

Trabajo Práctico N° 0: **Introducción a la Programación.**

Ejercicio 1.

Implementar un programa que lea por teclado dos números enteros e imprima en pantalla los valores leídos en orden inverso. Por ejemplo, si se ingresan los números 4 y 8, debe mostrar el mensaje: Se ingresaron los valores 8 y 4.

```
program TP0_E1;
{$codepage UTF8}
uses crt;
var
  num1, num2: int16;
begin
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num1);
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num2);
  textcolor(green); write('Se ingresaron los valores '); textcolor(red); write(num2);
  textcolor(green); write(' y '); textcolor(red); write(num1);
end.
```

Ejercicio 2.

Modificar el programa anterior para que el mensaje de salida muestre la suma de ambos números:

(a) Utilizando una variable adicional.

```
program TP0_E2a;
{$codepage UTF8}
uses crt;
var
    num1, num2: int16;
    suma: int32;
begin
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num1);
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num2);
    suma:=num1+num2;
    textcolor(green); write('La suma de los valores ingresados es '); textcolor(red);
    write(suma);
end.
```

(b) Sin utilizar una variable adicional.

```
program TP0_E2b;
{$codepage UTF8}
uses crt;
var
    num1, num2: int16;
begin
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num1);
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num2);
    textcolor(green); write('La suma de los valores ingresados es '); textcolor(red);
    write(num1+num2);
end.
```

Ejercicio 3.

Implementar un programa que lea dos números reales e imprima el resultado de la división de los mismos con una precisión de dos decimales. Por ejemplo, si se ingresan los valores 4,5 y 7,2, debe imprimir: “El resultado de dividir 4,5 por 7,2 es 0,62”.

```
program TP0_E3;
{$codepage UTF8}
uses crt;
var
  num1, num2: real;
begin
  textcolor(green); write('Introducir número real como dividendo: ');
  textcolor(yellow); readln(num1);
  textcolor(green); write('Introducir número real como divisor: ');
  textcolor(yellow); readln(num2);
  textcolor(green); write('El resultado de dividir '); textcolor(red); write(num1:0:2);
  textcolor(green); write(' por '); textcolor(red); write(num2:0:2); textcolor(green); write('
es '); textcolor(red); write(num1/num2:0:2);
end.
```

Ejercicio 4.

Implementar un programa que lea el diámetro D de un círculo e imprima:

- El radio (R) del círculo (la mitad del diámetro).
- El área del círculo. Para calcular el área de un círculo, se debe utilizar la fórmula $PI * R^2$.
- El perímetro del círculo. Para calcular el perímetro del círculo, se debe utilizar la fórmula $D * PI$ (o también $R * 2 * PI$).

```
program TP0_E4;
{$codepage UTF8}
uses crt;
var
  diametro, radio, area, perimetro: real;
begin
  textcolor(green); write('Introducir diámetro de un círculo: ');
  textcolor(yellow); readln(diametro);
  radio:=diametro/2;
  textcolor(green); write('El radio del círculo es '); textcolor(red); writeln(radio:0:2);
  area:=pi*sqr(diametro/2);
  textcolor(green); write('El área del círculo es '); textcolor(red); writeln(area:0:2);
  perimetro:=pi*diametro;
  textcolor(green); write('El perímetro del círculo es '); textcolor(red);
  write(perimetro:0:2);
end.
```

Ejercicio 5.

Un kiosquero debe vender una cantidad X de caramelos entre Y clientes, dividiendo cantidades iguales entre todos los clientes. Los que le sobren se los quedará para él.

- Realizar un programa que lea la cantidad de caramelos que posee el kiosquero (X), la cantidad de clientes (Y) e imprima en pantalla un mensaje informando la cantidad de caramelos que le corresponderá a cada cliente y la cantidad de caramelos que se quedará para sí mismo.
- Imprimir en pantalla el dinero que deberá cobrar el kiosquero si cada caramelo tiene un valor de \$1,60.

```
program TP0_E5;
{$codepage UTF8}
uses crt;
const
    precio=1.6;
var
    tot_caramelos, tot_clientes, caramelos_cliente, caramelos_kiosquero, caramelos_vendidos:
    int16;
begin
    textcolor(green); write('Introducir cantidad de caramelos que posee el kiosquero: ');
    textcolor(yellow); readln(tot_caramelos);
    textcolor(green); write('Introducir cantidad de clientes que tiene el kiosquero: ');
    textcolor(yellow); readln(tot_clientes);
    caramelos_cliente:=tot_caramelos div tot_clientes;
    caramelos_kiosquero:=tot_caramelos mod tot_clientes;
    caramelos_vendidos:=tot_caramelos-caramelos_kiosquero;
    textcolor(green); write('La cantidad de caramelos que le corresponderá a cada cliente es ');
    textcolor(red); writeln(caramelos_cliente);
    textcolor(green); write('La cantidad de caramelos que se quedará el kiosquero es ');
    textcolor(red); writeln(caramelos_kiosquero);
    textcolor(green); write('El dinero que deberá cobrar el kiosquero si cada caramelo tiene un
    valor de $'); textcolor(yellow); write(precio:0:2); textcolor(green); write(' es $');
    textcolor(red); write(caramelos_vendidos*precio:0:2);
end.
```

Ejercicio 6.

Realizar un programa que informe el valor total en pesos de una transacción en dólares. Para ello, el programa debe leer el monto total en dólares de la transacción, el valor del dólar al día de la fecha y el porcentaje (en pesos) de la comisión que cobra el banco por la transacción. Por ejemplo, si la transacción se realiza por 10 dólares, el dólar tiene un valor de 189,32 pesos y el banco cobra un 4% de comisión, entonces, el programa deberá informar: “La transacción será de 1968,93 pesos argentinos” (resultado de multiplicar $10 * 189,32$ y adicionarle el 4%).

```
program TP0_E6;
{$codepage UTF8}
uses crt;
var
    monto_dolares, valor_dolar, comision, monto_pesos: real;
begin
    textcolor(green); write('Introducir monto total en dólares de la transacción: ');
    textcolor(yellow); readln(monto_dolares);
    textcolor(green); write('Introducir valor del dólar al día de la fecha: ');
    textcolor(yellow); readln(valor_dolar);
    textcolor(green); write('Introducir valor de la comisión que cobra el banco por la
transacción (en porcentaje): ');
    textcolor(yellow); readln(comision);
    monto_pesos:=monto_dolares*valor_dolar*(1+comision/100);
    textcolor(green); write('La transacción será de '); textcolor(red); write(monto_pesos:0:2);
    textcolor(green); write(' pesos argentinos');
end.
```

Trabajo Práctico N° 1.1: **Estructuras de Control (if y while).**

Ejercicio 1.

Realizar un programa que lea 2 números enteros desde teclado e informe en pantalla cuál de los dos números es el mayor. Si son iguales, debe informar en pantalla lo siguiente: “Los números leídos son iguales”.

```
program TP1_E1;
{$codepage UTF8}
uses crt;
var
  num1, num2: int16;
begin
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num1);
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num2);
  if (num1>num2) then
  begin
    textcolor(green); write('El número mayor es '); textcolor(red); write(num1);
  end
  else
    if (num2>num1) then
    begin
      textcolor(green); write('El número mayor es '); textcolor(red); write(num2);
    end
    else
    begin
      textcolor(red); write('Los números leídos son iguales');
    end
  end
end.
```

Ejercicio 2.

Realizar un programa que lea un número real e imprima su valor absoluto. El valor absoluto de un número X se escribe $|X|$ y se define como: $|X| = X$ cuando X es mayor o igual a cero; $|X| = -X$ cuando X es menor a cero.

```
program TP1_E2;
{$codepage UTF8}
uses crt;
var
  num: real;
begin
  textcolor(green); write('Introducir número real: ');
  textcolor(yellow); readln(num);
  if (num<0) then
    num:=-num;
  textcolor(green); write('El valor absoluto del número es '); textcolor(red); write(num:0:2);
end.
```


Ejercicio 3.

Realizar un programa que lea 3 números enteros y los imprima en orden descendente. Por ejemplo, si se ingresan los valores 4, -10 y 12, deberá imprimir: “12 4 -10”.

```
program TP1_E3;
{$codepage UTF8}
uses crt;
var
  num1, num2, num3: int16;
begin
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num1);
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num2);
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num3);
  if ((num1>=num3) and (num2>=num3)) then
  begin
    textcolor(green); write('El ordenamiento descendente es '); textcolor(red); write(num1);
    write(' '); write(num2); write(' '); write(num3);
  end
  else if ((num1>=num3) and (num3>=num2)) then
  begin
    textcolor(green); write('El ordenamiento descendente es '); textcolor(red); write(num1);
    write(' '); write(num3); write(' '); write(num2);
  end
  else if ((num2>=num1) and (num1>=num3)) then
  begin
    textcolor(green); write('El ordenamiento descendente es '); textcolor(red); write(num2);
    write(' '); write(num1); write(' '); write(num3);
  end
  else if ((num2>=num3) and (num3>=num1)) then
  begin
    textcolor(green); write('El ordenamiento descendente es '); textcolor(red); write(num2);
    write(' '); write(num3); write(' '); write(num1);
  end
  else if ((num3>=num1) and (num1>=num2)) then
  begin
    textcolor(green); write('El ordenamiento descendente es '); textcolor(red); write(num3);
    write(' '); write(num1); write(' '); write(num2);
  end
  else
  begin
    textcolor(green); write('El ordenamiento descendente es '); textcolor(red); write(num3);
    write(' '); write(num2); write(' '); write(num1);
  end;
end.
```

Ejercicio 4.

Realizar un programa que lea un número real X . Luego, deberá leer números reales hasta que se ingrese uno cuyo valor sea, exactamente, el doble de X (el primer número leído).

```
program TP1_E4;
{$codepage UTF8}
uses crt;
const
    multiplo=2;
var
    i: int16;
    num1, num2: real;
begin
    i:=1;
    textcolor(green); write('Introducir número real: ');
    textcolor(yellow); readln(num1);
    textcolor(green); write('Introducir otro número real ',i,': ');
    textcolor(yellow); readln(num2);
    while (num2<>(multiplo*num1)) do
    begin
        i:=i+1;
        textcolor(green); write('Introducir otro número real ',i,': ');
        textcolor(yellow); readln(num2);
    end;
    textcolor(green); write('El número introducido ('); textcolor(red); write(num2:0:2);
textcolor(green); write(') es igual al inicial ('); textcolor(red); write(num1:0:2);
textcolor(green); write(') multiplicado por '); textcolor(red); write(multiplo);
end.
```

Ejercicio 5.

Modificar el ejercicio anterior para que, luego de leer el número X , se lean, a lo sumo, 10 números reales. La lectura deberá finalizar al ingresar un valor que sea el doble de X o al leer el décimo número, en cuyo caso deberá informarse: “No se ha ingresado el doble de X ”.

```
program TP1_E5;
{$codepage UTF8}
uses crt;
const
  num_total=10;
  multiplo=2;
var
  i: int8;
  num1, num2: real;
begin
  i:=1;
  textcolor(green); write('Introducir número real: ');
  textcolor(yellow); readln(num1);
  textcolor(green); write('Introducir otro número real ',i,': ');
  textcolor(yellow); readln(num2);
  while ((num2<>(num1*multiplo)) and (i<num_total)) do
  begin
    i:=i+1;
    textcolor(green); write('Introducir otro número real ',i,': ');
    textcolor(yellow); readln(num2);
  end;
  if (num2=(num1*multiplo)) then
  begin
    textcolor(green); write('El número introducido ('); textcolor(red); write(num2:0:2);
    textcolor(green); write(') es igual al inicial ('); textcolor(red); write(num1:0:2);
    textcolor(green); write(') multiplicado por '); textcolor(red); write(multiplo);
  end
  else
  begin
    textcolor(green); write('No se ha ingresado el doble de '); textcolor(red);
    write(num1:0:2);
  end;
end.
```

Ejercicio 6.

Realizar un programa que lea el número de legajo y el promedio de cada alumno de la facultad. La lectura finaliza cuando se ingresa el legajo -1, que no debe procesarse. Por ejemplo, se lee la siguiente secuencia: 33423, 8.40, 19003, 6.43, -1. En el ejemplo anterior, se leyó el legajo 33422, cuyo promedio fue 8.40, luego se leyó el legajo 19003, cuyo promedio fue 6.43 y, finalmente, el legajo -1 (para el cual no es necesario leer un promedio). Al finalizar la lectura, informar:

- La cantidad de alumnos leída (en el ejemplo anterior, se debería informar 2).
- La cantidad de alumnos cuyo promedio supera 6.5 (en el ejemplo anterior, se debería informar 1).
- El porcentaje de alumnos destacados (alumnos con promedio mayor a 8.5) cuyos legajos sean menor al valor 2500 (en el ejemplo anterior, se debería informar 0%).

```
program TP1_E6;
{$codepage UTF8}
uses crt;
const
  legajo_salida=-1;
  promedio_corte1=6.5;
  promedio_corte2=8.5;
  legajo_corte=2500;
var
  legajo, alumnos_total, alumnos_corte1, alumnos_corte2: int16;
  promedio, alumnos_corte2_porc: real;
begin
  alumnos_total:=0;
  alumnos_corte1:=0;
  alumnos_corte2:=0; alumnos_corte2_porc:=0;
  textcolor(green); write('Introducir número de legajo: ');
  textcolor(yellow); readln(legajo);
  while (legajo<>legajo_salida) do
  begin
    textcolor(green); write('Introducir promedio de ese legajo: ');
    textcolor(yellow); readln(promedio);
    alumnos_total:=alumnos_total+1;
    if (promedio>promedio_corte1) then
      alumnos_corte1:=alumnos_corte1+1;
    if ((promedio>promedio_corte2) and (legajo<legajo_corte)) then
      alumnos_corte2:=alumnos_corte2+1;
    textcolor(green); write('Introducir número de legajo: ');
    textcolor(yellow); readln(legajo);
  end;
  alumnos_corte2_porc:=alumnos_corte2/alumnos_total*100;
  textcolor(green); write('La cantidad de alumnos leída es '); textcolor(red);
  write(alumnos_total);
  textcolor(green); write('La cantidad de alumnos con promedio superior a ');
  textcolor(yellow); write(promedio_corte1:0:2); textcolor(green); write(' es ');
  textcolor(red); write(alumnos_corte1);
  textcolor(green); write('El porcentaje de alumnos destacados (alumnos con promedio mayor a ');
  textcolor(yellow); write(promedio_corte2:0:2); textcolor(green); write(') cuyos legajos
son menor al valor '); textcolor(red); write(legajo_corte); textcolor(green); write(' es del
'); textcolor(red); write(alumnos_corte2_porc:0:2); textcolor(green); write('%')
end.
```

Ejercicio 7.

Realizar un programa que lea el código, el precio actual y el nuevo precio de los productos de un almacén. La lectura finaliza al ingresar el producto con el código 32767, el cual debe procesarse. Para cada producto leído, el programa deberá indicar si el nuevo precio del producto supera en un 10% al precio anterior.

Por ejemplo:

- Si se ingresa el código 10382, con precio actual 40 y nuevo precio 44, deberá imprimir: "El aumento de precio del producto 10382 no supera el 10%".
- Si se ingresa el código 32767, con precio actual 30 y nuevo precio 33,01, deberá imprimir: "El aumento de precio del producto 32767 es superior al 10%".

```

program TP1_E7;
{$codepage UTF8}
uses crt;
const
    producto_salida=32767;
    porcentaje_corte=10.0;
var
    producto: int16;
    precio_anterior, precio_nuevo, variacion: real;
begin
    producto:=0;
    while (producto<>producto_salida) do
    begin
        textcolor(green); write('Introducir código de producto del producto: ');
        textcolor(yellow); readln(producto);
        textcolor(green); write('Introducir precio actual del producto: ');
        textcolor(yellow); readln(precio_anterior);
        textcolor(green); write('Introducir precio nuevo del producto: ');
        textcolor(yellow); readln(precio_nuevo);
        variacion:=(precio_nuevo/precio_anterior-1)*100;
        if (variacion<=porcentaje_corte) then
        begin
            textcolor(green); write('El aumento de precio del producto '); textcolor(red);
            write(producto); textcolor(green); write(' no supera el '); textcolor(yellow);
            write(porcentaje_corte:0:2); textcolor(green); writeln('%');
        end
        else
        begin
            textcolor(green); write('El aumento de precio del producto '); textcolor(red);
            write(producto); textcolor(green); write(' es superior al '); textcolor(yellow);
            write(porcentaje_corte:0:2); textcolor(green); writeln('%');
        end;
    end;
end.

```

Ejercicio 8.

Realizar un programa que lea tres caracteres e informe si los tres eran letras vocales o si, al menos, uno de ellos no lo era. Por ejemplo, si se leen los caracteres “a e o”, deberá informar “Los tres caracteres son vocales” y, si se leen los caracteres “z a g”, deberá informar “Al menos un caracter no era vocal”.

```
program TP1_E8;
{$codepage UTF8}
uses crt;
var
  vocales: int8;
  letra1, letra2, letra3: char;
begin
  vocales:=0;
  textcolor(green); write('Introducir letra: ');
  textcolor(yellow); readln(letra1);
  textcolor(green); write('Introducir letra: ');
  textcolor(yellow); readln(letra2);
  textcolor(green); write('Introducir letra: ');
  textcolor(yellow); readln(letra3);
  if ((letra1='a') or (letra1='e') or (letra1='i') or (letra1='o') or (letra1='u')) then
    vocales:=vocales+1;
  if ((letra2='a') or (letra2='e') or (letra2='i') or (letra2='o') or (letra2='u')) then
    vocales:=vocales+1;
  if ((letra3='a') or (letra3='e') or (letra3='i') or (letra3='o') or (letra3='u')) then
    vocales:=vocales+1;
  if (vocales=3) then
  begin
    textcolor(red); write('Los tres caracteres son vocales');
  end
  else
  begin
    textcolor(red); write('Al menos un caracter no es vocal');
  end;
end.
```

Ejercicio 9.

Realizar un programa que lea un caracter, que puede ser “+” (suma) o “-” (resta); si se ingresa otro caracter, debe informar un error y finalizar. Una vez leído el caracter de suma o resta, deberá leerse una secuencia de números enteros que finaliza con 0. El programa deberá aplicar la operación leída con la secuencia de números e imprimir el resultado final. Por ejemplo:

- Si se lee el caracter “-” y la secuencia 4 3 5 -6 0, deberá imprimir: $2 = (4 - 3 - 5 - (-6))$.
- Si se lee el caracter “+” y la secuencia -10 5 6 -1 0, deberá imprimir $0 = (-10 + 5 + 6 + (-1))$.

```

program TP1_E9;
{$codepage UTF8}
uses crt;
const
    num_salida=0;
var
    num, total: int16;
    operacion: char;
begin
    textcolor(green); write('Seleccionar operación ("+" o "-"): ');
    textcolor(yellow); readln(operacion);
    if ((operacion='+') or (operacion='-')) then
    begin
        textcolor(green); write('Introducir número entero: ');
        textcolor(yellow); readln(num);
        total:=0;
        while (num<>num_salida) do
        begin
            if (operacion='+') then
                total:=total+num
            else
                total:=total-num;
            textcolor(green); write('Introducir número entero: ');
            textcolor(yellow); readln(num);
        end;
        textcolor(green); write('El resultado de la operación es '); textcolor(red); write(total);
    end
    else
    begin
        textcolor(red); write('ERROR. La operación es inválida')
    end;
end.

```

Trabajo Práctico N° 1.2: **Estructuras de Control (for y repeat-until).**

Ejercicio 1.

(a) *Realizar un programa que lea 10 números enteros e informe la suma total de los números leídos.*

```
program TP1_E1a;
{$codepage UTF8}
uses crt;
const
  num_total=10;
var
  i: int8;
  num: int16;
  suma: int32;
begin
  suma:=0;
  for i:= 1 to num_total do
  begin
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num);
    suma:=suma+num;
  end;
  textcolor(green); write('La suma total de los números leídos es '); textcolor(red);
  write(suma);
end.
```

(b) *Modificar el ejercicio anterior para que, además, informe la cantidad de números mayores a 5.*

```
program TP1_E1b;
{$codepage UTF8}
uses crt;
const
  num_total=10;
  num_corte=5;
var
  i, nums_corte: int8;
  num: int16;
  suma: int32;
begin
  suma:=0;
  nums_corte:=0;
  for i:= 1 to num_total do
  begin
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num);
    suma:=suma+num;
    if (num>num_corte) then
      nums_corte:=nums_corte+1;
    end;
  end;
  textcolor(green); write('La suma total de los números leídos es '); textcolor(red);
  writeln(suma);
  textcolor(green); write('La cantidad de números leídos mayores a '); textcolor(yellow);
  write(num_corte); textcolor(green); write(' es '); textcolor(red); write(nums_corte);
```


end.

Ejercicio 2.

(a) Realizar un programa que lea 10 números e informe cuál fue el mayor número leído. Por ejemplo, si se lee la secuencia 3 5 6 2 3 10 98 8 -12 9, deberá informar: “El mayor número leído fue el 98”.

```
program TP1_E2a;
{$codepage UTF8}
uses crt;
const
    num_total=10;
var
    i: int8;
    num, max: int16;
begin
    max:=low(int16);
    for i:= 1 to num_total do
    begin
        textcolor(green); write('Introducir número entero: ');
        textcolor(yellow); readln(num);
        if (num>max) then
            max:=num;
        end;
        textcolor(green); write('El mayor número leído fue el '); textcolor(red); write(max);
    end.
end.
```

(b) Modificar el programa anterior para que, además de informar el mayor número leído, se informe el número de orden, dentro de la secuencia, en el que fue leído. Por ejemplo, si se lee la misma secuencia 3 5 6 2 3 10 98 8 -12 9, deberá informar: “El mayor número leído fue el 98, en la posición 7”.

```
program TP1_E2b;
{$codepage UTF8}
uses crt;
const
    num_total=10;
var
    i, pos: int8;
    num, max: int16;
begin
    max:=low(int16);
    for i:= 1 to num_total do
    begin
        textcolor(green); write('Introducir número entero: ');
        textcolor(yellow); readln(num);
        if (num>max) then
        begin
            max:=num;
            pos:=i;
        end;
    end;
    textcolor(green); write('El mayor número leído fue el '); textcolor(red); write(max);
    textcolor(green); write(', en la posición '); textcolor(red); write(pos);
end.
end.
```

Ejercicio 3.

Realizar un programa que lea desde teclado la información de alumnos ingresantes a la carrera Analista en TIC. De cada alumno, se lee nombre y nota obtenida en el módulo EPA (la nota es un número entre 1 y 10). La lectura finaliza cuando se lee el nombre “Zidane Zinedine”, que debe procesarse. Al finalizar la lectura, informar:

- La cantidad de alumnos aprobados (nota 8 o mayor).
- La cantidad de alumnos que obtuvieron un 7 como nota.

```
program TP1_E3;
{$codepage UTF8}
uses crt;
const
  nombre_salida='Zidane Zinedine';
  nota_corte1=8;
  nota_corte2=7;
var
  nota, alumnos_corte1, alumnos_corte2: int8;
  nombre: string;
begin
  alumnos_corte1:=0;
  alumnos_corte2:=0;
  repeat
    textcolor(green); write('Introducir apellido y nombre del alumno: ');
    textcolor(yellow); readln(nombre);
    textcolor(green); write('Introducir nota obtenida en el módulo de EPA por el alumno: ');
    textcolor(yellow); readln(nota);
    if (nota>=nota_corte1) then
      alumnos_corte1:=alumnos_corte1+1
    else
      if (nota=nota_corte2) then
        alumnos_corte2:=alumnos_corte2+1;
    until (nombre=nombre_salida);
    textcolor(green); write('La cantidad de alumnos aprobados (nota 8 o mayor) es ');
    textcolor(red); writeln(alumnos_corte1);
    textcolor(green); write('La cantidad de alumnos que obtuvieron un 7 como nota es ');
    textcolor(red); write(alumnos_corte2);
  end.
```

Ejercicio 4.

(a) Realizar un programa que lea 1000 números enteros desde teclado. Informar en pantalla cuáles son los dos números mínimos leídos.

```
program TP1_E4a;
{$codepage UTF8}
uses crt;
const
    num_total=1000;
var
    i, num, min1, min2: int16;
begin
    min1:=high(int16); min2:=high(int16);
    for i:= 1 to num_total do
        begin
            textcolor(green); write('Introducir número entero: ');
            textcolor(yellow); readln(num);
            if (num<min1) then
                begin
                    min2:=min1;
                    min1:=num;
                end
            else
                if (num<min2) then
                    min2:=num;
                end;
            textcolor(green); write('Los dos números mínimos leídos son '); textcolor(red); write(min1);
            textcolor(green); write(' y '); textcolor(red); write(min2);
        end.
end.
```

(b) Modificar el ejercicio anterior para que, en vez de leer 1000 números, la lectura finalice al leer el número 0, el cual debe procesarse.

```
program TP1_E4b;
{$codepage UTF8}
uses crt;
const
    num_salida=0;
var
    num, min1, min2: int16;
begin
    min1:=high(int16); min2:=high(int16);
    repeat
        textcolor(green); write('Introducir número entero: ');
        textcolor(yellow); readln(num);
        if (num<min1) then
            begin
                min2:=min1;
                min1:=num;
            end
        else
            if (num<min2) then
                min2:=num;
            end;
    until (num=num_salida);
    textcolor(green); write('Los dos números mínimos leídos son '); textcolor(red); write(min1);
    textcolor(green); write(' y '); textcolor(red); write(min2);
end.
```

(c) Modificar el ejercicio anterior para que, en vez de leer 1000 números, la lectura finalice al leer el número 0, el cual no debe procesarse.

```
program TP1_E4c;
{$codepage UTF8}
uses crt;
const
  num_salida=0;
var
  num, min1, min2: int16;
begin
  min1:=high(int16); min2:=high(int16);
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num);
  while (num<>num_salida) do
  begin
    if (num<min1) then
    begin
      min2:=min1;
      min1:=num;
    end
    else
      if (num<min2) then
      begin
        min2:=num;
        textcolor(green); write('Introducir número entero: ');
        textcolor(yellow); readln(num);
      end;
    textcolor(green); write('Los dos números mínimos leídos son '); textcolor(red); write(min1);
    textcolor(green); write(' y '); textcolor(red); write(min2);
  end.
```

Ejercicio 5.

Realizar un programa que lea números enteros desde teclado. La lectura debe finalizar cuando se ingrese el número 100, el cual debe procesarse. Informar en pantalla:

- El número máximo leído.
- El número mínimo leído.
- La suma total de los números leídos.

```
program TP1_E5;
{$codepage UTF8}
uses crt;
const
  num_salida=100;
var
  num, num_max, num_min, suma: int16;
begin
  num_max:=low(int16);
  num_min:=high(int16);
  suma:=0;
  repeat
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num);
    if (num>num_max) then
      num_max:=num;
    if (num<num_min) then
      num_min:=num;
    suma:=suma+num;
  until (num=num_salida);
  textcolor(green); write('El número máximo leído es '); textcolor(red); writeln(num_max);
  textcolor(green); write('El número mínimo leído es '); textcolor(red); writeln(num_min);
  textcolor(green); write('La suma total de los números leídos es '); textcolor(red);
  write(suma);
end.
```

Ejercicio 6.

Realizar un programa que lea información de 200 productos de un supermercado. De cada producto, se lee código y precio (cada código es un número entre 1 y 200). Informar en pantalla:

- Los códigos de los dos productos más baratos.
- La cantidad de productos de más de 16 pesos con código par.

```
program TP1_E6;
{$codepage UTF8}
uses crt;
const
  productos_total=200;
  precio_corte=16.0;
var
  i, producto, producto_min1, producto_min2, productos_corte: int16;
  precio, precio_min1, precio_min2: real;
begin
  precio_min1:=9999999; precio_min2:=9999999; producto_min1:=0; producto_min2:=0;
  productos_corte:=0;
  for i:= 1 to productos_total do
    begin
      textcolor(green); write('Introducir código de producto del producto: ');
      textcolor(yellow); readln(producto);
      textcolor(green); write('Introducir precio del producto: ');
      textcolor(yellow); readln(precio);
      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;
        if ((precio>precio_corte) and (producto mod 2=0)) then
          productos_corte:=productos_corte+1;
        end;
      textcolor(green); write('Los códigos de los dos productos más baratos son ');
      textcolor(red); write(producto_min1); textcolor(green); write(' y '); textcolor(red);
      writeln(producto_min2);
      textcolor(green); write('La cantidad de productos de más de '); textcolor(yellow);
      write(precio_corte:0:2); textcolor(green); write(' pesos con código par es '); textcolor(red);
      write(productos_corte);
    end.
end.
```

Ejercicio 7.

Realizar un programa que lea desde teclado información de autos de carrera. Para cada uno de los autos, se lee el nombre del piloto y el tiempo total que le tomó finalizar la carrera. En la carrera, participaron 100 autos. Informar en pantalla:

- Los nombres de los dos pilotos que finalizaron en los dos primeros puestos.
- Los nombres de los dos pilotos que finalizaron en los dos últimos puestos.

```
program TP1_E7;
{$codepage UTF8}
uses crt;
const
  autos_total=100;
var
  tiempo, tiempo_min1, tiempo_min2, tiempo_max1, tiempo_max2: int8;
  i: int16;
  nombre, nombre_min1, nombre_min2, nombre_max1, nombre_max2: string;
begin
  tiempo_min1:=high(int8); tiempo_min2:=high(int8); nombre_min1:=''; nombre_min2:='';
  tiempo_max1:=low(int8); tiempo_max2:=low(int8); nombre_max1:=''; nombre_max2:='';
  for i:= 1 to autos_total do
  begin
    textcolor(green); write('Introducir nombre del piloto: ');
    textcolor(yellow); readln(nombre);
    textcolor(green); write('Introducir tiempo total que le tomó finalizar la carrera al
piloto: ');
    textcolor(yellow); readln(tiempo);
    if (tiempo<tiempo_min1) then
    begin
      tiempo_min2:=tiempo_min1;
      nombre_min2:=nombre_min1;
      tiempo_min1:=tiempo;
      nombre_min1:=nombre;
    end
    else
      if (tiempo<tiempo_min2) then
      begin
        tiempo_min2:=tiempo;
        nombre_min2:=nombre;
      end;
    if (tiempo>tiempo_max1) then
    begin
      tiempo_max2:=tiempo_max1;
      nombre_max2:=nombre_max1;
      tiempo_max1:=tiempo;
      nombre_max1:=nombre;
    end
    else
      if (tiempo>tiempo_max2) then
      begin
        tiempo_max2:=tiempo;
        nombre_max2:=nombre;
      end;
    end;
    textcolor(green); write('Los nombres de los dos pilotos que finalizaron en los dos primeros
puestos son '); textcolor(red); write(nombre_min1); textcolor(green); write(' y ');
    textcolor(red); writeln(nombre_min2);
    textcolor(green); write('Los nombres de los dos pilotos que finalizaron en los dos últimos
puestos son '); textcolor(red); write(nombre_max2); textcolor(green); write(' y ');
    textcolor(red); write(nombre_max1);
  end.
end.
```


Ejercicio 8.

(a) Un local de ropa desea analizar las ventas realizadas en el último mes. Para ello, se lee por cada día del mes, los montos de las ventas realizadas. La lectura de montos para cada día finaliza cuando se lee el monto 0. Se asume un mes de 31 días. Informar la cantidad de ventas por cada día y el monto total acumulado en ventas de todo el mes.

```
program TP1_E8a;
{$codepage UTF8}
uses crt;
const
    monto_salida=0;
    dias_total=31;
var
    i, ventas_dia: int8;
    monto, monto_total: real;
begin
    monto_total:=0;
    for i:= 1 to dias_total do
        begin
            textcolor(green); write('Introducir monto de venta realizada el día ',i,' del mes: ');
            textcolor(yellow); readln(monto);
            ventas_dia:=0;
            while (monto<>monto_salida) do
                begin
                    ventas_dia:=ventas_dia+1;
                    monto_total:=monto_total+monto;
                    textcolor(green); write('Introducir monto de venta realizada el día ',i,' del mes: ');
                    textcolor(yellow); readln(monto);
                end;
            textcolor(green); write('La cantidad de ventas del día ',i,' del mes fue ');
            textcolor(red); writeln(ventas_dia);
            textcolor(green); write('El monto total acumulado en ventas de todo el mes fue $');
            textcolor(red); write(monto_total);
        end.
end.
```

(b) Modificar el ejercicio anterior para que, además, informe el día en el que se realizó la mayor cantidad de ventas.

```
program TP1_E8b;
{$codepage UTF8}
uses crt;
const
    monto_salida=0;
    dias_total=31;
var
    i, ventas_dia, ventas_max, dia_max: int8;
    monto, monto_total: real;
begin
    monto_total:=0;
    ventas_max:=low(int8); dia_max:=0;
    for i:= 1 to dias_total do
        begin
            textcolor(green); write('Introducir monto de venta realizada del día ',i,' del mes: ');
            textcolor(yellow); readln(monto);
            ventas_dia:=0;
            while (monto<>monto_salida) do
                begin
```

```
ventas_dia:=ventas_dia+1;
monto_total:=monto_total+monto;
textcolor(green); write('Introducir monto de venta realizada del día ',i,' del mes: ');
textcolor(yellow); readln(monto);
end;
textcolor(green); write('La cantidad de ventas del día ',i,' del mes fue ');
textcolor(red); writeln(ventas_dia);
if (ventas_dia>ventas_max) then
begin
ventas_max:=ventas_dia;
dia_max:=i;
end;
end;
textcolor(green); write('El monto total acumulado en ventas de todo el mes fue $');
textcolor(red); writeln(monto_total);
textcolor(green); write('El día en el que se realizó la mayor cantidad de ventas fue el ');
textcolor(red); write(dia_max); textcolor(green); write(' del mes');
end.
```

Trabajo Práctico N° 1.3: **Estructuras de Control (Adicionales).**

Ejercicio 1.

Realizar un programa que analice las inversiones de las empresas más grandes del país. Para cada empresa, se lee su código (un número entero), la cantidad de inversiones que tiene y el monto dedicado a cada una de las inversiones. La lectura finaliza al ingresar la empresa con código 100, que debe procesarse. El programa deberá informar:

- Para cada empresa, el monto promedio de sus inversiones.
- Código de la empresa con mayor monto total invertido.
- Cantidad de empresas con inversiones de más de \$50.000.

```

program TP1_E1;
{$codepage UTF8}
uses crt;
const
    empresa_salida=100;
    monto_corte=50000.0;
var
    i, empresa, inversiones, empresa_max, empresas_corte: int16;
    monto, monto_total, monto_max: real;
begin
    monto_max:=-9999999; empresa_max:=0;
    empresas_corte:=0;
    repeat
        textcolor(green); write('Introducir código de empresa de la empresa: ');
        textcolor(yellow); readln(empresa);
        textcolor(green); write('Introducir cantidad de inversiones de la empresa: ');
        textcolor(yellow); readln(inversiones);
        if (inversiones>0) then
            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;
                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);
                if (monto_total>monto_max) then
                    begin
                        monto_max:=monto_total;
                        empresa_max:=empresa;
                    end;
                if (monto_total>monto_corte) then
                    empresas_corte:=empresas_corte+1;
                end;
            until (empresa=empresa_salida);
        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.

La cátedra de CADP está analizando los resultados de las autoevaluaciones que realizaron los alumnos durante el cuatrimestre. Realizar un programa que lea, para cada alumno, su legajo, su condición (I para INGRESANTE, R para RECURSANTE) y la nota obtenida en cada una de las 5 autoevaluaciones. Si un alumno no realizó alguna autoevaluación en tiempo y forma, se le cargará la nota -1. La lectura finaliza al ingresar el legajo -1. Una vez ingresados todos los datos, el programa debe informar:

- Cantidad de alumnos INGRESANTES en condiciones de rendir el parcial y porcentaje sobre el total de alumnos INGRESANTES.
- Cantidad de alumnos RECURSANTES en condiciones de rendir el parcial y porcentaje sobre el total de alumnos RECURSANTES.
- Cantidad de alumnos que aprobaron todas las autoevaluaciones.
- Cantidad de alumnos cuya nota promedio fue mayor a 6.5 puntos.
- Cantidad de alumnos que obtuvieron cero puntos en, al menos, una autoevaluación.
- Código de los dos alumnos con mayor cantidad de autoevaluaciones con nota 10 (diez).
- Código de los dos alumnos con mayor cantidad de autoevaluaciones con nota 0 (cero).

Nota: Recordar que, para poder rendir el EXAMEN PARCIAL, el alumno deberá obtener “Presente” en, al menos, el 75% del total de las autoevaluaciones propuestas. Se considera “Presente” la autoevaluación que se entrega en tiempo y forma y con, al menos, el 40% de respuestas correctas.

