=1+random(1000);
    registro_corredor.pais:=pais_corte+random_string(random(10));
    registro_corredor.ciudad_ini:=ciudad_corte+random_string(random(10));
    registro_corredor.ciudad_fin:=ciudad_corte+random_string(random(10));
  end;
end;
procedure agregar_ordenado_lista_corredores(var lista_corredores: t_lista_corredores;
registro_corredor: t_registro_corredor);
var
  anterior, actual, nuevo: t_lista_corredores;
begin
  new(nuevo);
  nuevo^.ele:=registro_corredor;
  anterior:=lista_corredores; actual:=lista_corredores;
  while ((actual<>nil) and (actual^.ele.ciudad_ini<nuevo^.ele.ciudad_ini)) do
  begin
    anterior:=actual;
    actual:=actual^.sig;
  end;
  if (actual=lista_corredores) then
    lista_corredores:=nuevo
  else
    anterior^.sig:=nuevo;
  nuevo^.sig:=actual;
end;
procedure cargar_lista_corredores(var lista_corredores: t_lista_corredores);
var
  registro_corredor: t_registro_corredor;
begin
  leer_corredor(registro_corredor);
  while (registro_corredor.distancia<>distancia_salida) do
  begin
    agregar_ordenado_lista_corredores(lista_corredores,registro_corredor);
    leer_corredor(registro_corredor);
  end;
end;
procedure actualizar_maximos(corredores_ciudad: int16; ciudad: string; distancia_ciudad: real;
var corredores_max: int16; var ciudad_max: string; var distancia_max: real);
begin
  if (corredores_ciudad>corredores_max) then
  begin
    corredores_max:=corredores_ciudad;
    ciudad_max:=ciudad;
    distancia_max:=distancia_ciudad;
  end;
end;
procedure procesar_lista_corredores(lista_corredores: t_lista_corredores; var
corredores_total, corredores_distinta_ciudad: int16; var distancia_total, tiempo_total,
distancia_max, distancia_prom_corte, tiempo_prom_corte: real; var ciudad_max: string);
var
  corredores_ciudad, corredores_max, corredores_corte_pais: int16;
  distancia_ciudad, distancia_corte_pais, distancia_corte_ciudad, tiempo_corte_ciudad: real;
  ciudad: string;
begin
  corredores_max:=low(int16);
  corredores_corte_pais:=0; distancia_corte_pais:=0;
  distancia_corte_ciudad:=0; tiempo_corte_ciudad:=0;
  while (lista_corredores<>nil) do
  begin
    ciudad:=lista_corredores^.ele.ciudad_ini;
    corredores_ciudad:=0; distancia_ciudad:=0;
    while ((lista_corredores<>nil) and (lista_corredores^.ele.ciudad_ini=ciudad)) do
    begin
```

```
      corredores_total:=corredores_total+1;
      distancia_total:=distancia_total+lista_corredores^.ele.distancia;
      tiempo_total:=distancia_total+lista_corredores^.ele.tiempo;
      corredores_ciudad:=corredores_ciudad+1;
      distancia_ciudad:=distancia_ciudad+lista_corredores^.ele.distancia;
      if (lista_corredores^.ele.pais=pais_corte) then
      begin
        corredores_corte_pais:=corredores_corte_pais+1;
        distancia_corte_pais:=distancia_corte_pais+lista_corredores^.ele.distancia;
      end;
      if (lista_corredores^.ele.ciudad_ini<>lista_corredores^.ele.ciudad_fin) then
        corredores_distinta_ciudad:=corredores_distinta_ciudad+1;
      if (lista_corredores^.ele.ciudad_ini=ciudad_corte) then
      begin
        distancia_corte_ciudad:=distancia_corte_ciudad+lista_corredores^.ele.distancia;
        tiempo_corte_ciudad:=tiempo_corte_ciudad+lista_corredores^.ele.tiempo;
      end;
      lista_corredores:=lista_corredores^.sig;
    end;
    actualizar_maximos(corredores_ciudad,ciudad,distancia_ciudad,corredores_max,ciudad_max,dis
tancia_max);
  end;
  if (corredores_corte_pais>0) then
    distancia_prom_corte:=distancia_corte_pais/corredores_corte_pais;
  if (distancia_corte_ciudad>0) then
    tiempo_prom_corte:=tiempo_corte_ciudad/distancia_corte_ciudad;
end;
var
  lista_corredores: t_lista_corredores;
  corredores_total, corredores_distinta_ciudad: int16;
  distancia_total, tiempo_total, distancia_max, distancia_prom_corte, tiempo_prom_corte: real;
  ciudad_max: string;
begin
  randomize;
  lista_corredores:=nil;
  corredores_total:=0; distancia_total:=0; tiempo_total:=0;
