Ejercicio:
Realice un programa que lea para 50 restaurantes el puntaje que un evaluador le da cada día de la semana. Se pide informar para cada restaurante el día de la semana que obtuvo la nota máxima.

```
Program uno;
Type
notas = 0..10;
dias = 1..5;
Var
notaRes, max: notas;
diaMax:dias;
i:integer; d:dias;
```

```
Begin
for i:= 1 to 50 do begin
    max:= 0;
    for d:= 1 to 5 do begin
        read (notaRes);
        if (notaRes>=max) then begin
            diaMax:= d;
            max:= notaRes;
        end;
    end;

case diaMax of
1: write ("La nota máxima fue lunes");
2: write ("La nota máxima fue martes");
……. –los días que faltan--
end;
end;
End.
```

Realice un programa que lea números enteros y para cada número par, indique su dígito mayor. El programa termina cuando se lee el número 25.

```
Prgram uno;

function espar (n:integer): boolean;
Begin
   espar:= (n MOD 2 = 0);
end;

Procedure maximo (n:integer; var
max:integer);
Var
   dig:integer;
Begin
   max:= -1;
   while (n <> 0) do
   begin
      dig:= n MOD 10;
      if (dig > max) then max:= dig;
         n:= n DIV 10;
   end;
end;
```

```
Var
num,max:integer;

Begin
   read (num);
   while (num <> 25) do
   begin
      if (espar (num) ) then begin
         maximo (num, max);
         Write ('El dígito máximo de',
          num 'es', max);
      end;
      read (num);
   end;
end.
```

REGISTRO - OPERACIÓN - CARGAR REGISTRO

```
Procedure leer (var l:lugar);
begin
    read (l.nombre);
    read(l.provincia);
    read(l.cantV);
    read (l.descripcion);
end;
```

REGISTRO - OPERACIÓN - IMPRIMIR REGISTRO

```
Procedure imprimir(l:lugar);
begin
    write(l.nombre);
    write (l.provincia);
    write(l.cantV);
    write (l.descripcion);
end;
```

REGISTRO - OPERACIÓN - COMPARACION REGISTRO

```
Function iguales (l1,l2: lugar): boolean;
Begin
    if ( (l1.nombre=l2.nombre)and (l1.provincia = l2.provincia) and (l1.cantV=l2.cantV) and
            (l1.descripcion = l2.descripcion)) then
                    iguales:= true
    else
                    iguales:= false;
End;
```

Se pide realizar un programa que lea lugares hasta leer uno con nombre “XXX”. Al finalizar informar la cantidad de lugares de la provincia “Buenos Aires”, y la cantidad de lugares que tuvieron entre 1000 y 2000 visitantes (inclusive).

```
Program uno;

Type
lugar= record
…
end;

Procedure leer(var lu:lugar);
begin
   …
end;

Var
   cantBs,cantVisi:integer;
   l:lugar;
```

```
Begin
   leer(l); cantBs:=0;
   cantVisi:=0;
   while( l.nombre <> ”XXX”) do
   begin
      if (l.provincia= “Buenos Aires”) then
         cantBs:= cantBs+1;
      if (l.cantV>=1000) and (l.cantV<=2000) then
         cantVisi:= cantVisi+1;
      leer(l);
   end;
   write (cantBs,cantVisi);
end.
```

Realice un programa que lea lugares hasta leer uno de la provincia XXX e informe la nombre del lugar con más visitantes y la cantidad de lugares que fueron descubiertos en el verano del año 2005 (21/12/2004-21/3/2005).

```
Program dos;

Fucntion esDescubierto (l:lugar): boolean;
Begin
if ( ( (l.fec.dia>= 21) and (l.fec.mes = 12) and
(a.fec.año = 2004)) OR
( ( (l.fec.dia< 21) and (l.fec.mes = 3) and
(l.fec.año = 2005)) OR
( ( (l.fec.mes = 1) OR (l.fec.mes = 2) and
(l.fec.año = 2005)) then
   esDescubierto:= true

else
   esDescubierto:= false;

End;

Var
   l:lugar;
   cantDesc:integer;
   max:integer;
   maxNombre:string;
```

```
Begin
   leer(l);
   max:= -1;
   cantDesc:=0;
   while (l.nombre<>’XXX’) do
   begin
      if (esDescubierto(l) ) then
         cantDesc:= cantDesc + 1;
      if (l.cantV > max) then
      begin
         max:= l.cantV;
         maxNombre:=l.nombre;
      end;
      leer(l);
   end;
   write(maxNombre); write (cantDesc);
End.
```