```
program TP1_E2;
{$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;
var
    i, nota, presente, nota_total, notas_cero, notas_diez, notas_diez_max1, notas_diez_max2,
    notas_cero_max1, notas_cero_max2: int8;
    legajo, 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;
    condicion: char;
begin
    ingresantes_parcial:=0; ingresantes_total:=0;
    recursantes_parcial:=0; recursantes_total:=0;
    alumnos_autoeva:=0;
    alumnos_corte:=0;
    alumnos_cero:=0;
    notas_diez_max1:=0; notas_diez_max2:=0; legajo_diez_max1:=0; legajo_diez_max2:=0;
    notas_cero_max1:=0; notas_cero_max2:=0; legajo_cero_max1:=0; legajo_cero_max2:=0;
    textcolor(green); write('Introducir legajo del alumno: ');
    textcolor(yellow); readln(legajo);
    while (legajo<>legajo_salida) do
    begin
        {Introducir condición (I para INGRESANTE, R para RECURSANTE) del alumno}
```

```

textcolor(green); write('Introducir condición (I para INGRESANTE, R para RECURSANTE) del
alumno: ');
textcolor(yellow); readln(condicion);
{Nota obtenida por el alumno en cada una de las 5 autoevaluaciones}
presente:=0; nota_total:=0; notas_cero:=0; notas_diez:=0;
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;
{Cantidad de alumnos INGRESANTES en condiciones de rendir el parcial}
if (condicion=condicion_i) then
begin
  if (presente>=presente_corte*autoeva_total) then
    ingresantes_parcial:=ingresantes_parcial+1;
  ingresantes_total:=ingresantes_total+1;
end
{Cantidad de alumnos RECURSANTES en condiciones de rendir el parcial}
else
begin
  if (presente>=presente_corte*autoeva_total) then
    recursantes_parcial:=recursantes_parcial+1;
  recursantes_total:=recursantes_total+1;
end;
{Cantidad de alumnos que aprobaron todas las autoevaluaciones}
if (presente=autoeva_total) then
  alumnos_autoeva:=alumnos_autoeva+1;
{Cantidad de alumnos cuya nota promedio fue mayor a 6.5 puntos}
if (nota_total/autoeva_total>promedio_corte) then
  alumnos_corte:=alumnos_corte+1;
{Cantidad de alumnos que obtuvieron cero puntos en, al menos, una autoevaluación}
if (notas_cero>=1) then
  alumnos_cero:=alumnos_cero+1;
{Código de los dos alumnos con mayor cantidad de autoevaluaciones con nota 10 (diez)}
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;
{Código de los dos alumnos con mayor cantidad de autoevaluaciones con nota 0 (cero)}
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;
    {Introducir legajo del alumno}
    textcolor(green); write('Introducir legajo del alumno: ');
    textcolor(yellow); readln(legajo);
end;
if ((ingresantes_total>0) or (recursantes_total>0)) then
begin
    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.
end.
```

Ejercicio 3.

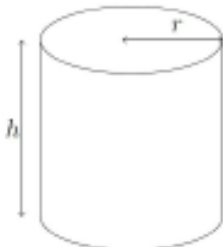
Un fabricante de tanques de agua está analizando las ventas de sus tanques durante el 2020. La empresa fabrica tanques a medida, que pueden ser rectangulares (tanques “R”) o cilíndricos (tanques “C”).

- De cada tanque R, se conoce su ancho (A), su largo (B) y su alto (C).
- De cada tanque C, se conoce su radio y su alto.

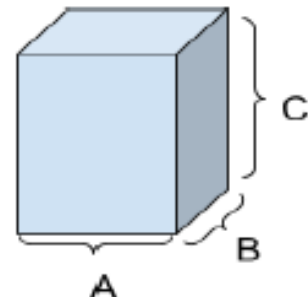
Todas las medidas se ingresan en metros. Realizar un programa que lea la información de los tanques vendidos por la empresa. La lectura finaliza al ingresar un tanque de tipo ‘Z’. Al finalizar la lectura, el programa debe informar:

- Volumen de los dos mayores tanques vendidos.
- Volumen promedio de todos los tanques cilíndricos vendidos.
- Volumen promedio de todos los tanques rectangulares vendidos.
- Cantidad de tanques cuyo alto sea menor a 1.40 metros.
- Cantidad de tanques cuyo volumen sea menor a 800 metros cúbicos.

Recordar: las fórmulas para el cálculo de volumen (V) del cilindro y del paralelepípedo rectangular son:



$$V = \pi \cdot r^2 \cdot h$$



$$V = A \cdot B \cdot C$$

```
program TP1_E3;
{$codepage UTF8}
uses crt;
const
    tanque_r='R'; tanque_c='C';
    tanque_salida='Z';
    alto_corte=1.40;
    volumen_corte=800.0;
var
    tanques_c, tanques_r, tanques_corte_alto, tanques_corte_volumen: int8;
    radio, alto, ancho, largo, volumen, volumen_max1, volumen_max2, volumen_total_c,
    volumen_total_r: real;
    tanque: char;
begin
    volumen_max1:=0; volumen_max2:=0;
    volumen_total_c:=0; tanques_c:=0;
    volumen_total_r:=0; tanques_r:=0;
    tanques_corte_alto:=0;
    tanques_corte_volumen:=0;
    textcolor(green); write('Introducir tipo de tanque vendido (R o C) por el fabricante: ');
    textcolor(yellow); readln(tanque);
    while (tanque<>tanque_salida) do
```

```
begin
  if (tanque=tanque_c) then
    begin
      textcolor(green); write('Introducir radio del tanque vendido ',tanque,' por el
fabricante: ');
      textcolor(yellow); readln(radio);
      textcolor(green); write('Introducir alto del tanque vendido ',tanque,' por el
fabricante: ');
      textcolor(yellow); readln(alto);
      volumen:=pi*radio*radio*alto;
      volumen_total_c:=volumen_total_c+volumen;
      tanques_c:=tanques_c+1;
    end
  else
    begin
      textcolor(green); write('Introducir ancho del tanque vendido ',tanque,' por el
fabricante: ');
      textcolor(yellow); readln(ancho);
      textcolor(green); write('Introducir largo del tanque vendido ',tanque,' por el
fabricante: ');
      textcolor(yellow); readln(largo);
      textcolor(green); write('Introducir alto del tanque vendido ',tanque,' por el
fabricante: ');
      textcolor(yellow); readln(alto);
      volumen:=ancho*largo*alto;
      volumen_total_r:=volumen_total_r+volumen;
      tanques_r:=tanques_r+1;
    end;
  if (volumen>volumen_max1) then
    begin
      volumen_max2:=volumen_max1;
      volumen_max1:=volumen;
    end
  else
    if (volumen>volumen_max2) then
      volumen_max2:=volumen;
    if (alto<alto_corte) then
      tanques_corte_alto:=tanques_corte_alto+1;
    if (volumen<volumen_corte) then
      tanques_corte_volumen:=tanques_corte_volumen+1;
    textcolor(green); write('Introducir tipo de tanque vendido (R o C) por el fabricante: ');
    textcolor(yellow); readln(tanque);
  end;
  if ((tanques_c>0) or (tanques_r>0)) then
    begin
      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;
    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 cilindricos (C) o rectangulares (R) vendidos');
end;
end.
```

Trabajo Práctico N° 2.1: **Funciones y Procedimientos.**

Ejercicio 1.

Dado el siguiente programa, indicar qué imprime:

```
program TP2_E1;  
{ $codepage UTF8 }  
uses crt;  
var  
  a, b: integer;  
procedure uno;  
var  
  b: integer;  
begin  
  b:=3;  
  writeln(b);  
end;  
begin  
  a:=1;  
  b:=2;  
  uno;  
  writeln(b,a);  
end.
```

Este programa imprime 3, 2 y 1.

Ejercicio 2.

Dado el siguiente programa, indicar qué imprime:

```
program TP2_E2;  
{ $codepage UTF8 }  
uses crt;  
var  
  a, b: integer;  
procedure uno;  
begin  
  b:=3;  
  writeln(b);  
end;  
begin  
  a:=1;  
  b:=2;  
  uno;  
  writeln(b,a);  
end.
```

Este programa imprime 3, 3 y 1.

Ejercicio 3.

Dado el siguiente programa, indicar cuál es error y su causa:

```
program TP2_E3;  
{$codepage UTF8}  
uses crt;  
var  
    a: integer;  
procedure uno;  
var  
    b: integer;  
begin  
    b:=2;  
    writeln(b);  
end;  
begin  
    a:=1;  
    uno;  
    writeln(b,a);  
end.
```

El error se da en el `writeln(b, a)`, ya que no se encuentra definida la variable global “b” en el programa.

Ejercicio 4.

Dados los siguientes programas, explicar las diferencias:

```
program TP2_E4a;  
{ $codepage UTF8 }  
uses crt;  
var  
  a, b: integer;  
procedure uno;  
begin  
  a:=1;  
  writeln(a);  
end;  
begin  
  a:=1;  
  b:=2;  
  uno;  
  writeln(b,a);  
end.
```

```
program TP2_E4b;  
{ $codepage UTF8 }  
uses crt;  
procedure uno;  
begin  
  a:=1;  
  writeln(a);  
end;  
var  
  a, b: integer;  
begin  
  a:=1;  
  b:=2;  
  uno;  
  writeln(b,a);  
end.
```

La diferencia es que, en el primer programa, se declaran variables globales “a” y “b” (antes del proceso “uno”), mientras que, en el segundo programa, se declaran, en cambio, variables locales al programa (después del proceso “uno”), lo cual provoca un error en el proceso “uno”, ya que éste no recibe como parámetros a estas variables locales al programa.

Ejercicio 5.

Dado el siguiente programa, indicar cuál es el error:

```
program TP2_E5;  
{$codepage UTF8}  
function cuatro: integer;  
begin  
    cuatro:=4;  
end;  
var  
    a: integer;  
begin  
    cuatro;  
    writeln(a);  
end.
```

El error es que la variable “a” no se encuentra inicializada.

Ejercicio 6.

(a) Realizar un módulo que lea de teclado números enteros hasta que llegue un valor negativo. Al finalizar la lectura, el módulo debe imprimir en pantalla cuál fue el número par más alto.

```
procedure num_par_mayor;
begin
  num_max:=low(int16);
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num);
  while (num>=num_salida) do
  begin
    if ((num mod 2=0) and (num>num_max)) then
      num_max:=num;
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num);
  end;
  textcolor(green); write('El número par más alto fue '); textcolor(red); write(num_max);
end;
```

(b) Implementar un programa que invoque al módulo del inciso a.

```
program TP2_E6;
{$codepage UTF8}
uses crt;
const
  num_salida=0;
var
  num, num_max: int16;
procedure num_par_mayor;
begin
  num_max:=low(int16);
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num);
  while (num>=num_salida) do
  begin
    if ((num mod 2=0) and (num>num_max)) then
      num_max:=num;
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num);
  end;
  textcolor(green); write('El número par más alto fue '); textcolor(red); write(num_max);
end;
begin
  num_par_mayor;
end.
```

Ejercicio 7.

Dado el siguiente programa:

```

program TP2_E7;
var
    suma, cant: integer;
function calcularPromedio: real;
var
    prom: real;
begin
    if (cant=0) then
        prom:=-1
    else
        prom:=suma/cant;
end;
begin
    readln(suma);
    readln(cant);
    if (calcularPromedio<>-1) then
    begin
        cant:=0;
        writeln('El promedio es: ', calcularPromedio);
    end
    else
        writeln('Dividir por cero no parece ser una buena idea');
end.

```

(a) *La función calcularPromedio calcula y retorna el promedio entre las variables globales suma y cant, pero parece incompleta. ¿Qué se debería agregar para que funcione correctamente?*

```

program TP2_E7a;
{$codepage UTF8}
uses crt;
var
    suma, cant: int16;
function calcularPromedio: real;
var
    prom: real;
begin
    if (cant=0) then
        prom:=-1
    else
        prom:=suma/cant;
        calcularPromedio:=prom;
end;
begin
    textcolor(green); write('Introducir suma: ');
    textcolor(yellow); readln(suma);
    textcolor(green); write('Introducir cantidad: ');
    textcolor(yellow); readln(cant);
    if (calcularPromedio<>-1) then
    begin
        textcolor(green); write('El promedio es '); textcolor(red); write(calcularPromedio:0:2);
    end
    else
    begin
        textcolor(green); write('Dividir por cero no parece ser una buena idea');
    end;
end.

```


(b) En el programa principal, la función `calcularPromedio` es invocada dos veces, pero esto podría mejorarse. ¿Cómo debería modificarse el programa principal para invocar a dicha función una única vez?

```
program TP2_E7b;
{$codepage UTF8}
uses crt;
var
    suma, cant: int16;
    prom: real;
function calcularPromedio: real;
begin
    if (cant=0) then
        prom:=-1
    else
        prom:=suma/cant;
        calcularPromedio:=prom;
    end;
begin
    prom:=0;
    textcolor(green); write('Introducir suma: ');
    textcolor(yellow); readln(suma);
    textcolor(green); write('Introducir cantidad: ');
    textcolor(yellow); readln(cant);
    if (calcularPromedio<>-1) then
        begin
            textcolor(green); write('El promedio es '); textcolor(red); write(prom:0:2);
        end
    else
        begin
            textcolor(green); write('Dividir por cero no parece ser una buena idea');
        end;
    end;
end.
```

(c) Si se leen por teclado los valores 48 (variable `suma`) y 6 (variable `cant`), ¿qué resultado imprime el programa? Considerar las tres posibilidades:

(i) El programa original.

Si se leen por teclado los valores 48 (variable “`suma`”) y 6 (variable “`cant`”), el resultado que imprime el programa es ninguno.

(ii) El programa luego de realizar la modificación del inciso (a).

Si se leen por teclado los valores 48 (variable “`suma`”) y 6 (variable “`cant`”), el resultado que imprime el programa es 8.

(iii) El programa luego de realizar las modificaciones de los incisos (a) y (b).

Si se leen por teclado los valores 48 (variable “*suma*”) y 6 (variable “*cant*”), el resultado que imprime el programa es 8.

Ejercicio 8.

Dado el siguiente programa:

```

program TP2_E8;
procedure leer;
var
  letra: char;
function analizarLetra: boolean;
begin
  if ((letra>='a') and (letra<='z')) then
    analizarLetra:=true
  else
    if ((letra>='A') and (letra<='Z')) then
      analizarLetra:=false;
end;
begin
  readln(letra);
  if (analizarLetra) then
    writeln('Se trata de una minúscula')
  else
    writeln('Se trata de una mayúscula');
end;
var
  ok: boolean;
begin
  leer;
  ok:=analizarLetra;
  if (ok) then
    writeln('Gracias, vuelva prontoss');
end.

```

(a) La función *analizarLetra* fue declarada como un submódulo dentro del procedimiento *leer*. Pero esto puede traer problemas en el código del programa principal.

(i) ¿Qué clase de problema se encuentra?

El problema encontrado es que no será posible invocar a la función “*analizarLetra*” en el programa principal.

(ii) ¿Cómo se puede resolver el problema para que el programa compile y funcione correctamente?

```

program TP2_E8a;
{$codepage UTF8}
uses crt;
var
  letra: char;
function analizarLetra: boolean;
begin
  if ((letra>='a') and (letra<='z')) then
    analizarLetra:=true
  else
    if ((letra>='A') and (letra<='Z')) then
      analizarLetra:=false;
end;

```

```

procedure leer;
begin
  textcolor(green); write('Introducir letra: ');
  textcolor(yellow); readln(letra);
  if (analizarLetra=true) then
  begin
    textcolor(green); writeln('Se trata de una minúscula');
  end
  else
  begin
    textcolor(green); writeln('Se trata de una mayúscula');
  end;
end;
var
  ok: boolean;
begin
  leer;
  ok:=analizarLetra;
  if (ok=true) then
  begin
    textcolor(green); write('Gracias, vuelva pronto');
  end;
end.

```

(b) La función *analizarLetra* parece incompleta, ya que no cubre algunos valores posibles de la variable *letra*.

(i) ¿De qué valores se trata?

Se trata de aquellos caracteres que no son letras.

(ii) ¿Qué sucede en nuestro programa si se ingresa uno de estos valores?

Lo que sucede en el programa si se ingresa uno de estos valores es que el proceso “*leer*”, por medio de la función “*analizarLetra*”, indica que es una letra minúscula (dado que el *boolean* que retorna la función está inicializado en *true*).

(iii) ¿Cómo se puede resolver este problema?

```

program TP2_E8b;
{$codepage UTF8}
uses crt;
var
  letra: char;
function analizarLetra: int8;
begin
  if ((letra>='a') and (letra<='z')) then
    analizarLetra:=1
  else
    if ((letra>='A') and (letra<='Z')) then
      analizarLetra:=2
    else
      analizarLetra:=-1;

```

```
end;
procedure leer;
begin
  textcolor(green); write('Introducir letra: ');
  textcolor(yellow); readln(letra);
  if (analizarLetra=1) then
  begin
    textcolor(green); writeln('Se trata de una minúscula');
  end
  else
    if (analizarLetra=2) then
    begin
      textcolor(green); writeln('Se trata de una mayúscula');
    end
    else
    begin
      textcolor(green); writeln('No es una letra');
    end;
  end;
end;
var
  ok: int8;
begin
  leer;
  ok:=analizarLetra;
  if (ok=1) then
  begin
    textcolor(green); write('Gracias, vuelva pronto');
  end;
end.
```

Trabajo Práctico N° 2.2: Funciones, Procedimientos y Parámetros.

Ejercicio 1.

Responder las preguntas en relación al siguiente programa:

```
program TP2_E1;
{$codepage UTF8}
uses crt;
procedure suma(num1: integer; var num2: integer);
begin
    num2:=num1+num2;
    num1:=0;
end;
var
    i, x: integer;
begin
    read(x);
    for i:= 1 to 5 do
        suma(i,x);
        write(x);
    end.
end.
```

(a) ¿Qué imprime si se lee el valor 10 en la variable x?

```
program TP2_E1a;
{$codepage UTF8}
uses crt;
procedure suma(num1: integer; var num2: integer);
begin
    num2:=num1+num2;
    num1:=0;
end;
var
    i, x: integer;
begin
    read(x);
    for i:= 1 to 5 do
        suma(i,x);
        write(x);
    end.
end.
```

Si se lee el valor 10 en la variable “x”, se imprime el número 25.

(b) ¿Qué imprime si se lee el valor 10 en la variable x y se cambia el encabezado del procedure por: procedure suma(num1: integer; num2: integer);?

```
program TP2_E1b;
{$codepage UTF8}
uses crt;
procedure suma(num1: integer; num2: integer);
begin
    num2:=num1+num2;
    num1:=0;
end;
```

```
var
  i, x: integer;
begin
  read(x);
  for i:= 1 to 5 do
    suma(i,x);
  write(x);
end.
```

Si se lee el valor 10 en la variable “x” y se cambia el encabezado del *procedure* por *procedure suma(num1: integer; num2:integer)*, se imprime el número 10.

(c) ¿Qué sucede si se cambia el encabezado del *procedure* por: *procedure suma(var num1: integer; var num2: integer);*?

```
program TP2_E1c;
{$codepage UTF8}
uses crt;
procedure suma(var num1: integer; var num2: integer);
begin
  num2:=num1+num2;
  num1:=0;
end;
var
  i, x: integer;
begin
  read(x);
  for i:= 1 to 5 do
    suma(i,x);
  write(x);
end.
```

Lo que sucede si se cambia el encabezado del *procedure* por *procedure suma(var num1: integer; var num2: integer)* es que se generará un error, ya que no es posible modificar el valor de una variable índice (en este caso, “i”).

Ejercicio 2.

Responder la pregunta en relación al siguiente programa:

```
program TP2_E2;
{$codepage UTF8}
uses crt;
procedure digParesImpares(num: integer; var par, impar: integer);
var
  dig: integer;
begin
  while (num<>0) do
  begin
    dig:=num mod 10;
    if (dig mod 2=0) then
      par:=par+1
    else
      impar:=impar+1;
    num:=num div 10;
  end;
end;
var
  dato, par, impar, total, cant: integer;
begin
  par:=0;
  impar:=0;
  repeat
    read(dato);
    digParesImpares(dato,par,impar);
  until (dato=100);
  writeln('Pares: ',par,' e Impares: ',impar);
end.
```

¿Qué imprime si se lee la secuencia de valores 250, 35, 100?

Si se lee la secuencia de valores 250, 35 y 100, el programa imprime “Pares: 4 e Impares: 4”.

Ejercicio 3.

Encontrar los 6 errores que existen en el siguiente programa. Utilizar los comentarios entre llaves como guía, indicar en qué línea se encuentra cada error y en qué consiste.

Con errores:

```
program TP2_E3;
{$codepage UTF8}
uses crt;
{Suma los números entre a y b y retorna el resultado en c}
procedure sumar(a, b, c: integer)
var
    suma: integer;
begin
    for i:= a to b do
        suma:=suma+i;
        c:=c+suma;
    end;
var
    result: integer;
begin
    result:=0;
    readln(a); readln(b);
    sumar(a,b,0);
    write('La suma total es ',result);
    {Averigua si el resultado final estuvo entre 10 y 30}
    ok:=((result>=10) or (result<=30));
    if (not ok) then
        write('La suma no quedó entre 10 y 30');
end.
```

Sin errores:

```
program TP2_E3;
{$codepage UTF8}
uses crt;
{Suma los números entre a y b y retorna el resultado en c}
procedure sumar(a, b: integer; var c: integer);
var
    i, suma: integer;
begin
    suma:=0;
    for i:= a to b do
        suma:=suma+i;
        c:=c+suma;
    end;
var
    result, a, b: integer;
    ok: boolean;
begin
    result:=0;
    readln(a); readln(b);
    sumar(a,b,result);
    write('La suma total es ',result);
    {Averigua si el resultado final estuvo entre 10 y 30}
    ok:=((result>=10) or (result<=30));
    if (not ok) then
        write('La suma no quedó entre 10 y 30');
end.
```

Los 6 errores que existen en el programa son:

1. Línea 3: En el *procedure* “*sumar*”, falta “;” al final de la instrucción.
2. Línea 3: En el *procedure* “*sumar*”, el parámetro “*c*” debe ser por referencia.
3. Línea 5: En el *procedure* “*sumar*”, falta crear la variable local al proceso correspondiente al índice “*i*”.
4. Línea 7: En el *procedure* “*sumar*”, falta inicializar la variable “*suma*”.
5. Línea 13: En el programa principal, falta crear las variables locales al programa “*a*” y “*b*”, como *integer*, y “*ok*”, como *boolean*.
6. Línea 17: En el programa principal, en la invocación del *procedure* “*sumar*”, se debe pasar la variable local al programa “*result*” como parámetro por valor.

Ejercicio 4.

El siguiente programa intenta resolver un enunciado. Sin embargo, el código posee 5 errores. Indicar en qué línea se encuentra cada error y en qué consiste el error.

Enunciado: Realizar un programa que lea datos de 130 programadores Java de una empresa. De cada programador, se lee el número de legajo y el salario actual. El programa debe imprimir el total del dinero destinado por mes al pago de salarios y el salario del empleado mayor legajo.

Con errores:

```
program TP2_E4;
{$codepage UTF8}
uses crt;
procedure leerDatos(var legajo: integer; salario: real);
begin
  writeln('Ingresar el número de legajo y el salario');
  read(legajo);
  read(salario);
end;
procedure actualizarMaximo(nuevoLegajo: integer; nuevoSalario: real; var maxLegajo: integer);
var
  maxSalario: real;
begin
  if (nuevoLegajo>maxLegajo) then
  begin
    maxLegajo:=nuevoLegajo;
    maxSalario:=nuevoSalario;
  end;
end;
var
  legajo, maxLegajo, i: integer;
  salario, maxSalario: real;
begin
  sumaSalarios:=0;
  for i:= 1 to 130 do
  begin
    leerDatos(salario,legajo);
    actualizarMaximo(legajo,salario,maxLegajo);
    sumaSalarios:=sumaSalarios+salario;
  end;
  writeln('En todo el mes se gastan ',sumaSalarios,' pesos');
  writeln('El salario del empleado más nuevo es ',maxSalario);
end.
```

Sin errores:

```
program TP2_E4;
{$codepage UTF8}
uses crt;
procedure leerDatos(var legajo: integer; var salario: real);
begin
  writeln('Ingresar el número de legajo y el salario');
  read(legajo);
  read(salario);
end;
procedure actualizarMaximo(nuevoLegajo: integer; nuevoSalario: real; var maxLegajo: integer;
var maxSalario: real);
begin
  if (nuevoLegajo>maxLegajo) then
```

```
begin
    maxLegajo:=nuevoLegajo;
    maxSalario:=nuevoSalario;
end;
end;
var
    legajo, maxLegajo, i: integer;
    salario, maxSalario, sumaSalarios: real;
begin
    maxLegajo:=0; maxSalario:=0;
    sumaSalarios:=0;
    for i:= 1 to 130 do
        begin
            leerDatos(legajo,salario);
            actualizarMaximo(legajo,salario,maxLegajo,maxSalario);
            sumaSalarios:=sumaSalarios+salario;
        end;
    writeln('En todo el mes se gastan ',sumaSalarios,' pesos');
    writeln('El salario del empleado más nuevo es ',maxSalario);
end.
```

Los 5 errores que posee el código son:

1. Línea 2: En el *procedure* “*leerDatos*”, el parámetro “*salario*” debe ser por referencia.
2. Línea 8: En el *procedure* “*actualizarMaximo*”, falta pasar la variable local al programa “*maxSalario*” como parámetro por referencia, por lo que no se debe crear la variable local al proceso homónima.
3. Línea 19: En el programa principal, falta crear la variable local al programa “*sumaSalarios*”.
4. Línea 21: En el programa principal, falta inicializar las variables locales al programa “*maxLegajo*” y “*maxSalario*”.
5. Línea 23: En el programa principal, en la invocación del *procedure* “*leerDatos*”, el orden de los parámetros está invertido.

Ejercicio 5.

(a) Realizar un módulo que reciba un par de números (*numA*, *numB*) y retorne si *numB* es el doble de *numA*.

```
function cumple_criterio(numA, numB: int16): boolean;
begin
    cumple_criterio:=(numB=multiplo_corte*numA);
end;
```

(b) Utilizando el módulo realizado en el inciso (a), realizar un programa que lea secuencias de pares de números hasta encontrar el par (0,0), e informe la cantidad total de pares de números leídos y la cantidad de pares en las que *numB* es el doble de *numA*. Ejemplo: si se lee la siguiente secuencia (1,2) (3,4) (9,3) (7,14) (0,0), el programa debe informar los valores 4 (cantidad de pares leídos) y 2 (cantidad de pares en los que *numB* es el doble de *numA*).

```
program TP2_E5;
{$codepage UTF8}
uses crt;
const
    numA_salida=0; numB_salida=0;
    multiplo_corte=2;
function cumple_criterio(numA, numB: int16): boolean;
begin
    cumple_criterio:=(numB=multiplo_corte*numA);
end;
var
    pares_total, doble: int8;
    numA, numB: int16;
begin
    pares_total:=0; doble:=0;
    textcolor(green); write('Introducir número entero A: ');
    textcolor(yellow); readln(numA);
    textcolor(green); write('Introducir número entero B: ');
    textcolor(yellow); readln(numB);
    while ((numA<>numA_salida) or (numB<>numB_salida)) do
    begin
        if (cumple_criterio(numA,numB)=true) then
            doble:=doble+1;
        pares_total:=pares_total+1;
        textcolor(green); write('Introducir número entero A: ');
        textcolor(yellow); readln(numA);
        textcolor(green); write('Introducir número entero B: ');
        textcolor(yellow); readln(numB);
    end;
    textcolor(green); write('La cantidad total de pares leídos es '); textcolor(red);
    writeln(pares_total);
    textcolor(green); write('La cantidad de pares en las que numB es el doble de numA es ');
    textcolor(red); write(doble);
end.
```

Ejercicio 6.

Realizar un programa modularizado que lea datos de 100 productos de una tienda de ropa. Para cada producto, debe leer el precio, código y tipo (pantalón, remera, camisa, medias, campera, etc.). Informar:

- Código de los dos productos más baratos.
- Código del producto de tipo “pantalón” más caro.
- Precio promedio.

```
program TP2_E6;
{$codepage UTF8}
uses crt;
const
    productos_total=3;
    tipo_corte='pantalon';
procedure leer_producto(var precio: real; var producto: int16; var tipo: string);
begin
    textcolor(green); write('Introducir precio del producto: ');
    textcolor(yellow); readln(precio);
    textcolor(green); write('Introducir código de producto del producto: ');
    textcolor(yellow); readln(producto);
    textcolor(green); write('Introducir tipo de producto: ');
    textcolor(yellow); readln(tipo);
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;
end;
procedure actualizar_maximo(precio: real; producto: int16; tipo: string; var precio_max: real;
var producto_max: int16);
begin
    if ((tipo=tipo_corte) and (precio>precio_max)) then
        begin
            precio_max:=precio;
            producto_max:=producto;
        end;
end;
procedure leer_productos(var producto_min1, producto_min2, producto_max: int16; var
precio_prom: real);
var
    i: int8;
    producto: int16;
    precio, precio_min1, precio_min2, precio_max, precio_total: real;
    tipo: string;
begin
    precio_min1:=9999999; precio_min2:=9999999;
    precio_max:=-9999999;
    precio_total:=0;
    for i:= 1 to productos_total do
```

```
begin
    leer_producto(precio,producto,tipo);
    actualizar_minimos(precio,producto,precio_min1,precio_min2,producto_min1,producto_min2);
    actualizar_maximo(precio,producto,tipo,precio_max,producto_max);
    precio_total:=precio_total+precio;
end;
precio_prom:=precio_total/productos_total;
end;
var
    producto_min1, producto_min2, producto_max: int16;
    precio_prom: real;
begin
    producto_min1:=0; producto_min2:=0;
    producto_max:=0;
    precio_prom:=0;
    leer_productos(producto_min1,producto_min2,producto_max,precio_prom);
    textcolor(green); write('Los códigos de los dos productos más baratos son ');
textcolor(red); write(producto_min1); textcolor(green); write(' y '); textcolor(red);
writeln(producto_min2);
    textcolor(green); write('El código del producto de tipo '); textcolor(yellow);
write(tipo_corte); textcolor(green); write(' más caro es '); textcolor(red);
writeln(producto_max);
    textcolor(green); write('El precio promedio es $'); textcolor(red); write(precio_prom:0:2);
end.
```

Ejercicio 7.

(a) Realizar un módulo que reciba como parámetro un número entero y retorne la cantidad de dígitos que posee y la suma de los mismos.

```
procedure cantidad_suma_digitos(num: int16; var digitos, suma: int16);
var
  digito: int8;
begin
  while (num>0) do
  begin
    digito:=num mod 10;
    digitos:=digitos+1;
    suma:=suma+digito;
    num:=num div 10;
  end;
end;
```

(b) Utilizando el módulo anterior, realizar un programa que lea una secuencia de números e imprima la cantidad total de dígitos leídos. La lectura finaliza al leer un número cuyos dígitos suman, exactamente, 10, el cual debe procesarse.

```
program TP2_E7;
{$codepage UTF8}
uses crt;
const
  suma_salida=10;
procedure cantidad_suma_digitos(num: int16; var digitos, suma: int16);
var
  digito: int8;
begin
  while (num>0) do
  begin
    digito:=num mod 10;
    digitos:=digitos+1;
    suma:=suma+digito;
    num:=num div 10;
  end;
end;
procedure cantidad_digitos_total(var digitos_total: int16);
var
  num, digitos, suma: int16;
begin
  repeat
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num);
    digitos:=0; suma:=0;
    cantidad_suma_digitos(num,digitos,suma);
    digitos_total:=digitos_total+digitos;
  until (suma=suma_salida);
end;
var
  digitos_total: int16;
begin
  digitos_total:=0;
  cantidad_digitos_total(digitos_total);
  textcolor(green); write('La cantidad total de dígitos leídos es '); textcolor(red);
  write(digitos_total);
end.
```


Ejercicio 8.

Realizar un programa modularizado que lea secuencia de números enteros. La lectura finaliza cuando llega el número 123456, el cual no debe procesarse. Informar en pantalla, para cada número, la suma de sus dígitos pares y la cantidad de dígitos impares que posee.

```
program TP2_E8;
{$codepage UTF8}
uses crt;
const
    num_salida=123456;
procedure suma_pares_cantidad_impares(num: int32; var suma_pares, cantidad_impares: int16);
var
    digito: int8;
begin
    while (num>0) do
    begin
        digito:=num mod 10;
        if (digito mod 2=0) then
            suma_pares:=suma_pares+digito
        else
            cantidad_impares:=cantidad_impares+1;
        num:=num div 10;
    end;
end;
var
    suma_pares, cantidad_impares: int16;
    num: int32;
begin
    textcolor(green); write('Introducir número entero: ');
    textcolor(yellow); readln(num);
    while (num<>num_salida) do
    begin
        suma_pares:=0; cantidad_impares:=0;
        suma_pares_cantidad_impares(num,suma_pares,cantidad_impares);
        textcolor(green); write('La suma de sus dígitos pares es '); textcolor(red);
        writeln(suma_pares);
        textcolor(green); write('La cantidad de dígitos impares que posee es '); textcolor(red);
        writeln(cantidad_impares);
        textcolor(green); write('Introducir número entero: ');
        textcolor(yellow); readln(num);
    end;
end.
```

Ejercicio 9.

Realizar un programa modularizado que lea información de alumnos de una facultad. Para cada alumno, se lee: número de inscripción, apellido y nombre. La lectura finaliza cuando se ingresa el alumno con número de inscripción 1200, que debe procesarse. Se pide calcular e informar:

- Apellido de los dos alumnos con número de inscripción más chico.
- Nombre de los dos alumnos con número de inscripción más grande.
- Porcentaje de alumnos con número de inscripción par.

```

program TP2_E9;
{$codepage UTF8}
uses crt;
const
    num_salida=1200;
procedure leer_alumno(var num: int32; var apellido, nombre: string);
begin
    textcolor(green); write('Introducir número de inscripción: ');
    textcolor(yellow); readln(num);
    textcolor(green); write('Introducir apellido: ');
    textcolor(yellow); readln(apellido);
    textcolor(green); write('Introducir nombre: ');
    textcolor(yellow); readln(nombre);
end;
procedure actualizar_minimos(num: int32; apellido: string; var num_min1, num_min2: int32; var
apellido_min1, apellido_min2: string);
begin
    if (num<num_min1) then
    begin
        num_min2:=num_min1;
        apellido_min2:=apellido_min1;
        num_min1:=num;
        apellido_min1:=apellido;
    end
    else
        if (num<num_min2) then
        begin
            num_min2:=num;
            apellido_min2:=apellido;
        end;
    end;
end;
procedure actualizar_maximos(num: int32; nombre: string; var num_max1, num_max2: int32; var
nombre_max1, nombre_max2: string);
begin
    if (num>num_max1) then
    begin
        num_max2:=num_max1;
        nombre_max2:=nombre_max1;
        num_max1:=num;
        nombre_max1:=nombre;
    end
    else
        if (num>num_max2) then
        begin
            num_max2:=num;
            nombre_max2:=nombre;
        end;
    end;
end;
procedure leer_alumnos(var apellido_min1, apellido_min2, nombre_max1, nombre_max2: string; var
porcentaje_par: real);
var

```

```
alumnos_par, alumnos_total: int16;  
num, num_min1, num_min2, num_max1, num_max2: int32;  
apellido, nombre: string;  
begin  
  alumnos_par:=0; alumnos_total:=0;  
  num_min1:=high(int32); num_min2:=high(int32);  
  num_max1:=low(int32); num_max2:=low(int32);  
  repeat  
    leer_alumno(num,apellido,nombre);  
    actualizar_minimos(num,apellido,num_min1,num_min2,apellido_min1,apellido_min2);  
    actualizar_maximos(num,nombre,num_max1,num_max2,nombre_max1,nombre_max2);  
    alumnos_total:=alumnos_total+1;  
    if (num mod 2=0) then  
      alumnos_par:=alumnos_par+1;  
    until (num=num_salida);  
  porcentaje_par:=alumnos_par/alumnos_total*100;  
end;  
var  
  porcentaje_par: real;  
  apellido_min1, apellido_min2, nombre_max1, nombre_max2: string;  
begin  
  apellido_min1:=''; apellido_min2:='';  
  nombre_max1:=''; nombre_max2:='';  
  porcentaje_par:=0;  
  leer_alumnos(apellido_min1,apellido_min2,nombre_max1,nombre_max2,porcentaje_par);  
  textcolor(green); write('Los apellidos de los dos alumnos con número de inscripción más  
chico son '); textcolor(red); write(apellido_min1); textcolor(green); write(' y ');  
textcolor(red); writeln(apellido_min2);  
  textcolor(green); write('Los nombres de los dos alumnos con número de inscripción más grande  
son '); textcolor(red); write(nombre_max1); textcolor(green); write(' y '); textcolor(red);  
writeln(nombre_max2);  
  textcolor(green); write('El porcentaje de alumnos con número de inscripción par es ');  
textcolor(red); write(porcentaje_par:0:2); textcolor(green); write('%');  
end.
```

Ejercicio 10.

Realizar un programa modularizado que lea una secuencia de caracteres y verifique si cumple con el patrón A\$B#, donde:

- A es una secuencia de sólo letras vocales.
- B es una secuencia de sólo caracteres alfabéticos sin letras vocales.
- Los caracteres \$ y # seguro existen.

Nota: En caso de no cumplir, informar qué parte del patrón no se cumplió.

```

program TP2_E10;
{$codepage UTF8}
uses crt;
function leer_secuencia(secuencia: string): string;
begin
    textcolor(green); write('Introducir secuencia de caracteres: ');
    textcolor(yellow); readln(secuencia);
    leer_secuencia:=secuencia;
end;
function es_vocal(c: char): boolean;
begin
    es_vocal:=(c='A') or (c='E') or (c='I') or (c='O') or (c='U') or (c='a') or (c='e') or
(c='i') or (c='o') or (c='u');
end;
procedure parseo_string(var cumple_A, cumple_B, cumple_AB: boolean);
var
    i, j: int8;
    secuencia: string;
begin
    secuencia:='';
    secuencia:=leer_secuencia(secuencia);
    i:=1;
    while (secuencia[i]<>'$') do
        begin
            cumple_A:=cumple_A and (es_vocal(secuencia[i])=true);
            cumple_AB:=cumple_AB and (es_vocal(secuencia[i])=true);
            i:=i+1;
        end;
    j:=i+1;
    while (secuencia[j]<>'#') do
        begin
            cumple_B:=cumple_B and (es_vocal(secuencia[j])=false);
            cumple_AB:=cumple_AB and (es_vocal(secuencia[j])=false);
            j:=j+1;
        end;
    end;
var
    cumple_A, cumple_B, cumple_AB: boolean;
begin
    cumple_A:=true; cumple_B:=true; cumple_AB:=true;
    parseo_string(cumple_A,cumple_B,cumple_AB);
    if (cumple_AB=true) then
        begin
            textcolor(yellow); write('La secuencia cumple con el patrón A$B#');
        end
    else
        if ((cumple_A=false) and (cumple_B=true)) then
            begin
                textcolor(yellow); write('La secuencia no cumple con la parte A del patrón A$B#');
            end
        else
            if ((cumple_A=true) and (cumple_B=false)) then
                begin

```

```
        textcolor(yellow); write('La secuencia no cumple con la parte B del patrón A$B#');
    end
    else
    begin
        textcolor(yellow); write('La secuencia no cumple con las partes A y B del patrón
A$B#');
    end;
end.
```

Ejercicio 11.

Realizar un programa modularizado que lea una secuencia de caracteres y verifique si cumple con el patrón $A\%B^*$, donde:

- A es una secuencia de caracteres en la que no existe el carácter '\$'.
- B es una secuencia con la misma cantidad de caracteres que aparecen en A y en la que aparece, a lo sumo, 3 veces el carácter '@'.
- Los caracteres % y * seguro existen.

Nota: en caso de no cumplir, informar que parte del patrón no se cumplió.