  ciudad_max:=''; distancia_max:=0;
  distancia_prom_corte:=0;
  corredores_distinta_ciudad:=0;
  tiempo_prom_corte:=0;
  cargar_lista_corredores(lista_corredores);
  if (lista_corredores<>nil) then
  begin
    procesar_lista_corredores(lista_corredores,corredores_total,corredores_distinta_ciudad,dis
tancia_total,tiempo_total,distancia_max,distancia_prom_corte,tiempo_prom_corte,ciudad_max);
    textcolor(green); write('La cantidad total de corredores, la distancia total recorrida y
el tiempo total de carrera de todos los corredores son '); textcolor(red);
write(corredores_total); textcolor(green); write(', '); textcolor(red);
write(distancia_total:0:2); textcolor(green); write(' y '); textcolor(red);
write(tiempo_total:0:2); textcolor(green); writeln(', respectivamente');
    textcolor(green); write('El nombre de la ciudad que convocó la mayor cantidad de
corredores y la cantidad total de kilómetros recorridos por los corredores de esa ciudad es
'); textcolor(red); write(ciudad_max); textcolor(green); write(' y '); textcolor(red);
write(distancia_max:0:2); textcolor(green); writeln(', respectivamnte');
    textcolor(green); write('La distancia promedio recorrida por corredores de ');
textcolor(yellow); write(pais_corte); textcolor(green); write(' es '); textcolor(red);
writeln(distancia_prom_corte:0:2);
    textcolor(green); write('La cantidad de corredores que partieron de una ciudad y
finalizaron en otra ciudad es '); textcolor(red); writeln(corredores_distinta_ciudad);
    textcolor(green); write('El paso (cantidad de minutos por km) promedio de los corredores
de la ciudad de '); textcolor(yellow); write(ciudad_corte); textcolor(green); write(' es ');
textcolor(red); write(tiempo_prom_corte:0:2);
  end;
end.
```

Ejercicio 17.

Continuando con los 3 ejercicios adicionales de la Guía opcional de actividades adicionales, ahora, se sumará lo aprendido sobre listas para almacenar la información ingresada por teclado. Consideraciones importantes:

- *Los datos ingresados por teclado deberán almacenarse en una estructura de tipo lista apropiada.*
- *Una vez leídos y almacenados los datos, deberán procesarse (recorrer la lista) para resolver cada inciso. Al hacerlo, deberán reutilizarse los módulos ya implementados en las prácticas anteriores. En la medida de lo posible, la lista deberá recorrerse una única vez para resolver todos los incisos.*

Ejercicio 1:

```
program TP6_E17a;
{$codepage UTF8}
uses crt;
const
  empresa_salida=100;
  monto_corte=50000.0;
type
  t_registro_empresa=record
    empresa: int16;
    inversiones: int16;
    monto_total: real;
  end;
  t_lista_empresas=^t_nodo_empresas;
  t_nodo_empresas=record
    ele: t_registro_empresa;
    sig: t_lista_empresas;
  end;
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
    textcolor(green); write('Introducir monto de la inversión ',i,' de la empresa ',empresa,':
');
    textcolor(yellow); readln(monto);
    monto_total:=monto_total+monto;
  end;
end;
procedure leer_empresa(var registro_empresa: t_registro_empresa);
begin
  textcolor(green); write('Introducir código de empresa de la empresa: ');
  textcolor(yellow); readln(registro_empresa.empresa);
  textcolor(green); write('Introducir cantidad de inversiones de la empresa: ');
  textcolor(yellow); readln(registro_empresa.inversiones);
  if (registro_empresa.inversiones>0) then
    leer_inversiones(registro_empresa.empresa,registro_empresa.inversiones,registro_empresa.mo
nto_total);
end;
procedure agregar_adelante_lista_empresas(var lista_empresas: t_lista_empresas;
registro_empresa: t_registro_empresa);
var
  nuevo: t_lista_empresas;
```

```
begin
  new(nuevo);
  nuevo^.ele:=registro_empresa;
  nuevo^.sig:=lista_empresas;
  lista_empresas:=nuevo;
end;
procedure cargar_lista_empresas(var lista_empresas: t_lista_empresas);
var
  registro_empresa: t_registro_empresa;
begin
  repeat
    leer_empresa(registro_empresa);
