**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,precio\_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\*lista\_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\*lista\_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\_sonda.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\_construccion+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,email\_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\_alumnos[i]^.ele.transporte]+1;

        vector\_transportes2[vector\_alumnos[i]^.ele.transporte]:=vector\_transportes2[vector\_alumnos[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\_gasto,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,distancia\_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,distancia\_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.monto\_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^.ele.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,ingresantes\_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\_diez\_max2,legajo\_diez\_max1,legajo\_diez\_max2);

    calcular\_g(lista\_alumnos^.ele.notas\_cero,lista\_alumnos^.ele.legajo,notas\_cero\_max1,notas\_cero\_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\_parcial,recursantes\_total,alumnos\_autoeva,alumnos\_corte,alumnos\_cero,legajo\_diez\_max1,legajo\_diez\_max2,legajo\_cero\_max1,legajo\_cero\_max2);

    if (ingresantes\_total>0) then

    begin

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

    end

    else

    begin

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

    end;

    if (recursantes\_total>0) then

    begin

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

    end

    else

    begin

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

    end;

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

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

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

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

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

  end

  else

  begin

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

  end;

end.

Ejercicio 3:

program 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.alto;

    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\_total\_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\_total\_r,tanques\_c,tanques\_r,tanques\_corte\_alto,tanques\_corte\_volumen);

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

    if (tanques\_c>0) then

    begin

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

    end

    else

    begin

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

    end;

    if (tanques\_r>0) then

    begin

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

    end

    else

    begin

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

    end;

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

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

  end

  else

  begin

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

  end;

end.