```
program TP2_E11;
{$codepage UTF8}
uses crt;
const
    arrobascorte=3;
function leer_secuencia(sequencia: string): string;
begin
    textcolor(green); write('Introducir secuencia de caracteres: ');
    textcolor(yellow); readln(sequencia);
    leer_secuencia:=sequencia;
end;
procedure parseo_string(var cumple_A, cumple_B, cumple_AB: boolean);
var
    i, j, arrobasc: int8;
    sequencia: string;
begin
    sequencia:='';
    sequencia:=leer_secuencia(sequencia);
    i:=1; arrobasc:=0;
    while (sequencia[i]<>'%') do
        begin
            cumple_A:=cumple_A and (sequencia[i]<>'$');
            cumple_AB:=cumple_AB and (sequencia[i]<>'$');
            i:=i+1;
        end;
    j:=i+1;
    while (sequencia[j]<>'*') do
        begin
            if (sequencia[j]='@') then
                arrobasc:=arrobasc+1;
            cumple_B:=cumple_B and (arrobasc<=arrobascorte);
            cumple_AB:=cumple_AB and (arrobasc<=arrobascorte);
            j:=j+1;
        end;
    cumple_B:=cumple_B and (j/2=i);
    cumple_AB:=cumple_AB and (j/2=i);
end;
var
    cumple_A, cumple_B, cumple_AB: boolean;
begin
    cumple_A:=true; cumple_B:=true; cumple_AB:=true;
    parseo_string(cumple_A,cumple_B,cumple_AB);
    if (cumple_AB=true) then
        begin
            textcolor(yellow); write('La secuencia cumple con el patrón A%B*');
        end
    else
        if ((cumple_A=false) and (cumple_B=true)) then
            begin
                textcolor(yellow); write('La secuencia no cumple con la parte A del patrón A%B*');
            end
        else
```

```
if ((cumple_A=true) and (cumple_B=false)) then
begin
    textcolor(yellow); write('La secuencia no cumple con la parte B del patrón A%B*');
end
else
begin
    textcolor(yellow); write('La secuencia no cumple con las partes A y B del patrón
A%B*');
end;
end.
```

Ejercicio 12.

(a) Realizar un módulo que calcule el rendimiento económico de una plantación de soja. El módulo debe recibir la cantidad de hectáreas (ha) sembradas, el tipo de zona de siembra (1: zona muy fértil, 2: zona estándar, 3: zona árida) y el precio en U\$S de la tonelada de soja; y devolver el rendimiento económico esperado de dicha plantación. Para calcular el rendimiento económico esperado, debe considerar el siguiente rendimiento por tipo de zona:

Tipo de zona	Rendimiento por ha
1	6 toneladas por ha
2	2,6 toneladas por ha
3	1,4 toneladas por ha

```
function rendimiento_economico(ha, zona, precio: int16): real;
begin
  case zona of
    1: rendimiento_economico:=ha*6*precio;
    2: rendimiento_economico:=ha*2.6*precio;
    3: rendimiento_economico:=ha*1.4*precio;
  end;
end;
```

(b) ARBA desea procesar información obtenida de imágenes satelitales de campos sembrados con soja en la provincia de Buenos Aires. De cada campo, se lee: localidad, cantidad de hectáreas sembradas y el tipo de zona (1, 2 o 3). La lectura finaliza al leer un campo de 900 ha en la localidad 'Saladillo', que debe procesarse. El precio de la soja es de U\$S 320 por tn. Informar:

- La cantidad de campos de la localidad Tres de Febrero con rendimiento estimado superior a U\$S 10.000.
- La localidad del campo con mayor rendimiento económico esperado.
- La localidad del campo con menor rendimiento económico esperado.
- El rendimiento económico promedio.

```
program TP2_E12;
{$codepage UTF8}
uses crt;
const
  ha_salida=900; localidad_salida='Saladillo';
  precio=320.0;
  localidad_corte='Tres de Febrero'; rendimiento_corte=10000.0;
function rendimiento_economico(ha, zona: int16; precio: real): real;
begin
  case zona of
    1: rendimiento_economico:=ha*6*precio;
    2: rendimiento_economico:=ha*2.6*precio;
    3: rendimiento_economico:=ha*1.4*precio;
  end;
end;
procedure leer_campo(var localidad: string; var ha, zona: int16);
begin
  textcolor(green); write('Introducir localidad del campo: ');
  textcolor(yellow); readln(localidad);
```



```

    textcolor(green); write('Introducir cantidad de hectáreas sembradas del campo: ');
    textcolor(yellow); readln(ha);
    textcolor(green); write('Introducir tipo de zona del campo (1: zona muy fértil, 2: zona
estándar, 3: zona árida): ');
    textcolor(yellow); readln(zona);
end;
procedure actualizar_maximo(rendimiento: real; localidad: string; var rendimiento_max: real;
var localidad_max: string);
begin
    if (rendimiento>rendimiento_max) then
    begin
        rendimiento_max:=rendimiento;
        localidad_max:=localidad;
    end;
end;
procedure actualizar_minimo(rendimiento: real; localidad: string; var rendimiento_min: real;
var localidad_min: string);
begin
    if (rendimiento<rendimiento_min) then
    begin
        rendimiento_min:=rendimiento;
        localidad_min:=localidad;
    end;
end;
procedure leer_campos(var campos_corte: int16; var rendimiento_prom: real; var localidad_max,
localidad_min: string);
var
    ha, zona, campos_total: int16;
    rendimiento, rendimiento_max, rendimiento_min, rendimiento_total: real;
    localidad: string;
begin
    rendimiento_max:=-9999999;
    rendimiento_min:=9999999;
    rendimiento_total:=0; campos_total:=0;
    repeat
        leer_campo(localidad,ha,zona);
        rendimiento:=rendimiento_economico(ha,zona,precio);
        rendimiento_total:=rendimiento_total+rendimiento;
        campos_total:=campos_total+1;
        if ((localidad=localidad_corte) and (rendimiento>rendimiento_corte)) then
            campos_corte:=campos_corte+1;
        actualizar_maximo(rendimiento,localidad,rendimiento_max,localidad_max);
        actualizar_minimo(rendimiento,localidad,rendimiento_min,localidad_min);
    until ((localidad=localidad_salida) and (ha=ha_salida));
    rendimiento_prom:=rendimiento_total/campos_total;
end;
var
    campos_corte: int16;
    rendimiento_prom: real;
    localidad_max, localidad_min: string;
begin
    campos_corte:=0;
    localidad_max:=''; localidad_min:='';
    rendimiento_prom:=0;
    leer_campos(campos_corte,rendimiento_prom,localidad_max,localidad_min);
    textcolor(green); write('La cantidad de campos de la localidad Tres de Febrero con
rendimiento estimado superior a U$S '); textcolor(yellow); write(rendimiento_corte:0:2);
textcolor(green); write(' es '); textcolor(red); writeln(campos_corte);
    textcolor(green); write('La localidad del campo con mayor rendimiento económico esperado es
'); textcolor(red); writeln(localidad_max);
    textcolor(green); write('La localidad del campo con menor rendimiento económico esperado es
'); textcolor(red); writeln(localidad_min);
    textcolor(green); write('El rendimiento económico promedio es $'); textcolor(red);
write(rendimiento_prom:0:2);
end.

```

Ejercicio 13.

Dado el siguiente programa:

```
program TP2_E13;
{$codepage UTF8}
uses crt;
procedure intercambio(var num1, num2: integer);
var
    aux: integer;
begin
    aux:=num1;
    num1:=num2;
    num2:=aux;
end;
procedure sumar(num1: integer; var num2: integer);
begin
    num2:=num1+num2;
end;
var
    i, num1, num2: integer;
begin
    read(num1);
    read(num2);
    for i:= 1 to 3 do
    begin
        intercambio(num1,num2);
        sumar(i,num1);
    end;
    writeln(num1);
end.
```

(a) ¿Qué imprime si se leen los valores $num1 = 10$ y $num2 = 5$?

Si se leen, los valores $num1 = 10$ y $num2 = 5$, el programa imprime 9.

(b) ¿Qué imprime si se leen los valores $num1 = 5$ y $num2 = 10$?

Si se leen, los valores $num1 = 5$ y $num2 = 10$, el programa imprime 14.

Ejercicio 14.

Realizar un programa modularizado que lea 10 pares de números (X , Y) e informe, para cada par de números, la suma y el producto de todos los números entre X e Y . Por ejemplo, dado el par (3, 6), debe informar:

“La suma es 18” (obtenido de calcular $3+4+5+6$).

“El producto es 360” (obtenido de calcular $3*4*5*6$).

```
program TP2_E14;
{$codepage UTF8}
uses crt;
const
  pares_total=10;
procedure leer_numeros(var numX, numY: int16);
begin
  textcolor(green); write('Introducir número entero X: ');
  textcolor(yellow); readln(numX);
  textcolor(green); write('Introducir número entero Y: ');
  textcolor(yellow); readln(numY);
end;
procedure calcular_suma_producto(var numX, numY: int16; var suma, producto: real);
var
  i: int16;
begin
  suma:=0; producto:=1;
  leer_numeros(numX,numY);
  for i:= numX to numY do
  begin
    suma:=suma+i;
    producto:=producto*i;
  end;
end;
var
  i: int8;
  numX, numY: int16;
  suma, producto: real;
begin
  suma:=0; producto:=1;
  for i:= 1 to pares_total do
  begin
    calcular_suma_producto(numX,numY,suma,producto);
    textcolor(green); write('Para el par '); textcolor(red); write('(' ,numX ,', ' ,numY ,')');
    textcolor(green); write(', la suma es '); textcolor(red); write(suma:0:2); textcolor(green);
    write(' y el producto es '); textcolor(red); writeln(producto:0:2);
  end;
end.
```

Ejercicio 15.

Realizar un programa modularizado que lea información de 200 productos de un supermercado. De cada producto, se lee código y precio (cada código es un número entre 1 y 200). Informar en pantalla:

- Los códigos de los dos productos más baratos.
- La cantidad de productos de más de 16 pesos con código par.

```

program TP2_E15;
{$codepage UTF8}
uses crt;
const
    productos_total=200;
    precio_corte=16.0;
procedure leer_producto(var producto, precio: int16);
begin
    textcolor(green); write('Introducir código de producto del producto: ');
    textcolor(yellow); readln(producto);
    textcolor(green); write('Introducir precio del producto: ');
    textcolor(yellow); readln(precio);
end;
procedure actualizar_minimos(precio, producto: int16; var precio_min1, precio_min2,
    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;
end;
procedure leer_productos(var producto_min1, producto_min2, productos_corte: int16);
var
    i, producto, precio, precio_min1, precio_min2: int16;
begin
    precio_min1:=high(int16); precio_min2:=high(int16);
    for i:= 1 to productos_total do
    begin
        leer_producto(producto,precio);
        actualizar_minimos(precio,producto,precio_min1,precio_min2,producto_min1,producto_min2);
        if ((precio>precio_corte) and (producto mod 2=0)) then
            productos_corte:=productos_corte+1;
        end;
    end;
end;
var
    producto_min1, producto_min2, productos_corte: int16;
begin
    producto_min1:=0; producto_min2:=0;
    productos_corte:=0;
    leer_productos(producto_min1,producto_min2,productos_corte);
    textcolor(green); write('Los códigos de los dos productos más baratos son ');
    textcolor(red); write(producto_min1); textcolor(green); write(' y '); textcolor(red);
    write(producto_min2); textcolor(green); writeln(', respectivamente');

```

```
textcolor(green); write('La cantidad de productos de más de '); textcolor(yellow);  
write(precio_corte:0:2); textcolor(green); write(' pesos con código par es '); textcolor(red);  
write(productos_corte);  
end.
```

Ejercicio 16.

(a) Realizar un módulo que reciba como parámetro el radio de un círculo y retorne su diámetro y su perímetro.

```
procedure circulo(radio: real; var diametro, perimetro: real);
begin
    diametro:=radio*2;
    perimetro:=pi*diametro;
end;
```

(b) Utilizando el módulo anterior, realizar un programa que analice información de planetas obtenida del Telescopio Espacial Kepler. De cada planeta, se lee su nombre, su radio (medido en kilómetros) y la distancia (medida en años luz) a la Tierra. La lectura finaliza al leer un planeta con radio 0, que no debe procesarse. Informar:

- Nombre y distancia de los planetas que poseen un diámetro menor o igual que el de la Tierra (12.700 km) y mayor o igual que el de Marte (6.780 km).
- Cantidad de planetas con un perímetro superior al del planeta Júpiter (439.264 km).

```
program TP2_E16;
{$codepage UTF8}
uses crt;
const
    radio_salida=0;
    diametro_corte1=12700.0; diametro_corte2=6780.0;
    perimetro_corte=439264.0;
procedure circulo(radio: real; var diametro, perimetro: real);
begin
    diametro:=radio*2;
    perimetro:=pi*diametro;
end;
procedure leer_planeta(var nombre: string; var radio, distancia: real);
begin
    textcolor(green); write('Introducir nombre del planeta: ');
    textcolor(yellow); readln(nombre);
    textcolor(green); write('Introducir radio (medido en kilómetros) del planeta : ');
    textcolor(yellow); readln(radio);
    textcolor(green); write('Introducir distancia (medida en años luz) a la tierra del planeta: ');
    textcolor(yellow); readln(distancia);
end;
procedure leer_planetas(var planetas_corte: int16);
var
    radio, distancia, diametro, perimetro: real;
    nombre: string;
begin
    diametro:=0; perimetro:=0;
    leer_planeta(nombre,radio,distancia);
    while (radio<>radio_salida) do
    begin
        circulo(radio,diametro,perimetro);
        if ((diametro<=diametro_corte1) and (diametro>=diametro_corte2)) then
        begin
            textcolor(green); write('El planeta '); textcolor(red); write(nombre); textcolor(green);
            write(' tiene un diámetro menor o igual al de la Tierra ('); textcolor(yellow);
            write(diametro_corte1:0:2); textcolor(green); write(' km) y mayor o igual que el de Marte (');
            write(perimetro_corte:0:2); textcolor(green); write(' km)');
        end;
    end;
end;
```

```
textcolor(yellow); write(diametro_corte2:0:2); textcolor(green); write(' km), y queda a ');
textcolor(red); write(distancia:0:2); textcolor(green); writeln(' años luz de la Tierra');
    end;
    if (perimetro>perimetro_corte) then
        planetas_corte:=planetas_corte+1;
        leer_planeta(nombre,radio,distancia);
    end;
end;
var
    planetas_corte: int16;
begin
    planetas_corte:=0;
    leer_planetas(planetas_corte);
    textcolor(green); write('La cantidad de planetas con un perímetro superior al del planeta
Júpiter ('); textcolor(yellow); write(perimetro_corte:0:2); textcolor(green); write(' km) es
'); textcolor(red); write(planetas_corte);
end.
```

Ejercicio 17.

En la “Práctica 1 - Ejercicios Adicionales”, se resolvieron 3 problemas complejos sin utilizar módulos. Al carecer de herramientas para modularizar, esos programas resultaban difíciles de leer, de extender y de depurar.

(a) Analizar las soluciones a dichos problemas e identificar:

(i) ¿Qué porciones de su código podrían modularizarse? ¿En qué casos propondría una estructura de módulos anidada?

(ii) ¿Qué tipo de módulo (función o procedimiento) conviene utilizar en cada caso? ¿Existe algún caso en los que sólo un tipo de módulo es posible?

(iii) ¿Qué mecanismos de comunicación conviene utilizar entre los módulos propuestos?

(b) Implementar, nuevamente, los 3 programas, teniendo en cuenta los módulos propuestos en el inciso anterior.

Ejercicio 1:

```

program TP2_E17a;
{$codepage UTF8}
uses crt;
const
    empresa_salida=100;
    monto_corte=50000.0;
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 empresa, inversiones: int16; var monto_total: real);
begin
    textcolor(green); write('Introducir código de empresa de la empresa: ');
    textcolor(yellow); readln(empresa);
    textcolor(green); write('Introducir cantidad de inversiones de la empresa: ');
    textcolor(yellow); readln(inversiones);
    if (inversiones>0) then
        leer_inversiones(empresa,inversiones,monto_total);
    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 leer_empresas(var empresa_max, empresas_corte: int16);
var
  empresa, inversiones: int16;
  monto_total, monto_max: real;
begin
  monto_max:=-9999999;
  repeat
    leer_empresa(empresa,inversiones,monto_total);
    if (inversiones>0) then
    begin
      calcular_a(empresa,inversiones,monto_total);
      calcular_b(monto_total,empresa,monto_max,empresa_max);
      calcular_c(monto_total,empresas_corte);
    end;
  until (empresa=empresa_salida);
end;
var
  empresa_max, empresas_corte: int16;
begin
  empresa_max:=0;
  empresas_corte:=0;
  leer_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 TP2_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;
procedure leer_notas(var presente, nota_total, notas_cero, notas_diez: int8);
var
  i, nota: int8;
begin
  presente:=0; nota_total:=0; notas_cero:=0; notas_diez:=0;
  for i:= 1 to autoeva_total do
    begin

```

```

    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 legajo: int16; var condicion: char; var presente, nota_total,
notas_cero, notas_diez: int8);
begin
    textcolor(green); write('Introducir legajo del alumno: ');
    textcolor(yellow); readln(legajo);
    if (legajo<>legajo_salida) then
        begin
            textcolor(green); write('Introducir condición (I para INGRESANTE, R para RECURSANTE) del
alumno: ');
            textcolor(yellow); readln(condicion);
            leer_notas(presente,nota_total,notas_cero,notas_diez);
        end;
    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;
        end;
    end;
end;
procedure calcular_c(presente: int8; var alumnos_autoeva: int16);
begin
    if (presente=autoeva_total) then
        alumnos_autoeva:=alumnos_autoeva+1;
    end;
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;
end;
procedure calcular_e(notas_cero: int8; var alumnos_cero: int16);
begin
    if (notas_cero>=1) then
        alumnos_cero:=alumnos_cero+1;
    end;
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
    end;
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;
end;
procedure leer_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
    presente, nota_total, notas_cero, notas_diez, notas_diez_max1, notas_diez_max2,
notas_cero_max1, notas_cero_max2: int8;
    legajo: int16;
    condicion: char;
begin
    notas_diez_max1:=0; notas_diez_max2:=0;
    notas_cero_max1:=0; notas_cero_max2:=0;
    leer_alumno(legajo,condicion,presente,nota_total,notas_cero,notas_diez);
    while (legajo<>legajo_salida) do
        begin
            calcular_ab(condicion,presente,ingresantes_total,ingresantes_parcial,recursantes_total,rec
ursantes_parcial);
            calcular_c(presente,alumnos_autoeva);
            calcular_d(nota_total,alumnos_corte);
            calcular_e(notas_cero,alumnos_cero);
            calcular_f(notas_diez,legajo,notas_diez_max1,notas_diez_max2,legajo_diez_max1,legajo_diez_
max2);
            calcular_g(notas_cero,legajo,notas_cero_max1,notas_cero_max2,legajo_cero_max1,legajo_cero_
max2);
            leer_alumno(legajo,condicion,presente,nota_total,notas_cero,notas_diez);
        end;
    end;
end;
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;
begin
    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;
    leer_alumnos(ingresantes_parcial,ingresantes_total,recursantes_parcial,recursantes_total,alu
mnos_autoeva,alumnos_corte,alumnos_cero,legajo_diez_max1,legajo_diez_max2,legajo_cero_max1,leg
ajo_cero_max2);
    if ((ingresantes_total>0) or (recursantes_total>0)) then

```

```

begin
  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;
end.

```

Ejercicio 3:

```

program TP2_E17c;
{$codepage UTF8}
uses crt;
const
  tanque_r='R'; tanque_c='C';
  tanque_salida='Z';
  alto_corte=1.40;
  volumen_corte=800.0;
procedure leer_tanque(var tanque: char; var alto, volumen: real);
var
  radio, ancho, largo: real;
begin
  textcolor(green); write('Introducir tipo de tanque vendido (R o C) por el fabricante: ');
  textcolor(yellow); readln(tanque);
  if (tanque<>tanque_salida) then
    begin
      if (tanque=tanque_c) then
        begin

```

```

        textcolor(green); write('Introducir radio del tanque vendido ',tanque,' por el
fabricante: ');
        textcolor(yellow); readln(radio);
        textcolor(green); write('Introducir alto del tanque vendido ',tanque,' por el
fabricante: ');
        textcolor(yellow); readln(alto);
        volumen:=pi*radio*radio*alto;
    end
    else
    begin
        textcolor(green); write('Introducir ancho del tanque vendido ',tanque,' por el
fabricante: ');
        textcolor(yellow); readln(ancho);
        textcolor(green); write('Introducir largo del tanque vendido ',tanque,' por el
fabricante: ');
        textcolor(yellow); readln(largo);
        textcolor(green); write('Introducir alto del tanque vendido ',tanque,' por el
fabricante: ');
        textcolor(yellow); readln(alto);
        volumen:=ancho*largo*alto;
    end;
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 leer_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);
var
    volumen, alto: real;
    tanque: char;
begin
    leer_tanque(tanque,alto,volumen);
    while (tanque<>tanque_salida) do

```

```

begin
    calcular_a(volumen,volumen_max1,volumen_max2);
    calcular_bc(tanque,volumen,volumen_total_c,volumen_total_r,tanques_c,tanques_r);
    calcular_d(alto,tanques_corte_alto);
    calcular_e(volumen,tanques_corte_volumen);
    leer_tanque(tanque,alto,volumen);
end;
end;
var
    tanques_c, tanques_r, tanques_corte_alto, tanques_corte_volumen: int16;
    volumen_max1, volumen_max2, volumen_total_c, volumen_total_r: real;
begin
    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;
    leer_tanques(volumen_max1,volumen_max2,volumen_total_c,volumen_total_r,tanques_c,tanques_r,t
    anques_corte_alto,tanques_corte_volumen);
    if ((tanques_c>0) or (tanques_r>0)) then
        begin
            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.

```

Trabajo Práctico N° 3: Registros. Ejercicios con Corte de Control.

Ejercicio 1.

Dado el siguiente programa:

```
program TP3_E1;
{$codepage UTF8}
uses crt;
type
  str20=string[20];
  alumno=record
    codigo: integer;
    nombre: str20;
    promedio: real;
  end;
procedure leer(var alu: alumno);
begin
  textcolor(green); write('Introducir código de alumno del alumno: '); textcolor(yellow);
  readln(alu.codigo);
  if (alu.codigo<>0) then
  begin
    textcolor(green); write('Introducir nombre del alumno: '); textcolor(yellow);
    readln(alu.nombre);
    textcolor(green); write('Introducir promedio del alumno: '); textcolor(yellow);
    readln(alu.promedio);
  end;
end;
var
  a: alumno;
begin
end.
```

(a) Completar el programa principal para que lea información de alumnos (código, nombre, promedio) e informe la cantidad de alumnos leídos. La lectura finaliza cuando ingresa un alumno con código 0, que no debe procesarse. Nota: Utilizar el módulo leer.

```
program TP3_E1a;
{$codepage UTF8}
uses crt;
const
  alumno_salida=0;
type
  str20=string[20];
  alumno=record
    codigo: integer;
    nombre: str20;
    promedio: real;
  end;
procedure leer(var alu: alumno);
begin
  textcolor(green); write('Introducir código de alumno del alumno: '); textcolor(yellow);
  readln(alu.codigo);
  if (alu.codigo<>alumno_salida) then
  begin
    textcolor(green); write('Introducir nombre del alumno: '); textcolor(yellow);
    readln(alu.nombre);
```

```

    textcolor(green); write('Introducir promedio del alumno: '); textcolor(yellow);
readln(alu.promedio);
    end;
end;
var
    a: alumno;
    alumnos_leidos: integer;
begin
    alumnos_leidos:=0;
    leer(a);
    while (a.codigo<>alumno_salida) do
        begin
            alumnos_leidos:=alumnos_leidos+1;
            leer(a);
        end;
    textcolor(green); write('La cantidad de alumnos leidos es '); textcolor(red);
write(alumnos_leidos);
end.

```

(b) *Modificar al programa anterior para que, al finalizar la lectura de todos los alumnos, se informe también el nombre del alumno con mejor promedio.*

```

program TP3_E1b;
{$codepage UTF8}
uses crt;
const
    alumno_salida=0;
type
    str20=string[20];
    alumno=record
        codigo: integer;
        nombre: str20;
        promedio: real;
    end;
procedure leer(var alu: alumno);
begin
    textcolor(green); write('Introducir código de alumno del alumno: '); textcolor(yellow);
readln(alu.codigo);
    if (alu.codigo<>alumno_salida) then
        begin
            textcolor(green); write('Introducir nombre del alumno: '); textcolor(yellow);
readln(alu.nombre);
            textcolor(green); write('Introducir promedio del alumno: '); textcolor(yellow);
readln(alu.promedio);
        end;
    end;
procedure promedios(alu: alumno; var promedio_max: real; var alumno_max: str20);
begin
    if (alu.promedio>promedio_max) then
        begin
            promedio_max:=alu.promedio;
            alumno_max:=alu.nombre;
        end;
    end;
var
    a: alumno;
    alumnos_leidos: integer;
    promedio_max: real;
    alumno_max: str20;
begin
    alumnos_leidos:=0;
    leer(a);

```



```
while (a.codigo<>alumno_salida) do
begin
    alumnos_leidos:=alumnos_leidos+1;
    promedios(a,promedio_max,alumno_max);
    leer(a);
end;
textcolor(green); write('La cantidad de alumnos leidos es '); textcolor(red);
writeln(alumnos_leidos);
textcolor(green); write('El nombre del alumno con mejor promedio es '); textcolor(red);
write(alumno_max);
end.
```

Ejercicio 2.

El registro civil de La Plata ha solicitado un programa para analizar la distribución de casamientos durante el año 2019. Para ello, cuenta con información de las fechas de todos los casamientos realizados durante ese año.

(a) Analizar y definir un tipo de dato adecuado para almacenar la información de la fecha de cada casamiento.

```
type
  t_dia=1..31;
  t_mes=1..12;
  t_registro_casamiento=record
    dia: t_dia;
    mes: t_mes;
    anio: int16;
  end;
```

(b) Implementar un módulo que lea una fecha desde teclado y la retorne en un parámetro cuyo tipo es el definido en el inciso (a).

```
procedure leer_casamiento(var registro_casamiento: t_registro_casamiento);
begin
  textcolor(green); write('Introducir año del casamiento: '); textcolor(yellow);
  readln(registro_casamiento.anio);
  if (registro_casamiento.anio<>anio_salida) then
  begin
    textcolor(green); write('Introducir mes del casamiento: '); textcolor(yellow);
    readln(registro_casamiento.mes);
    textcolor(green); write('Introducir día del casamiento: '); textcolor(yellow);
    readln(registro_casamiento.dia);
  end;
```

(c) Implementar un programa que:

- Lea la fecha de todos los casamientos realizados en 2019. La lectura finaliza al ingresar el año 2020, que no debe procesarse.*
- Informe la cantidad de casamientos realizados durante los meses de verano (enero, febrero y marzo) y la cantidad de casamientos realizados en los primeros 10 días de cada mes. Nota: Utilizar el módulo realizado en (b) para la lectura de fecha.*

```
program TP3_E2;
{$codepage UTF8}
uses crt;
const
  anio_salida=2020;
  dias_total=31;
  meses_total=12;
  mes_corte1=1; mes_corte2=2; mes_corte3=3;
  dia_corte=10;
type
  t_dia=1..dias_total;
  t_mes=1..meses_total;
  t_registro_casamiento=record
```

```
dia: t_dia;
mes: t_mes;
anio: int16;
end;
procedure leer_casamiento(var registro_casamiento: t_registro_casamiento);
begin
    textcolor(green); write('Introducir año del casamiento: '); textcolor(yellow);
    readln(registro_casamiento.anio);
    if (registro_casamiento.anio<>anio_salida) then
    begin
        textcolor(green); write('Introducir mes del casamiento: '); textcolor(yellow);
        readln(registro_casamiento.mes);
        textcolor(green); write('Introducir día del casamiento: '); textcolor(yellow);
        readln(registro_casamiento.dia);
    end;
end;
procedure leer_casamientos(var casamientos_corte_mes, casamientos_corte_dia: int16);
var
    registro_casamiento: t_registro_casamiento;
begin
    leer_casamiento(registro_casamiento);
    while (registro_casamiento.anio<>anio_salida) do
    begin
        if ((registro_casamiento.mes=mes_corte1) or (registro_casamiento.mes=mes_corte2) or
            (registro_casamiento.mes=mes_corte3)) then
            casamientos_corte_mes:=casamientos_corte_mes+1;
            if (registro_casamiento.dia<=dia_corte) then
                casamientos_corte_dia:=casamientos_corte_dia+1;
            leer_casamiento(registro_casamiento);
        end;
    end;
end;
var
    casamientos_corte_mes, casamientos_corte_dia: int16;
begin
    casamientos_corte_mes:=0;
    casamientos_corte_dia:=0;
    leer_casamientos(casamientos_corte_mes,casamientos_corte_dia);
    textcolor(green); write('La cantidad de casamientos realizados durante los meses de verano
(enero, febrero y marzo) es '); textcolor(red); writeln(casamientos_corte_mes);
    textcolor(green); write('La cantidad de casamientos realizados en los primeros ');
    textcolor(yellow); write(dia_corte); textcolor(green); write(' días de cada mes es ');
    textcolor(red); write(casamientos_corte_dia);
end.
```

Ejercicio 3.

El Ministerio de Educación desea realizar un relevamiento de las 2400 escuelas primarias de la provincia de Bs. As., con el objetivo de evaluar si se cumple la proporción de alumnos por docente calculada por la UNESCO para el año 2015 (1 docente cada 23,435 alumnos). Para ello, se cuenta con información de: CUE (código único de establecimiento), nombre del establecimiento, cantidad de docentes, cantidad de alumnos, localidad. Se pide implementar un programa que procese la información y determine:

- Cantidad de escuelas de La Plata con una relación de alumnos por docente superior a la sugerida por UNESCO.
- CUE y nombre de las dos escuelas con mejor relación entre docentes y alumnos.

El programa debe utilizar:

- Un módulo para la lectura de la información de la escuela.
- Un módulo para determinar la relación docente-alumno (esa relación se obtiene del cociente entre la cantidad de alumnos y la cantidad de docentes).

```
program TP3_E3;
{$codepage UTF8}
uses crt;
const
    escuelas_total=2400;
    localidad_corte='La Plata'; ratio_corte=23.435;
type
    t_registro_escuela=record
        escuela: int16;
        nombre: string;
        docentes: int16;
        alumnos: int16;
        localidad: string;
    end;
procedure leer_escuela(var registro_escuela: t_registro_escuela);
begin
    textcolor(green); write('Introducir CUE (Código Único de Establecimiento) de la escuela: ');
    textcolor(yellow); readln(registro_escuela.escuela);
    textcolor(green); write('Introducir nombre de la escuela: '); textcolor(yellow);
    readln(registro_escuela.nombre);
    textcolor(green); write('Introducir cantidad de docentes de la escuela: ');
    textcolor(yellow); readln(registro_escuela.docentes);
    textcolor(green); write('Introducir cantidad de alumnos de la escuela: ');
    textcolor(yellow); readln(registro_escuela.alumnos);
    textcolor(green); write('Introducir localidad de la escuela: '); textcolor(yellow);
    readln(registro_escuela.localidad);
end;
function ratio_alumnos_docente(registro_escuela: t_registro_escuela): real;
begin
    ratio_alumnos_docente:=registro_escuela.alumnos/registro_escuela.docentes;
end;
procedure actualizar_minimos(ratio: real; registro_escuela: t_registro_escuela; var
ratio_min1, ratio_min2: real; var escuela_min1, escuela_min2: int16; var nombre_min1,
nombre_min2: string);
begin
    if (ratio<ratio_min1) then
    begin
        ratio_min2:=ratio_min1;
        escuela_min2:=escuela_min1;
        nombre_min2:=nombre_min1;
        nombre_min1:=registro_escuela.nombre;
        escuela_min1:=registro_escuela.escuela;
```

```

end
else
  if (ratio<ratio_min2) then
    begin
      ratio_min2:=ratio;
      escuela_min2:=registro_escuela.escuela;
      nombre_min2:=registro_escuela.nombre;
    end;
end;
end;
procedure leer_escuelas(var escuelas_corte, escuela_min1, escuela_min2: int16; var
nombre_min1, nombre_min2: string);
var
  registro_escuela: t_registro_escuela;
  i: int16;
  ratio, ratio_min1, ratio_min2: real;
begin
  ratio:=0;
  ratio_min1:=9999999; ratio_min2:=9999999;
  for i:= 1 to escuelas_total do
    begin
      leer_escuela(registro_escuela);
      ratio:=ratio_alumnos_docente(registro_escuela);
      actualizar_minimos(ratio,registro_escuela,ratio_min1,ratio_min2,escuela_min1,escuela_min2,
nombre_min1,nombre_min2);
      if ((registro_escuela.localidad=localidad_corte) and (ratio>ratio_corte)) then
        escuelas_corte:=escuelas_corte+1;
      end;
    end;
  end;
var
  escuelas_corte, escuela_min1, escuela_min2: int16;
  nombre_min1, nombre_min2: string;
begin
  escuelas_corte:=0;
  escuela_min1:=0; escuela_min2:=0; nombre_min1:=''; nombre_min2:='';
  leer_escuelas(escuelas_corte,escuela_min1,escuela_min2,nombre_min1,nombre_min2);
  textcolor(green); write('La cantidad de escuelas de La Plata con una relación de alumnos por
docente superior a la sugerida por UNESCO ( '); textcolor(yellow); write(ratio_corte:0:2);
textcolor(green); write(') es '); textcolor(red); writeln(escuelas_corte);
  textcolor(green); write('Los CUEs de las dos escuelas con mejor relación entre docentes y
alumnos son '); textcolor(red); write(escuela_min1); textcolor(green); write(' y ');
textcolor(red); writeln(escuela_min2);
  textcolor(green); write('Los nombres de las dos escuelas con mejor relación entre docentes y
alumnos son '); textcolor(red); write(nombre_min1); textcolor(green); write(' y ');
textcolor(red); writeln(nombre_min2);
end.

```

Ejercicio 4.

Una compañía de telefonía celular debe realizar la facturación mensual de sus 9300 clientes con planes de consumo ilimitados (clientes que pagan por lo que consumen). Para cada cliente, se conoce su código de cliente y cantidad de líneas a su nombre. De cada línea, se tiene el número de teléfono, la cantidad de minutos consumidos y la cantidad de MB consumidos en el mes. Se pide implementar un programa que lea los datos de los clientes de la compañía e informe el monto total a facturar para cada uno. Para ello, se requiere:

- Realizar un módulo que lea la información de una línea de teléfono.
- Realizar un módulo que reciba los datos de un cliente, lea la información de todas sus líneas (utilizando el módulo desarrollado en el inciso (a)) y retorne la cantidad total de minutos y la cantidad total de MB a facturar del cliente.

Nota: Para realizar los cálculos tener en cuenta que cada minuto cuesta \$3,40 y cada MB consumido cuesta \$1,35.

```

program TP3_E4;
{$codepage UTF8}
uses crt;
const
  clientes_total=9300;
  costo_minuto=3.40; costo_MB=1.35;
type
  t_cliente=1..clientes_total;
  t_registro_cliente=record
    cliente: int16;
    lineas: int8;
  end;
  t_registro_linea=record
    numero: int16;
    minutos: int16;
    MBs: int16;
  end;
procedure leer_cliente(var registro_cliente: t_registro_cliente);
begin
  textcolor(green); write('Introducir código de cliente del cliente: '); textcolor(yellow);
  readln(registro_cliente.cliente);
  textcolor(green); write('Introducir cantidad de líneas del cliente: '); textcolor(yellow);
  readln(registro_cliente.lineas);
end;
procedure leer_linea(var registro_linea: t_registro_linea);
begin
  textcolor(green); write('Introducir número de teléfono de la línea: '); textcolor(yellow);
  readln(registro_linea.numero);
  textcolor(green); write('Introducir cantidad de minutos consumidos en el mes de la línea: '); textcolor(yellow); readln(registro_linea.minutos);
  textcolor(green); write('Introducir cantidad de MB consumidos en el mes de la línea: '); textcolor(yellow); readln(registro_linea.MBs);
end;
procedure leer_lineas(lineas: int8; var minutos_cliente, MBs_cliente: int16);
var
  registro_linea: t_registro_linea;
  i: int8;
begin
  for i:= 1 to lineas do
    begin
      leer_linea(registro_linea);
      minutos_cliente:=minutos_cliente+registro_linea.minutos;
      MBs_cliente:=MBs_cliente+registro_linea.MBs;
    end;
  end;
end;

```

```
    end;
end;
var
    registro_cliente: t_registro_cliente;
    i: t_cliente;
    minutos_cliente, MBs_cliente: int16;
    monto_cliente: real;
begin
    for i:= 1 to clientes_total do
        begin
            minutos_cliente:=0; MBs_cliente:=0;
            leer_cliente(registro_cliente);
            leer_lineas(registro_cliente.lineas,minutos_cliente,MBs_cliente);
            monto_cliente:=minutos_cliente*costo_minuto+MBs_cliente*costo_MB;
            textcolor(green); write('El monto total a facturar del cliente '); textcolor(yellow);
write(i); textcolor(green); write(' es '); textcolor(red); writeln(monto_cliente:0:2);
        end;
    end.
end.
```

Ejercicio 5.

Realizar un programa que lea información de autos que están a la venta en una concesionaria. De cada auto, se lee: marca, modelo y precio. La lectura finaliza cuando se ingresa la marca “ZZZ”, que no debe procesarse. La información se ingresa ordenada por marca. Se pide calcular e informar:

- El precio promedio por marca.
- Marca y modelo del auto más caro.

```

program TP3_E5;
{$codepage UTF8}
uses crt;
const
    marca_salida='ZZZ';
type
    t_registro_auto=record
        marca: string;
        modelo: string;
        precio: real;
    end;
procedure leer_auto(var registro_auto: t_registro_auto);
begin
    textcolor(green); write('Introducir marca del auto: '); textcolor(yellow);
    readln(registro_auto.marca);
    if (registro_auto.marca<>marca_salida) then
    begin
        textcolor(green); write('Introducir modelo del auto: '); textcolor(yellow);
        readln(registro_auto.modelo);
        textcolor(green); write('Introducir precio del auto: '); textcolor(yellow);
        readln(registro_auto.precio);
    end;
end;
procedure actualizar_maximos(registro_auto: t_registro_auto; var precio_max: real; var
marca_max, modelo_max: string);
begin
    if (registro_auto.precio>precio_max) then
    begin
        precio_max:=registro_auto.precio;
        marca_max:=registro_auto.marca;
        modelo_max:=registro_auto.modelo;
    end;
end;
procedure leer_autos(var marca_max, modelo_max: string);
var
    registro_auto: t_registro_auto;
    autos_total: int32;
    precio_total, precio_prom, precio_max: real;
    marca: string;
begin
    precio_max:=-99999999;
    leer_auto(registro_auto);
    while (registro_auto.marca<>marca_salida) do
    begin
        marca:=registro_auto.marca;
        precio_total:=0; autos_total:=0; precio_prom:=0;
        while ((registro_auto.marca<>marca_salida) and ((registro_auto.marca=marca))) do
        begin
            precio_total:=precio_total+registro_auto.precio;
            autos_total:=autos_total+1;
            actualizar_maximos(registro_auto,precio_max,marca_max,modelo_max);
            leer_auto(registro_auto);
        end;
    end;
end;

```



```
    precio_prom:=precio_total/autos_total;
    textcolor(green); write('El precio promedio de la marca '); textcolor(red); write(marca);
textcolor(green); write(' es '); textcolor(red); writeln(precio_prom:0:2);
    end;
end;
var
    marca_max, modelo_max: string;
begin
    marca_max:=''; modelo_max:='';
    leer_autos(marca_max,modelo_max);
    textcolor(green); write('La marca y el modelo del auto más caro son '); textcolor(red);
write(marca_max); textcolor(green); write(' y '); textcolor(red); write(modelo_max);
textcolor(green); write(', respectivamente');
end.
```

Ejercicio 6.

Una empresa importadora de microprocesadores desea implementar un sistema de software para analizar la información de los productos que mantiene, actualmente, en stock. Para ello, se conoce la siguiente información de los microprocesadores: marca (Intel, AMD, NVidia, etc.), línea (Xeon, Core i7, Opteron, Atom, Centrino, etc.), cantidad de cores o núcleos de procesamiento (1, 2, 4, 8), velocidad del reloj (medida en Ghz) y tamaño en nanómetros (nm) de los transistores (14, 22, 32, 45, etc.). La información de los microprocesadores se lee de forma consecutiva por marca de procesador y la lectura finaliza al ingresar un procesador con 0 cores (que no debe procesarse). Se pide implementar un programa que lea información de los microprocesadores de la empresa importadora e informe:

- Marca y línea de todos los procesadores de más de 2 cores con transistores de, a lo sumo, 22 nm.
- Las dos marcas con mayor cantidad de procesadores con transistores de 14 nm.
- Cantidad de procesadores multicore (de más de un core) de Intel o AMD, cuyos relojes alcancen velocidades de, al menos, 2 Ghz.