    agregar_adelante_lista_empresas(lista_empresas,registro_empresa);
  until (lista_empresas^.ele.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_lista_empresas(lista_empresas: t_lista_empresas; var empresa_max,
empresas_corte: int16);
var
  monto_max: real;
begin
  monto_max:=-9999999;
  while (lista_empresas<>nil) do
  begin
    if (lista_empresas^.ele.inversiones>0) then
    begin
      calcular_a(lista_empresas^.ele.empresa,lista_empresas^.ele.inversiones,lista_empresas^.e
le.monto_total);
      calcular_b(lista_empresas^.ele.monto_total,lista_empresas^.ele.empresa,monto_max,empresa
_max);
      calcular_c(lista_empresas^.ele.monto_total,empresas_corte);
    end;
    lista_empresas:=lista_empresas^.sig;
  end;
end;
var
  lista_empresas: t_lista_empresas;
  empresa_max, empresas_corte: int16;
begin
  lista_empresas:=nil;
  empresa_max:=0;
  empresas_corte:=0;
  cargar_lista_empresas(lista_empresas);
  procesar_lista_empresas(lista_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 TP6_E17b;
{$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_registro_alumno=record
    legajo: int16;
    condicion: char;
    presente: int8;
    notas_cero: int8;
    notas_diez: int8;
    nota_total: int8;
  end;
  t_lista_alumnos=^t_nodo_alumnos;
  t_nodo_alumnos=record
    ele: t_registro_alumno;
    sig: t_lista_alumnos;
  end;
procedure leer_notas(var presente, notas_cero, notas_diez, nota_total: int8);
var
  i, nota: int8;
begin
  for i:= 1 to autoeva_total do
  begin
    textcolor(green); write('Introducir nota de autoevaluación ',i,' del alumno: ');
    textcolor(yellow); readln(nota);
    if ((nota<>nota_incumple) and (nota>=nota_corte)) then
      presente:=presente+1;
    if (nota=nota_cero) then
      notas_cero:=notas_cero+1;
    if (nota=nota_diez) then
      notas_diez:=notas_diez+1;
    if (nota<>nota_incumple) then
      nota_total:=nota_total+nota;
  end;
end;
procedure leer_alumno(var registro_alumno: t_registro_alumno);
begin
  textcolor(green); write('Introducir legajo del alumno: ');
  textcolor(yellow); readln(registro_alumno.legajo);
  if (registro_alumno.legajo<>legajo_salida) then
  begin
    textcolor(green); write('Introducir condición (I para INGRESANTE, R para RECURSANTE) del
alumno: ');
    textcolor(yellow); readln(registro_alumno.condicion);
    registro_alumno.presente:=0; registro_alumno.notas_cero:=0; registro_alumno.notas_diez:=0;
registro_alumno.nota_total:=0;
```

```
    leer_notas(registro_alumno.presente,registro_alumno.notas_cero,registro_alumno.notas_diez,
registro_alumno.nota_total);
  end;
end;
procedure agregar_adelante_lista_alumnos(var lista_alumnos: t_lista_alumnos; registro_alumno:
t_registro_alumno);
var
  nuevo: t_lista_alumnos;
begin
  new(nuevo);
  nuevo^.ele:=registro_alumno;
  nuevo^.sig:=lista_alumnos;
  lista_alumnos:=nuevo;
end;
procedure cargar_lista_alumnos(var lista_alumnos: t_lista_alumnos);
var
  registro_alumno: t_registro_alumno;
begin
  leer_alumno(registro_alumno);
  while (registro_alumno.legajo<>legajo_salida) do
  begin
    agregar_adelante_lista_alumnos(lista_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_lista_alumnos(lista_alumnos: t_lista_alumnos; 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
  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;
  while (lista_alumnos<>nil) do
  begin
    calcular_ab(lista_alumnos^.ele.condicion,lista_alumnos^.ele.presente,ingresantes_total,ing
resantes_parcial,recursantes_total,recursantes_parcial);
    calcular_c(lista_alumnos^.ele.presente,alumnos_autoeva);
    calcular_d(lista_alumnos^.ele.nota_total,alumnos_corte);
    calcular_e(lista_alumnos^.ele.notas_cero,alumnos_cero);
    calcular_f(lista_alumnos^.ele.notas_diez,lista_alumnos^.ele.legajo,notas_diez_max1,notas_d
iez_max2,legajo_diez_max1,legajo_diez_max2);
    calcular_g(lista_alumnos^.ele.notas_cero,lista_alumnos^.ele.legajo,notas_cero_max1,notas_c