Se pide realizar un programa que lea lugares el lugar con mayor cantidad de visitantes y la provincia con menor cantidad de lugares turísticos. La lectura termina cuando llega el lugar de nombre XXX. La lectura es ordenada por provincia.

```
Var

l:lugar; sum,max,min:integer;
actual,nomMax,nomMin:string;

Begin
   leer(l);
   max:= -1;
   while (l.nombre <> 'XXX') do
   begin
      actual:= l.provincia; sum:= 0;
      while (actual = l.provincia) do
      begin
         if (l.cantV >= max) then
         begin
            max:= l.cantV;
            nomMax:= L.nombre;
         end;
         sum:= sum + 1;
         leer(l);
      end;
      if (sum < min) then
      begin
         min:= sum ;
         nomMin:= actual;
      end;
      {END del While de actual}
   end;
   write (nomMin);
   write (nomMax);
end.
```

VECTORES - RECORRIDO TOTAL

Realice un programa que llene un vector de 7 elementos enteros positivos y luego informe la cantidad de números que son pares.

```
Program uno;

const
   tam = 7;

type
   numeros= array [1..tam] of integer;

var
   VN: numeros; cantP:integer;

Procedure llenarNumeros (var num:numeros);
var i:integer;
begin
   for i:= 1 to tam do
      read (num[i]);
end;
```

```
function pares (num:numeros):integer;
var
i:integer;
cant:integer;
begin
   cant:=0;
   for i:= 1 to tam do
      if (num[i] MOD 2 = 0) then
         cant:= cant + 1;
         pares:= cant;
end;

begin
   llenarNumeros (VN);
   cantP:= pares(VN);
   write ("La cantidad de números pares es:",
cantP);
end.
```

VECTORES - RECORRIDO PARCIAL

Informe la posición del primer número que es par.

```
function posicion(num:numeros):integer;
var
   i,pos:integer;
   par:boolean;

begin
   pos:=1; par:= false;
   while (not par) do
      if (num[pos] MOD 2 = 0) then
         par:= true
      else
         pos: = pos + 1;
   posicion:= pos;
end;
```

VECTORES - DIMENSION LOGICA Y FISICA

Devolver la cantidad de elementos (dimensión lógica) cargados en el arreglo (tam es un Const).

```
Procedure llenarNumeros(var n:numeros;var cant:integer);
var
   num:integer;
begin
   cant:=0;
   read(num);
   while (num <> 99) and (cant <= tam) do
   begin
      cant:= cant + 1;
      n[cant]:= num;
      read(num);
   end;
end;
```

Encontrar el máximo valor de un arreglo.

```
function máximo (n:numeros, cantidad:integer):integer;
var
   i,max:integer;
begin
   max:= -9999;
   for i:= 1 to cantidad do
      if (n[i] > max)then
         max:= n[i];
         maximo:= max;
end;
```

ARREGLOS - AGREGAR

Dado un arreglo de números enteros (200 elementos como máximo) realice un programa que lea un número y lo agregue en el arreglo. Agregar siempre es al final.

```
Program uno;
const
   tam = 200;
type
   numeros= array [1..tam] of integer;
var
   VN: numeros; dim,valor:integer;
   ok:boolean;
begin
   read (valor);
   llenarNumeros (VN,dim); //ya está hecho
   agregar(VN, dim, valor, ok);
end.
```

```
Procedure agregar (var vec:numeros; var dimL:integer;
num:integer; var ok:boolean);

begin
   ok:= false;
   if ( (dimL + 1) <= tam) then
   begin
      vec[dimL+1]:= num;
      dimL:= dimL + 1;
      ok:= true;
   end;
end;
```