```
program TP3_E6;
{$codepage UTF8}
uses crt;
const
    cores_salida=0;
    cores_corte=2;
    transistores_corte1=22;
    transistores_corte2=14;
    velocidad_corte=2.0;
type
    t_registro_procesador=record
        cores: int16;
        marca: string;
        linea: string;
        velocidad: real;
        transistores: int16;
    end;
procedure leer_procesador(var registro_procesador: t_registro_procesador);
begin
    textcolor(green); write('Introducir cantidad de cores o núcleos de procesamiento del
procesador (1, 2, 4, 8): '); textcolor(yellow); readln(registro_procesador.cores);
    if (registro_procesador.cores<>cores_salida) then
        begin
            textcolor(green); write('Introducir marca del procesador (Intel, AMD, NVidia): ');
            textcolor(yellow); readln(registro_procesador.marca);
            textcolor(green); write('Introducir línea del procesador (Xeon, Core i7, Opteron, Atom,
Centrino): '); textcolor(yellow); readln(registro_procesador.linea);
            textcolor(green); write('Introducir velocidad del reloj (medida en Ghz): ');
            textcolor(yellow); readln(registro_procesador.velocidad);
            textcolor(green); write('Introducir tamaño en nanómetros (nm) de los transistores (14, 22,
32, 45): '); textcolor(yellow); readln(registro_procesador.transistores);
        end;
end;
procedure actualizar_maximos(transistores_marca: int16; marca: string; var transistores_max1,
transistores_max2: int16; var marca_max1, marca_max2: string);
begin
    if (transistores_marca>transistores_max1) then
        begin
            transistores_max2:=transistores_max1;
            marca_max2:=marca_max1;
            transistores_max1:=transistores_marca;
```

```

    marca_max1:=marca;
end
else
    if (transistores_marca>transistores_max2) then
    begin
        transistores_max2:=transistores_marca;
        marca_max2:=marca;
    end;
end;
procedure leer_procesadores(var procesadores_corte: int16; var marca_max1, marca_max2:
string);
var
    registro_procesador: t_registro_procesador;
    transistores_marca, transistores_max1, transistores_max2: int16;
    marca: string;
begin
    transistores_max1:=low(int16); transistores_max2:=low(int16);
    leer_procesador(registro_procesador);
    while (registro_procesador.cores<>cores_salida) do
    begin
        marca:=registro_procesador.marca;
        transistores_marca:=0;
        while ((registro_procesador.cores<>cores_salida) and (registro_procesador.marca=marca)) do
        begin
            if ((registro_procesador.cores>cores_corte) and
(registro_procesador.transistores<=transistores_corte1)) then
            begin
                textcolor(green); write('La marca y la línea de este procesador con más de ');
                textcolor(yellow); write(cores_corte); textcolor(green); write(' cores con transistores de, a
lo sumo, '); textcolor(yellow); write(transistores_corte1); textcolor(green); write(' nm. son
'); textcolor(red); write(registro_procesador.marca); textcolor(green); write(' y ');
                textcolor(red); writeln(registro_procesador.linea); textcolor(green); writeln(',
respectivamente');
            end;
            if (registro_procesador.transistores=transistores_corte2) then
                transistores_marca:=transistores_marca+1;
            if ((registro_procesador.cores>=cores_corte) and ((registro_procesador.marca='Intel') or
(registro_procesador.marca='AMD')) and (registro_procesador.velocidad>=velocidad_corte)) then
                procesadores_corte:=procesadores_corte+1;
                leer_procesador(registro_procesador);
            end;
            actualizar_maximos(transistores_marca,marca,transistores_max1,transistores_max2,marca_max1
,marca_max2);
        end;
    end;
end;
var
    procesadores_corte: int16;
    marca_max1, marca_max2: string;
begin
    marca_max1:=''; marca_max2:='';
    procesadores_corte:=0;
    leer_procesadores(procesadores_corte,marca_max1,marca_max2);
    textcolor(green); write('Las dos marcas con mayor cantidad de procesadores con transistores
de '); textcolor(yellow); write(transistores_corte2); textcolor(green); write(' nm. son ');
    textcolor(red); write(marca_max1); textcolor(green); write(' y '); textcolor(red);
    writeln(marca_max2);
    textcolor(green); write('La cantidad de procesadores multicore (de más de un core) de Intel
o AMD, cuyos relojes alcancen velocidades de, al menos, '); textcolor(yellow);
    write(velocidad_corte:0:2); textcolor(green); write(' Ghz es '); textcolor(red);
    write(procesadores_corte);
end.

```

Ejercicio 7.

Realizar un programa que lea información de centros de investigación de Universidades Nacionales. De cada centro, se lee su nombre abreviado (ej., LIDI, LIFIA, LINTI), la universidad a la que pertenece, la cantidad de investigadores y la cantidad de becarios que poseen. La información se lee de forma consecutiva por universidad y la lectura finaliza al leer un centro con 0 investigadores, que no debe procesarse. Informar:

- Cantidad total de centros para cada universidad.
- Universidad con mayor cantidad de investigadores en sus centros.
- Los dos centros con menor cantidad de becarios.

```

program TP3_E7;
{$codepage UTF8}
uses crt;
const
    investigadores_salida=0;
type
    t_registro_centro=record
        centro: string;
        universidad: string;
        investigadores: int16;
        becarios: int16;
    end;
procedure leer_centro(var registro_centro: t_registro_centro);
begin
    textcolor(green); write('Introducir cantidad de investigadores que posee el centro: ');
    textcolor(yellow); readln(registro_centro.investigadores);
    if (registro_centro.investigadores<>investigadores_salida) then
    begin
        textcolor(green); write('Introducir nombre abreviado del centro: '); textcolor(yellow);
        readln(registro_centro.centro);
        textcolor(green); write('Introducir universidad a la que pertenece el centro: ');
        textcolor(yellow); readln(registro_centro.universidad);
        textcolor(green); write('Introducir cantidad de becarios que posee el centro: ');
        textcolor(yellow); readln(registro_centro.becarios);
    end;
end;
procedure actualizar_minimos(becarios: int16; centro: string; var becarios_min1,
    becarios_min2: int16; var centro_min1, centro_min2: string);
begin
    if (becarios<becarios_min1) then
    begin
        becarios_min2:=becarios_min1;
        centro_min2:=centro_min1;
        becarios_min1:=becarios;
        centro_min1:=centro;
    end
    else
    if (becarios<becarios_min2) then
    begin
        becarios_min2:=becarios;
        centro_min2:=centro;
    end;
end;
procedure actualizar_maximo(investigadores_universidad: int16; universidad: string; var
    investigadores_max: int16; var universidad_max: string);
begin
    if (investigadores_universidad>investigadores_max) then
    begin
        investigadores_max:=investigadores_universidad;
        universidad_max:=universidad;
    end;
end;

```

```

    end;
end;
procedure leer_centros(var universidad_max, centro_min1, centro_min2: string);
var
    registro_centro: t_registro_centro;
    centros_universidad, investigadores_universidad, investigadores_max, becarios_min1,
    becarios_min2: int16;
    universidad: string;
begin
    investigadores_max:=low(int16);
    becarios_min1:=high(int16); becarios_min2:=high(int16);
    leer_centro(registro_centro);
    while (registro_centro.investigadores<>investigadores_salida) do
    begin
        universidad:=registro_centro.universidad;
        centros_universidad:=0;
        investigadores_universidad:=0;
        while ((registro_centro.investigadores<>investigadores_salida) and
(registro_centro.universidad=universidad)) do
        begin
            centros_universidad:=centros_universidad+1;
            investigadores_universidad:=investigadores_universidad+registro_centro.investigadores;
            actualizar_minimos(registro_centro.becarios,registro_centro.centro,becarios_min1,becario
s_min2,centro_min1,centro_min2);
            leer_centro(registro_centro);
        end;
        textcolor(green); write('La cantidad total de centros de la universidad ');
        textcolor(red); write(universidad); textcolor(green); write(' es '); textcolor(red);
        writeln(centros_universidad);
        actualizar_maximo(investigadores_universidad,universidad,investigadores_max,universidad_ma
x);
    end;
end;
var
    universidad_max, centro_min1, centro_min2: string;
begin
    universidad_max:='';
    centro_min1:=''; centro_min2:='';
    leer_centros(universidad_max,centro_min1,centro_min2);
    textcolor(green); write('La universidad con mayor cantidad de investigadores en sus centros
es '); textcolor(red); writeln(universidad_max);
    textcolor(green); write('Los dos centros con menor cantidad de becarios son ');
    textcolor(red); write(centro_min1); textcolor(green); write(' y '); textcolor(red);
    write(centro_min2);
end.

```

Ejercicio 8.

La Comisión Provincial por la Memoria desea analizar la información de los proyectos presentados en el programa Jóvenes y Memoria durante la convocatoria 2020. Cada proyecto, posee un código único, un título, el docente coordinador (DNI, nombre y apellido, email), la cantidad de alumnos que participan del proyecto, el nombre de la escuela y la localidad a la que pertenece. Cada escuela, puede presentar más de un proyecto. La información se ingresa ordenada consecutivamente por localidad y, para cada localidad, por escuela. Realizar un programa que lea la información de los proyectos hasta que se ingrese el proyecto con código -1 (que no debe procesarse), e informe:

- Cantidad total de escuelas que participan en la convocatoria 2020 y cantidad de escuelas por cada localidad.
- Nombres de las dos escuelas con mayor cantidad de alumnos participantes.
- Título de los proyectos de la localidad de Daireaux cuyo código posee igual cantidad de dígitos pares e impares.

```

program TP3_E8;
{$codepage UTF8}
uses crt;
const
    proyecto_salida=-1;
    localidad_corte='Daireaux';
type
    t_registro_docente=record
        dni: int32;
        nombre: string;
        apellido: string;
        email: string;
    end;
    t_registro_proyecto=record
        proyecto: int16;
        titulo: string;
        docente: t_registro_docente;
        alumnos: int16;
        escuela: string;
        localidad: string;
    end;
procedure leer_docente(var registro_docente: t_registro_docente);
begin
    textcolor(green); write('Introducir DNI del docente: '); textcolor(yellow);
    readln(registro_docente.dni);
    textcolor(green); write('Introducir nombre del docente: '); textcolor(yellow);
    readln(registro_docente.nombre);
    textcolor(green); write('Introducir apellido del docente: '); textcolor(yellow);
    readln(registro_docente.apellido);
    textcolor(green); write('Introducir email del docente: '); textcolor(yellow);
    readln(registro_docente.email);
end;
procedure leer_proyecto(var registro_proyecto: t_registro_proyecto);
begin
    textcolor(green); write('Introducir código único del proyecto: '); textcolor(yellow);
    readln(registro_proyecto.proyecto);
    if (registro_proyecto.proyecto<>proyecto_salida) then
        begin
            textcolor(green); write('Introducir título del proyecto: '); textcolor(yellow);
            readln(registro_proyecto.titulo);
            leer_docente(registro_proyecto.docente);
        end;
    end;
end;

```

```

    textcolor(green); write('Introducir cantidad de alumnos que participan del proyecto: ');
textcolor(yellow); readln(registro_proyecto.alumnos);
    textcolor(green); write('Introducir nombre de la escuela del proyecto: ');
textcolor(yellow); readln(registro_proyecto.escuela);
    textcolor(green); write('Introducir localidad a la que pertenece el proyecto: ');
textcolor(yellow); readln(registro_proyecto.localidad);
    end;
end;
procedure actualizar_maximos(alumnos: int16; escuela: string; var alumnos_max1, alumnos_max2:
int16; var escuela_max1, escuela_max2: string);
begin
    if (alumnos>alumnos_max1) then
    begin
        alumnos_max2:=alumnos_max1;
        escuela_max2:=escuela_max1;
        alumnos_max1:=alumnos;
        escuela_max1:=escuela;
    end
    else
        if (alumnos>alumnos_max2) then
        begin
            alumnos_max2:=alumnos;
            escuela_max2:=escuela;
        end;
    end;
end;
function contar_pares_impares(proyecto: int16): boolean;
var
    pares, impares: int16;
begin
    pares:=0; impares:=0;
    while (proyecto<>0) do
    begin
        if (proyecto mod 2=0) then
            pares:=pares+1
        else
            impares:=impares+1;
        proyecto:=proyecto div 10;
    end;
    if (pares=impares) then
        contar_pares_impares:=true
    else
        contar_pares_impares:=false;
    end;
end;
procedure leer_proyectos(var escuelas_total: int16; var escuela_max1, escuela_max2: string);
var
    registro_proyecto: t_registro_proyecto;
    escuelas_localidad, alumnos_escuela, alumnos_max1, alumnos_max2: int16;
    localidad, escuela: string;
begin
    alumnos_max1:=low(int16); alumnos_max2:=low(int16);
    leer_proyecto(registro_proyecto);
    while (registro_proyecto.proyecto<>proyecto_salida) do
    begin
        localidad:=registro_proyecto.localidad;
        escuelas_localidad:=0;
        while ((registro_proyecto.proyecto<>proyecto_salida) and
(registro_proyecto.localidad=localidad)) do
        begin
            escuela:=registro_proyecto.escuela;
            escuelas_localidad:=escuelas_localidad+1;
            alumnos_escuela:=0;
            while ((registro_proyecto.escuela=escuela) and (registro_proyecto.localidad=localidad)
and (registro_proyecto.proyecto<>proyecto_salida)) do
            begin
                alumnos_escuela:=alumnos_escuela+registro_proyecto.alumnos;

```

```

        if ((registro_proyecto.localidad=localidad_corte) and
(contar_pares_impares(registro_proyecto.proyecto)=true)) then
            begin
                textcolor(green); write('El título de este proyecto de la localidad ');
textcolor(yellow); write(localidad_corte); textcolor(green); write(', cuyo código posee igual
cantidad de dígitos pares e impares, es '); textcolor(red); writeln(registro_proyecto.titulo);
                end;
                leer_proyecto(registro_proyecto);
            end;
            escuelas_total:=escuelas_total+escuelas_localidad;
            actualizar_maximos(alumnos_escuela,escuela,alumnos_max1,alumnos_max2,escuela_max1,escuel
a_max2);
        end;
        textcolor(green); write('La cantidad de escuelas de la localidad '); textcolor(red);
write(localidad); textcolor(green); write(' es '); textcolor(red);
writeln(escuelas_localidad);
    end;
end;
var
    escuelas_total: int16;
    escuela_max1, escuela_max2: string;
begin
    escuelas_total:=0;
    escuela_max1:=''; escuela_max2:='';
    leer_proyectos(escuelas_total,escuela_max1,escuela_max2);
    textcolor(green); write('La cantidad total de escuelas que participan en la convocatoria
2020 es '); textcolor(red); writeln(escuelas_total);
    textcolor(green); write('Los nombres de las dos escuelas con mayor cantidad de alumnos
participantes son '); textcolor(red); write(escuela_max1); textcolor(green); write(' y ');
textcolor(red); write(escuela_max2);
end.

```


Ejercicio 9.

Realizar un programa que lea información de los candidatos ganadores de las últimas elecciones a intendente de la provincia de Buenos Aires. Para cada candidato, se lee: localidad, apellido del candidato, cantidad de votos obtenidos y cantidad de votantes de la localidad. La lectura finaliza al leer la localidad 'Zárate', que debe procesarse. Informar:

- El intendente que obtuvo la mayor cantidad de votos en la elección.
- El intendente que obtuvo el mayor porcentaje de votos de la elección.

```

program TP3_E9;
{$codepage UTF8}
uses crt;
const
    localidad_salida='Zarate';
type
    t_registro_candidato=record
        localidad: string;
        apellido: string;
        votos: int16;
        votantes: int16;
    end;
procedure leer_candidato(var registro_candidato: t_registro_candidato);
begin
    textcolor(green); write('Introducir localidad del candidato: '); textcolor(yellow);
    readln(registro_candidato.localidad);
    if (registro_candidato.localidad<>localidad_salida) then
    begin
        textcolor(green); write('Introducir apellido del candidato: '); textcolor(yellow);
        readln(registro_candidato.apellido);
        textcolor(green); write('Introducir cantidad de votos obtenidos del candidato: ');
        textcolor(yellow); readln(registro_candidato.votos);
        textcolor(green); write('Introducir cantidad de votantes de la localidad: ');
        textcolor(yellow); readln(registro_candidato.votantes);
    end;
end;
procedure actualizar_maximo_cantidad(votos: int16; intendente: string; var votos_cantidad:
int16; var intendente_cantidad: string);
begin
    if (votos>votos_cantidad) then
    begin
        votos_cantidad:=votos;
        intendente_cantidad:=intendente;
    end;
end;
procedure actualizar_maximo_porcentaje(porcentaje: real; intendente: string; var
votos_porcentaje: real; var intendente_porcentaje: string);
begin
    if (porcentaje>votos_porcentaje) then
    begin
        votos_porcentaje:=porcentaje;
        intendente_porcentaje:=intendente;
    end;
end;
procedure leer_candidatos(var intendente_cantidad, intendente_porcentaje: string);
var
    registro_candidato: t_registro_candidato;
    votos_cantidad: int16;
    porcentaje, votos_porcentaje: real;
begin
    votos_cantidad:=low(int16);

```

```
votos_porcentaje:=-999999;
leer_candidato(registro_candidato);
while (registro_candidato.localidad<>localidad_salida) do
begin
    actualizar_maximo_cantidad(registro_candidato.votos,registro_candidato.apellido,votos_cant
idad,intendente_cantidad);
    porcentaje:=registro_candidato.votos/registro_candidato.votantes*100;
    actualizar_maximo_porcentaje(porcentaje,registro_candidato.apellido,votos_porcentaje,inten
dente_porcentaje);
    leer_candidato(registro_candidato);
end;
end;
var
    intendente_cantidad, intendente_porcentaje: string;
begin
    intendente_cantidad:='';
    intendente_porcentaje:='';
    leer_candidatos(intendente_cantidad,intendente_porcentaje);
    textcolor(green); write('El intendente que obtuvo la mayor cantidad de votos en la elección
es '); textcolor(red); writeln(intendente_cantidad);
    textcolor(green); write('El intendente que obtuvo el mayor porcentaje de votos en la
elección es '); textcolor(red); write(intendente_porcentaje);
end.
```

Ejercicio 10.

Un centro de investigación de la UNLP está organizando la información de las 320 especies de plantas con las que trabajan. Para cada especie, se ingresa su nombre científico, tiempo promedio de vida (en meses), tipo de planta (por ej., árbol, conífera, arbusto, helecho, musgo, etc.), clima (templado, continental, subtropical, desértico, etc.) y países en el mundo donde se las encuentra. La información de las plantas se ingresa ordenada por tipo de planta y, para cada planta, la lectura de países donde se las encuentra finaliza al ingresar el país 'zzz'. Al finalizar la lectura, informar:

- El tipo de planta con menor cantidad de plantas.
- El tiempo promedio de vida de las plantas de cada tipo.
- El nombre científico de las dos plantas más longevas.
- Los nombres de las plantas nativas de Argentina que se encuentran en regiones con clima subtropical.
- El nombre de la planta que se encuentra en más países.

```
program TP3_E10;
{$codepage UTF8}
uses crt;
const
  plantas_total=320;
  pais_salida='zzz';
  pais_corte='Argentina'; clima_corte='subtropical';
type
  t_registro_planta=record
    nombre: string;
    vida: int16;
    tipo: string;
    clima: string;
    pais: string;
  end;
procedure leer_planta(var registro_planta: t_registro_planta);
begin
  textcolor(green); write('Introducir país en el mundo donde se encuentra la especie de
planta: '); textcolor(yellow); readln(registro_planta.pais);
  if (registro_planta.pais<>pais_salida) then
  begin
    textcolor(green); write('Introducir nombre científico de la especie de planta: ');
    textcolor(yellow); readln(registro_planta.nombre);
    textcolor(green); write('Introducir tiempo promedio de vida (en meses) de la especie de
planta: '); textcolor(yellow); readln(registro_planta.vida);
    textcolor(green); write('Introducir tipo de planta (árbol, conífera, arbusto, helecho,
musgo, etc.) de la especie de planta: '); textcolor(yellow); readln(registro_planta.tipo);
    textcolor(green); write('Introducir clima (templado, continental, subtropical, desértico,
etc.) de la especie de planta: '); textcolor(yellow); readln(registro_planta.clima);
  end;
end;
procedure actualizar_minimo(plantas_tipo: int16; tipo: string; var plantas_min: int16; var
tipo_min: string);
begin
  if (plantas_tipo<plantas_min) then
  begin
    plantas_min:=plantas_tipo;
    tipo_min:=tipo;
  end;
end;
function calcular_tiempo_promedio(vida_tipo, plantas_tipo_paises: int16): real;
begin
  calcular_tiempo_promedio:=vida_tipo/plantas_tipo_paises;
```

```

end;
procedure actualizar_maximos(vida: int16; planta: string; var vida_max1, vida_max2: int16; var
planta_max1, planta_max2: string);
begin
    if (vida>vida_max1) then
        begin
            vida_max2:=vida_max1;
            planta_max2:=planta_max1;
            vida_max1:=vida;
            planta_max1:=planta;
        end
    else
        if (vida>vida_max2) then
            begin
                vida_max2:=vida;
                planta_max2:=planta;
            end;
    end;
end;
procedure actualizar_maximo(paises_planta: int16; planta: string; var paises_max3: int16; var
planta_max3: string);
begin
    if (paises_planta>paises_max3) then
        begin
            paises_max3:=paises_planta;
            planta_max3:=planta;
        end;
end;
procedure leer_plantas(var tipo_min, planta_max1, planta_max2, planta_max3: string);
var
    registro_planta: t_registro_planta;
    plantas, plantas_tipo, plantas_min, vida_tipo, plantas_tipo_paises, vida_max1, vida_max2,
paises_planta, paises_max3: int16;
    tipo, planta: string;
begin
    plantas:=0;
    plantas_min:=high(int16);
    vida_max1:=low(int16); vida_max2:=low(int16);
    paises_max3:=low(int16);
    leer_planta(registro_planta);
    while (plantas<plantas_total) do
        begin
            tipo:=registro_planta.tipo;
            plantas_tipo:=0;
            vida_tipo:=0; plantas_tipo_paises:=0;
            while ((plantas<plantas_total) and (registro_planta.tipo=tipo)) do
                begin
                    planta:=registro_planta.nombre;
                    plantas_tipo:=plantas_tipo+1;
                    paises_planta:=0;
                    while ((plantas<plantas_total) and (registro_planta.tipo=tipo) and
(registro_planta.pais<>pais_salida)) do
                        begin
                            vida_tipo:=vida_tipo+registro_planta.vida;
                            plantas_tipo_paises:=plantas_tipo_paises+1;
                            actualizar_maximos(registro_planta.vida,registro_planta.nombre,vida_max1,vida_max2,pla
nta_max1,planta_max2);
                            if ((registro_planta.pais=pais_corte) and (registro_planta.clima=clima_corte)) then
                                begin
                                    textcolor(green); write('El nombre de la planta nativa de '); textcolor(yellow);
write(pais_corte); textcolor(green); write(' que se encuentran en una región con clima ');
textcolor(yellow); write(clima_corte); textcolor(green); write(' es '); textcolor(red);
writeln(registro_planta.nombre);
                                end;
                                paises_planta:=paises_planta+1;
                                leer_planta(registro_planta);
                            end;
                        end;
                    end;
                end;
            end;
        end;
    end;

```

```
    actualizar_maximo(países_planta,planta,países_max3,planta_max3);
    plantas:=plantas+1;
    if (plantas<plantas_total) then
        leer_planta(registro_planta);
    end;
    actualizar_minimo(plantas_tipo,tipo,plantas_min,tipo_min);
    textcolor(green); write('El tiempo de vida promedio de las plantas de tipo ');
textcolor(red); write(tipo); textcolor(green); write(' es '); textcolor(red);
writeln(calcular_tiempo_promedio(vida_tipo,plantas_tipo_países):0:2);
    end;
end;
var
    tipo_min, planta_max1, planta_max2, planta_max3: string;
begin
    tipo_min:='';
    planta_max1:=''; planta_max2:='';
    planta_max3:='';
    leer_plantas(tipo_min,planta_max1,planta_max2,planta_max3);
    textcolor(green); write('El tipo de planta con menor cantidad de plantas es ');
textcolor(red); writeln(tipo_min);
    textcolor(green); write('Los nombres científicos de las dos plantas más longevas son ');
textcolor(red); write(planta_max1); textcolor(green); write(' y '); textcolor(red);
writeln(planta_max2);
    textcolor(green); write('El nombre de la planta que se encuentra en más países es ');
textcolor(red); write(planta_max3);
end.
```

Ejercicio 11.

Una compañía de vuelos internacionales está analizando la información de todos los vuelos realizados por sus aviones durante todo el año 2019. De cada vuelo, se conoce el código de avión, país de salida, país de llegada, cantidad de kilómetros recorridos y porcentaje de ocupación del avión. La información se ingresa ordenada por código de avión y, para cada avión, por país de salida. La lectura finaliza al ingresar el código 44. Informar:

- Los dos aviones que más kilómetros recorrieron y los dos aviones que menos kilómetros recorrieron.
- El avión que salió desde más países diferentes.
- La cantidad de vuelos de más de 5.000 km que no alcanzaron el 60% de ocupación del avión.
- La cantidad de vuelos de menos de 10.000 km que llegaron a Australia o a Nueva Zelanda.

```

program TP3_E11;
{$codepage UTF8}
uses crt;
const
  avion_salida=44;
  kms_corte1=5000.0; ocupacion_corte=60.0;
  kms_corte2=10000.0; pais_corte1='Australia'; pais_corte2='Nueva Zelanda';
type
  t_registro_vuelo=record
    avion: int16;
    pais_salida: string;
    pais_llegada: string;
    kms: real;
    ocupacion: real;
  end;
procedure leer_vuelo(var registro_vuelo: t_registro_vuelo);
begin
  textcolor(green); write('Introducir código de avión del vuelo: '); textcolor(yellow);
  readln(registro_vuelo.avion);
  if (registro_vuelo.avion<>avion_salida) then
  begin
    textcolor(green); write('Introducir país de salida del vuelo: '); textcolor(yellow);
    readln(registro_vuelo.pais_salida);
    textcolor(green); write('Introducir país de llegada del vuelo: '); textcolor(yellow);
    readln(registro_vuelo.pais_llegada);
    textcolor(green); write('Introducir cantidad de kilómetros recorridos del vuelo: ');
    textcolor(yellow); readln(registro_vuelo.kms);
    textcolor(green); write('Introducir porcentaje de ocupación del avión del vuelo: ');
    textcolor(yellow); readln(registro_vuelo.ocupacion);
  end;
end;
procedure actualizar_maximos(kms_avion: real; avion: int16; var kms_max1, kms_max2: real; var
  avion_max1, avion_max2: int16);
begin
  if (kms_avion>kms_max1) then
  begin
    kms_max2:=kms_max1;
    avion_max2:=avion_max1;
    kms_max1:=kms_avion;
    avion_max1:=avion;
  end
  else
    if (kms_avion>kms_max2) then

```

```

begin
    kms_max2:=kms_avion;
    avion_max2:=avion;
end;
end;
procedure actualizar_minimos(var kms_avion: real; avion: int16; var kms_min1, kms_min2: real;
var avion_min1, avion_min2: int16);
begin
    if (kms_avion<kms_min1) then
    begin
        kms_min2:=kms_min1;
        avion_min2:=avion_min1;
        kms_min1:=kms_avion;
        avion_min1:=avion;
    end
    else
        if (kms_avion<kms_min2) then
        begin
            kms_min2:=kms_avion;
            avion_min2:=avion;
        end;
    end;
end;
procedure actualizar_maximo(países_avion, avion: int16; var países_max3, avion_max3: int16);
begin
    if (países_avion>países_max3) then
    begin
        países_max3:=países_avion;
        avion_max3:=avion;
    end;
end;
procedure leer_vuelos(var avion_max1, avion_max2, avion_min1, avion_min2, avion_max3,
vuelos_corte1, vuelos_corte2: int16);
var
    registro_vuelo: t_registro_vuelo;
    avion, países_avion, países_max3: int16;
    kms_avion, kms_max1, kms_max2, kms_min1, kms_min2: real;
    país: string;
begin
    kms_max1:=-9999999; kms_max2:=-9999999;
    kms_min1:=9999999; kms_min2:=9999999;
    países_max3:=low(int16);
    leer_vuelo(registro_vuelo);
    while (registro_vuelo.avion<>avion_salida) do
    begin
        avion:=registro_vuelo.avion;
        kms_avion:=0;
        países_avion:=0;
        while ((registro_vuelo.avion<>avion_salida) and (registro_vuelo.avion=avion)) do
        begin
            país:=registro_vuelo.país_salida;
            países_avion:=países_avion+1;
            while ((registro_vuelo.avion<>avion_salida) and (registro_vuelo.avion=avion) and
(registro_vuelo.país_salida=país)) do
            begin
                kms_avion:=kms_avion+registro_vuelo.kms;
                if ((registro_vuelo.kms>kms_corte1) and (registro_vuelo.ocupacion<ocupacion_corte))
then
                    vuelos_corte1:=vuelos_corte1+1;
                    if ((registro_vuelo.kms<kms_corte2) and ((registro_vuelo.país_llegada=país_corte1) or
(registro_vuelo.país_llegada=país_corte2))) then
                        vuelos_corte2:=vuelos_corte2+1;
                        leer_vuelo(registro_vuelo);
                    end;
                end;
            end;
        end;
        actualizar_maximos(kms_avion,avion,kms_max1,kms_max2,avion_max1,avion_max2);
        actualizar_minimos(kms_avion,avion,kms_min1,kms_min2,avion_min1,avion_min2);
    end;
end;

```

```
    actualizar_maximo(países_avion,avion,países_max3,avion_max3);
end;
end;
var
    avion_max1, avion_max2, avion_min1, avion_min2, avion_max3, vuelos_corte1, vuelos_corte2:
    int16;
begin
    avion_max1:=0; avion_max2:=0; avion_min1:=0; avion_min2:=0;
    avion_max3:=0;
    vuelos_corte1:=0;
    vuelos_corte2:=0;
    leer_vuelos(avion_max1,avion_max2,avion_min1,avion_min2,avion_max3,vuelos_corte1,vuelos_corte2);
    textcolor(green); write('Los dos aviones que más kilómetros recorrieron son ');
    textcolor(red); write(avion_max1); textcolor(green); write(' y '); textcolor(red);
    writeln(avion_max2);
    textcolor(green); write('Los dos aviones que menos kilómetros recorrieron son ');
    textcolor(red); write(avion_min1); textcolor(green); write(' y '); textcolor(red);
    writeln(avion_min2);
    textcolor(green); write('El avión que salió de más países diferentes es '); textcolor(red);
    writeln(avion_max3);
    textcolor(green); write('La cantidad de vuelos de más de '); textcolor(yellow);
    write(kms_corte1:0:2); textcolor(green); write(' kms que no alcanzaron el ');
    textcolor(yellow); write(ocupacion_corte:0:2); textcolor(green); write('% de ocupación del
    avión es '); textcolor(red); writeln(vuelos_corte1);
    textcolor(green); write('La cantidad de vuelos de menos de '); textcolor(yellow);
    write(kms_corte2:0:2); textcolor(green); write(' kms que llegaron a '); textcolor(yellow);
    write(pais_corte1); textcolor(green); write(' o a '); textcolor(red); write(pais_corte2);
    textcolor(green); write(' es '); textcolor(red); write(vuelos_corte2);
end.
```


Ejercicio 12.

En la “Práctica 1 - Ejercicios Adicionales”, se resolvieron 3 problemas complejos sin utilizar módulos. Al carecer de herramientas para modularizar, esos programas resultaban difíciles de leer, de extender y de depurar. En la “Práctica 2 (parte 2) - Ejercicios Adicionales”, se adaptaron los 3 problemas para utilizar módulos y, así, organizar mejor el programa. Ahora, podemos incluir los registros y, así, seguir mejorando nuestros programas. Para cada caso, analizar:

- ¿Qué entidades del programa conviene representar como registros?
- ¿Qué atributos de cada entidad deben incluirse en los registros?
- ¿Qué cambios deben realizarse en los módulos implementados en la Práctica 2 para aprovechar los nuevos tipos de datos? ¿Conviene seguir utilizando los mismos módulos en todos los casos?

Una vez realizado el análisis, modificar los 3 problemas, utilizando registros para representar los datos del programa. Al finalizar cada problema, comparar la solución usando registros y módulos con la solución sin registros y con módulos (Práctica 2) y con la solución sin registros ni módulos (Práctica 1).

- ¿Qué diferencias observa?
- ¿Qué similitudes encuentra?

Ejercicio 1:

```
program TP3_E12a;
{$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;
procedure leer_inversiones(empresa: 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.monto_total);
end;
```

```

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 leer_empresas(var empresa_max, empresas_corte: int16);
var
    registro_empresa: t_registro_empresa;
    monto_max: real;
begin
    monto_max:=-9999999;
    repeat
        leer_empresa(registro_empresa);
        calcular_a(registro_empresa.empresa,registro_empresa.inversiones,registro_empresa.monto_to
tal);
        calcular_b(registro_empresa.monto_total,registro_empresa.empresa,monto_max,empresa_max);
        calcular_c(registro_empresa.monto_total,empresas_corte);
    until (registro_empresa.empresa=empresa_salida);
end;
var
    empresa_max, empresas_corte: int16;
begin
    empresa_max:=0;
    empresas_corte:=0;
    leer_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 TP3_E12b;
{$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;
type
    t_registro_alumno=record

```

```
legajo: int16;
condicion: char;
presente: int8;
notas_cero: int8;
notas_diez: int8;
nota_total: int8;
end;
procedure leer_notas(var presente, notas_cero, notas_diez, nota_total: int8);
var
  i, nota: int8;
begin
  presente:=0; nota_total:=0; notas_cero:=0; notas_diez:=0;
  for i:= 1 to autoeva_total do
    begin
      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;
  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);
      leer_notas(registro_alumno.presente,registro_alumno.notas_cero,registro_alumno.notas_diez,
registro_alumno.nota_total);
    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;
  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;
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;
end;
procedure leer_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
  registro_alumno: t_registro_alumno;
  notas_diez_max1, notas_diez_max2, notas_cero_max1, notas_cero_max2: int8;
begin
  notas_diez_max1:=0; notas_diez_max2:=0;
  notas_cero_max1:=0; notas_cero_max2:=0;
  leer_alumno(registro_alumno);
  while (registro_alumno.legajo<>legajo_salida) do
    begin
      calcular_ab(registro_alumno.condicion,registro_alumno.presente,ingresantes_total,ingresant
es_parcial,recursantes_total,recursantes_parcial);
      calcular_c(registro_alumno.presente,alumnos_autoeva);
      calcular_d(registro_alumno.nota_total,alumnos_corte);
      calcular_e(registro_alumno.notas_cero,alumnos_cero);
      calcular_f(registro_alumno.notas_diez,registro_alumno.legajo,notas_diez_max1,notas_diez_ma
x2,legajo_diez_max1,legajo_diez_max2);
      calcular_g(registro_alumno.notas_cero,registro_alumno.legajo,notas_cero_max1,notas_cero_ma
x2,legajo_cero_max1,legajo_cero_max2);
      leer_alumno(registro_alumno);
    end;
  end;
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;

```

```

begin
  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;
  leer_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) or (recursantes_total>0)) then
    begin
      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 TP3_E12c;
{$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;
procedure leer_tanque(var registro_tanque: t_registro_tanque);
begin
    textcolor(green); write('Introducir tipo de tanque vendido (R o C) por el fabricante: ');
    textcolor(yellow); readln(registro_tanque.tanque);
    if (registro_tanque.tanque<>'tanque_salida') then
    begin
        if (registro_tanque.tanque='tanque_c') then
        begin
            textcolor(green); write('Introducir radio del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.radio);
            textcolor(green); write('Introducir alto del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.alto);
            registro_tanque.volumen:=pi*registro_tanque.radio*registro_tanque.radio*registro_tanque.
alto;
        end
        else
        begin
            textcolor(green); write('Introducir ancho del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.ancho);
            textcolor(green); write('Introducir largo del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.largo);
            textcolor(green); write('Introducir alto del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.alto);
            registro_tanque.volumen:=registro_tanque.ancho*registro_tanque.largo*registro_tanque.alt
o;
        end;
    end;
end;
procedure 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;
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;
    end
end;

```

```

    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 leer_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);
var
    registro_tanque: t_registro_tanque;
begin
    leer_tanque(registro_tanque);
    while (registro_tanque.tanque<>tanque_salida) do
        begin
            calcular_a(registro_tanque.volumen,volumen_max1,volumen_max2);
            calcular_bc(registro_tanque.tanque,registro_tanque.volumen,volumen_total_c,volumen_total_r,
tanques_c,tanques_r);
            calcular_d(registro_tanque.alto,tanques_corte_alto);
            calcular_e(registro_tanque.volumen,tanques_corte_volumen);
            leer_tanque(registro_tanque);
        end;
    end;
var
    tanques_c, tanques_r, tanques_corte_alto, tanques_corte_volumen: int16;
    volumen_max1, volumen_max2, volumen_total_c, volumen_total_r: real;
begin
    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;
    leer_tanques(volumen_max1,volumen_max2,volumen_total_c,volumen_total_r,tanques_c,tanques_r,t
anques_corte_alto,tanques_corte_volumen);
    if ((tanques_c>0) or (tanques_r>0)) then
        begin
            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.
```


Trabajo Práctico N° 4.1: **Vectores (Parte 1).**

Ejercicio 1.

```
program TP4_E1;
{$codepage UTF8}
uses crt;
type
  vnums=array[1..10] of integer;
var
  numeros: vnums;
  i: integer;
begin
  for i:= 1 to 10 do
    numeros[i]:=i;
  for i:= 2 to 10 do
    numeros[i]:=numeros[i]+numeros[i-1]
end.
```

(a) ¿Qué valores toma la variable *numeros* al finalizar el primer bloque *for*?

(b) Al terminar el programa, ¿con qué valores finaliza la variable *numeros*?