ero_max2,legajo_cero_max1,legajo_cero_max2);
    lista_alumnos:=lista_alumnos^.sig;
  end;
end;
var
  lista_alumnos: t_lista_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
  lista_alumnos:=nil;
  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_lista_alumnos(lista_alumnos);
  if (lista_alumnos<>nil) then
  begin
    procesar_lista_alumnos(lista_alumnos,ingresantes_parcial,ingresantes_total,recursantes_par
cial,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 TP6_E17c;
{$codepage UTF8}
uses crt;
const
  tanque_r='R'; tanque_c='C';
  tanque_salida='Z';
  alto_corte=1.40;
  volumen_corte=800.0;
type
  t_registro_tanque=record
    tanque: char;
    radio: real;
    alto: real;
    ancho: real;
    largo: real;
    volumen: real;
  end;
  t_lista_tanques=^t_nodo_tanques;
  t_nodo_tanques=record
```

```
    ele: t_registro_tanque;
    sig: t_lista_tanques;
  end;
procedure leer_tanque(var registro_tanque: t_registro_tanque);
begin
  textcolor(green); write('Introducir tipo de tanque vendido (R o C) por el fabricante: ');
  textcolor(yellow); readln(registro_tanque.tanque);
  if (registro_tanque.tanque<>tanque_salida) then
  begin
    if (registro_tanque.tanque=tanque_c) then
    begin
      textcolor(green); write('Introducir radio del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
      textcolor(yellow); readln(registro_tanque.radio);
      textcolor(green); write('Introducir alto del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
      textcolor(yellow); readln(registro_tanque.alto);
      registro_tanque.volumen:=pi*registro_tanque.radio*registro_tanque.radio*registro_tanque.
alto;
    end
    else
    begin
      textcolor(green); write('Introducir ancho del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
      textcolor(yellow); readln(registro_tanque.ancho);
      textcolor(green); write('Introducir largo del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
      textcolor(yellow); readln(registro_tanque.largo);
      textcolor(green); write('Introducir alto del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
      textcolor(yellow); readln(registro_tanque.alto);
      registro_tanque.volumen:=registro_tanque.ancho*registro_tanque.largo*registro_tanque.alt
o;
    end;
  end;
end;
procedure agregar_adelante_lista_tanques(var lista_tanques: t_lista_tanques; registro_tanque:
t_registro_tanque);
var
  nuevo: t_lista_tanques;
begin
  new(nuevo);
  nuevo^.ele:=registro_tanque;
  nuevo^.sig:=lista_tanques;
  lista_tanques:=nuevo;
end;
procedure cargar_lista_tanques(var lista_tanques: t_lista_tanques);
var
  registro_tanque: t_registro_tanque;
begin
  leer_tanque(registro_tanque);
  while (registro_tanque.tanque<>tanque_salida) do
  begin
    agregar_adelante_lista_tanques(lista_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_lista_tanques(lista_tanques: t_lista_tanques; var volumen_max1,
volumen_max2, volumen_total_c, volumen_total_r: real; var tanques_c, tanques_r,
tanques_corte_alto, tanques_corte_volumen: int16);
begin
  while (lista_tanques<>nil) do
  begin
    calcular_a(lista_tanques^.ele.volumen,volumen_max1,volumen_max2);
    calcular_bc(lista_tanques^.ele.tanque,lista_tanques^.ele.volumen,volumen_total_c,volumen_t
otal_r,tanques_c,tanques_r);
    calcular_d(lista_tanques^.ele.alto,tanques_corte_alto);
    calcular_e(lista_tanques^.ele.volumen,tanques_corte_volumen);
    lista_tanques:=lista_tanques^.sig;
  end;
end;
var
  lista_tanques: t_lista_tanques;
  tanques_c, tanques_r, tanques_corte_alto, tanques_corte_volumen: int16;
  volumen_max1, volumen_max2, volumen_total_c, volumen_total_r: real;
begin
  lista_tanques:=nil;
  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_lista_tanques(lista_tanques);
  if (lista_tanques<>nil) then
  begin
    procesar_lista_tanques(lista_tanques,volumen_max1,volumen_max2,volumen_total_c,volumen_tot
al_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.
```