ARREGLOS - INSERTAR

Dado un arreglo de números enteros (a lo sumo 200 números) realice un programa que lea un número y una posición e inserte el número en el arreglo en la posición leída.

```
Program uno;
const
   tam = 200;
type
   numeros= array [1..tam] of integer;
var
   VN: numeros; dim, pos, valor :integer;
   ok :boolean;
begin
   read (valor); read (pos);
   llenarNumeros (VN, dim); //ya está hecho
   insertar( VN, dim, valor, pos,ok);
end.
```

```
Procedure insertar (var vec:numeros; var dimL:integer;
num:integer, pos:integer; var ok:boolean);
begin
   ok:=false;
   if ( (dimL + 1) <= tam) and (pos=>1) and (pos<=dimL))
   then begin
      for i:= dimL downto pos do
         vec[i+1]:= vec[i];
      vec[pos]:= num;
      dimL:= dimL + 1;
      ok:= true;
   end;
end;
```

ARREGLOS - INSERTAR

Supongamos que se quiere insertar un nombre a un vector ordenado de nombres, de manera que siga ordenado el arreglo.

```
Procedure INSERTAR (var nombres: arreglo; var dimL: integer; elem: nom; var exito: boolean);
var i : integer;
begin
   if (dimF > dimL) then begin
      exito := true;
      i:= 1;
         { Buscar posición a insertar}
      while (i<=dimL) and (nombres[ i ] < elem) do
         i:= I +1;
      for j:= dimL downto i do
         nombres [ j +1 ] := nombres [ j ] ;
      nombres [ i ] := elem;
      { Actualizar cantidad de elementos }
      dimL := dimL + 1;
   end
   else éxito := false;
end;
```

Dado un arreglo de 200 caracteres como máximo se pide generar un nuevo arreglo que contenga todas las vocales del arreglo original.

```
Program uno;
const
   tam = 200;
type
   letras= array [1..tam] of char;
var
   a1,a2: letras; dim1,dim2:integer;
begin
   llenarArreglo (a1,dim1); //ya está hecho
   procesar(a1,dim1, a2,dim2);
end.

Function esVocal(letra:char): boolean;
Var
i:integer;
Begin
if ((letra='a') or (letra='e') or (letra='i') or
(letra='o') or (letra='u')) then
esVocal:= true
else esVocal:= false;
end;
```

```
Procedure procesar(a1:letras; dim1:integer;
var a2:letras,var dim2:integer);
Var
   i:integer;
Begin
   dim2:=0;
   for i:= 1 to dim1 do begin
      if (esVocal(a1[i])) then
      begin
         a2[dim2+1]:= a1[i];
         dim2:= dim2 + 1;
      end;
   end;
end;
```

ARREGLOS - BORRAR

Dado un arreglo de números enteros (a lo sumo 200 elementos) realice un programa que lea una posición y elimine del arreglo el número ubicado en la posición leída.

```
Program uno;
const
   tam = 200;
type
   numeros= array [1..tam] of integer;
var
   VN: numeros; dim,pos,valor:integer;
   ok:boolean;
begin
   read (valor); read (pos;)
   llenarNumeros (VN,dim); //ya está hecho
   eliminar(VN, dim, pos, ok);
end.
```

```
Procedure borrar ( var vec :numeros; var dimL :integer;
var ok :boolean; pos :integer);
begin
   ok:=false;
   if ( (pos => 1) and (pos<=dimL) ) then
   begin
      for i:= pos to (dimL-1) do
         vec[i]:= vec[i+1];
      dimL:= dimL - 1;
      ok:= true;
   end;
end;
```

```
Procedure BORRARELEM
(var nombres: vector; var dimL: integer; elem : nom; var exito: boolean);
var i , j: integer;
begin
i:= 1
while (i<=dimL) and (nombres[ i ] <> elem) do
i:= i + 1
if (i> dimL) then exito := false
else begin
for j:= i to dimL-1 do
nombres [ j ] := nombres [ j +1];
dimL := dimL -1;
exito:= true;
end
```

BUSCAR - ARREGLOS

```
Function Buscar ( x: TipoElem; v:vector;
dimL: Indice) : Indice;
var pos:Indice;
Begin
pos:=1;
while (pos <= dimL) and (x <> v[pos]) do
pos:=pos+1;
if (pos > dimL) then pos:=0;
Buscar := pos;
end;
```