```
program TP4_E1ab;
{$codepage UTF8}
uses crt;
type
  vnums=array[1..10] of integer;
var
  numeros: vnums;
  i: integer;
begin
  for i:= 1 to 10 do
    begin
      numeros[i]:=i;
      if (i<10) then
        write(numeros[i],', ')
      else
        writeln(numeros[i])
    end;
  for i:= 2 to 10 do
    begin
      numeros[i]:=numeros[i]+numeros[i-1];
      if (i<10) then
        write(numeros[i],', ')
      else
        writeln(numeros[i])
    end;
end.
```

Los valores que toma la variable “*numeros*” al finalizar el primer bloque *for* son 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

Al terminar el programa, la variable “*numeros*” finaliza con los valores 1, 3, 6, 10, 15, 21, 28, 36, 45, 55.

(c) Si se desea cambiar la línea 11 por la sentencia: *for i:=1 to 9 do*, ¿cómo debe modificarse el código para que la variable *números* contenga los mismos valores que en (1.b)?

```
program TP4_E1c;
{$codepage UTF8}
uses crt;
type
  vnums=array[1..10] of integer;
var
  numeros: vnums;
  i: integer;
begin
  for i:= 1 to 10 do
  begin
    numeros[i]:=i;
    if (i<10) then
      write(numeros[i],', ')
    else
      writeln(numeros[i])
    end;
  for i:= 1 to 9 do
  begin
    numeros[i+1]:=numeros[i+1]+numeros[i];
    if (i+1<10) then
      write(numeros[i+1],', ')
    else
      writeln(numeros[i+1])
    end;
  end.
end.
```

(d) ¿Qué valores están contenidos en la variable *numeros* si las líneas 11 y 12 se reemplazan por *for i:= 1 to 9 do numeros[i+1]:=numeros[i];*?

```
program TP4_E1d;
{$codepage UTF8}
uses crt;
type
  vnums=array[1..10] of integer;
var
  numeros: vnums;
  i: integer;
begin
  for i:= 1 to 10 do
  begin
    numeros[i]:=i;
    if (i<10) then
      write(numeros[i],', ')
    else
      writeln(numeros[i])
    end;
  for i:= 1 to 9 do
  begin
    numeros[i+1]:=numeros[i];
    if (i<9) then
      write(numeros[i],', ')
    else
      writeln(numeros[i])
    end;
  end.
end.
```

Los valores que están contenidos en la variable “*numeros*” si las líneas 11 y 12 se reemplazan por *for i:= 1 to 9 do numeros[i+1]:=numeros[i]* son 1, 1, 1, 1, 1, 1, 1, 1, 1, 1.

Ejercicio 2.

Dado el siguiente programa, completar las líneas indicadas, considerando que:

- El módulo `cargarVector` debe leer números reales y almacenarlos en el vector que se pasa como parámetro. Al finalizar, debe retornar el vector y su dimensión lógica. La lectura finaliza cuando se ingresa el valor 0 (que no debe procesarse) o cuando el vector está completo.
- El módulo `modificarVectorySumar` debe devolver el vector con todos sus elementos incrementados con el valor n y también debe devolver la suma de todos los elementos del vector.

```

program TP4_E2;
{$codepage UTF8}
uses crt;
const
    cant_datos=150;
type
    vdatos=array[1..cant_datos] of real;
procedure cargarVector(var v: vdatos; var dimL: integer);
var
    num_real: int16;
begin
    textcolor(green); write('Introducir número real: ');
    textcolor(yellow); readln(num_real);
    while ((num_real<>0) and (dimL<cant_datos)) do
        begin
            dimL:=dimL+1;
            v[dimL]:=num_real;
            textcolor(green); write('Introducir número real: ');
            textcolor(yellow); readln(num_real);
        end;
    end;
procedure modificarVectorySumar(var v: vdatos; dimL: integer; n: real; var suma: real);
var
    i: int16;
begin
    for i:= 1 to dimL do
        begin
            v[i]:=v[i]+n;
            suma:=suma+v[i];
        end;
    end;
var
    datos: vdatos;
    dim: integer;
    num, suma: real;
begin
    dim:=0; suma:=0;
    cargarVector(datos,dim);
    textcolor(green); write('Introducir valor a sumar: ');
    textcolor(yellow); readln(num);
    modificarVectorySumar(datos,dim,num,suma);
    textcolor(green); write('La suma de los valores es '); textcolor(red); writeln(suma:0:2);
    textcolor(green); write('Se procesaron '); textcolor(red); write(dim); textcolor(green);
    write(' números');
end.

```

Ejercicio 3.

Se dispone de un vector con números enteros, de dimensión física $dimF$ y dimensión lógica $dimL$.

(a) Realizar un módulo que imprima el vector desde la primera posición hasta la última.

```
procedure imprimir_1adimL(vector_numeros: t_vector_numeros; dimL: int16);
var
  i: int16;
begin
  for i:= 1 to dimL do
    if (i<dimL) then
      write(vector_numeros[i],', ')
    else
      writeln(vector_numeros[i]);
  end;
```

(b) Realizar un módulo que imprima el vector desde la última posición hasta la primera.

```
procedure imprimir_dimLa1(vector_numeros: t_vector_numeros; dimL: int16);
var
  i: int16;
begin
  for i:= dimL downto 1 do
    if (i>1) then
      write(vector_numeros[i],', ')
    else
      writeln(vector_numeros[i]);
  end;
```

(c) Realizar un módulo que imprima el vector desde la mitad ($dimL \text{ DIV } 2$) hacia la primera posición y desde la mitad más uno hacia la última posición.

```
procedure imprimir_dimLdiv2(vector_numeros: t_vector_numeros; dimL: int16);
var
  i, dimLdiv2, dimLdiv2mas1: int16;
begin
  dimLdiv2:=dimL div 2; dimLdiv2mas1:=dimLdiv2+1;
  for i:= dimLdiv2 downto 1 do
    if (i>1) then
      write(vector_numeros[i],', ')
    else
      writeln(vector_numeros[i]);
  for i:= dimLdiv2mas1 to dimL do
    if (i<dimL) then
      write(vector_numeros[i],', ')
    else
      writeln(vector_numeros[i]);
  end;
```

(d) Realizar un módulo que reciba el vector, una posición X y otra posición Y , e imprima el vector desde la posición X hasta la Y . Asumir que tanto X como Y son menores o igual

a la dimensión lógica. Y considerar que, dependiendo de los valores de X e Y , podría ser necesario recorrer hacia adelante o hacia atrás.

```
procedure imprimir_general(vector_numeros: t_vector_numeros; numX, numY: int16);
var
  i: int16;
begin
  if (numX <= numY) then
    for i := numX to numY do
      if (i < numY) then
        write(vector_numeros[i], ', ')
      else
        writeln(vector_numeros[i])
    else
      for i := numX downto numY do
        if (i > numY) then
          write(vector_numeros[i], ', ')
        else
          writeln(vector_numeros[i]);
end;
```

(e) Utilizando el módulo implementado en el inciso anterior, volver a realizar los incisos a, b y c.

```
program TP4_E3;
{$codepage UTF8}
uses crt;
type
  t_vector_numeros = array of int16;
procedure crear_vector_numeros(var vector_numeros: t_vector_numeros; dimF: int16);
begin
  setLength(vector_numeros, dimF);
end;
procedure cargar_vector_numeros(var vector_numeros: t_vector_numeros; dimL: int16);
var
  i: int16;
begin
  for i := 1 to dimL do
    vector_numeros[i] := random(high(int16));
end;
procedure imprimir_1adimL(vector_numeros: t_vector_numeros; dimL: int16);
var
  i: int16;
begin
  for i := 1 to dimL do
    if (i < dimL) then
      write(vector_numeros[i], ', ')
    else
      writeln(vector_numeros[i]);
end;
procedure imprimir_dimLa1(vector_numeros: t_vector_numeros; dimL: int16);
var
  i: int16;
begin
  for i := dimL downto 1 do
    if (i > 1) then
      write(vector_numeros[i], ', ')
    else
      writeln(vector_numeros[i]);
end;
```

```
procedure imprimir_dimLdiv2(vector_numeros: t_vector_numeros; dimL: int16);
var
    i, dimLdiv2, dimLdiv2mas1: int16;
begin
    dimLdiv2:=dimL div 2; dimLdiv2mas1:=dimLdiv2+1;
    for i:= dimLdiv2 downto 1 do
        if (i>1) then
            write(vector_numeros[i],', ')
        else
            writeln(vector_numeros[i]);
    for i:= dimLdiv2mas1 to dimL do
        if (i<dimL) then
            write(vector_numeros[i],', ')
        else
            writeln(vector_numeros[i]);
    end;
procedure imprimir_general(vector_numeros: t_vector_numeros; numX, numY: int16);
var
    i: int16;
begin
    if (numX<=numY) then
        for i:= numX to numY do
            if (i<numY) then
                write(vector_numeros[i],', ')
            else
                writeln(vector_numeros[i])
        else
            for i:= numX downto numY do
                if (i>numY) then
                    write(vector_numeros[i],', ')
                else
                    writeln(vector_numeros[i]);
    end;
var
    vector_numeros: t_vector_numeros;
    dimF, dimL: int16;
begin
    randomize;
    textcolor(green); write('Introducir dimensión física del vector: ');
    textcolor(yellow); readln(dimF);
    textcolor(green); write('Introducir dimensión lógica del vector: ');
    textcolor(yellow); readln(dimL);
    crear_vector_numeros(vector_numeros,dimF);
    if (dimL>0) then
        begin
            cargar_vector_numeros(vector_numeros,dimL);
            imprimir_1adimL(vector_numeros,dimL);
            imprimir_dimLa1(vector_numeros,dimL);
            imprimir_dimLdiv2(vector_numeros,dimL);
            imprimir_general(vector_numeros,1,dimL);
            imprimir_general(vector_numeros,dimL,1);
            imprimir_general(vector_numeros,dimL div 2,1);
            imprimir_general(vector_numeros,dimL div 2+1,dimL);
        end;
    end.
```

Ejercicio 4.

Se dispone de un vector con 100 números enteros. Implementar los siguientes módulos:

(a) posicion: *dado un número X y el vector de números, retorna la posición del número X en dicho vector o el valor -1 en caso de no encontrarse.*

```
function posicion(vector_numeros: t_vector_numeros; numX: int16): int16;
var
  pos: int16;
begin
  pos:=1;
  while ((pos<=num_total) and (vector_numeros[pos]<>numX)) do
    pos:=pos+1;
  if (pos<=num_total) then
    posicion:=pos
  else
    posicion:=-1;
  end;
end;
```

(b) intercambio: *recibe dos valores x e y (entre 1 y 100) y el vector de números y retorna el mismo vector, donde se intercambiaron los valores de las posiciones x e y.*

```
procedure intercambio(var vector_numeros: t_vector_numeros; numX, numY: int16);
var
  num_aux: int16;
begin
  num_aux:=vector_numeros[numX];
  vector_numeros[numX]:=vector_numeros[numY];
  vector_numeros[numY]:=num_aux;
end;
```

(c) sumaVector: *retorna la suma de todos los elementos del vector.*

```
function sumaVector(vector_numeros: t_vector_numeros): int16;
var
  i: t_numero;
  suma: int16;
begin
  suma:=0;
  for i:= 1 to num_total do
    suma:=suma+vector_numeros[i];
  sumaVector:=suma;
end;
```

(d) promedio: *devuelve el valor promedio de los elementos del vector.*

```
function promedio(vector_numeros: t_vector_numeros): real;
begin
  promedio:=sumaVector(vector_numeros)/num_total;
end;
```


(e) *elementoMaximo*: retorna la posición del mayor elemento del vector.

```
function elementoMaximo(vector_numeros: t_vector_numeros): int16;
var
  i: t_numero;
  ele_max, pos_max: int16;
begin
  ele_max:=low(int16); pos_max:=0;
  for i:= 1 to num_total do
    if (vector_numeros[i]>ele_max) then
      begin
        ele_max:=vector_numeros[i];
        pos_max:=i;
      end;
  end;
  elementoMaximo:=pos_max;
end;
```

(f) *elementoMinimo*: retorna la posición del menor elemento del vector.

```
function elementoMinimo(vector_numeros: t_vector_numeros): int16;
var
  i: t_numero;
  ele_min, pos_min: int16;
begin
  ele_min:=high(int16); pos_min:=0;
  for i:= 1 to num_total do
    if (vector_numeros[i]<ele_min) then
      begin
        ele_min:=vector_numeros[i];
        pos_min:=i;
      end;
  end;
  elementoMinimo:=pos_min;
end;
```

```
program TP4_E4;
{$codepage UTF8}
uses crt;
const
  num_total=100;
type
  t_numero=1..num_total;
  t_vector_numeros=array[t_numero] of int16;
procedure cargar_vector_numeros(var vector_numeros: t_vector_numeros);
var
  i: t_numero;
begin
  for i:= 1 to num_total do
    vector_numeros[i]:=random(high(int16));
  end;
function posicion(vector_numeros: t_vector_numeros; numX: int16): int16;
var
  pos: int16;
begin
  pos:=1;
  while ((pos<=num_total) and (vector_numeros[pos]<>numX)) do
    pos:=pos+1;
  if (pos<=num_total) then
    posicion:=pos
```

```

    else
        posicion:=-1;
    end;
procedure intercambio(var vector_numeros: t_vector_numeros; numX, numY: int16);
var
    num_aux: int16;
begin
    num_aux:=vector_numeros[numX];
    vector_numeros[numX]:=vector_numeros[numY];
    vector_numeros[numY]:=num_aux;
end;
function sumaVector(vector_numeros: t_vector_numeros): int16;
var
    i: t_numero;
    suma: int16;
begin
    suma:=0;
    for i:= 1 to num_total do
        suma:=suma+vector_numeros[i];
    sumaVector:=suma;
end;
function promedio(vector_numeros: t_vector_numeros): real;
begin
    promedio:=sumaVector(vector_numeros)/num_total;
end;
function elementoMaximo(vector_numeros: t_vector_numeros): int16;
var
    i: t_numero;
    ele_max, pos_max: int16;
begin
    ele_max:=low(int16); pos_max:=0;
    for i:= 1 to num_total do
        if (vector_numeros[i]>ele_max) then
            begin
                ele_max:=vector_numeros[i];
                pos_max:=i;
            end;
        elementoMaximo:=pos_max;
    end;
function elementoMinimo(vector_numeros: t_vector_numeros): int16;
var
    i: t_numero;
    ele_min, pos_min: int16;
begin
    ele_min:=high(int16); pos_min:=0;
    for i:= 1 to num_total do
        if (vector_numeros[i]<ele_min) then
            begin
                ele_min:=vector_numeros[i];
                pos_min:=i;
            end;
        elementoMinimo:=pos_min;
    end;
var
    vector_numeros: t_vector_numeros;
    numX, numY: int16;
begin
    cargar_vector_numeros(vector_numeros);
    textcolor(green); write('Introducir número entero X para buscar su posición (si existe) en
el vector: ');
    textcolor(yellow); readln(numX);
    textcolor(green); write('La posición del número '); textcolor(red); write(numX);
textcolor(green); write(' en el vector es '); textcolor(red);
writeln(posicion(vector_numeros,numX));
    textcolor(green); write('Introducir número entero X (entre 1 y 100): ');
    textcolor(yellow); readln(numX);

```

```
    textcolor(green); write('Introducir número entero Y (entre 1 y 100): ');
    textcolor(yellow); readln(numY);
    textcolor(green); write('Pre-intercambio, en las posiciones '); textcolor(red); write(numX);
    textcolor(green); write(' y '); textcolor(red); write(numY); textcolor(green); write(', se
    tienen los valores '); textcolor(red); write(vector_numeros[numX]); textcolor(green); write('
    y '); textcolor(red); write(vector_numeros[numY]); textcolor(green); writeln(',
    respectivamente');
    intercambio(vector_numeros,numX,numY);
    textcolor(green); write('Post-intercambio, en las posiciones '); textcolor(red);
    write(numX); textcolor(green); write(' y '); textcolor(red); write(numY); textcolor(green);
    write(', se tienen los valores '); textcolor(red); write(vector_numeros[numX]);
    textcolor(green); write(' y '); textcolor(red); write(vector_numeros[numY]); textcolor(green);
    writeln(', respectivamente');
    textcolor(green); write('La suma de todos los elementos del vector es '); textcolor(red);
    writeln(sumaVector(vector_numeros));
    textcolor(green); write('El valor promedio de los elementos del vector es ');
    textcolor(red); writeln(promedio(vector_numeros):0:2);
    textcolor(green); write('La posición del mayor elemento del vector es '); textcolor(red);
    writeln(elementoMaximo(vector_numeros));
    textcolor(green); write('La posición del menor elemento del vector es '); textcolor(red);
    write(elementoMinimo(vector_numeros));
end.
```

Ejercicio 5.

Utilizando los módulos implementados en el Ejercicio 4, realizar un programa que lea números enteros desde teclado (a lo sumo, 100) y los almacene en un vector. La carga finaliza al leer el número 0. Al finalizar la carga, se debe intercambiar la posición del mayor elemento por la del menor elemento e informar la operación realizada de la siguiente manera: “El elemento máximo ... que se encontraba en la posición ... fue intercambiado con el elemento mínimo ... que se encontraba en la posición ...”.

```

program TP4_E5;
{$codepage UTF8}
uses crt;
const
    num_total=100;
    num_salida=0;
type
    t_numero=1..num_total;
    t_vector_numeros=array[t_numero] of int16;
procedure cargar_vector_numeros(var vector_numeros: t_vector_numeros; var numeros: int16);
var
    num: int16;
begin
    textcolor(green); write('Introducir número entero para la posición ',numeros+1,' del vector: ');
    textcolor(yellow); readln(num);
    while ((num<>num_salida) and (numeros<num_total)) do
    begin
        numeros:=numeros+1;
        vector_numeros[numeros]:=num;
        textcolor(green); write('Introducir número entero para la posición ',numeros+1,' del vector: ');
        textcolor(yellow); readln(num);
    end;
end;
procedure elementoMaximo(vector_numeros: t_vector_numeros; numeros: int16; var ele_max, pos_max: int16);
var
    i: t_numero;
begin
    for i:= 1 to numeros do
        if (vector_numeros[i]>ele_max) then
        begin
            ele_max:=vector_numeros[i];
            pos_max:=i;
        end;
    end;
end;
procedure elementoMinimo(vector_numeros: t_vector_numeros; numeros: int16; var ele_min, pos_min: int16);
var
    i: t_numero;
begin
    for i:= 1 to numeros do
        if (vector_numeros[i]<ele_min) then
        begin
            ele_min:=vector_numeros[i];
            pos_min:=i;
        end;
    end;
end;
procedure intercambio(var vector_numeros: t_vector_numeros; pos_max, pos_min: int16);
var
    num_aux: int16;
begin

```

```
    num_aux:=vector_numeros[pos_max];
    vector_numeros[pos_max]:=vector_numeros[pos_min];
    vector_numeros[pos_min]:=num_aux;
end;
var
    vector_numeros: t_vector_numeros;
    numeros, ele_max, pos_max, ele_min, pos_min: int16;
begin
    numeros:=0;
    ele_max:=low(int16); pos_max:=0;
    ele_min:=high(int16); pos_min:=0;
    cargar_vector_numeros(vector_numeros,numeros);
    if (numeros>0) then
    begin
        elementoMaximo(vector_numeros,numeros,ele_max,pos_max);
        elementoMinimo(vector_numeros,numeros,ele_min,pos_min);
        intercambio(vector_numeros,pos_max,pos_min);
        textcolor(green); write('El elemento máximo '); textcolor(red); write(ele_max);
    textcolor(green); write(', que se encontraba en la posición '); textcolor(red);
    write(pos_max); textcolor(green); write(', fue intercambiado con el elemento mínimo ');
    textcolor(red); write(ele_min); textcolor(green); write(', que se encontraba en la posición
    '); textcolor(red); write(pos_min);
    end;
end.
```

Ejercicio 6.

Dado que en la solución anterior se recorre dos veces el vector (una para calcular el elemento máximo y otra para el mínimo), implementar un único módulo que recorra una única vez el vector y devuelva ambas posiciones.

```

program TP4_E6;
{$codepage UTF8}
uses crt;
const
  num_total=100;
  num_salida=0;
type
  t_numero=1..num_total;
  t_vector_numeros=array[t_numero] of int16;
procedure cargar_vector_numeros(var vector_numeros: t_vector_numeros; var numeros: int16);
var
  num: int16;
begin
  textcolor(green); write('Introducir número entero para la posición ',numeros+1,' del vector: ');
  textcolor(yellow); readln(num);
  while ((num<>num_salida) and (numeros<num_total)) do
  begin
    numeros:=numeros+1;
    vector_numeros[numeros]:=num;
    textcolor(green); write('Introducir número entero para la posición ',numeros+1,' del vector: ');
    textcolor(yellow); readln(num);
  end;
end;
procedure elementosMaximoYMinimo(vector_numeros: t_vector_numeros; numeros: int16; var
ele_max, pos_max, ele_min, pos_min: int16);
var
  i: t_numero;
begin
  for i:= 1 to numeros do
  begin
    if (vector_numeros[i]>ele_max) then
    begin
      ele_max:=vector_numeros[i];
      pos_max:=i;
    end;
    if (vector_numeros[i]<ele_min) then
    begin
      ele_min:=vector_numeros[i];
      pos_min:=i;
    end;
  end;
end;
procedure intercambio(var vector_numeros: t_vector_numeros; pos_max, pos_min: int16);
var
  num_aux: int16;
begin
  num_aux:=vector_numeros[pos_max];
  vector_numeros[pos_max]:=vector_numeros[pos_min];
  vector_numeros[pos_min]:=num_aux;
end;
var
  vector_numeros: t_vector_numeros;
  numeros, ele_max, pos_max, ele_min, pos_min: int16;
begin
  numeros:=0;

```

```
ele_max:=low(int16); pos_max:=0;
ele_min:=high(int16); pos_min:=0;
cargar_vector_numeros(vector_numeros,numeros);
if (numeros>0) then
begin
  elementosMaximoYMinimo(vector_numeros,numeros,ele_max,pos_max,ele_min,pos_min);
  intercambio(vector_numeros,pos_max,pos_min);
  textcolor(green); write('El elemento máximo '); textcolor(red); write(ele_max);
textcolor(green); write(', que se encontraba en la posición '); textcolor(red);
write(pos_max); textcolor(green); write(', fue intercambiado con el elemento mínimo ');
textcolor(red); write(ele_min); textcolor(green); write(', que se encontraba en la posición
'); textcolor(red); write(pos_min);
  end;
end.
```

Ejercicio 7.

Realizar un programa que lea números enteros desde teclado hasta que se ingrese el valor -1 (que no debe procesarse) e informe:

- la cantidad de ocurrencias de cada dígito procesado.
- el dígito más leído.
- los dígitos que no tuvieron ocurrencias.

Por ejemplo, si la secuencia que se lee es: 63 34 99 94 96 -1, el programa deberá informar:

- Número 3: 2 veces
- Número 4: 2 veces
- Número 6: 2 veces
- Número 9: 4 veces
- El dígito más leído fue el 9
- Los dígitos que no tuvieron ocurrencias son: 0, 1, 2, 5, 7, 8

```

program TP4_E7;
{$codepage UTF8}
uses crt;
const
  digitos_total=9;
  num_salida=-1;
  num_corte=0;
type
  t_digito=0..digitos_total;
  t_vector_digitos=array[t_digito] of int16;
procedure inicializar_vector_digitos(var vector_digitos: t_vector_digitos);
var
  i: t_digito;
begin
  for i:= 1 to digitos_total do
    vector_digitos[i]:=0;
  end;
procedure descomponer_numero(var vector_digitos: t_vector_digitos; num: int16);
var
  digito: t_digito;
begin
  if (num=0) then
    vector_digitos[0]:=vector_digitos[0]+1
  else
    while (num<>0) do
      begin
        digito:=num mod 10;
        vector_digitos[digito]:=vector_digitos[digito]+1;
        num:=num div 10;
      end;
  end;
procedure cargar_vector_digitos(var vector_digitos: t_vector_digitos);
var
  num: int16;
begin
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(num);
  while (num<>num_salida) do
    begin
      descomponer_numero(vector_digitos,num);
      textcolor(green); write('Introducir número entero: ');
      textcolor(yellow); readln(num);
    end;
end;

```



```
end;
procedure digito_mas_Leido(num: int16; digito: int8; var num_max: int16; var digito_max:
int8);
begin
    if (num>num_max) then
        begin
            num_max:=num;
            digito_max:=digito;
        end;
    end;
end;
procedure digitos_sin_ocurrencias(num: int16; digito: int8; var vector_digitos2:
t_vector_digitos; var dimL: int8);
begin
    if (num=num_corte) then
        begin
            dimL:=dimL+1;
            vector_digitos2[dimL]:=digito;
        end;
    end;
end;
procedure procesar_vector_digitos(vector_digitos: t_vector_digitos);
var
    i: t_digito;
    digito_max, dimL: int8;
    num_max: int16;
    vector_digitos2: t_vector_digitos;
begin
    num_max:=low(int16); digito_max:=-1;
    dimL:=0;
    for i:= 0 to digitos_total do
        begin
            textcolor(green); write('Número ',i,': '); textcolor(red); write(vector_digitos[i]);
textcolor(green); writeln(' veces');
            digito_mas_Leido(vector_digitos[i],i,num_max,digito_max);
            digitos_sin_ocurrencias(vector_digitos[i],i,vector_digitos2,dimL);
        end;
        textcolor(green); write('El dígito más leído fue el '); textcolor(red); writeln(digito_max);
        if (dimL>0) then
            begin
                textcolor(green); write('Los dígitos que no tuvieron ocurrencias son: ');
                for i:= 1 to dimL do
                    begin
                        if (i<dimL) then
                            begin
                                textcolor(red); write(vector_digitos2[i]); textcolor(green); write(', ');
                            end
                        else
                            begin
                                textcolor(red); write(vector_digitos2[i]);
                            end;
                        end;
                    end;
                end;
            end
        else
            begin
                textcolor(red); write('No hay dígitos sin ocurrencias');
            end;
        end;
    end;
var
    vector_digitos: t_vector_digitos;
begin
    inicializar_vector_digitos(vector_digitos);
    cargar_vector_digitos(vector_digitos);
    procesar_vector_digitos(vector_digitos);
end.
```

Ejercicio 8.

Realizar un programa que lea y almacene la información de 400 alumnos ingresantes a la Facultad de Informática de la UNLP en el año 2020. De cada alumno, se lee: número de inscripción, DNI, apellido, nombre y año de nacimiento. Una vez leída y almacenada toda la información, calcular e informar:

- El porcentaje de alumnos con DNI compuesto sólo por dígitos pares.
- Apellido y nombre de los dos alumnos de mayor edad.

```

program TP4_E8;
{$codepage UTF8}
uses crt;
const
    alumnos_total=400;
type
    t_alumno=1..alumnos_total;
    t_registro_alumno=record
        numero: int16;
        dni: int32;
        apellido: string;
        nombre: string;
        nacimiento: int16;
    end;
    t_vector_alumnos=array[t_alumno] of t_registro_alumno;
procedure leer_alumno(var registro_alumno: t_registro_alumno);
begin
    textcolor(green); write('Introducir número de inscripción del alumno: ');
    textcolor(yellow); readln(registro_alumno.numero);
    textcolor(green); write('Introducir DNI del alumno: ');
    textcolor(yellow); readln(registro_alumno.dni);
    textcolor(green); write('Introducir apellido del alumno: ');
    textcolor(yellow); readln(registro_alumno.apellido);
    textcolor(green); write('Introducir nombre del alumno: ');
    textcolor(yellow); readln(registro_alumno.nombre);
    textcolor(green); write('Introducir año de nacimiento del alumno: ');
    textcolor(yellow); readln(registro_alumno.nacimiento);
end;
procedure cargar_vector_alumnos(var vector_alumnos: t_vector_alumnos);
var
    registro_alumno: t_registro_alumno;
    i: t_alumno;
begin
    for i:= 1 to alumnos_total do
        begin
            leer_alumno(registro_alumno);
            vector_alumnos[i]:=registro_alumno;
        end;
    end;
function hay_impar(dni: int32): boolean;
var
    digito: int8;
    impar: boolean;
begin
    impar:=false;
    while ((dni<>0) and (impar<>true)) do
        begin
            digito:=dni mod 10;
            impar:=(digito mod 2<>0);
            dni:=dni div 10;
        end;
    hay_impar:=impar;
end;

```

```

procedure actualizar_minimos(nacimiento: int16; apellido, nombre: string; var
nacimiento_min1, nacimiento_min2: int16; var apellido_min1, nombre_min1, apellido_min2,
nombre_min2: string);
begin
    if (nacimiento < nacimiento_min1) then
        begin
            nacimiento_min2 := nacimiento_min1;
            apellido_min2 := apellido_min1;
            nombre_min2 := nombre_min1;
            nacimiento_min1 := nacimiento;
            apellido_min1 := apellido;
            nombre_min1 := nombre;
        end
    else
        if (nacimiento < nacimiento_min2) then
            begin
                nacimiento_min2 := nacimiento;
                apellido_min2 := apellido;
                nombre_min2 := nombre;
            end;
    end;
end;

procedure procesar_vector_alumnos(vector_alumnos: t_vector_alumnos; var porcentaje_pares:
real; var apellido_min1, nombre_min1, apellido_min2, nombre_min2: string);
var
    i: t_alumno;
    alumnos_pares, nacimiento_min1, nacimiento_min2: int16;
begin
    alumnos_pares := 0;
    nacimiento_min1 := high(int16); nacimiento_min2 := high(int16);
    for i := 1 to alumnos_total do
        begin
            if (hay_impar(vector_alumnos[i].dni) = false) then
                alumnos_pares := alumnos_pares + 1;
                actualizar_minimos(vector_alumnos[i].nacimiento, vector_alumnos[i].apellido, vector_alumnos[
i].nombre, nacimiento_min1, nacimiento_min2, apellido_min1, nombre_min1, apellido_min2, nombre_min2)
            ;
        end;
    porcentaje_pares := alumnos_pares / alumnos_total * 100;
end;

var
    vector_alumnos: t_vector_alumnos;
    porcentaje_pares: real;
    apellido_min1, nombre_min1, apellido_min2, nombre_min2: string;
begin
    porcentaje_pares := 0;
    apellido_min1 := ''; nombre_min1 := ''; apellido_min2 := ''; nombre_min2 := '';
    cargar_vector_alumnos(vector_alumnos);
    procesar_vector_alumnos(vector_alumnos, porcentaje_pares, apellido_min1, nombre_min1, apellido_m
in2, nombre_min2);
    textcolor(green); write('El porcentaje de alumnos con DNI compuesto sólo por dígitos pares
es '); textcolor(red); write(porcentaje_pares:0:2); textcolor(green); writeln('%');
    textcolor(green); write('El apellido y nombre de los dos alumnos de mayor edad son ');
    textcolor(red); write(apellido_min1, ' ', nombre_min1); textcolor(green); write(' y ');
    textcolor(red); write(apellido_min2, ' ', nombre_min2);
end.

```

Ejercicio 9.

Modificar la solución del punto anterior considerando que el programa lea y almacene la información de, a lo sumo, 400 alumnos. La lectura finaliza cuando se ingresa el DNI -1 (que no debe procesarse).

```

program TP4_E9;
{$codepage UTF8}
uses crt;
const
  alumnos_total=400;
  dni_salida=-1;
type
  t_alumno=1..alumnos_total;
  t_registro_alumno=record
    numero: int16;
    dni: int32;
    apellido: string;
    nombre: string;
    nacimiento: int16;
  end;
  t_vector_alumnos=array[t_alumno] of t_registro_alumno;
procedure leer_alumno(var registro_alumno: t_registro_alumno);
begin
  textcolor(green); write('Introducir DNI del alumno: ');
  textcolor(yellow); readln(registro_alumno.dni);
  if (registro_alumno.dni<>dni_salida) then
  begin
    textcolor(green); write('Introducir número de inscripción del alumno: ');
    textcolor(yellow); readln(registro_alumno.numero);
    textcolor(green); write('Introducir apellido del alumno: ');
    textcolor(yellow); readln(registro_alumno.apellido);
    textcolor(green); write('Introducir nombre del alumno: ');
    textcolor(yellow); readln(registro_alumno.nombre);
    textcolor(green); write('Introducir año de nacimiento del alumno: ');
    textcolor(yellow); readln(registro_alumno.nacimiento);
  end;
end;
procedure cargar_vector_alumnos(var vector_alumnos: t_vector_alumnos; var alumnos: int16);
var
  registro_alumno: t_registro_alumno;
begin
  leer_alumno(registro_alumno);
  while ((registro_alumno.dni<>dni_salida) and (alumnos<alumnos_total)) do
  begin
    alumnos:=alumnos+1;
    vector_alumnos[alumnos]:=registro_alumno;
    leer_alumno(registro_alumno);
  end;
end;
function hay_impar(dni: int32): boolean;
var
  digito: int8;
  impar: boolean;
begin
  impar:=false;
  while ((dni<>0) and (impar<>true)) do
  begin
    digito:=dni mod 10;
    impar:=(digito mod 2<>0);
    dni:=dni div 10;
  end;
  hay_impar:=impar;

```

```

end;
procedure actualizar_minimos(nacimiento: int16; apellido, nombre: string; var
nacimiento_min1, nacimiento_min2: int16; var apellido_min1, nombre_min1, apellido_min2,
nombre_min2: string);
begin
    if (nacimiento < nacimiento_min1) then
        begin
            nacimiento_min2 := nacimiento_min1;
            apellido_min2 := apellido_min1;
            nombre_min2 := nombre_min1;
            nacimiento_min1 := nacimiento;
            apellido_min1 := apellido;
            nombre_min1 := nombre;
        end
    else
        if (nacimiento < nacimiento_min2) then
            begin
                nacimiento_min2 := nacimiento;
                apellido_min2 := apellido;
                nombre_min2 := nombre;
            end;
    end;
end;
procedure procesar_vector_alumnos(vector_alumnos: t_vector_alumnos; alumnos: int16; var
porcentaje_pares: real; var apellido_min1, nombre_min1, apellido_min2, nombre_min2: string);
var
    i: t_alumno;
    alumnos_pares, nacimiento_min1, nacimiento_min2: int16;
begin
    alumnos_pares := 0;
    nacimiento_min1 := high(int16); nacimiento_min2 := high(int16);
    for i := 1 to alumnos do
        begin
            if (hay_impar(vector_alumnos[i].dni) = false) then
                alumnos_pares := alumnos_pares + 1;
                actualizar_minimos(vector_alumnos[i].nacimiento, vector_alumnos[i].apellido, vector_alumnos[
i].nombre, nacimiento_min1, nacimiento_min2, apellido_min1, nombre_min1, apellido_min2, nombre_min2)
            ;
        end;
    porcentaje_pares := alumnos_pares / alumnos * 100;
end;
var
    vector_alumnos: t_vector_alumnos;
    alumnos: int16;
    porcentaje_pares: real;
    apellido_min1, nombre_min1, apellido_min2, nombre_min2: string;
begin
    alumnos := 0;
    porcentaje_pares := 0;
    apellido_min1 := ''; nombre_min1 := ''; apellido_min2 := ''; nombre_min2 := '';
    cargar_vector_alumnos(vector_alumnos, alumnos);
    if (alumnos > 0) then
        begin
            procesar_vector_alumnos(vector_alumnos, alumnos, porcentaje_pares, apellido_min1, nombre_min1,
apellido_min2, nombre_min2);
            textcolor(green); write('El porcentaje de alumnos con DNI compuesto sólo por dígitos pares
es '); textcolor(red); write(porcentaje_pares:0:2); textcolor(green); writeln('%');
            textcolor(green); write('El apellido y nombre de los dos alumnos de mayor edad son ');
textcolor(red); write(apellido_min1, ' ', nombre_min1); textcolor(green); write(' y ');
textcolor(red); write(apellido_min2, ' ', nombre_min2);
        end;
    end.

```

Ejercicio 10.

Realizar un programa que lea y almacene el salario de los empleados de una empresa de turismo (a lo sumo, 300 empleados). La carga finaliza cuando se lea el salario 0 (que no debe procesarse) o cuando se completa el vector. Una vez finalizada la carga de datos, se pide:

- Realizar un módulo que incremente el salario de cada empleado en un 15%. Para ello, implementar un módulo que reciba como parámetro un valor real X , el vector de valores reales y su dimensión lógica y retorne el mismo vector en el cual cada elemento fue multiplicado por el valor X .
- Realizar un módulo que muestre en pantalla el sueldo promedio de los empleados de la empresa.

```
program TP4_E10;
{$codepage UTF8}
uses crt;
const
    empleados_total=300;
    salario_salida=0;
    incremento=1.15;
type
    t_empleado=1..empleados_total;
    t_vector_salarios=array[t_empleado] of real;
procedure inicializar_vector_salarios(var vector_salarios: t_vector_salarios);
var
    i: t_empleado;
begin
    for i:= 1 to empleados_total do
        vector_salarios[i]:=0;
    end;
procedure cargar_vector_salarios(var vector_salarios: t_vector_salarios; var empleados:
int16);
var
    salario: real;
begin
    textcolor(green); write('Introducir salario del empleado ',empleados+1,': $');
    textcolor(yellow); readln(salario);
    while ((salario<>salario_salida) and (empleados<empleados_total)) do
        begin
            empleados:=empleados+1;
            vector_salarios[empleados]:=salario;
            textcolor(green); write('Introducir salario del empleado ',empleados+1,': $');
            textcolor(yellow); readln(salario);
        end;
    end;
procedure incrementar_salarios(incremento: real; var vector_salarios: t_vector_salarios;
empleados: int16);
var
    i: t_empleado;
begin
    for i:= 1 to empleados do
        vector_salarios[i]:=vector_salarios[i]*incremento;
    end;
procedure calcular_salario_promedio(vector_salarios: t_vector_salarios; empleados: int16);
var
    i: t_empleado;
    salario_total, salario_prom: real;
begin
    salario_total:=0;
    for i:= 1 to empleados do
```

```
    salario_total:=salario_total+vector_salarios[i];
    salario_prom:=salario_total/empleados;
    textcolor(green); write('El sueldo promedio de los empleados de la empresa es $');
    textcolor(red); write(salario_prom:0:2);
end;
var
    vector_salarios: t_vector_salarios;
    empleados: int16;
begin
    empleados:=0;
    inicializar_vector_salarios(vector_salarios);
    cargar_vector_salarios(vector_salarios,empleados);
    if (empleados>0) then
        begin
            incrementar_salarios(incremento,vector_salarios,empleados);
            calcular_salario_promedio(vector_salarios,empleados);
        end;
    end;
end.
```

Ejercicio 11.

El colectivo de fotógrafos ArgenPics desea conocer los gustos de sus seguidores en las redes sociales. Para ello, para cada una de las 200 fotos publicadas en su página de Facebook, cuenta con la siguiente información: título de la foto, el autor de la foto, cantidad de Me gusta, cantidad de clics y cantidad de comentarios de usuarios. Realizar un programa que lea y almacene esta información. Una vez finalizada la lectura, el programa debe procesar los datos e informar:

- Título de la foto más vista (la que posee mayor cantidad de clics).
- Cantidad total de Me gusta recibidas a las fotos cuyo autor es el fotógrafo “Art Vandelay”.
- Cantidad de comentarios recibidos para cada una de las fotos publicadas en esa página.

```
program TP4_E11;
{$codepage UTF8}
uses crt;
const
  fotos_total=200;
  autor_corte='Art Vandelay';
type
  t_foto=1..fotos_total;
  t_registro_foto=record
    titulo: string;
    autor: string;
    megusta: int16;
    clics: int16;
    comentarios: int16;
  end;
  t_vector_fotos=array[t_foto] of t_registro_foto;
procedure leer_foto(var registro_foto: t_registro_foto);
begin
  textcolor(green); write('Introducir título de la foto: ');
  textcolor(yellow); readln(registro_foto.titulo);
  textcolor(green); write('Introducir autor de la foto: ');
  textcolor(yellow); readln(registro_foto.autor);
  textcolor(green); write('Introducir cantidad de Me gusta de la foto: ');
  textcolor(yellow); readln(registro_foto.megusta);
  textcolor(green); write('Introducir cantidad de clics de la foto: ');
  textcolor(yellow); readln(registro_foto.clics);
  textcolor(green); write('Introducir cantidad de comentarios de usuarios en la foto: ');
  textcolor(yellow); readln(registro_foto.comentarios);
end;
procedure cargar_vector_fotos(var vector_fotos: t_vector_fotos);
var
  registro_foto: t_registro_foto;
  i: t_foto;
begin
  for i:= 1 to fotos_total do
    begin
      leer_foto(registro_foto);
      vector_fotos[i]:=registro_foto;
    end;
  end;
procedure actualizar_maximo(clics: int16; titulo: string; var clics_max: int16; var
titulo_max: string);
begin
  if (clics>clics_max) then
    begin
      clics_max:=clics;
```



```
        titulo_max:=titulo;
    end;
end;
procedure procesar_vector_fotos(vector_fotos: t_vector_fotos; var titulo_max: string; var
megusta_corte: int16);
var
    i: t_foto;
    clics_max: int16;
begin
    clics_max:=low(int16);
    for i:= 1 to fotos_total do
        begin
            actualizar_maximo(vector_fotos[i].clics,vector_fotos[i].titulo,clics_max,titulo_max);
            if (vector_fotos[i].autor=autor_corte) then
                megusta_corte:=megusta_corte+1;
                textcolor(green); write('La cantidad de comentarios recibidos de la foto ');
textcolor(red); write(vector_fotos[i].titulo); textcolor(green); write(' es ');
textcolor(red); writeln(vector_fotos[i].comentarios);
            end;
        end;
    end;
var
    vector_fotos: t_vector_fotos;
    megusta_corte: int16;
    titulo_max: string;
begin
    titulo_max:='';
    megusta_corte:=0;
    cargar_vector_fotos(vector_fotos);
    procesar_vector_fotos(vector_fotos,titulo_max,megusta_corte);
    textcolor(green); write('El título de la foto más vista (la que posee mayor cantidad de
clics) es '); textcolor(red); writeln(titulo_max);
    textcolor(green); write('La cantidad total de Me gusta recibidas a las fotos cuyo autor es
el fotógrafo '); textcolor(yellow); write(autor_corte); textcolor(green); write('" es ');
textcolor(red); write(megusta_corte);
end.
```

Ejercicio 12.

En astrofísica, una galaxia se identifica por su nombre, su tipo (1. elíptica; 2. espiral; 3. lenticular; 4. irregular), su masa (medida en kg) y la distancia en pársecs (pc) medida desde la Tierra. La Unión Astronómica Internacional cuenta con datos correspondientes a las 53 galaxias que componen el Grupo Local. Realizar un programa que lea y almacene estos datos y, una vez finalizada la carga, informe:

- La cantidad de galaxias de cada tipo.
- La masa total acumulada de las 3 galaxias principales (la Vía Láctea, Andrómeda y Triángulo) y el porcentaje que esto representa respecto a la masa de todas las galaxias.
- La cantidad de galaxias no irregulares que se encuentran a menos de 1000 pc.
- El nombre de las dos galaxias con mayor masa y el de las dos galaxias con menor masa.

```

program TP4_E12;
{$codepage UTF8}
uses crt;
const
    galaxias_total=53;
    tipo_ini=1; tipo_fin=4;
    galaxia_corte1='la Via Lactea'; galaxia_corte2='Andromeda'; galaxia_corte3='Triangulo';
    distancia_corte=1000.0;
type
    t_galaxia=1..galaxias_total;
    t_tipo=tipo_ini..tipo_fin;
    t_registro_galaxia=record
        nombre: string;
        tipo: t_tipo;
        masa: real;
        distancia: real;
    end;
    t_vector_galaxias=array[t_galaxia] of t_registro_galaxia;
    t_vector_tipos=array[t_tipo] of int16;
procedure inicializar_vector_tipos(var vector_tipos: t_vector_tipos);
var
    i: t_tipo;
begin
    for i:= tipo_ini to tipo_fin do
        vector_tipos[i]:=0;
    end;
procedure leer_galaxia(var registro_galaxia: t_registro_galaxia);
begin
    textcolor(green); write('Introducir nombre de la galaxia: ');
    textcolor(yellow); readln(registro_galaxia.nombre);
    textcolor(green); write('Introducir tipo (1. elíptica; 2. espiral; 3. lenticular; 4.
irregular) de la galaxia: ');
    textcolor(yellow); readln(registro_galaxia.tipo);
    textcolor(green); write('Introducir masa (medida en kg) de la galaxia: ');
    textcolor(yellow); readln(registro_galaxia.masa);
    textcolor(green); write('Introducir distancia en pársecs (pc) medida desde la Tierra de la
galaxia: ');
    textcolor(yellow); readln(registro_galaxia.distancia);
end;
procedure cargar_vector_galaxias(var vector_galaxias: t_vector_galaxias);
var
    registro_galaxia: t_registro_galaxia;
    i: t_galaxia;
begin
    for i:= 1 to galaxias_total do

```

```

begin
    leer_galaxia(registro_galaxia);
    vector_galaxias[i]:=registro_galaxia;
end;
end;
procedure actualizar_maximos(masa: real; nombre: string; var masa_max1, masa_max2: real; var
nombre_max1, nombre_max2: string);
begin
    if (masa>masa_max1) then
    begin
        masa_max2:=masa_max1;
        nombre_max2:=nombre_max1;
        masa_max1:=masa;
        nombre_max1:=nombre;
    end
    else
        if (masa>masa_max2) then
        begin
            masa_max2:=masa;
            nombre_max2:=nombre;
        end;
    end;
end;
procedure actualizar_minimos(masa: real; nombre: string; var masa_min1, masa_min2: real; var
nombre_min1, nombre_min2: string);
begin
    if (masa<masa_min1) then
    begin
        masa_min2:=masa_min1;
        nombre_min2:=nombre_min1;
        masa_min1:=masa;
        nombre_min1:=nombre;
    end
    else
        if (masa<masa_min2) then
        begin
            masa_min2:=masa;
            nombre_min2:=nombre;
        end;
    end;
end;
procedure procesar_vector_galaxias(vector_galaxias: t_vector_galaxias; var vector_tipos:
t_vector_tipos; var masa_principales, porcentaje_principales: real; var
galaxias_no_irregulares: int8; var nombre_max1, nombre_max2, nombre_min1, nombre_min2:
string);
var
    i: t_galaxia;
    masa_total, masa_max1, masa_max2, masa_min1, masa_min2: real;
begin
    masa_total:=0;
    masa_max1:=-9999999; masa_max2:=-9999999;
    masa_min1:=9999999; masa_min2:=9999999;
    for i:= 1 to galaxias_total do
    begin
        vector_tipos[vector_galaxias[i].tipo]:=vector_tipos[vector_galaxias[i].tipo]+1;
        masa_total:=masa_total+vector_galaxias[i].masa;
        if ((vector_galaxias[i].nombre=galaxia_corte1) or
(vector_galaxias[i].nombre=galaxia_corte2) or (vector_galaxias[i].nombre=galaxia_corte3)) then
            masa_principales:=masa_principales+vector_galaxias[i].masa;
        if ((vector_galaxias[i].tipo<>4) and (vector_galaxias[i].distancia<distancia_corte)) then
            galaxias_no_irregulares:=galaxias_no_irregulares+1;
        actualizar_maximos(vector_galaxias[i].masa,vector_galaxias[i].nombre,masa_max1,masa_max2,n
ombre_max1,nombre_max2);
        actualizar_minimos(vector_galaxias[i].masa,vector_galaxias[i].nombre,masa_min1,masa_min2,n
ombre_min1,nombre_min2);
    end;
    porcentaje_principales:=masa_principales/masa_total*100;
end;

```

```

var
  vector_galaxias: t_vector_galaxias;
  vector_tipos: t_vector_tipos;
  galaxias_no_irregulares: int8;
  masa_principales, porcentaje_principales: real;
  nombre_max1, nombre_max2, nombre_min1, nombre_min2: string;
begin
  inicializar_vector_tipos(vector_tipos);
  masa_principales:=0; porcentaje_principales:=0;
  galaxias_no_irregulares:=0;
  nombre_max1:= ''; nombre_max2:= '';
  nombre_min1:= ''; nombre_min2:= '';
  cargar_vector_galaxias(vector_galaxias);
  procesar_vector_galaxias(vector_galaxias,vector_tipos,masa_principales,porcentaje_principale
s,galaxias_no_irregulares,nombre_max1,nombre_max2,nombre_min1,nombre_min2);
  textcolor(green); write('La cantidad de galaxias de cada tipo (1. elíptica; 2. espiral; 3.
lenticular; 4. irregular) es '); textcolor(red); write(vector_tipos[1],', ',vector_tipos[2],',
',vector_tipos[3],', ',vector_tipos[4]); textcolor(green); writeln(', respectivamente');
  textcolor(green); write('La masa total acumulada de las 3 galaxias principales (');
textcolor(yellow); write(galaxia_corte1,', ',galaxia_corte2,', ',galaxia_corte3);
textcolor(green); write(') y el porcentaje que esto representa respecto a la masa de todas las
galaxias es '); textcolor(red); write(masa_principales:0:2); textcolor(green); write(' y ');
textcolor(red); write(porcentaje_principales:0:2); textcolor(green); writeln('%
respectivamente');
  textcolor(green); write('La cantidad de galaxias no irregulares que se encuentran a menos de
'); textcolor(yellow); write(distancia_corte:0:2); textcolor(green); write(' pc es ');
textcolor(red); writeln(galaxias_no_irregulares);
  textcolor(green); write('Los nombres de las dos galaxias con mayor masa son ');
textcolor(red); write(nombre_max1); textcolor(green); write(' y '); textcolor(red);
writeln(nombre_max2);
  textcolor(green); write('Los nombres de las dos galaxias con menor masa son ');
textcolor(red); write(nombre_min1); textcolor(green); write(' y '); textcolor(red);
write(nombre_min2);
end.

```

Ejercicio 13.

El Grupo Intergubernamental de Expertos sobre el Cambio Climático de la ONU (IPCC) realiza todos los años mediciones de temperatura en 100 puntos diferentes del planeta y, para cada uno de estos lugares, obtiene un promedio anual. Este relevamiento se viene realizando desde hace 50 años y, con todos los datos recolectados, el IPCC se encuentra en condiciones de realizar análisis estadísticos. Realizar un programa que lea y almacene los datos correspondientes a las mediciones de los últimos 50 años (la información se ingresa ordenada por año). Una vez finalizada la carga de la información, obtener:

- *El año con mayor temperatura promedio a nivel mundial.*
- *El año con la mayor temperatura detectada en algún punto del planeta en los últimos 50 años.*

```

program TP4_E13;
{$codepage UTF8}
uses crt;
const
  puntos_total=100;
  anio_ini=1974; anio_fin=2023;
type
  t_punto=1..puntos_total;
  t_anio=anio_ini..anio_fin;
  t_vector_puntos=array[t_punto] of real;
  t_vector_anios=array[t_anio] of t_vector_puntos;
procedure inicializar_vector_anios(var vector_anios: t_vector_anios);
var
  i: t_anio;
  j: t_punto;
begin
  for i:= anio_ini to anio_fin do
    for j:= 1 to puntos_total do
      vector_anios[i][j]:=0;
    end;
  end;
procedure cargar_vector_anios(var vector_anios: t_vector_anios);
var
  i: t_anio;
  j: t_punto;
  temperatura: real;
begin
  for i:= anio_ini to anio_fin do
    for j:= 1 to puntos_total do
      begin
        textcolor(green); write('Introducir temperatura del año ',i,' en el punto ',j,': ');
        textcolor(yellow); readln(temperatura);
        vector_anios[i][j]:=temperatura;
      end;
    end;
  end;
procedure actualizar_maximo1(promedio_anio: real; anio: int16; var promedio_max: real; var
anio_max1: int16);
begin
  if (promedio_anio>promedio_max) then
    begin
      promedio_max:=promedio_anio;
      anio_max1:=anio;
    end;
  end;
procedure actualizar_maximo2(temperatura: real; anio: int16; var temperatura_max: real; var
anio_max2: int16);
begin

```

```
if (temperatura>temperatura_max) then
begin
    temperatura_max:=temperatura;
    anio_max2:=anio;
end;
end;
procedure procesar_vector_anios(vector_anios: t_vector_anios; var anio_max1, anio_max2:
int16);
var
    i: t_anio;
    j: t_punto;
    temperatura_anio, promedio_anio, promedio_max, temperatura_max: real;
begin
    promedio_max:=-9999999;
    temperatura_max:=-9999999;
    for i:= anio_ini to anio_fin do
    begin
        temperatura_anio:=0;
        for j:= 1 to puntos_total do
        begin
            temperatura_anio:=temperatura_anio+vector_anios[i][j];
            actualizar_maximo2(vector_anios[i][j],i,temperatura_max,anio_max2);
        end;
        promedio_anio:=temperatura_anio/puntos_total;
        actualizar_maximo1(promedio_anio,i,promedio_max,anio_max1);
    end;
end;
var
    vector_anios: t_vector_anios;
    anio_max1, anio_max2: int16;
begin
    anio_max1:=0; anio_max2:=0;
    inicializar_vector_anios(vector_anios);
    cargar_vector_anios(vector_anios);
    procesar_vector_anios(vector_anios,anio_max1,anio_max2);
    textcolor(green); write('El año con mayor temperatura promedio a nivel mundial es ');
    textcolor(red); writeln(anio_max1);
    textcolor(green); write('El año con la mayor temperatura detectada en algún punto del
planeta en los últimos 50 años es '); textcolor(red); write(anio_max2);
end.
```

Ejercicio 14.

El repositorio de código fuente más grande en la actualidad, GitHub, desea estimar el monto invertido en los proyectos que aloja. Para ello, dispone de una tabla con información de los desarrolladores que participan en un proyecto de software, junto al valor promedio que se paga por hora de trabajo:

CÓDIGO	ROL DEL DESARROLLADOR	VALOR/HORA (USD)
1	Analista Funcional	35,20
2	Programador	27,45
3	Administrador de bases de datos	31,03
4	Arquitecto de software	44,28
5	Administrador de redes y seguridad	39,87

Nota: los valores/hora se incluyen a modo de ejemplo

Realizar un programa que procese la información de los desarrolladores que participaron en los 1000 proyectos de software más activos durante el año 2017. De cada participante, se conoce su país de residencia, código de proyecto (1 a 1000), el nombre del proyecto en el que participó, el rol que cumplió en dicho proyecto (1 a 5) y la cantidad de horas trabajadas. La lectura finaliza al ingresar el código de proyecto -1, que no debe procesarse. Al finalizar la lectura, el programa debe informar:

- El monto total invertido en desarrolladores con residencia en Argentina.
- La cantidad total de horas trabajadas por Administradores de bases de datos.
- El código del proyecto con menor monto invertido.
- La cantidad de Arquitectos de software de cada proyecto.

```

program TP4_E14;
{$codepage UTF8}
uses crt;
const
    proyectos_total=1000;
    rol_ini=1; rol_fin=5;
    proyecto_salida=-1;
    pais_corte='Argentina';
    rol_corte1=3;
    rol_corte2=4;
type
    t_proyecto=1..proyectos_total;
    t_rol=rol_ini..rol_fin;
    t_registro_participante=record
        pais: string;
        proyecto: int16;
        nombre: string;
        rol: t_rol;
        horas: int16;
    end;
    t_registro_proyecto=record

```

```

    monto: real;
    cantidad: int16;
end;
t_vector_proyectos=array[t_proyecto] of t_registro_proyecto;
t_vector_salarios=array[t_rol] of real;
procedure cargar_vector_salarios(var vector_salarios: t_vector_salarios);
begin
    vector_salarios[1]:=35.20;
    vector_salarios[2]:=27.45;
    vector_salarios[3]:=31.03;
    vector_salarios[4]:=44.28;
    vector_salarios[5]:=39.87;
end;
procedure inicializar_vector_proyectos(var vector_proyectos: t_vector_proyectos);
var
    i: t_proyecto;
begin
    for i:= 1 to proyectos_total do
        begin
            vector_proyectos[i].monto:=0;
            vector_proyectos[i].cantidad:=0;
        end;
    end;
end;
procedure leer_participante(var registro_participante: t_registro_participante);
begin
    textcolor(green); write('Introducir código del proyecto (1 a 1000) en el que participó el
participante: ');
    textcolor(yellow); readln(registro_participante.proyecto);
    if (registro_participante.proyecto<>proyecto_salida) then
        begin
            textcolor(green); write('Introducir país de residencia del participante: ');
            textcolor(yellow); readln(registro_participante.pais);
            textcolor(green); write('Introducir nombre del proyecto en el que participó el
participante: ');
            textcolor(yellow); readln(registro_participante.nombre);
            textcolor(green); write('Introducir rol que cumplió en dicho proyecto (1 a 5) el
participante: ');
            textcolor(yellow); readln(registro_participante.rol);
            textcolor(green); write('Introducir cantidad de horas trabajadas del participante: ');
            textcolor(yellow); readln(registro_participante.horas);
        end;
    end;
end;
procedure cargar_vector_proyectos(var vector_proyectos: t_vector_proyectos; var monto_corte:
real; var horas_corte: int16; vector_salarios: t_vector_salarios);
var
    registro_participante: t_registro_participante;
begin
    leer_participante(registro_participante);
    while (registro_participante.proyecto<>proyecto_salida) do
        begin
            if (registro_participante.pais=pais_corte) then
                monto_corte:=monto_corte+vector_salarios[registro_participante.rol]*registro_participant
e.horas;
            if (registro_participante.rol=rol_corte1) then
                horas_corte:=horas_corte+registro_participante.horas;
                vector_proyectos[registro_participante.proyecto].monto:=vector_proyectos[registro_particip
ante.proyecto].monto+vector_salarios[registro_participante.rol]*registro_participante.horas;
            if (registro_participante.rol=rol_corte2) then
                vector_proyectos[registro_participante.proyecto].cantidad:=vector_proyectos[registro_par
ticipante.proyecto].cantidad+1;
                leer_participante(registro_participante);
            end;
        end;
    end;
end;
procedure procesar_vector_proyectos(vector_proyectos: t_vector_proyectos; var proyecto_min:
int16);
var

```



```
i: t_proyecto;
monto_min: real;
begin
  monto_min:=9999999;
  for i:= 1 to proyectos_total do
    begin
      if ((vector_proyectos[i].monto>0) and (vector_proyectos[i].monto<monto_min)) then
        begin
          monto_min:=vector_proyectos[i].monto;
          proyecto_min:=i;
        end;
      textcolor(green); write('La cantidad de Arquitectos de software del proyecto ',i,' es ');
      textcolor(red); writeln(vector_proyectos[i].cantidad);
    end;
  end;
var
  vector_salarios: t_vector_salarios;
  vector_proyectos: t_vector_proyectos;
  horas_corte, proyecto_min: int16;
  monto_corte: real;
begin
  cargar_vector_salarios(vector_salarios);
  monto_corte:=0;
  horas_corte:=0;
  proyecto_min:=0;
  inicializar_vector_proyectos(vector_proyectos);
  cargar_vector_proyectos(vector_proyectos,monto_corte,horas_corte,vector_salarios);
  textcolor(green); write('El monto total invertido en desarrolladores con residencia en ');
  textcolor(yellow); write(pais_corte); textcolor(green); write(' es '); textcolor(red);
  writeln(monto_corte:0:2);
  textcolor(green); write('La cantidad total de horas trabajadas por Administradores de bases
de datos es '); textcolor(red); writeln(horas_corte);
  procesar_vector_proyectos(vector_proyectos,proyecto_min);
  textcolor(green); write('El código de proyecto con menor monto invertido es ');
  textcolor(red); write(proyecto_min);
end.
```

Trabajo Práctico N° 4.2: Vectores (Parte 2).

Ejercicio 1.

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

```
function buscar_desordenado_vector_numeros(vector_numeros: t_vector_numeros; numero: int16):
boolean;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=num_total) and (vector_numeros[pos]<>numero)) do
        pos:=pos+1;
    buscar_desordenado_vector_numeros:=(pos<=num_total);
end;
```

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

```
function buscar_ordenado_vector_numeros(vector_numeros: t_vector_numeros; numero: int16):
boolean;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=num_total) and (vector_numeros[pos]<numero)) do
        pos:=pos+1;
    buscar_ordenado_vector_numeros:=((pos<=num_total) and (vector_numeros[pos]=numero));
end;
```

```
program TP4_E1;
{$codepage UTF8}
uses crt;
const
    num_total=500;
type
    t_numero=1..num_total;
    t_vector_numeros=array[t_numero] of int16;
procedure cargar_vector_numeros(var vector_numeros: t_vector_numeros);
var
    i: t_numero;
begin
    for i:= 1 to num_total do
        vector_numeros[i]:=random(high(int16));
    end;
function buscar_desordenado_vector_numeros(vector_numeros: t_vector_numeros; numero: int16):
boolean;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=num_total) and (vector_numeros[pos]<>numero)) do
        pos:=pos+1;
    buscar_desordenado_vector_numeros:=(pos<=num_total);
```

```
end;
procedure ordenar_vector_numeros(var vector_numeros: t_vector_numeros);
var
    i, j, k: t_numero;
    item: int16;
begin
    for i:= 1 to (num_total-1) do
        begin
            k:=i;
            for j:= (i+1) to num_total do
                if (vector_numeros[j]<vector_numeros[k]) then
                    k:=j;
                if (k<>i) then
                    begin
                        item:=vector_numeros[k];
                        vector_numeros[k]:=vector_numeros[i];
                        vector_numeros[i]:=item;
                    end;
                end;
            end;
        end;
    end;
function buscar_ordenado_vector_numeros(vector_numeros: t_vector_numeros; numero: int16):
boolean;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=num_total) and (vector_numeros[pos]<numero)) do
        pos:=pos+1;
    buscar_ordenado_vector_numeros:=((pos<=num_total) and (vector_numeros[pos]=numero));
end;
var
    vector_numeros: t_vector_numeros;
    numero: int16;
begin
    randomize;
    cargar_vector_numeros(vector_numeros);
    textcolor(green); write('Introducir número que se desea buscar en el vector: ');
    textcolor(yellow); readln(numero);
    textcolor(green); write('¿El número '); textcolor(yellow); write(numero); textcolor(green);
write(' se encontró en el vector (desordenado)? '); textcolor(red);
writeln(buscar_desordenado_vector_numeros(vector_numeros,numero));
    ordenar_vector_numeros(vector_numeros);
    textcolor(green); write('¿El número '); textcolor(yellow); write(numero); textcolor(green);
write(' se encontró en el vector (ordenado)? '); textcolor(red);
write(buscar_ordenado_vector_numeros(vector_numeros,numero));
end.
```

Ejercicio 2.

Realizar un programa que resuelva los siguientes incisos:

- Lea nombres de alumnos y los almacene en un vector de, a lo sumo, 500 elementos. La lectura finaliza cuando se lee el nombre 'ZZZ', que no debe procesarse.
- Lea un nombre y elimine la primera ocurrencia de dicho nombre en el vector.
- Lea un nombre y lo inserte en la posición 4 del vector.
- Lea un nombre y lo agregue al vector.

Nota: Realizar todas las validaciones necesarias.

```

program TP4_E2;
{$codepage UTF8}
uses crt;
const
    nombres_total=500;
    nombre_salida='ZZZ';
    pos_corte=4;
type
    t_nombre=1..nombres_total;
    t_vector_nombres=array[t_nombre] of string;
procedure inicializar_vector_nombres(var vector_nombres: t_vector_nombres);
var
    i: t_nombre;
begin
    for i:= 1 to nombres_total do
        vector_nombres[i]:='';
    end;
procedure cargar_vector_nombres(var vector_nombres: t_vector_nombres; var nombres: int16);
var
    nombre: string;
begin
    textcolor(green); write('Introducir nombre del alumno: ');
    textcolor(yellow); readln(nombre);
    while ((nombre<>nombre_salida) and (nombres<nombres_total)) do
        begin
            nombres:=nombres+1;
            vector_nombres[nombres]:=nombre;
            textcolor(green); write('Introducir nombre del alumno: ');
            textcolor(yellow); readln(nombre);
        end;
end;
function buscar_desordenado_vector_nombres(vector_nombres: t_vector_nombres; nombres: int16;
nombre: string): int16;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=nombres) and (vector_nombres[pos]<>nombre)) do
        pos:=pos+1;
    if (pos<=nombres) then
        buscar_desordenado_vector_nombres:=pos
    else
        buscar_desordenado_vector_nombres:=-1;
    end;
procedure eliminar_vector_nombres(var vector_nombres: t_vector_nombres; var nombres: int16;
nombre: string; pos: int16);
var
    i: t_nombre;
begin
    if ((pos>=1) and (pos<=nombres)) then
        begin

```

```

    for i:= pos to (nombres-1) do
        vector_nombres[i]:=vector_nombres[i+1];
        nombres:=nombres-1;
    end;
end;
procedure insertar_vector_nombres(var vector_nombres: t_vector_nombres; var nombres: int16;
nombre: string; pos: int16);
var
    i: t_nombre;
begin
    if ((nombres<nombres_total) and ((pos>=1) and (pos<=nombres))) then
    begin
        for i:= nombres downto pos do
            vector_nombres[i+1]:=vector_nombres[i];
            vector_nombres[pos_corte]:=nombre;
            nombres:=nombres+1;
        end;
    end;
end;
procedure agregar_vector_nombres(var vector_nombres: t_vector_nombres; var nombres: int16;
nombre: string);
begin
    if (nombres<nombres_total) then
    begin
        nombres:=nombres+1;
        vector_nombres[nombres]:=nombre;
    end;
end;
var
    vector_nombres: t_vector_nombres;
    nombre: string;
    nombres: int16;
begin
    nombres:=0;
    inicializar_vector_nombres(vector_nombres);
    cargar_vector_nombres(vector_nombres,nombres);
    if (nombres>0) then
    begin
        textcolor(green); write('Introducir nombre cuya primera ocurrencia se desea eliminar del
vector: ');
        textcolor(yellow); readln(nombre);
        eliminar_vector_nombres(vector_nombres,nombres,nombre,buscar_desordenado_vector_nombres(vect
or_nombres,nombres,nombre));
        textcolor(green); write('Introducir nombre para insertarlo en la posición ');
        textcolor(yellow); write(pos_corte); textcolor(green); write(' del vector: ');
        textcolor(yellow); readln(nombre);
        insertar_vector_nombres(vector_nombres,nombres,nombre,pos_corte);
        textcolor(green); write('Introducir nombre para agregarlo en el vector: ');
        textcolor(yellow); readln(nombre);
        agregar_vector_nombres(vector_nombres,nombres,nombre);
    end;
end.

```

Ejercicio 3.

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

- Realizar un programa que lea y almacene la información de los viajes (a lo sumo, 200). La lectura finaliza cuando se ingresa una distancia recorrida igual a 0 km, que no debe procesarse.
- Realizar un módulo que reciba el vector generado en (a) e informe:
 - El monto promedio transportado de los viajes realizados.
 - La distancia recorrida y el día del mes en que se realizó el viaje que transportó menos dinero.
 - La cantidad de viajes realizados cada día del mes.
- Realizar un módulo que reciba el vector generado en (a) y elimine todos los viajes cuya distancia recorrida sea igual a 100 km.

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

```
program TP4_E3;
{$codepage UTF8}
uses crt;
const
  dia_ini=1; dia_fin=31;
  viajes_total=200;
  distancia_salida=0;
  distancia_corte=100;
type
  t_viaje=1..viajes_total;
  t_dia=dia_ini..dia_fin;
  t_registro_viaje=record
    dia: t_dia;
    monto: real;
    distancia: real;
  end;
  t_vector_viajes=array[t_viaje] of t_registro_viaje;
  t_vector_dias=array[t_dia] of int16;
procedure inicializar_vector_dias(var vector_dias: t_vector_dias);
var
  i: t_dia;
begin
  for i:= dia_ini to dia_fin do
    vector_dias[i]:=0;
  end;
procedure leer_viaje(var registro_viaje: t_registro_viaje);
begin
  textcolor(green); write('Introducir distancia recorrida por el camión (medida en kilómetros)
del viaje: ');
  textcolor(yellow); readln(registro_viaje.distancia);
  if (registro_viaje.distancia<>distancia_salida) then
  begin
    textcolor(green); write('Introducir día del mes de marzo del viaje: ');
    textcolor(yellow); readln(registro_viaje.dia);
    textcolor(green); write('Introducir monto de dinero transportado del viaje: ');
    textcolor(yellow); readln(registro_viaje.monto);
  end;
end;
procedure cargar_vector_viajes(var vector_viajes: t_vector_viajes; var viajes: int16);
```

```

var
    registro_viaje: t_registro_viaje;
begin
    leer_viaje(registro_viaje);
    while ((registro_viaje.distancia<>distancia_salida) and (viajes<viajes_total)) do
        begin
            viajes:=viajes+1;
            vector_viajes[viajes]:=registro_viaje;
            leer_viaje(registro_viaje);
        end;
    end;
procedure actualizar_minimo(monto: real; dia: t_dia; distancia: real; var monto_min: real; var
dia_min: int8; var distancia_min: real);
begin
    if (monto<monto_min) then
        begin
            monto_min:=monto;
            dia_min:=dia;
            distancia_min:=distancia;
        end;
    end;
procedure calcular_informar_vector_viajes(vector_viajes: t_vector_viajes; viajes: int16);
var
    vector_dias: t_vector_dias;
    i: t_viaje;
    j: t_dia;
    dia_min: int8;
    monto_total, monto_prom, monto_min, distancia_min: real;
begin
    monto_total:=0; monto_prom:=0;
    monto_min:=999999; distancia_min:=0; dia_min:=0;
    inicializar_vector_dias(vector_dias);
    for i:= 1 to viajes do
        begin
            monto_total:=monto_total+vector_viajes[i].monto;
            actualizar_minimo(vector_viajes[i].monto,vector_viajes[i].dia,vector_viajes[i].distancia,m
onto_min,dia_min,distancia_min);
            vector_dias[vector_viajes[i].dia]:=vector_dias[vector_viajes[i].dia]+1;
        end;
    monto_prom:=monto_total/viajes;
    textcolor(green); write('El monto promedio de los viajes realizados es '); textcolor(red);
writeln(monto_prom:0:2);
    textcolor(green); write('La distancia recorrida y el día del mes en que se realizó el viaje
que transportó menos dinero son '); textcolor(red); write(distancia_min:0:2);
    textcolor(green); write(' y '); textcolor(red); write(dia_min); textcolor(green); writeln(',
respectivamente');
    for j:= dia_ini to dia_fin do
        begin
            textcolor(green); write('La cantidad de viajes realizados el día ',i,' del mes de marzo es
'); textcolor(red); writeln(vector_dias[i]);
        end;
    end;
procedure buscar_desordenado_vector_viajes(vector_viajes: t_vector_viajes; viajes: int16;
distancia: real; var pos: int16);
begin
    while ((pos<=viajes) and (vector_viajes[pos].distancia<>distancia)) do
        pos:=pos+1;
        if (pos>viajes) then
            pos:=-1;
        end;
procedure eliminar_vector_viajes(var vector_viajes: t_vector_viajes; var viajes: int16; pos:
int16);
var
    i: t_viaje;
begin
    if ((pos>=1) and (pos<=viajes)) then

```

```
begin
  for i:= pos to (viajes-1) do
    vector_viajes[i]:=vector_viajes[i+1];
    viajes:=viajes-1;
  end;
end;
procedure buscar_eliminar_vector_viajes(var vector_viajes: t_vector_viajes; var viajes:
int16);
var
  pos: int16;
begin
  pos:=0;
  buscar_desordenado_vector_viajes(vector_viajes,viajes,distancia_corte,pos);
  while ((pos>=1) and (pos<=viajes)) do
    begin
      eliminar_vector_viajes(vector_viajes,viajes,pos);
      buscar_desordenado_vector_viajes(vector_viajes,viajes,distancia_corte,pos);
    end;
  end;
var
  vector_viajes: t_vector_viajes;
  viajes: int16;
begin
  viajes:=0;
  cargar_vector_viajes(vector_viajes,viajes);
  if (viajes>0) then
    begin
      calcular_informar_vector_viajes(vector_viajes,viajes);
      buscar_eliminar_vector_viajes(vector_viajes,viajes);
    end;
  end.
end.
```


Ejercicio 4.

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

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

```
function buscar_ordenado1_vector_alumnos(vector_alumnos: t_vector_alumnos; alumnos, numero:
int16): int16;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=alumnos) and (vector_alumnos[pos].numero<numero)) do
        pos:=pos+1;
        buscar_ordenado1_vector_alumnos:=pos;
    end;
function calcular_a(vector_alumnos: t_vector_alumnos; alumnos, numero: int16): int16;
begin
    calcular_a:=buscar_ordenado1_vector_alumnos(vector_alumnos,alumnos,numero);
end;
```

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

```
procedure insertar_vector_alumnos(var vector_alumnos: t_vector_alumnos; var alumnos: int16;
registro_alumno: t_registro_alumno; pos: int16);
var
    i: t_alumno;
begin
    if ((alumnos<alumnos_total) and ((pos>=1) and (pos<=alumnos))) then
        for i:= alumnos downto pos do
            vector_alumnos[i+1]:=vector_alumnos[i];
        end;
        if ((alumnos<alumnos_total) and ((pos>=1) and (pos<=alumnos+1))) then
            begin
                vector_alumnos[pos]:=registro_alumno;
                alumnos:=alumnos+1;
            end;
        end;
    end;
procedure agregar_vector_alumnos(var vector_alumnos: t_vector_alumnos; alumnos: int16;
registro_alumno: t_registro_alumno);
begin
    vector_alumnos[alumnos+1]:=registro_alumno;
end;
procedure calcular_b(var vector_alumnos: t_vector_alumnos; var alumnos: int16;
registro_alumno: t_registro_alumno);
var
    pos: int16;
begin
    pos:=0;
    if (alumnos<alumnos_total) then
        begin
            pos:=buscar_ordenado1_vector_alumnos(vector_alumnos,alumnos,registro_alumno.numero);
            insertar_vector_alumnos(vector_alumnos,alumnos,registro_alumno,pos);
        end;
    end;
end;
```

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

```
procedure calcular_c(var vector_alumnos: t_vector_alumnos; var alumnos: int16; pos: int16);
var
  i: t_alumno;
begin
  if ((pos>=1) and (pos<=alumnos)) then
  begin
    for i:= pos to (alumnos-1) do
      vector_alumnos[i]:=vector_alumnos[i+1];
    alumnos:=alumnos-1;
  end;
end;
```

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

```
function buscar_ordenado2_vector_alumnos(vector_alumnos: t_vector_alumnos; alumnos, numero:
int16): int16;
var
  pos: int16;
begin
  pos:=1;
  while ((pos<=alumnos) and (vector_alumnos[pos].numero<numero)) do
    pos:=pos+1;
  if ((pos<=alumnos) and (vector_alumnos[pos].numero=numero)) then
    buscar_ordenado2_vector_alumnos:=pos
  else
    buscar_ordenado2_vector_alumnos:=-1;
  end;
procedure calcular_d(var vector_alumnos: t_vector_alumnos; var alumnos: int16; numero: int16);
begin
  calcular_c(vector_alumnos,alumnos,buscar_ordenado2_vector_alumnos(vector_alumnos,alumnos,num
ero));
end;
```

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

```
procedure buscar_desordenado_vector_alumnos(vector_alumnos: t_vector_alumnos; alumnos: int16;
var pos: int16);
begin
  while ((pos<=alumnos) and (vector_alumnos[pos].asistencias<>asistencias_corte)) do
    pos:=pos+1;
  if (pos>alumnos) then
    pos:=-1;
  end;
procedure calcular_e(var vector_alumnos: t_vector_alumnos; var alumnos: int16);
var
  pos: int16;
begin
  pos:=0;
  buscar_desordenado_vector_alumnos(vector_alumnos,alumnos,pos);
  while ((pos>=1) and (pos<=alumnos)) do
  begin
```

```

    calcular_c(vector_alumnos,alumnos,pos);
    buscar_desordenado_vector_alumnos(vector_alumnos,alumnos,pos);
end;
end;

```

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

```

program TP4_E4;
{$codepage UTF8}
uses crt;
const
    alumnos_total=1000;
    numero_salida=-1;
    asistencias_corte=0;
type
    t_alumno=1..alumnos_total;
    t_registro_alumno=record
        numero: int16;
        apellido: string;
        nombre: string;
        asistencias: int8;
    end;
    t_vector_alumnos=array[t_alumno] of t_registro_alumno;
procedure leer_alumno(var registro_alumno: t_registro_alumno);
begin
    textcolor(green); write('Introducir número de alumno del alumno: ');
    textcolor(yellow); readln(registro_alumno.numero);
    if (registro_alumno.numero<>numero_salida) then
    begin
        textcolor(green); write('Introducir apellido del alumno: ');
        textcolor(yellow); readln(registro_alumno.apellido);
        textcolor(green); write('Introducir nombre del alumno: ');
        textcolor(yellow); readln(registro_alumno.nombre);
        textcolor(green); write('Introducir cantidad de asistencias a clase del alumno: ');
        textcolor(yellow); readln(registro_alumno.asistencias);
    end;
end;
function buscar_ordenado1_vector_alumnos(vector_alumnos: t_vector_alumnos; alumnos, numero:
int16): int16;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=alumnos) and (vector_alumnos[pos].numero<numero)) do
        pos:=pos+1;
    buscar_ordenado1_vector_alumnos:=pos;
end;
procedure insertar_vector_alumnos(var vector_alumnos: t_vector_alumnos; var alumnos: int16;
registro_alumno: t_registro_alumno; pos: int16);
var
    i: t_alumno;
begin
    if ((alumnos<alumnos_total) and ((pos>=1) and (pos<=alumnos))) then
        for i:= alumnos downto pos do
            vector_alumnos[i+1]:=vector_alumnos[i];
        if ((alumnos<alumnos_total) and ((pos>=1) and (pos<=alumnos+1))) then
        begin
            vector_alumnos[pos]:=registro_alumno;
            alumnos:=alumnos+1;
        end;
    end;
end;

```