BUSCAR - ARREGLOS DESORDENADOS

Dado un arreglo de números enteros (a lo sumo 200) realice un programa que lea un número y devuelva verdadero si el número se encuentra y falso en caso contrario.

```
Program uno;
const
   tam = 200;
type
   numeros= array [1..tam] of integer;
var
   VN: numeros; dim, num, valor :integer;
begin
   read (num);
   llenarNumeros (VN,dim); //ya está hecho
   if (buscar(VN,dim,num)) then
      write (num, “Esta en el arreglo”)
   else (num,”No se encuentra en el arreglo”);
end.
```

```
Function buscar (vec:numeros; dimL, num:integer):boolean;
Var ok:boolean; pos:integer;
begin
   ok:=false; pos:= 1;
   while ( (pos <=dimL) and (not ok) ) do
   begin
      if (vec[pos] = num) then
         ok:=true
      else
      pos:= pos + 1;
   end;
   buscar:= ok;
end;
```

BUSCAR - ARREGLOS ORDENADOS

```
Program uno;
const
   tam = 200;
type
   numeros= array [1..tam] of integer;
var
   VN: numeros; dim, num, valor :integer;
begin
   read (num);
   llenarNumeros (VN,dim); //ya está hecho
   if (buscar(VN,dim,num)) then
      write (num,“Esta en el     arreglo”)
   else (num,”No se encuentra en el arreglo”);
end.
```

```
Function buscar (vec:numeros; dimL, num:integer):boolean;
Var pos:integer; ok:boolean
Begin
    pos:= 1;
    while ( (pos <=dimL) and (vec[pos] < num) ) do
       pos:= pos + 1;
       if (pos<=dimL) and (vec[pos] = num) then
          buscar := true
       else buscar:= false;
end;
```

BUSCAR - ARREGLOS ORDENADOS - SECUENCIAL

```
Function BuscoOrdenado ( x: integer; v:Vector; dimL: Indice): Indice;
var pos : Indice;
begin
   pos:=1;
   while (pos<=dimL) and (x<v[pos]) do
      pos:=pos+1;
   if ( pos > dimL ) or (v [pos] > x) then pos:=0;
BuscoOrdenado:= pos;
end;
```

BUSCAR - ARREGLOS ORDENADOS - BUSQUEDA DICOTOMICA

```
Procedure BusquedaBin ( var v: Vector; var j: Indice; dimL: Indice, x : TipoElem) ;
Var pri, ult, medio : Indice ;
Begin
   j :=0 ;
   pri:= 1 ;
   ult:= dimL;
   medio := (pri + ult ) div 2 ;
   While ( pri < = ult ) and ( x <> v [medio]) do begin
      If ( x < v [ medio ] ) then
         ult:= medio -1 ;
      else pri:= medio+1 ;
      medio := ( pri + ult ) div 2 ;
   end ;
   If pri < = ult Then
      j := medio
   Else j := 0 ;
```

ARREGLOS - ORDENACION - METODO DE SELECCION

```
Procedure Ordenar ( var v: tVector; dimLog: indice );
var i, j, p: indice;
item : tipoElem;
begin
for i:=1 to dimLog-1 do begin {busca el mínimo v[p] entre v[i], ..., v[N] }
p := i;
for j := i+1 to dimLog do
if v[ j ] < v[ p ] then p:=j;
{intercambia v[i] y v[p] }
item := v[ p ];
v[ p ] := v[ i ];
v[ i ] := item;
end;
end;
```

La Facultad de Informática está procesando la información de sus alumnos de primer año, para esto lee una sucesión de datos de a lo sumo 800 alumnos. De cada alumno se conoce nombre, código de materia que está cursando (1..1000) y promedio del alumno en la carrera (suponga que el alumno cursa una sola materia). La lectura termina con el alumno de nombre 'ZZZ'. Además la Facultad dispone de una tabla en la que para código de materia se tiene guardado el nombre de la misma y el año en que empezó a dictarse.