```

procedure cargar_vector_alumnos(var vector_alumnos: t_vector_alumnos; var alumnos: int16);
var
    registro_alumno: t_registro_alumno;
    pos: int16;
begin
    pos:=0;
    leer_alumno(registro_alumno);
    while ((registro_alumno.numero<>numero_salida) and (alumnos<alumnos_total)) do
    begin
        pos:=buscar_ordenado1_vector_alumnos(vector_alumnos,alumnos,registro_alumno.numero);
        insertar_vector_alumnos(vector_alumnos,alumnos,registro_alumno,pos);
        leer_alumno(registro_alumno);
    end;
end;
function calcular_a(vector_alumnos: t_vector_alumnos; alumnos, numero: int16): int16;
begin
    calcular_a:=buscar_ordenado1_vector_alumnos(vector_alumnos,alumnos,numero);
end;
procedure calcular_b(var vector_alumnos: t_vector_alumnos; var alumnos: int16;
registro_alumno: t_registro_alumno);
var
    pos: int16;
begin
    pos:=0;
    if (alumnos<alumnos_total) then
    begin
        pos:=buscar_ordenado1_vector_alumnos(vector_alumnos,alumnos,registro_alumno.numero);
        insertar_vector_alumnos(vector_alumnos,alumnos,registro_alumno,pos);
    end;
end;
procedure calcular_c(var vector_alumnos: t_vector_alumnos; var alumnos: int16; pos: int16);
var
    i: t_alumno;
begin
    if ((pos>=1) and (pos<=alumnos)) then
    begin
        for i:= pos to (alumnos-1) do
            vector_alumnos[i]:=vector_alumnos[i+1];
            alumnos:=alumnos-1;
        end;
    end;
end;
function buscar_ordenado2_vector_alumnos(vector_alumnos: t_vector_alumnos; alumnos, numero:
int16): int16;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=alumnos) and (vector_alumnos[pos].numero<numero)) do
        pos:=pos+1;
    if ((pos<=alumnos) and (vector_alumnos[pos].numero=numero)) then
        buscar_ordenado2_vector_alumnos:=pos
    else
        buscar_ordenado2_vector_alumnos:=-1;
    end;
end;
procedure calcular_d(var vector_alumnos: t_vector_alumnos; var alumnos: int16; numero: int16);
begin
    calcular_c(vector_alumnos,alumnos,buscar_ordenado2_vector_alumnos(vector_alumnos,alumnos,num
ero));
end;
procedure buscar_desordenado_vector_alumnos(vector_alumnos: t_vector_alumnos; alumnos: int16;
var pos: int16);
begin
    while ((pos<=alumnos) and (vector_alumnos[pos].asistencias<>asistencias_corte)) do
        pos:=pos+1;
    if (pos>alumnos) then
        pos:=-1;

```

```
end;
procedure calcular_e(var vector_alumnos: t_vector_alumnos; var alumnos: int16);
var
    pos: int16;
begin
    pos:=0;
    buscar_desordenado_vector_alumnos(vector_alumnos,alumnos,pos);
    while ((pos>=1) and (pos<=alumnos)) do
    begin
        calcular_c(vector_alumnos,alumnos,pos);
        buscar_desordenado_vector_alumnos(vector_alumnos,alumnos,pos);
    end;
end;
var
    registro_alumno: t_registro_alumno;
    vector_alumnos: t_vector_alumnos;
    alumnos, pos, numero: int16;
begin
    alumnos:=0;
    cargar_vector_alumnos(vector_alumnos,alumnos);
    if (alumnos>0) then
    begin
        textcolor(green); write('Introducir número de alumno del alumno del cual se desea conocer
su posición en el vector: ');
        textcolor(yellow); readln(numero);
        textcolor(green); write('La posición en el vector del alumno con número de alumno ');
        textcolor(yellow); write(numero); textcolor(green); write(' es '); textcolor(red);
        writeln(calcular_a(vector_alumnos,alumnos,numero));
        textcolor(green); writeln('Leer un alumno para insertarlo en el vector: ');
        leer_alumno(registro_alumno);
        calcular_b(vector_alumnos,alumnos,registro_alumno);
        textcolor(green); write('Introducir posición de un alumno dentro del vector que se desea
eliminar del vector: ');
        textcolor(yellow); readln(pos);
        calcular_c(vector_alumnos,alumnos,pos);
        textcolor(green); write('Introducir número de alumno del alumno que se desea eliminar del
vector: ');
        textcolor(yellow); readln(numero);
        calcular_d(vector_alumnos,alumnos,numero);
        calcular_e(vector_alumnos,alumnos);
    end;
end.
```

Ejercicio 5.

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

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

```
program TP4_E5;
{$codepage UTF8}
uses crt;
const
    clientes_total=500;
    ciudades_total=2400;
    mes_ini=1; mes_fin=12;
    anio_ini=2001; anio_fin=2020;
    cat_ini='A'; cat_fin='F';
    ciudad_ini=1; ciudad_fin=10;
type
    t_cliente=1..clientes_total;
    t_mes=mes_ini..mes_fin;
    t_anio=anio_ini..anio_fin;
    t_categoria=cat_ini..cat_fin;
    t_ciudad1=1..ciudades_total;
    t_ciudad2=ciudad_ini..ciudad_fin;
    t_registro_cliente=record
        fecha: int16;
        categoria: t_categoria;
        ciudad: t_ciudad1;
        monto: real;
    end;
    t_registro_ciudad=record
        ciudad: int16;
        clientes: int16;
    end;
    t_vector_clientes=array[t_cliente] of t_registro_cliente;
    t_vector_meses=array[t_mes] of int16;
    t_vector_anios=array[t_anio] of int16;
    t_vector_categorias=array[t_categoria] of int16;
    t_vector_ciudades1=array[t_ciudad1] of int16;
    t_vector_ciudades2=array[t_ciudad2] of t_registro_ciudad;
procedure inicializar_vectores(var vector_meses1, vector_meses2: t_vector_meses; var
vector_anios: t_vector_anios; var vector_categorias: t_vector_categorias; var
vector_ciudades1: t_vector_ciudades1; var vector_ciudades2: t_vector_ciudades2);
var
    i: t_mes;
    j: t_anio;
    k: t_categoria;
    l: t_ciudad1;
    m: t_ciudad2;
```

```
begin
  for i:= mes_ini to mes_fin do
    begin
      vector_meses1[i]:=0;
      vector_meses2[i]:=0;
    end;
  for j:= anio_ini to anio_fin do
    vector_anios[j]:=0;
  for k:= cat_ini to cat_fin do
    vector_categorias[k]:=0;
  for l:= 1 to ciudades_total do
    vector_ciudades1[l]:=0;
  for m:= ciudad_ini to ciudad_fin do
    begin
      vector_ciudades2[m].ciudad:=0;
      vector_ciudades2[m].clientes:=0;
    end;
  end;
end;
procedure leer_cliente(var registro_cliente: t_registro_cliente);
begin
  registro_cliente.fecha:=((anio_ini-1)*12+1)+random((anio_fin-anio_ini)*12);
  registro_cliente.categoria:=chr(ord('A')+random(6));
  registro_cliente.ciudad:=1+random(ciudades_total);
  registro_cliente.monto:=1+random(9999999);
end;
function buscar_ordenado_vector_clientes(vector_clientes: t_vector_clientes; clientes, fecha:
int16): int16;
var
  pos: int16;
begin
  pos:=1;
  while ((pos<=clientes) and (vector_clientes[pos].fecha<fecha)) do
    pos:=pos+1;
  buscar_ordenado_vector_clientes:=pos;
end;
procedure insertar_vector_clientes(var vector_clientes: t_vector_clientes; var clientes:
int16; registro_cliente: t_registro_cliente; pos: int16);
var
  i: t_cliente;
begin
  if ((clientes<clientes_total) and ((pos>=1) and (pos<=clientes))) then
    for i:= clientes downto pos do
      vector_clientes[i+1]:=vector_clientes[i];
    if ((clientes<clientes_total) and ((pos>=1) and (pos<=clientes+1))) then
      begin
        vector_clientes[pos]:=registro_cliente;
        clientes:=clientes+1;
      end;
    end;
end;
procedure cargar_vector_clientes(var vector_clientes: t_vector_clientes; var monto_prom:
real);
var
  registro_cliente: t_registro_cliente;
  i: t_cliente;
  clientes, pos: int16;
  monto_total: real;
begin
  clientes:=0; pos:=0;
  monto_total:=0;
  for i:= 1 to clientes_total do
    begin
      leer_cliente(registro_cliente);
      pos:=buscar_ordenado_vector_clientes(vector_clientes,clientes,registro_cliente.fecha);
      insertar_vector_clientes(vector_clientes,clientes,registro_cliente,pos);
      monto_total:=monto_total+vector_clientes[i].monto;
    end;
  end;
```

```

    monto_prom:=monto_total/clientes_total;
end;
procedure agregar_vector_meses1(fecha: int16; var vector_meses1: t_vector_meses);
begin
    vector_meses1[fecha mod 12]:=vector_meses1[fecha mod 12]+1;
end;
procedure agregar_vector_anios(fecha: int16; var vector_anios: t_vector_anios);
begin
    vector_anios[fecha div 12]:=vector_anios[fecha div 12]+1;
end;
procedure agregar_vector_categorias(categoria: t_categoria; var vector_categorias:
t_vector_categorias);
begin
    vector_categorias[categoria]:=vector_categorias[categoria]+1;
end;
procedure agregar_vector_ciudades1(ciudad: t_ciudad1; var vector_ciudades1:
t_vector_ciudades1);
begin
    vector_ciudades1[ciudad]:=vector_ciudades1[ciudad]+1;
end;
procedure agregar_vector_meses2(fecha: int16; monto, monto_prom: real; var vector_meses2:
t_vector_meses);
begin
    if (monto>monto_prom) then
        vector_meses2[fecha mod 12]:=vector_meses2[fecha mod 12]+1;
end;
procedure actualizar_maximo(vector_anios: t_vector_anios; var anio_max: int16);
var
    i: t_anio;
    num_max: int16;
begin
    num_max:=low(int16);
    for i:= anio_ini to anio_fin do
        if (vector_anios[i]>num_max) then
            begin
                num_max:=vector_anios[i];
                anio_max:=i;
            end;
        end;
end;
function buscar_ordenado_vector_ciudades2(vector_ciudades2: t_vector_ciudades2; ciudades,
clientes: int16): int16;
var
    pos: int16;
begin
    pos:=1;
    while ((pos<=ciudades) and (vector_ciudades2[pos].clientes>clientes)) do
        pos:=pos+1;
    buscar_ordenado_vector_ciudades2:=pos;
end;
procedure insertar_vector_ciudades2(var vector_ciudades2: t_vector_ciudades2; var ciudades:
int16; registro_ciudad: t_registro_ciudad; pos: int16);
var
    i: t_ciudad2;
begin
    if ((ciudades<ciudad_fin) and ((pos>1) and (pos<=ciudades))) then
        for i:= ciudades downto pos do
            vector_ciudades2[i+1]:=vector_ciudades2[i];
        if ((ciudades<ciudad_fin) and ((pos>1) and (pos<=ciudades+1))) then
            begin
                vector_ciudades2[pos]:=registro_ciudad;
                ciudades:=ciudades+1;
            end;
        end;
end;
procedure actualizar_maximos(vector_ciudades1: t_vector_ciudades1; var vector_ciudades2:
t_vector_ciudades2);
var

```



```

registro_ciudad: t_registro_ciudad;
i: t_ciudad1;
ciudades, pos: int16;
begin
  ciudades:=0; pos:=0;
  for i:= 1 to ciudades_total do
    begin
      pos:=buscar_ordenado_vector_ciudades2(vector_ciudades2,ciudades,vector_ciudades1[i]);
      if (pos<=ciudad_fin) then
        begin
          if (ciudades=ciudad_fin) then
            ciudades:=ciudades-1;
          registro_ciudad.ciudad:=i;
          registro_ciudad.clientes:=vector_ciudades1[i];
          insertar_vector_ciudades2(vector_ciudades2,ciudades,registro_ciudad,pos);
        end;
      end;
    end;
  end;
procedure procesar_vector_clientes(vector_clientes: t_vector_clientes; monto_prom: real; var
vector_meses1, vector_meses2: t_vector_meses; var vector_anios: t_vector_anios; var anio_max:
int16; var vector_categorias: t_vector_categorias; var vector_ciudades1: t_vector_ciudades1;
var vector_ciudades2: t_vector_ciudades2);
var
  i: t_cliente;
begin
  for i:= 1 to clientes_total do
    begin
      agregar_vector_meses1(vector_clientes[i].fecha,vector_meses1);
      agregar_vector_anios(vector_clientes[i].fecha,vector_anios);
      agregar_vector_categorias(vector_clientes[i].categoria,vector_categorias);
      agregar_vector_ciudades1(vector_clientes[i].ciudad,vector_ciudades1);
      agregar_vector_meses2(vector_clientes[i].fecha,vector_clientes[i].monto,monto_prom,vector_
meses2);
    end;
    actualizar_maximo(vector_anios,anio_max);
    actualizar_maximos(vector_ciudades1,vector_ciudades2);
  end;
var
  vector_clientes: t_vector_clientes;
  vector_meses1, vector_meses2: t_vector_meses;
  vector_anios: t_vector_anios;
  vector_categorias: t_vector_categorias;
  vector_ciudades1: t_vector_ciudades1;
  vector_ciudades2: t_vector_ciudades2;
  anio_max: int16;
  monto_prom: real;
begin
  randomize;
  anio_max:=0;
  monto_prom:=0;
  inicializar_vectores(vector_meses1,vector_meses2,vector_anios,vector_categorias,vector_ciuda
des1,vector_ciudades2);
  cargar_vector_clientes(vector_clientes,monto_prom);
  procesar_vector_clientes(vector_clientes,monto_prom,vector_meses1,vector_meses2,vector_anios
,anio_max,vector_categorias,vector_ciudades1,vector_ciudades2);
end.

```

Ejercicio 6.

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

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

```

program TP4_E6;
{$codepage UTF8}
uses crt;
const
  computadoras_total=10000;
  computadora_salida=-1;
  version_corte1='18.04'; version_corte2='16.04';
  computadora_corte1=0; computadora_corte2=500;
type
  t_computadora=1..computadoras_total;
  t_registro_computadora=record
    computadora: int16;
    version: string;
    paquetes: int16;
    cuentas: int16;
  end;
  t_vector_computadoras=array[t_computadora] of t_registro_computadora;
procedure leer_computadora(var registro_computadora: t_registro_computadora);
begin
  textcolor(green); write('Introducir código de computadora de la computadora: ');
  textcolor(yellow); readln(registro_computadora.computadora);
  if (registro_computadora.computadora<>computadora_salida) then
  begin
    textcolor(green); write('Introducir versión de Ubuntu que utiliza la computadora: ');
    textcolor(yellow); readln(registro_computadora.version);
    textcolor(green); write('Introducir cantidad de paquetes instalados de la computadora: ');
    textcolor(yellow); readln(registro_computadora.paquetes);
    textcolor(green); write('Introducir cantidad de cuentas de usuario que posee la
computadora: ');
    textcolor(yellow); readln(registro_computadora.cuentas);
  end;
end;
function buscar_ordenado_vector_computadoras(vector_computadoras: t_vector_computadoras;
computadoras, computadora: int16): int16;
var
  pos: int16;
begin
  pos:=1;
  while ((pos<=computadoras) and (vector_computadoras[pos].computadora<computadora)) do
    pos:=pos+1;
  buscar_ordenado_vector_computadoras:=pos;
end;

```

```

procedure insertar_vector_computadoras(var vector_computadoras: t_vector_computadoras; var
computadoras: int16; registro_computadora: t_registro_computadora; pos: int16);
var
  i: t_computadora;
begin
  if ((computadoras<computadoras_total) and ((pos>=1) and (pos<=computadoras))) then
    for i:= computadoras downto pos do
      vector_computadoras[i+1]:=vector_computadoras[i];
    if ((computadoras<computadoras_total) and ((pos>=1) and (pos<=computadoras+1))) then
      begin
        vector_computadoras[pos]:=registro_computadora;
        computadoras:=computadoras+1;
      end;
    end;
end;
procedure cargar_vector_computadoras(var vector_computadoras: t_vector_computadoras; var
computadoras: int16);
var
  registro_computadora: t_registro_computadora;
  pos: int16;
begin
  pos:=0;
  leer_computadora(registro_computadora);
  while ((registro_computadora.computadora<>computadora_salida) and
(computadoras<computadoras_total)) do
    begin
      pos:=buscar_ordenado_vector_computadoras(vector_computadoras,computadoras,registro_computa
dora.computadora);
      insertar_vector_computadoras(vector_computadoras,computadoras,registro_computadora,pos);
      leer_computadora(registro_computadora);
    end;
  end;
end;
procedure actualizar_maximo(paquetes: int16; version: string; var paquetes_max: int16; var
version_max: string);
begin
  if (paquetes>paquetes_max) then
    begin
      paquetes_max:=paquetes;
      version_max:=version;
    end;
  end;
end;
procedure eliminar_vector_computadoras(var vector_computadoras: t_vector_computadoras; var
computadoras: int16; pos: int16);
var
  i: t_computadora;
begin
  if ((pos>=1) and (pos<=computadoras)) then
    begin
      for i:= pos to (computadoras-1) do
        vector_computadoras[i]:=vector_computadoras[i+1];
      computadoras:=computadoras-1;
    end;
  end;
end;
procedure procesar_vector_computadoras(var vector_computadoras: t_vector_computadoras; var
computadoras, versiones_corte: int16; var cuentas_prom: real; var version_max: string);
var
  pos: int16;
  computadoras_aux, cuentas_total, paquetes_max: int16;
begin
  pos:=1;
  computadoras_aux:=computadoras;
  cuentas_total:=0;
  paquetes_max:=low(int16);
  while ((pos>=1) and (pos<=computadoras)) do
    begin
      if ((vector_computadoras[pos].version=version_corte1) or
(vector_computadoras[pos].version=version_corte2)) then

```

```

    versiones_corte:=versiones_corte+1;
    cuentas_total:=cuentas_total+vector_computadoras[pos].cuentas;
    actualizar_maximo(vector_computadoras[pos].paquetes,vector_computadoras[pos].version,paquetes_max,version_max);
    if ((vector_computadoras[pos].computadora>computadora_corte1) and
(vector_computadoras[pos].computadora<computadora_corte2)) then
        begin
            eliminar_vector_computadoras(vector_computadoras,computadoras,pos);
            pos:=pos-1;
        end;
        pos:=pos+1;
    end;
    cuentas_prom:=cuentas_total/computadoras_aux;
end;
var
    vector_computadoras: t_vector_computadoras;
    computadoras, versiones_corte: int16;
    cuentas_prom: real;
    version_max: string;
begin
    computadoras:=0;
    versiones_corte:=0;
    cuentas_prom:=0;
    version_max:='';
    cargar_vector_computadoras(vector_computadoras,computadoras);
    if (computadoras>0) then
        begin
            procesar_vector_computadoras(vector_computadoras,computadoras,versiones_corte,cuentas_prom,version_max);
            textcolor(green); write('La cantidad de computadoras que utilizan las versiones ');
            textcolor(yellow); write(version_corte1); textcolor(green); write(' o '); textcolor(yellow);
            write(version_corte2); textcolor(green); write(' es '); textcolor(red);
            writeln(versiones_corte);
            textcolor(green); write('El promedio de cuentas de usuario por computadora es ');
            textcolor(red); writeln(cuentas_prom:0:2);
            textcolor(green); write('La versión de Ubuntu de la computadora con mayor cantidad de
paquetes instalados es '); textcolor(red); write(version_max);
        end;
    end.

```

Ejercicio 7.

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

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

Ejercicio 1:

```
program TP4_E7a;
{$codepage UTF8}
uses crt;
const
    empresa_salida=100;
    monto_corte=50000.0;
    empresas_total=5000;
type
    t_empresa=1..empresas_total;
    t_registro_empresa=record
        empresa: int16;
        inversiones: int16;
        monto_total: real;
    end;
    t_vector_empresas=array[t_empresa] of t_registro_empresa;
procedure leer_inversiones(empresa: 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 cargar_vector_empresas(var vector_empresas: t_vector_empresas; var empresas: int16);
var
    registro_empresa: t_registro_empresa;
```

```

begin
  repeat
    leer_empresa(registro_empresa);
    empresas:=empresas+1;
    vector_empresas[empresas]:=registro_empresa;
  until (vector_empresas[empresas].empresa=empresa_salida);
end;
procedure calcular_a(empresa, inversiones: int16; monto_total: real);
begin
  textcolor(green); write('El monto promedio de las inversiones de la empresa ');
  textcolor(yellow); write(empresa); textcolor(green); write(' es '); textcolor(red);
  writeln(monto_total/inversiones:0:2);
end;
procedure calcular_b(monto_total: real; empresa: int16; var monto_max: real; var empresa_max:
int16);
begin
  if (monto_total>monto_max) then
  begin
    monto_max:=monto_total;
    empresa_max:=empresa;
  end;
end;
procedure calcular_c(monto_total: real; var empresas_corte: int16);
begin
  if (monto_total>monto_corte) then
    empresas_corte:=empresas_corte+1;
end;
procedure procesar_vector_empresas(vector_empresas: t_vector_empresas; empresas: int16; var
empresa_max, empresas_corte: int16);
var
  i: t_empresa;
  monto_max: real;
begin
  monto_max:=-9999999;
  for i:= 1 to empresas do
    if (vector_empresas[i].inversiones>0) then
    begin
      calcular_a(vector_empresas[i].empresa,vector_empresas[i].inversiones,vector_empresas[i].
monto_total);
      calcular_b(vector_empresas[i].monto_total,vector_empresas[i].empresa,monto_max,empresa_m
ax);
      calcular_c(vector_empresas[i].monto_total,empresas_corte);
    end;
  end;
var
  vector_empresas: t_vector_empresas;
  empresas, empresa_max, empresas_corte: int16;
begin
  empresas:=0;
  empresa_max:=0;
  empresas_corte:=0;
  cargar_vector_empresas(vector_empresas,empresas);
  procesar_vector_empresas(vector_empresas,empresas,empresa_max,empresas_corte);
  textcolor(green); write('El código de la empresa con mayor monto total invertido es ');
  textcolor(red); writeln(empresa_max);
  textcolor(green); write('La cantidad de empresas con inversiones de más de $');
  textcolor(yellow); write(monto_corte:0:2); textcolor(green); write(' es '); textcolor(red);
  write(empresas_corte);
end.

```

Ejercicio 2:

```

program TP4_E7b;
{$codepage UTF8}
uses crt;

```

```

const
    condicion_i='I'; condicion_r='R';
    autoeva_total=5;
    nota_incumple=-1;
    legajo_salida=-1;
    nota_corte=4;
    promedio_corte=6.5;
    nota_cero=0;
    nota_diez=10;
    presente_corte=0.75;
    alumnos_total=5000;
type
    t_alumno=1..alumnos_total;
    t_registro_alumno=record
        legajo: int16;
        condicion: char;
        presente: int8;
        notas_cero: int8;
        notas_diez: int8;
        nota_total: int8;
    end;
    t_vector_alumnos=array[t_alumno] of t_registro_alumno;
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;
    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 cargar_vector_alumnos(var vector_alumnos: t_vector_alumnos; var alumnos: int16);
var
    registro_alumno: t_registro_alumno;
begin
    leer_alumno(registro_alumno);
    while (registro_alumno.legajo<>legajo_salida) do
        begin
            alumnos:=alumnos+1;
            vector_alumnos[alumnos]:=registro_alumno;
            leer_alumno(registro_alumno);
        end;
    end;

```

```
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;
        end;
end;
procedure calcular_c(presente: int8; var alumnos_autoeva: int16);
begin
    if (presente=autoeva_total) then
        alumnos_autoeva:=alumnos_autoeva+1;
    end;
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;
end;
procedure calcular_e(notas_cero: int8; var alumnos_cero: int16);
begin
    if (notas_cero>=1) then
        alumnos_cero:=alumnos_cero+1;
    end;
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;
    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;
    end;
end;
```



```

procedure procesar_vector_alumnos(vector_alumnos: t_vector_alumnos; alumnos: int16; var
ingresantes_parcial, ingresantes_total, recursantes_parcial, recursantes_total,
alumnos_autoeva, alumnos_corte, alumnos_cero, legajo_diez_max1, legajo_diez_max2,
legajo_cero_max1, legajo_cero_max2: int16);
var
  i: t_alumno;
  notas_diez_max1, notas_diez_max2, notas_cero_max1, notas_cero_max2: int8;
begin
  notas_diez_max1:=0; notas_diez_max2:=0;
  notas_cero_max1:=0; notas_cero_max2:=0;
  for i:= 1 to alumnos do
    begin
      calcular_ab(vector_alumnos[i].condicion,vector_alumnos[i].presente,ingresantes_total,ingre
santes_parcial,recursantes_total,recursantes_parcial);
      calcular_c(vector_alumnos[i].presente,alumnos_autoeva);
      calcular_d(vector_alumnos[i].nota_total,alumnos_corte);
      calcular_e(vector_alumnos[i].notas_cero,alumnos_cero);
      calcular_f(vector_alumnos[i].notas_diez,vector_alumnos[i].legajo,notas_diez_max1,notas_die
z_max2,legajo_diez_max1,legajo_diez_max2);
      calcular_g(vector_alumnos[i].notas_cero,vector_alumnos[i].legajo,notas_cero_max1,notas_cer
o_max2,legajo_cero_max1,legajo_cero_max2);
    end;
  end;
var
  vector_alumnos: t_vector_alumnos;
  alumnos, ingresantes_parcial, ingresantes_total, recursantes_parcial, recursantes_total,
  alumnos_autoeva, alumnos_corte, alumnos_cero, legajo_diez_max1, legajo_diez_max2,
  legajo_cero_max1, legajo_cero_max2: int16;
begin
  alumnos:=0;
  ingresantes_parcial:=0; ingresantes_total:=0;
  recursantes_parcial:=0; recursantes_total:=0;
  alumnos_autoeva:=0;
  alumnos_corte:=0;
  alumnos_cero:=0;
  legajo_diez_max1:=0; legajo_diez_max2:=0;
  legajo_cero_max1:=0; legajo_cero_max2:=0;
  cargar_vector_alumnos(vector_alumnos,alumnos);
  if (alumnos>0) then
    begin
      procesar_vector_alumnos(vector_alumnos,alumnos,ingresantes_parcial,ingresantes_total,recur
santes_parcial,recursantes_total,alumnos_autoeva,alumnos_corte,alumnos_cero,legajo_diez_max1,l
egajo_diez_max2,legajo_cero_max1,legajo_cero_max2);
      if (ingresantes_total>0) then
        begin
          textcolor(green); write('La cantidad de alumnos INGRESANTES en condiciones de rendir el
parcial y el porcentaje sobre el total de alumnos INGRESANTES son '); textcolor(red);
write(ingresantes_parcial); textcolor(green); write(' y '); textcolor(red);
write(ingresantes_parcial/ingresantes_total*100:0:2); textcolor(green); writeln('%
respectivamente');
        end
      else
        begin
          textcolor(red); writeln('No hay alumnos INGRESANTES (I)');
        end;
      if (recursantes_total>0) then
        begin
          textcolor(green); write('La cantidad de alumnos RECURSANTES en condiciones de rendir el
parcial y el porcentaje sobre el total de alumnos RECURSANTES son '); textcolor(red);
write(recursantes_parcial); textcolor(green); write(' y '); textcolor(red);
write(recursantes_parcial/recursantes_total*100:0:2); textcolor(green); writeln('%
respectivamente');
        end
      else
        begin
          textcolor(red); writeln('No hay alumnos RECURSANTES (R)');
        end;
      end;
    end;
  end;
end;

```

```

end;
textcolor(green); write('La cantidad de alumnos que aprobaron todas las autoevaluaciones
es '); textcolor(red); writeln(alumnos_autoeva);
textcolor(green); write('La cantidad de alumnos cuya nota promedio fue mayor a ');
textcolor(yellow); write(promedio_corte:0:2); textcolor(green); write(' puntos es ');
textcolor(red); writeln(alumnos_corte);
textcolor(green); write('La cantidad de alumnos que obtuvieron cero puntos en, al menos,
una autoevaluación es '); textcolor(red); writeln(alumnos_cero);
textcolor(green); write('Los legajos de los dos alumnos con mayor cantidad de
autoevaluaciones con nota 10 (diez) son '); textcolor(red); write(legajo_diez_max1);
textcolor(green); write(' y '); textcolor(red); writeln(legajo_diez_max2);
textcolor(green); write('Los legajos de los dos alumnos con mayor cantidad de
autoevaluaciones con nota 0 (cero) son '); textcolor(red); write(legajo_cero_max1);
textcolor(green); write(' y '); textcolor(red); write(legajo_cero_max2);
end
else
begin
textcolor(red); write('No hay alumnos INGRESANTES (I) o RECURSANTES (R)');
end;
end.

```

Ejercicio 3:

```

program TP4_E7c;
{$codepage UTF8}
uses crt;
const
tanque_r='R'; tanque_c='C';
tanque_salida='Z';
alto_corte=1.40;
volumen_corte=800.0;
tanques_total=5000;
type
t_tanque=1..tanques_total;
t_registro_tanque=record
tanque: char;
radio: real;
alto: real;
ancho: real;
largo: real;
volumen: real;
end;
t_vector_tanques=array[t_tanque] of t_registro_tanque;
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.ancha*registro_tanque.largo*registro_tanque.alt
o;
    end;
end;
end;
procedure cargar_vector_tanques(var vector_tanques: t_vector_tanques; var tanques: int16);
var
    registro_tanque: t_registro_tanque;
begin
    leer_tanque(registro_tanque);
    while (registro_tanque.tanque<>tanque_salida) do
    begin
        tanques:=tanques+1;
        vector_tanques[tanques]:=registro_tanque;
        leer_tanque(registro_tanque);
    end;
end;
procedure calcular_a(volumen: real; var volumen_max1, volumen_max2: real);
begin
    if (volumen>volumen_max1) then
    begin
        volumen_max2:=volumen_max1;
        volumen_max1:=volumen;
    end
    else
        if (volumen>volumen_max2) then
            volumen_max2:=volumen;
end;
procedure calcular_bc(tanque: char; volumen: real; var volumen_total_c, volumen_total_r: real;
var tanques_c, tanques_r: int16);
begin
    if (tanque=tanque_c) then
    begin
        volumen_total_c:=volumen_total_c+volumen;
        tanques_c:=tanques_c+1;
    end
    else
    begin
        volumen_total_r:=volumen_total_r+volumen;
        tanques_r:=tanques_r+1;
    end;
end;
procedure calcular_d(alto: real; var tanques_corte_alto: int16);
begin
    if (alto<alto_corte) then
        tanques_corte_alto:=tanques_corte_alto+1;
end;
procedure calcular_e(volumen: real; var tanques_corte_volumen: int16);
begin
    if (volumen<volumen_corte) then
        tanques_corte_volumen:=tanques_corte_volumen+1;
end;
procedure procesar_vector_tanques(vector_tanques: t_vector_tanques; tanques: int16; var
volumen_max1, volumen_max2, volumen_total_c, volumen_total_r: real; var tanques_c, tanques_r,
tanques_corte_alto, tanques_corte_volumen: int16);
var
    i: t_tanque;
begin
    for i:= 1 to tanques do
    begin

```

```

    calcular_a(vector_tanques[i].volumen,volumen_max1,volumen_max2);
    calcular_bc(vector_tanques[i].tanque,vector_tanques[i].volumen,volumen_total_c,volumen_tot
al_r,tanques_c,tanques_r);
    calcular_d(vector_tanques[i].alto,tanques_corte_alto);
    calcular_e(vector_tanques[i].volumen,tanques_corte_volumen);
  end;
end;
var
  vector_tanques: t_vector_tanques;
  tanques, tanques_c, tanques_r, tanques_corte_alto, tanques_corte_volumen: int16;
  volumen_max1, volumen_max2, volumen_total_c, volumen_total_r: real;
begin
  tanques:=0;
  volumen_max1:=0; volumen_max2:=0;
  tanques_c:=0; volumen_total_c:=0;
  tanques_r:=0; volumen_total_r:=0;
  tanques_corte_alto:=0;
  tanques_corte_volumen:=0;
  cargar_vector_tanques(vector_tanques,tanques);
  if (tanques>0) then
    begin
      procesar_vector_tanques(vector_tanques,tanques,volumen_max1,volumen_max2,volumen_total_c,v
olumen_total_r,tanques_c,tanques_r,tanques_corte_alto,tanques_corte_volumen);
      textcolor(green); write('El volumen de los mayores tanques vendidos es '); textcolor(red);
write(volumen_max1:0:2); textcolor(green); write(' y '); textcolor(red);
writeln(volumen_max2:0:2);
      if (tanques_c>0) then
        begin
          textcolor(green); write('El volumen promedio de todos los tanques cilíndricos (C)
vendidos es '); textcolor(red); writeln(volumen_total_c/tanques_c:0:2);
        end
      else
        begin
          textcolor(red); writeln('No hay tanques cilíndricos (C) vendidos');
        end;
      if (tanques_r>0) then
        begin
          textcolor(green); write('El volumen promedio de todos los tanques rectangulares (R)
vendidos es '); textcolor(red); writeln(volumen_total_r/tanques_r:0:2);
        end
      else
        begin
          textcolor(red); writeln('No hay tanques rectangulares (R) vendidos');
        end;
      textcolor(green); write('La cantidad de tanques cuyo alto es menor a ');
textcolor(yellow); write(alto_corte:0:2); textcolor(green); write(' metros es ');
textcolor(red); writeln(tanques_corte_alto);
      textcolor(green); write('La cantidad de tanques cuyo volumen es menor a ');
textcolor(yellow); write(volumen_corte:0:2); textcolor(green); write(' metros cúbicos es ');
textcolor(red); write(tanques_corte_volumen);
    end
  else
    begin
      textcolor(red); write('No hay tanques cilíndricos (C) o rectangulares (R) vendidos');
    end;
  end.

```

Trabajo Práctico N° 5: **Punteros.**

Para algunos ejercicios de la parte práctica, se utilizará la función de Pascal “sizeof”, que recibe como parámetro una variable de cualquier tipo y retorna la cantidad de bytes que dicha variable ocupa en la memoria principal. Se presenta la siguiente tabla, que indica la cantidad de bytes que ocupa la representación interna de distintos tipos de datos en un compilador de Pascal típico. Se recomienda graficar cada una de las situaciones planteadas a partir de una prueba de escritorio.

TIPO	CANTIDAD DE BYTES
Entero	2 bytes
Real	4 bytes
Char	1 byte
String	Tantos bytes como indique la longitud del String + 1
Record	La suma de las longitudes de los campos del registro
Puntero	4 bytes
Boolean	1 byte

Tabla de referencia de tamaño de los tipos de datos de Pascal (estos valores pueden variar entre diferentes implementaciones del compilador)

Ejercicio 1.

Indicar los valores que imprime el siguiente programa en Pascal. Justificar mediante prueba de escritorio.

```

program TP5_E1;
{$codepage UTF8}
uses crt;
type
  cadena=string[50];
  puntero_cadena=^cadena;
var
  pc: puntero_cadena;
begin
  writeln(sizeof(pc), ' bytes');
  new(pc);
  writeln(sizeof(pc), ' bytes');
  pc^:='un nuevo nombre';
  writeln(sizeof(pc), ' bytes');
  writeln(sizeof(pc^), ' bytes');
  pc^:='otro nuevo nombre distinto al anterior';
  writeln(sizeof(pc^), ' bytes');
end.

```

Instrucciones	pc	pc^	write
writeln(sizeof(pc), ' bytes');			4 bytes
new(pc);	XXX		
writeln(sizeof(pc), ' bytes');			4 bytes
pc^:='un nuevo nombre';		‘un nuevo nombre’	
writeln(sizeof(pc), ' bytes');			4 bytes
writeln(sizeof(pc^), ' bytes');			51 bytes
pc^:='otro nuevo nombre distinto al anterior';		‘otro nuevo nombre distinto al anterior’	
writeln(sizeof(pc^), ' bytes');			51 bytes

Ejercicio 2.

Indicar los valores que imprime el siguiente programa en Pascal. Justificar mediante prueba de escritorio.

```

program TP5_E2;
{$codepage UTF8}
uses crt;
type
  cadena=string[9];
  producto=record
    codigo: integer;
    descripcion: cadena;
    precio: real;
  end;
  puntero_producto=^producto;
var
  p: puntero_producto;
  prod: producto;
begin
  writeln(sizeof(p),' bytes');
  writeln(sizeof(prod),' bytes');
  new(p);
  writeln(sizeof(p),' bytes');
  p^.codigo:=1;
  p^.descripcion:='nuevo producto';
  writeln(sizeof(p^),' bytes');
  p^.precio:=200;
  writeln(sizeof(p^),' bytes');
  prod.codigo:=2;
  prod.descripcion:='otro nuevo producto';
  writeln(sizeof(prod),' bytes');
end.

```

Instrucciones	p	p^	prod	write
writeln(sizeof(p),' bytes');				4 bytes
writeln(sizeof(prod),' bytes');				16 (24) bytes
new(p);	XXX			
writeln(sizeof(p),' bytes');				4 bytes
p^.codigo:=1;		.codigo=1		
p^.descripcion:='nuevo producto';		.descripcion='nuevo pro'		warning
writeln(sizeof(p^),' bytes');				16 (24) bytes
p^.precio:=200;		.precio=200		
writeln(sizeof(p^),' bytes');				16 (24) bytes
prod.codigo:=2;			.codigo=2	
prod.descripcion:='otro nuevo producto';			.descripcion='otro nuev'	warning
writeln(sizeof(prod),' bytes');				16 (24) bytes

Ejercicio 3.

Indicar los valores que imprime el siguiente programa en Pascal. Justificar mediante prueba de escritorio.

```
program TP5_E3;
{$codepage UTF8}
uses crt;
type
  numeros=array[1..10000] of integer;
  puntero_numeros=^numeros;
var
  n: puntero_numeros;
  num: numeros;
  i: integer;
begin
  writeln(sizeof(n), ' bytes');
  writeln(sizeof(num), ' bytes');
  new(n);
  writeln(sizeof(n^), ' bytes');
  for i:= 1 to 5000 do
    n^[i]:=i;
  writeln(sizeof(n^), ' bytes');
end.
```

Instrucciones	n	n^	num	i	write
writeln(sizeof(n), ' bytes');					4 bytes
writeln(sizeof(num), ' bytes');					20000 bytes
new(n);	XXX				
writeln(sizeof(n^), ' bytes');					20000 bytes
for i:= 1 to 5000 do n^[i]:=i;			n^[1..5000]=[1..5000]	[1..5000]	
writeln(sizeof(n^), ' bytes');					20000 bytes

Ejercicio 4.

Indicar los valores que imprime el siguiente programa en Pascal. Justificar mediante prueba de escritorio.

(a)

```
program TP5_E4a;
{$codepage UTF8}
uses crt;
type
  cadena=string[50];
  puntero_cadena=^cadena;
var
  pc: puntero_cadena;
begin
  pc^:='un nuevo texto';
  new(pc);
  writeln(pc^);
end.
```

Instrucciones	pc	pc^	write
pc^:='un nuevo texto';		error	
new(pc);	XXX		
writeln(pc^);			basura

(b)

```
program TP5_E4b;
{$codepage UTF8}
uses crt;
type
  cadena=string[50];
  puntero_cadena=^cadena;
var
  pc: puntero_cadena;
begin
  new(pc);
  pc^:='un nuevo nombre';
  writeln(sizeof(pc^), ' bytes');
  writeln(pc^);
  dispose(pc);
  pc^:='otro nuevo nombre';
end.
```

Instrucciones	pc	pc^	write
new(pc);	XXX		
pc^:='un nuevo nombre';		'un nuevo nombre'	
writeln(sizeof(pc^), ' bytes');			51 bytes
writeln(pc^);			un nuevo nombre
dispose(pc);	elimina puntero y libera memoria		
pc^:='otro nuevo nombre';		error	

(c)

```

program TP5_E4c;
{$codepage UTF8}
uses crt;
type
  cadena=string[50];
  puntero_cadena=^cadena;
procedure cambiarTexto(pun: puntero_cadena);
begin
  pun^:='Otro texto';
end;
var
  pc: puntero_cadena;
begin
  new(pc);
  pc^:='Un texto';
  writeln(pc^);
  cambiarTexto(pc);
  writeln(pc^);
end.

```

Instrucciones	pc	pc^	write
new(pc);	XXX		
pc^:='Un texto';		'Un texto'	
writeln(pc^);			Un texto
cambiarTexto(pc);		'Otro texto'	
writeln(pc^);			Otro texto

(d)

```

program TP5_E4d;
{$codepage UTF8}
uses crt;
type
  cadena=string[50];
  puntero_cadena=^cadena;
procedure cambiarTexto(pun: puntero_cadena);
begin
  new(pun);
  pun^:='Otro texto';
end;
var
  pc: puntero_cadena;
begin
  new(pc);
  pc^:='Un texto';
  writeln(pc^);
  cambiarTexto(pc);
  writeln(pc^);
end.

```

Instrucciones	pc	pc^	copia pc	copia pc^	write
new(pc);	XXX				
pc^:='Un texto';		'Un texto'			

writeln(pc^);					Un texto
cambiarTexto(pc);			YYY	‘Otro texto’	
writeln(pc^);					Un texto

Ejercicio 5.

De acuerdo a los valores de la tabla que indica la cantidad de bytes que ocupa la representación interna de cada tipo de dato en Pascal, calcular el tamaño de memoria en los puntos señalados a partir de (I), suponiendo que las variables del programa ya están declaradas y se cuenta con 400.000 bytes libres. Justificar mediante prueba de escritorio.

```
program TP5_E5;
{$codepage UTF8}
uses crt;
type
  TEmpleado=record
    sucursal: char;
    apellido: string[25];
    correoElectronico: string[40];
    sueldo: real;
  end;
Str50=string[50];
var
  alguien: TEmpleado;
  PtrEmpleado: ^TEmpleado;
begin
  {Suponer que, en este punto, se cuenta con 400.000 bytes de memoria disponible (I)}
  readln(alguien.apellido);
  {Pensar si la lectura anterior ha hecho variar la cantidad de memoria (II)}
  new(PtrEmpleado);
  {¿Cuánta memoria disponible hay ahora? (III)}
  readln(PtrEmpleado^.Sucursal,PtrEmpleado^.apellido);
  readln(PtrEmpleado^.correoElectronico,PtrEmpleado^.sueldo);
  {¿Cuánta memoria disponible hay ahora? (IV)}
  dispose(PtrEmpleado);
  {¿Cuánta memoria disponible hay ahora? (V)}
end.
```

Instrucciones	Memoria
<code>readln(alguien.apellido);</code>	400.000 bytes
<code>new(PtrEmpleado);</code>	399.928 (399.924) bytes (400.000 - 72)
<code>readln(PtrEmpleado^.Sucursal,PtrEmpleado^.apellido);</code>	399.928 (399.924) bytes
<code>readln(PtrEmpleado^.correoElectrónico,PtrEmpleado^.sueldo);</code>	399.928 (399.924) bytes
<code>dispose(PtrEmpleado);</code>	400.000 bytes (399.928 + 72)

Ejercicio 6.

Realizar un programa que ocupe 50 KB de memoria en total. Para ello:

(a) El programa debe utilizar sólo memoria estática.

```

program TP5_E6a;
{$codepage UTF8}
uses crt;
const
  KB=50; bytes=51200;
  vector_total=25600;
type
  t_vector=array[1..vector_total] of int16;
var
  vector: t_vector;
begin
  textcolor(green); write('La memoria estática ocupada por vector es '); textcolor(red);
  write(sizeof(vector)/1024:0:2); textcolor(green); write(' KB');
end.

```

(b) El programa debe utilizar el 50% de memoria estática y el 50% de memoria dinámica.

```

program TP5_E6b;
{$codepage UTF8}
uses crt;
const
  KB=50; bytes=51200;
  vector_total=12798;
type
  t_vector=array[1..vector_total] of int16;
  t_string=string[3];
  t_registro_vector=record
    vector: t_vector;
    palabra: t_string;
  end;
  t_puntero_registro=^t_registro_vector;
var
  vector: t_vector;
  puntero_registro: t_puntero_registro;
begin
  new(puntero_registro);
  textcolor(green); write('La memoria estática ocupada por vector es '); textcolor(red);
  write(sizeof(vector)/1024:0:4); textcolor(green); writeln(' KB');
  textcolor(green); write('La memoria estática ocupada por puntero_registro es ');
  textcolor(red); write(sizeof(puntero_registro)/1024:0:4); textcolor(green); writeln(' KB');
  textcolor(green); write('La memoria dinámica ocupada por contenido puntero_registro es ');
  textcolor(red); write(sizeof(puntero_registro^)/1024:0:4); textcolor(green); write(' KB');
end.

```

(c) El programa debe minimizar tanto como sea posible el uso de la memoria estática (a lo sumo, 4 bytes).

```

program TP5_E6c;
{$codepage UTF8}

```

```
uses crt;
const
  KB=50; bytes=51200;
  vector_total=25596;
type
  t_vector=array[1..vector_total] of int16;
  t_puntero_vector=^t_vector;
var
  puntero_vector: t_puntero_vector;
begin
  new(puntero_vector);
  textcolor(green); write('La memoria estática ocupada por puntero_vector es ');
textcolor(red); write(sizeof(puntero_vector)/1024:0:4); textcolor(green); writeln(' KB');
  textcolor(green); write('La memoria dinámica ocupada por contenido puntero_vector es ');
textcolor(red); write(sizeof(puntero_vector^)/1024:0:4); textcolor(green); write(' KB');
end.
```

Ejercicio 7.

Se desea almacenar en memoria una secuencia de 2500 nombres de ciudades, cada nombre podrá tener una longitud máxima de 50 caracteres.

(a) Definir una estructura de datos estática que permita guardar la información leída. Calcular el tamaño de memoria que requiere la estructura.

```
program TP5_E7a;
{$codepage UTF8}
uses crt;
const
  longitud_ciudad=50;
  ciudades_total=2500;
type
  t_ciudad=string[longitud_ciudad];
  t_vector_ciudad=array[1..ciudades_total] of t_ciudad;
begin
  textcolor(green); write('El tamaño de memoria que requiere la estructura es ');
  textcolor(red); write(sizeof(t_vector_ciudad)); textcolor(green); write(' bytes');
end.
```

(b) Dado que un compilador de Pascal típico no permite manejar estructuras de datos estáticas que superen los 64 KB, pensar en utilizar un vector de punteros a palabras, como se indica en la siguiente estructura.

Type

```
Nombre=String[50];
Puntero=^Nombre;
ArrPunteros=array[1..2500] of Puntero;
```

Var

```
Punteros: ArrPunteros;
```

(i) Indicar cuál es el tamaño de la variable Punteros al comenzar el programa.

El tamaño de la variable “Punteros”, al comenzar el programa, es 10.000 bytes.

(ii) Escribir un módulo que permita reservar memoria para los 2500 nombres. ¿Cuál es la cantidad de memoria reservada después de ejecutar el módulo? ¿La misma corresponde a memoria estática o dinámica?

```
procedure reservar_memoria(var punteros: t_vector_ciudad);
var
  i: int16;
begin
  for i:= 1 to ciudades_total do
    new(punteros[i]);
  end;
```

La cantidad de memoria reservada después de ejecutar el módulo es 127.500 bytes, la cual corresponde a memoria dinámica.

(iii) Escribir un módulo para leer los nombres y almacenarlos en la estructura de la variable Punteros.

```
procedure leer_ciudades(var vector_ciudad: t_vector_ciudad);
var
  i: int16;
begin
  for i:= 1 to ciudades_total do
  begin
    textcolor(green); write('Introducir nombre de ciudad ',i,': ');
    textcolor(yellow); readln(vector_ciudad[i]^);
  end;
end;
```

```
program TP5_E7b;
{$codepage UTF8}
uses crt;
const
  longitud_ciudad=50;
  ciudades_total=2500;
type
  t_ciudad=string[longitud_ciudad];
  t_puntero_ciudad=^t_ciudad;
  t_vector_ciudad=array[1..ciudades_total] of t_puntero_ciudad;
procedure reservar_memoria(var vector_ciudad: t_vector_ciudad);
var
  i: int16;
begin
  for i:= 1 to ciudades_total do
    new(vector_ciudad[i]);
  end;
procedure leer_ciudades(var vector_ciudad: t_vector_ciudad);
var
  i: int16;
begin
  for i:= 1 to ciudades_total do
  begin
    textcolor(green); write('Introducir nombre de ciudad ',i,': ');
    textcolor(yellow); readln(vector_ciudad[i]^);
  end;
end;
var
  vector_ciudad: t_vector_ciudad;
  i: int16;
begin
  for i:= 1 to ciudades_total do
    vector_ciudad[i]:=nil;
    textcolor(green); write('El tamaño de la variable vector_ciudad es '); textcolor(red);
write(sizeof(vector_ciudad)); textcolor(green); writeln(' bytes');
    textcolor(green); write('El tamaño del contenido de la variable vector_ciudad es ');
textcolor(red); write(sizeof(vector_ciudad[1]^)*length(vector_ciudad)); textcolor(green);
writeln(' bytes');
    reservar_memoria(vector_ciudad);
    textcolor(green); write('El tamaño de la variable vector_ciudad es '); textcolor(red);
write(sizeof(vector_ciudad)); textcolor(green); writeln(' bytes');
    textcolor(green); write('El tamaño del contenido de la variable vector_ciudad es ');
textcolor(red); write(sizeof(vector_ciudad[1]^)*length(vector_ciudad)); textcolor(green);
writeln(' bytes');
```



```
leer_ciudades(vector_ciudad);
  textcolor(green); write('El tamaño de la variable vector_ciudad es '); textcolor(red);
write(sizeof(vector_ciudad)); textcolor(green); writeln(' bytes');
  textcolor(green); write('El tamaño del contenido de la variable vector_ciudad es ');
textcolor(red); write(sizeof(vector_ciudad[1])*length(vector_ciudad)); textcolor(green);
write(' bytes');
  for i:= 1 to ciudades_total do
    dispose(vector_ciudad[i]);
end.
```

Ejercicio 8.

Analizar el siguiente programa:

```

program TP5_E8;
{$codepage UTF8}
uses crt;
type
  datos=array[1..20] of integer;
  punt=^datos;
procedure procesarDatos(v: datos; var v2: datos);
var
  i, j: integer;
begin
  for i:= 1 to 20 do
    v2[21-i]:=v[i];
  end;
var
  vect: datos;
  pvect: punt;
  i: integer;
begin
  for i:= 1 to 20 do
    vect[i]:=i;
  new(pvect);
  for i:= 20 downto 1 do
    pvect^[i]:=0;
  procesarDatos(pvect^,vect);
  writeln('fin');
end.

```

Responder: ¿cuánta memoria en total ocupa el programa al ejecutarse? Considerar tanto variables estáticas como dinámicas, parámetros y variables locales de los módulos.

Hasta sentencia de la línea	Memoria estática	Memoria dinámica	Memoria total
(a) 18	46 bytes	0 bytes	46 bytes
(b) 20	46 bytes	0 bytes	46 bytes
(c) 23	46 bytes	40 bytes	86 bytes
(d) 11	0 bytes	0 bytes	0 bytes
(e) 25	46 bytes	40 bytes	86 bytes

Aclaración: Hasta la sentencia de la línea 24, tenemos 88 bytes en memoria dinámica, ya que se suman 40 bytes por el parámetro por valor, 4 bytes por el parámetro por referencia y 4 bytes por las variables locales al proceso “procesarDatos”.

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;
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(valor);
  while (valor <> 0) do
    begin
      armarNodo(pri, valor);
      textcolor(green); write('Introducir número entero: ');
      textcolor(yellow); readln(valor);
    end;
  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.

```

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;

```

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

```

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;

```

```

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:=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;
var
  pri: lista;
  valor: integer;
begin
  pri:=nil;
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(valor);
  while (valor<>0) do
    begin
      armarNodo(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);
    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: integer;
    nombre: string;
    apellido: string;
  end;
  nodo=record
    dato: persona;
    sig: lista;
  end;
procedure leerPersona(var 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(var l: lista; p: persona);
var
  aux: lista;
begin
  new(aux);
  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: 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;
{Programa principal}
var
  l: lista;
begin
  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).*

```
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;
end;
```

(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.*

```
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;
```

```
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;
            end;
            ult^.sig:=aux;
        end;
    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;
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;
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;
end;
var
    pri, ult: lista;
    valor: 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);
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);
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.

```
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;
```

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

```
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;
```

(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*.

```
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;
```

```
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: integer;
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: integer);
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: lista;
  valor, divisor: integer;
begin
  pri:=nil;
  textcolor(green); write('Introducir número entero: ');
  textcolor(yellow); readln(valor);
  while (valor<>0) do
    begin
      armarNodo(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.
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;
end;
procedure procesar_lista_productos(lista_productos: t_lista_productos; var porcentaje_debajo:
real; var producto_min1, producto_min2: int16);
var
    productos_total, productos_debajo: int16;
    precio_min1, precio_min2: real;
begin
    productos_total:=0; productos_debajo:=0;
    precio_min1:=9999999; precio_min2:=9999999;
    while (lista_productos<>nil) do
    begin
        productos_total:=productos_total+1;
        if (lista_productos^.ele.stock_actual<lista_productos^.ele.stock_minimo) then
            productos_debajo:=productos_debajo+1;
        if (contar_pares(lista_productos^.ele.producto)=true) then
        begin
            textcolor(green); write('La descripción es este producto con código compuesto por, al
menos, tres dígitos pares es '); textcolor(red); writeln(lista_productos^.ele.descripcion);
        end;
        actualizar_minimos(lista_productos^.ele.precio,lista_productos^.ele.producto,precio_min1,p
recio_min2,producto_min1,producto_min2);
        lista_productos:=lista_productos^.sig;
    end;
    porcentaje_debajo:=productos_debajo/productos_total*100;
end;
var
    lista_productos: t_lista_productos;
    producto_min1, producto_min2: int16;
    porcentaje_debajo: real;
begin

```



```
lista_productos:=nil;
porcentaje_debajo:=0;
producto_min1:=0; producto_min2:=0;
cargar_lista_productos(lista_productos);
if (lista_productos<>nil) then
begin
  procesar_lista_productos(lista_productos,porcentaje_debajo,producto_min1,producto_min2);
  textcolor(green); write('El porcentaje de productos con stock actual por debajo de su
stock mínimo es '); textcolor(red); write(porcentaje_debajo:0:2); textcolor(green);
writeln('%');
  textcolor(green); write('Los códigos de los dos productos más económicos 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 la sonda: ');
textcolor(yellow); readln(registro_sonda.costo_construccion);
textcolor(green); write('Introducir costo de mantenimiento mensual de la sonda: ');
textcolor(yellow); readln(registro_sonda.costo_mantenimiento);
textcolor(green); write('Introducir rango del espectro electromagnético sobre el que
realizará estudios la sonda (1. radio; 2. microondas; 3. infrarrojo; 4. luz visible; 5.
ultravioleta; 6. rayos X; 7. rayos gamma): ');
textcolor(yellow); readln(registro_sonda.rango);
end;
procedure agregar_adelante_lista_sondas(var lista_sondas: t_lista_sondas; registro_sonda:
t_registro_sonda);
var
    nuevo: t_lista_sondas;
begin
    new(nuevo);
    nuevo^.ele:=registro_sonda;
    nuevo^.sig:=lista_sondas;
    lista_sondas:=nuevo;
end;
procedure cargar_lista_sondas(var lista_sondas: t_lista_sondas; var duracion_prom, costo_prom:
real);
var
    registro_sonda: t_registro_sonda;
    sondas_total: int16;
    duracion_total, costo_total: real;
begin
    duracion_total:=0; sondas_total:=0;
    costo_total:=0;
    repeat
        leer_sonda(registro_sonda);
        agregar_adelante_lista_sondas(lista_sondas,registro_sonda);
        duracion_total:=duracion_total+lista_sondas^.ele.duracion;
        sondas_total:=sondas_total+1;
        costo_total:=costo_total+lista_sondas^.ele.costo_construccion;
    until (registro_sonda.nombre=nombre_salida);
    duracion_prom:=duracion_total/sondas_total;
    costo_prom:=costo_total/sondas_total;
end;
procedure actualizar_maximo(costo: real; nombre: string; var costo_max: real; var nombre_max:
string);
begin
    if (costo>costo_max) then
    begin
        costo_max:=costo;
        nombre_max:=nombre;
    end;
end;
procedure procesar_lista_sondas(lista_sondas: t_lista_sondas; duracion_prom, costo_prom: real;
var nombre_max: string; var vector_rangos: t_vector_rangos; var sondas_prom: int16);
var
    costo_sonda, costo_max: real;
begin
    costo_max:=-9999999;
    while (lista_sondas<>nil) do
    begin
        costo_sonda:=lista_sondas^.ele.costo_construccion+lista_sondas^.ele.costo_mantenimiento*li
sta_sondas^.ele.duracion;
        actualizar_maximo(costo_sonda,lista_sondas^.ele.nombre,costo_max,nombre_max);
        vector_rangos[lista_sondas^.ele.rango]:=vector_rangos[lista_sondas^.ele.rango]+1;
        if (lista_sondas^.ele.duracion>duracion_prom) then
            sondas_prom:=sondas_prom+1;
        if (lista_sondas^.ele.costo_construccion>costo_prom) then
        begin
            textcolor(green); write('El nombre de esta sonda cuyo costo de construcción supera el
costo promedio entre todas las sondas es '); textcolor(red);
writeln(lista_sondas^.ele.nombre);

```

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

Ejercicio 7.

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

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

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

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

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

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

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

```

    textcolor(green); write('Introducir costo de mantenimiento mensual de la sonda: ');
    textcolor(yellow); readln(registro_sonda.costos_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_registro_sonda;
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.costos_construccion;
    until (registro_sonda.nombre=nombre_salida);
    duracion_prom:=duracion_total/sondas_total;
    costo_prom:=costo_total/sondas_total;
end;
procedure actualizar_maximo(costos: real; nombre: string; var costo_max: real; var nombre_max:
string);
begin
    if (costos>costo_max) then
        begin
            costo_max:=costos;
            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
    costos_sonda, costo_max: real;
begin
    costo_max:=-9999999;
    while (lista_sondas<>nil) do
        begin
            costos_sonda:=lista_sondas^.ele.costos_construccion+lista_sondas^.ele.costos_mantenimiento*li
sta_sondas^.ele.duracion;
            actualizar_maximo(costos_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.costos_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;
end;

```

```

    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;
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);
end;

```

```
    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;
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;
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;
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;
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;
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;
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.

```
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;
```

(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.

```
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;
end;
```

(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.

```
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;

```

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

```

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;
end;

```

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

```

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;
end;

```

```

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
    end;
end;

```

```
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;
          end;
        end;
    end;
end;
```

```

        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;
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;
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;
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);
    end;
end;

```



```

    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.

```
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;
```

(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.

```
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;
```

(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.

```

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;
end;

```

(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;
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;
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;
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;
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; tamano_corte=5;
type
    t_registro_celular=record
        version: int16;
        tamano: 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.tamano:=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;
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.tamano<=tamano_corte));
end;
procedure procesar_lista_celulares(lista_celulares: t_lista_celulares; var celulares_corte:
int16; var tamano_prom: real);
var
    version, celulares_version, celulares_total: int16;
    tamano_total: real;
begin
    celulares_total:=0; tamano_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;
            tamano_total:=tamano_total+lista_celulares^.ele.tamano;
            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;
    tamano_prom:=tamano_total/celulares_total;
end;
var
    lista_celulares: t_lista_celulares;
    celulares_corte: int16;
    tamano_prom: real;
begin
    randomize;
    lista_celulares:=nil;
    celulares_corte:=0;
    tamano_prom:=0;
    cargar_lista_celulares(lista_celulares);
    if (lista_celulares<>nil) then
    begin
        procesar_lista_celulares(lista_celulares,celulares_corte,tamano_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(tamano_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(tamano_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='Económica';
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_rol=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_rols(var vector_rols: t_vector_rols);
var
  i: t_rol;
begin
  for i:= rol_ini to rol_fin do
    vector_rols[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;
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;
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;
end;
procedure procesar_vector_usuarios(vector_usuarios: t_vector_usuarios; var lista_usuarios:
t_lista_usuarios; var vector_rol: t_vector_rol; 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_rol[vector_usuarios[i].rol]:=vector_rol[vector_usuarios[i].rol]+1;
                actualizar_maximos(vector_usuarios[i].dias,vector_usuarios[i].email,dias_max1,dias_max2,em
ail_max1,email_max2);
            end;
        end;
    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_rol(vector_rol: t_vector_rol);
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_rol[i]);
    end;
  end;
var
  vector_usuarios: t_vector_usuarios;
  vector_rol: t_vector_rol;
  lista_usuarios: t_lista_usuarios;
  email_max1, email_max2: string;
begin
  randomize;
  lista_usuarios:=nil;
  inicializar_vector_rol(vector_rol);
  email_max1:=''; email_max2:='';
  cargar_vector_usuarios(vector_usuarios);
  procesar_vector_usuarios(vector_usuarios,lista_usuarios,vector_rol,email_max1,email_max2);
  if (lista_usuarios<>nil) then
    imprimir_lista_usuarios(lista_usuarios);
    imprimir_vector_rol(vector_rol);
    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;
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;
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;
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;
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;
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;
end;

```

```

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;
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;
    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];
                    end;
                end;
            end;
        end;
    end;
end;

```

```

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

```

Ejercicio 15.

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

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

Al finalizar, el programa debe imprimir en pantalla la lista de alumnos para cada turno.

Nota: La distribución de alumnos debe ser lo más equitativa posible.

```
program TP6_E15;
{$codepage UTF8}
uses crt;
const
  dni_salida=0;
  nota_ini=1; nota_fin=10;
  turno_ini=1; turno_fin=5;
  nota_corte1=8; nota_corte2=5;
type
  t_nota=nota_ini..nota_fin;
  t_turno=turno_ini..turno_fin;
  t_registro_alumno=record
    dni: int16;
    apellido: string;
    nota: t_nota;
  end;
  t_lista_alumnos=^t_nodo_alumnos;
  t_nodo_alumnos=record
    ele: t_registro_alumno;
    sig: t_lista_alumnos;
  end;
  t_vector_alumnos=array[t_turno] of t_lista_alumnos;
procedure inicializar_vector_alumnos(var vector_alumnos: t_vector_alumnos);
var
  i: t_turno;
begin
  for i:= turno_ini to turno_fin do
    vector_alumnos[i]:=nil;
  end;
function random_string(length: int8): string;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
  end;
  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;
```



```

end;
procedure agregar_adelante_lista_alumnos(var lista_alumnos: t_lista_alumnos; registro_alumno:
t_registro_alumno);
var
    nuevo: t_registro_alumno;
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;
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;
    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;
    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));
  end;
  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;
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: 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.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^.e
le.monto_total);
            calcular_b(lista_empresas^.ele.monto_total,lista_empresas^.ele.empresa,monto_max,empresa
_max);
            calcular_c(lista_empresas^.ele.monto_total,empresas_corte);
        end;
        lista_empresas:=lista_empresas^.sig;
    end;
end;
var
    lista_empresas: t_lista_empresas;
    empresa_max, empresas_corte: int16;
begin
    lista_empresas:=nil;
    empresa_max:=0;
    empresas_corte:=0;
    cargar_lista_empresas(lista_empresas);
    procesar_lista_empresas(lista_empresas,empresa_max,empresas_corte);
    textcolor(green); write('El código de la empresa con mayor monto total invertido es ');
    textcolor(red); writeln(empresa_max);

```

```

    textcolor(green); write('La cantidad de empresas con inversiones de más de $');
textcolor(yellow); write(monto_corte:0:2); textcolor(green); write(' es '); textcolor(red);
write(empresas_corte);
end.

```

Ejercicio 2:

```

program TP6_E17b;
{$codepage UTF8}
uses crt;
const
    condicion_i='I'; condicion_r='R';
    autoeva_total=5;
    nota_incumple=-1;
    legajo_salida=-1;
    nota_corte=4;
    promedio_corte=6.5;
    nota_cero=0;
    nota_diez=10;
    presente_corte=0.75;
    alumnos_total=5000;
type
    t_registro_alumno=record
        legajo: int16;
        condicion: char;
        presente: int8;
        notas_cero: int8;
        notas_diez: int8;
        nota_total: int8;
    end;
    t_lista_alumnos=^t_nodo_alumnos;
    t_nodo_alumnos=record
        ele: t_registro_alumno;
        sig: t_lista_alumnos;
    end;
procedure leer_notas(var presente, notas_cero, notas_diez, nota_total: int8);
var
    i, nota: int8;
begin
    for i:= 1 to autoeva_total do
        begin
            textcolor(green); write('Introducir nota de autoevaluación ',i,' del alumno: ');
            textcolor(yellow); readln(nota);
            if ((nota<>nota_incumple) and (nota>=nota_corte)) then
                presente:=presente+1;
            if (nota=nota_cero) then
                notas_cero:=notas_cero+1;
            if (nota=nota_diez) then
                notas_diez:=notas_diez+1;
            if (nota<>nota_incumple) then
                nota_total:=nota_total+nota;
            end;
        end;
    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;
        end;
    end;

```



```
    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;
    end;
end;
procedure calcular_c(presente: int8; var alumnos_autoeva: int16);
begin
    if (presente=autoeva_total) then
        alumnos_autoeva:=alumnos_autoeva+1;
    end;
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;
end;
procedure calcular_e(notas_cero: int8; var alumnos_cero: int16);
begin
    if (notas_cero>=1) then
        alumnos_cero:=alumnos_cero+1;
    end;
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
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;
end;
procedure procesar_lista_alumnos(lista_alumnos: t_lista_alumnos; var ingresantes_parcial,
ingresantes_total, recursantes_parcial, recursantes_total, alumnos_autoeva, alumnos_corte,
alumnos_cero, legajo_diez_max1, legajo_diez_max2, legajo_cero_max1, legajo_cero_max2: int16);
var
  notas_diez_max1, notas_diez_max2, notas_cero_max1, notas_cero_max2: int8;
begin
  notas_diez_max1:=0; notas_diez_max2:=0;
  notas_cero_max1:=0; notas_cero_max2:=0;
  while (lista_alumnos<>nil) do
    begin
      calcular_ab(lista_alumnos^.ele.condicion,lista_alumnos^.ele.presente,ingresantes_total,ing
resantes_parcial,recursantes_total,recursantes_parcial);
      calcular_c(lista_alumnos^.ele.presente,alumnos_autoeva);
      calcular_d(lista_alumnos^.ele.nota_total,alumnos_corte);
      calcular_e(lista_alumnos^.ele.notas_cero,alumnos_cero);
      calcular_f(lista_alumnos^.ele.notas_diez,lista_alumnos^.ele.legajo,notas_diez_max1,notas_d
iez_max2,legajo_diez_max1,legajo_diez_max2);
      calcular_g(lista_alumnos^.ele.notas_cero,lista_alumnos^.ele.legajo,notas_cero_max1,notas_c
ero_max2,legajo_cero_max1,legajo_cero_max2);
      lista_alumnos:=lista_alumnos^.sig;
    end;
  end;
var
  lista_alumnos: t_lista_alumnos;
  ingresantes_parcial, ingresantes_total, recursantes_parcial, recursantes_total,
alumnos_autoeva, alumnos_corte, alumnos_cero, legajo_diez_max1, legajo_diez_max2,
legajo_cero_max1, legajo_cero_max2: int16;
begin
  lista_alumnos:=nil;
  ingresantes_parcial:=0; ingresantes_total:=0;
  recursantes_parcial:=0; recursantes_total:=0;
  alumnos_autoeva:=0;
  alumnos_corte:=0;
  alumnos_cero:=0;
  legajo_diez_max1:=0; legajo_diez_max2:=0;
  legajo_cero_max1:=0; legajo_cero_max2:=0;
  cargar_lista_alumnos(lista_alumnos);
  if (lista_alumnos<>nil) then
    begin
      procesar_lista_alumnos(lista_alumnos,ingresantes_parcial,ingresantes_total,recursantes_par
cial,recursantes_total,alumnos_autoeva,alumnos_corte,alumnos_cero,legajo_diez_max1,legajo_diez
_max2,legajo_cero_max1,legajo_cero_max2);
    end;
  end;
end;

```

```

    if (ingresantes_total>0) then
    begin
        textcolor(green); write('La cantidad de alumnos INGRESANTES en condiciones de rendir el
parcial y el porcentaje sobre el total de alumnos INGRESANTES son '); textcolor(red);
write(ingresantes_parcial); textcolor(green); write(' y '); textcolor(red);
write(ingresantes_parcial/ingresantes_total*100:0:2); textcolor(green); writeln('%',
respectivamente');
    end
    else
    begin
        textcolor(red); writeln('No hay alumnos INGRESANTES (I)');
    end;
    if (recursantes_total>0) then
    begin
        textcolor(green); write('La cantidad de alumnos RECURSANTES en condiciones de rendir el
parcial y el porcentaje sobre el total de alumnos RECURSANTES son '); textcolor(red);
write(recursantes_parcial); textcolor(green); write(' y '); textcolor(red);
write(recursantes_parcial/recursantes_total*100:0:2); textcolor(green); writeln('%',
respectivamente');
    end
    else
    begin
        textcolor(red); writeln('No hay alumnos RECURSANTES (R)');
    end;
    textcolor(green); write('La cantidad de alumnos que aprobaron todas las autoevaluaciones
es '); textcolor(red); writeln(alumnos_autoeva);
    textcolor(green); write('La cantidad de alumnos cuya nota promedio fue mayor a ');
textcolor(yellow); write(promedio_corte:0:2); textcolor(green); write(' puntos es ');
textcolor(red); writeln(alumnos_corte);
    textcolor(green); write('La cantidad de alumnos que obtuvieron cero puntos en, al menos,
una autoevaluación es '); textcolor(red); writeln(alumnos_cero);
    textcolor(green); write('Los legajos de los dos alumnos con mayor cantidad de
autoevaluaciones con nota 10 (diez) son '); textcolor(red); write(legajo_diez_max1);
textcolor(green); write(' y '); textcolor(red); writeln(legajo_diez_max2);
    textcolor(green); write('Los legajos de los dos alumnos con mayor cantidad de
autoevaluaciones con nota 0 (cero) son '); textcolor(red); write(legajo_cero_max1);
textcolor(green); write(' y '); textcolor(red); write(legajo_cero_max2);
    end
    else
    begin
        textcolor(red); write('No hay alumnos INGRESANTES (I) o RECURSANTES (R)');
    end;
end.

```

Ejercicio 3:

```

program TP6_E17c;
{$codepage UTF8}
uses crt;
const
    tanque_r='R'; tanque_c='C';
    tanque_salida='Z';
    alto_corte=1.40;
    volumen_corte=800.0;
type
    t_registro_tanque=record
        tanque: char;
        radio: real;
        alto: real;
        ancho: real;
        largo: real;
        volumen: real;
    end;
    t_lista_tanques=^t_nodo_tanques;
    t_nodo_tanques=record

```

```

    ele: t_registro_tanque;
    sig: t_lista_tanques;
end;
procedure leer_tanque(var registro_tanque: t_registro_tanque);
begin
    textcolor(green); write('Introducir tipo de tanque vendido (R o C) por el fabricante: ');
    textcolor(yellow); readln(registro_tanque.tanque);
    if (registro_tanque.tanque<>'tanque_salida') then
    begin
        if (registro_tanque.tanque='tanque_c') then
        begin
            textcolor(green); write('Introducir radio del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.radio);
            textcolor(green); write('Introducir alto del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.alto);
            registro_tanque.volumen:=pi*registro_tanque.radio*registro_tanque.radio*registro_tanque.
alto;
        end
        else
        begin
            textcolor(green); write('Introducir ancho del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.ancho);
            textcolor(green); write('Introducir largo del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.largo);
            textcolor(green); write('Introducir alto del tanque vendido ',registro_tanque.tanque,'
por el fabricante: ');
            textcolor(yellow); readln(registro_tanque.alto);
            registro_tanque.volumen:=registro_tanque.ancho*registro_tanque.largo*registro_tanque.alt
o;
        end;
    end;
end;
procedure agregar_adelante_lista_tanques(var lista_tanques: t_lista_tanques; registro_tanque:
t_registro_tanque);
var
    nuevo: t_registro_tanque;
begin
    new(nuevo);
    nuevo^.ele:=registro_tanque;
    nuevo^.sig:=lista_tanques;
    lista_tanques:=nuevo;
end;
procedure cargar_lista_tanques(var lista_tanques: t_lista_tanques);
var
    registro_tanque: t_registro_tanque;
begin
    leer_tanque(registro_tanque);
    while (registro_tanque.tanque<>'tanque_salida') do
    begin
        agregar_adelante_lista_tanques(lista_tanques,registro_tanque);
        leer_tanque(registro_tanque);
    end;
end;
procedure calcular_a(volumen: real; var volumen_max1, volumen_max2: real);
begin
    if (volumen>volumen_max1) then
    begin
        volumen_max2:=volumen_max1;
        volumen_max1:=volumen;
    end
    else
    if (volumen>volumen_max2) then

```

```

    volumen_max2:=volumen;
end;
procedure calcular_bc(tanque: char; volumen: real; var volumen_total_c, volumen_total_r: real;
var tanques_c, tanques_r: int16);
begin
    if (tanque=tanque_c) then
    begin
        volumen_total_c:=volumen_total_c+volumen;
        tanques_c:=tanques_c+1;
    end
    else
    begin
        volumen_total_r:=volumen_total_r+volumen;
        tanques_r:=tanques_r+1;
    end;
end;
procedure calcular_d(alto: real; var tanques_corte_alto: int16);
begin
    if (alto<alto_corte) then
        tanques_corte_alto:=tanques_corte_alto+1;
end;
procedure calcular_e(volumen: real; var tanques_corte_volumen: int16);
begin
    if (volumen<volumen_corte) then
        tanques_corte_volumen:=tanques_corte_volumen+1;
end;
procedure procesar_lista_tanques(lista_tanques: t_lista_tanques; var volumen_max1,
volumen_max2, volumen_total_c, volumen_total_r: real; var tanques_c, tanques_r,
tanques_corte_alto, tanques_corte_volumen: int16);
begin
    while (lista_tanques<>nil) do
    begin
        calcular_a(lista_tanques^.ele.volumen,volumen_max1,volumen_max2);
        calcular_bc(lista_tanques^.ele.tanque,lista_tanques^.ele.volumen,volumen_total_c,volumen_t
otal_r,tanques_c,tanques_r);
        calcular_d(lista_tanques^.ele.alto,tanques_corte_alto);
        calcular_e(lista_tanques^.ele.volumen,tanques_corte_volumen);
        lista_tanques:=lista_tanques^.sig;
    end;
end;
var
    lista_tanques: t_lista_tanques;
    tanques_c, tanques_r, tanques_corte_alto, tanques_corte_volumen: int16;
    volumen_max1, volumen_max2, volumen_total_c, volumen_total_r: real;
begin
    lista_tanques:=nil;
    volumen_max1:=0; volumen_max2:=0;
    tanques_c:=0; volumen_total_c:=0;
    tanques_r:=0; volumen_total_r:=0;
    tanques_corte_alto:=0;
    tanques_corte_volumen:=0;
    cargar_lista_tanques(lista_tanques);
    if (lista_tanques<>nil) then
    begin
        procesar_lista_tanques(lista_tanques,volumen_max1,volumen_max2,volumen_total_c,volumen_tot
al_r,tanques_c,tanques_r,tanques_corte_alto,tanques_corte_volumen);
        textcolor(green); write('El volumen de los mayores tanques vendidos es '); textcolor(red);
write(volumen_max1:0:2); textcolor(green); write(' y '); textcolor(red);
writeln(volumen_max2:0:2);
        if (tanques_c>0) then
        begin
            textcolor(green); write('El volumen promedio de todos los tanques cilíndricos (C)
vendidos es '); textcolor(red); writeln(volumen_total_c/tanques_c:0:2);
        end
        else
        begin

```

```
    textcolor(red); writeln('No hay tanques cilíndricos (C) vendidos');
end;
if (tanques_r>0) then
begin
    textcolor(green); write('El volumen promedio de todos los tanques rectangulares (R)
vendidos es '); textcolor(red); writeln(volumen_total_r/tanques_r:0:2);
end
else
begin
    textcolor(red); writeln('No hay tanques rectangulares (R) vendidos');
end;
    textcolor(green); write('La cantidad de tanques cuyo alto es menor a ');
textcolor(yellow); write(alto_corte:0:2); textcolor(green); write(' metros es ');
textcolor(red); writeln(tanques_corte_alto);
    textcolor(green); write('La cantidad de tanques cuyo volumen es menor a ');
textcolor(yellow); write(volumen_corte:0:2); textcolor(green); write(' metros cúbicos es ');
textcolor(red); write(tanques_corte_volumen);
end
else
begin
    textcolor(red); write('No hay tanques cilíndricos (C) o rectangulares (R) vendidos');
end;
end.
```