Una vez finalizada la lectura de los alumnos se pide informar el nombre de la materia que más alumnos cursan.

```
Program uno;
Const
   tamA=800; tamC=1000;
Type
   alumno=record
      nombre:string;
      cod:integer;
      prom:real;
      end;
   materia=record
      nombre:string;
      año:integer;
   end;
alumnos= array[1..tamA] of alumno;
materias = array[1..tamC]of material;
contador= array[1..tamC]of integer;

Modulos…

Var
   cont: contador; mat:materias; alu:alumnos; dimL:integer;
   nombreMat,nombreAlu:string;
Begin
   inicializador(cont);
   llenarMaterias(mat); //ya está cargada
   llenarAlumnos (alu,dimL);
   procesar(alu,dimL,mat,cont,nombreAlu,nombreMat);
   write (“El alumno de major poromedio es”, nombreAlu);
   write (“La materia mas cursada ea”, nombreMat);
End.
```

```
Procedure inicializador (var a:contador);
Var i:integer;
Begin
   for i:= 1 to tamC do
      a[i]:= 0;
End;

Procedure llenarAlumnos (var a:alumnus; var dim:int);
Var
   Al:alumno;
Begin
   dim:= 0;
   leer(Al);
   while (Al.nombre<>”ZZZ”) and (dim<=tamA) do
   begin
      dim:= dim+1;
      a[dim]:= Al;
      leer(Al);
   End;
End;
Procedure leer(var a:alumno);
Begin
   Read (a.nombre);
   read (A.cod);
   read(A.prom);
End;
Procedure procesar ( a:alumnos; dim:int; mat:materias,
cont:Contador; var nomA,nomM:string);
Var
i:integer; max:real;
Al:alumno;
Begin
   max:= -1;
   for i:= 1 to dim do begin
      mejorProm(a[i],max,nomA);
      actualizar (a[i],cont);
   end;
   maximo (cont,mat,nomM);
End;
```

```
Procedure majoerProm(a:alumno; var promMax:real;
var nomA:string);
Begin
   If (a.prom > promMax) then begin
      promMax:= a.prom;
      nomA:= a.nombre;
   end;
end;
Procedure actualizar( a:alumno; cont:Contador);
Begin
   cont[a.cod] := cont[a.cod] + 1;
End;
```

```
Procedure maximo (cont: Contador; mat:materias;
nom:string);
Var i:integer; max:integer; pos:integer;
Begin
   for i:= 1 to tamM do begin
      if (cont[i] > max) then begin
         max:= cont[i];
      pos:= i;
   end;
   nom:= mat[pos].nombre;
end;
```

Se necesita conocer la cantidad de veces que aparece la temperatura con valor 10 en un vector de temperaturas. ¿Memoria utilizada por el módulo? ¿Tiempo empleado por el módulo?

```
Type
temperaturas = array [1..30] of real;

Function contar ( tem:temperaturas): integer;
Var i: 1..30; can10 : integer;
begin
can10 := 0;    {1}

{recorrido total del vector}

For i := 1 to 30 do     {2}
    If ( tem [i] = 10 ) then     {3}
        can10 := can10 + 1;     {4}

contar := can10;     {5}
end.
```

Cálculo Teórico del tiempo de ejecución:

- Las líneas {1} y {5} cuentan una unidad cada una, entonces tenemos 2
- La línea {3} evalúa una condición, cuenta 1 unidad y la línea {4} cuenta 2 unidades. Por la tanto, cada vez que se ejecutan utilizan 3 unidades -> 3 * 30
- La línea {2} tiene una inicialización, testeo de i <=30 e incremento de i, entonces 1 de la asignación, más 31 para todos los test y 30*2 para el incremento, entonces 1+ 31 + 60 = 92

Total = 2 + 90 + 92 (como máximo!!!)

{1} -> 1 asignación = 1 unidad de tiempo
{4} → 1 asignación = 1 unidad de tiempo
{3} → (1 asignación + 1 suma) * n = 2 * n
{2} → una asignación (i:=1) + testeos de i<=n + incremento de i (i:= i+1)

→ 1 + (n + 1) + 2*n = 3*n + 2

T(N) ==> total = 1 + 1 + 2*n + 3*n + 2 = 5*n + 4 ➔ O(n)