

Introducción a la Programación I – Práctico 7 (Arreglos)

```
Program elp7;  
{carga caracteres en un arreglo hasta que aparezca un * o hasta que se llene}  
Type TArreglo=array[1..50] of char;  
  
procedure cargar (var datos:TArreglo;dato1:char;ubicacion:integer);  
{ubica cada caracter en el arreglo}  
begin  
  datos[ubicacion]:=dato1;  
end;  
  
var arreglo:TArreglo;caracter:char;lugar:integer;  
begin  
  caracter:='a';  
  for lugar:=1 to 50 do arreglo[lugar]:='v';  
  lugar:=0;  
  while ((lugar) <50) and ((caracter) <> '*') do  
    begin  
      writeln('ingrese un caracter ');  
      readln(caracter);  
      lugar:= lugar+1;  
      cargar(arreglo, caracter, lugar);  
    end;  
  write('Elementos del arreglo: ');  
  for lugar:=1 to lugar do  
    begin  
      write(arreglo[lugar]);  
      write(', ');  
    end;  
end.  
  
/*****/
```

```
Program e2p7;  
{imprime la suma de un arreglo de dimension 100}  
Const max=100;  
Type TDatos=array[1..max] of real;  
  
Function suma (sumnum:TDatos):real;  
{suma todos los elemento de un arreglo}  
var indice:integer;sumaux:real;  
begin  
    sumaux:=0;  
    for indice:=1 to max do  
        sumaux:=sumaux+sumnum[indice];  
    suma:=sumaux;  
end;  
  
var datos:TDatos;lugar:integer;numero:real;  
begin  
    for lugar:=1 to max do  
        begin  
            read(numero);  
            datos[lugar]:=numero;  
        end;  
    writeln(suma(datos));  
end.  
  
/*****/
```

```
program e3p7;

const Max = 50;

type Arreglo1 = Array[1..Max] of Char;

function existeelemento (Ar1:Arreglo1; elemdato:Char):Boolean;

var Pos:Integer;

begin
  Pos:= 1;
  Existeelemento:= False;
  While (Pos <= Max) and
    (not existeelemento) do
    Begin
      If Ar1[Pos] = Elemdato Then
        Existeelemento:= True;
        Pos:= Pos + 1
      End;
    End;

var Miarreglo: Arreglo1;
  DADO: Char;
  Indice: Integer;

begin
  For indice:= 1 To Max do
    Miarreglo[Indice] := '1';
  Miarreglo[49] := 'z';
  Write('Ingrese el elemento que desea buscar en el arreglo: ');
  Readln(Dado);
  If existeelemento(Miarreglo, DADO) Then
    Writeln('El elemento se encuentra en el arreglo')
  Else
    Writeln('El elemento NO se ha encontrado en el arreglo');
end.

/*****/
```

```
Program e4p7;  
{ingresa caracteres en un arreglo y este continua ordenado, despues  
imprime el arreglo}  
const  
  min=1;  
  max=10;  
Type Tordenado=array [min..max] of char;  
  
Procedure insertar(var caracteres:Tordenado;dato:char;ubicacion:integer);  
{inserta un caracter en un arreglo y este continua ordenado}  
var numero:integer;  
begin  
  while (ubicacion > 0) and  
    ((dato)< caracteres[ubicacion]) do  
    begin  
      numero:=ubicacion + 1;  
      caracteres[numero]:=caracteres[ubicacion];  
      ubicacion:=ubicacion-1;  
    end;  
  numero:=ubicacion + 1;  
  caracteres[numero]:=dato;  
end;  
  
var listacaracteres:Tordenado;lugar:integer;caracter:char;  
begin  
  writeln('ingrese un caracter');  
  readln(listacaracteres[min]);  
  lugar:=min;  
  while lugar < max do  
    begin  
      writeln('ingrese un caracter');  
      readln(caracter);  
      insertar(listacaracteres,caracter,lugar);  
      lugar:=lugar+1;  
    end;  
  for lugar:=min to lugar do  
    begin  
      write(listacaracteres[lugar]);  
      write(', ');  
    end;  
end.  
  
/*****/
```

```
Program e6p7;  
{dado un arreglo de N caracteres, imprime un arreglo de N-1 valores  
booleanos, es true si el primer valor de cada par, es menor o igual al  
siguiente, sino es false}
```

```
const
```

```
  min=1;  
  max=10;
```

```
Type Ttof=array[min..(max-1)] of boolean;
```

```
Type Tnumeros=array[min..max] of integer;
```

```
Procedure cargar(var listado:Tnumeros);  
{ingresa Numeros hasta llenar el arreglo}
```

```
var simbolo,numero:integer;
```

```
begin
```

```
  for numero:=min to max do
```

```
    begin
```

```
      read(simbolo);
```

```
      listado[numero]:=simbolo;
```

```
    end;
```

```
end;
```

```
Function compara(primero:integer; segundo:integer):boolean;  
{devuelve true si el si el primer valor es menor o igual al  
segundo, y false, si es mayor que el segundo}
```

```
begin
```

```
  if (primero<=segundo) then
```

```
    compara:=true
```

```
  else
```

```
    compara:=false;
```

```
end;
```

```
Procedure comparar(listanumeros:Tnumeros;var listacomparada:Ttof);
```

```
{compara una lista de numeros, y devuelve un arreglo con los resultados}
```

```
var numero1,numero2,lugar:integer;
```

```
begin
```

```
  lugar:=min;
```

```
  numero1:=listanumeros[min];
```

```
  numero2:=listanumeros[min+1];
```

```
  while (lugar<max) do
```

```
    begin
```

```
      listacomparada[lugar]:=compara(numero1,numero2);
```

```
      lugar:=lugar+1;
```

```
      if lugar<max then
```

```
        begin
```

```
          numero1:=numero2;
```

```
          numero2:=listanumeros[lugar+1];
```

```
        end;
```

```
    end;
```

```
end;
```

```
var listavalores:Tnumeros; listatof:Ttof; espacio:integer;
begin
  cargar(listavalores);
  comparar(listavalores,listatof);
  for espacio:=min to (max-1) do
    writeln(listatof[espacio]);
end.

/*****/
```

```
Program e7p7;  
{imprime por pantalla true o false, dependiendo, si un arreglo  
ingresado por el usuario es capicua o no}
```

```
const
```

```
    min=1;  
    max=10;
```

```
Type Tnumeros=array[min..max] of integer;
```

```
Procedure cargar(var listado:Tnumeros);  
{ingresa Numeros hasta llenar el arreglo}
```

```
var indice,numero:integer;
```

```
begin
```

```
    for indice:=min to max do
```

```
        begin
```

```
            read(numero);
```

```
            listado[indice]:=numero;
```

```
        end;
```

```
end;
```

```
Function compara(listanumeros:Tnumeros):boolean;
```

```
{compara es true si un arreglo es capicua, sino false}
```

```
var numero1,numero2:integer;tof:boolean;
```

```
begin
```

```
    numero1:=min;
```

```
    numero2:=max;
```

```
    tof:=true;
```

```
    while ((numero1<numero2) and
```

```
        (tof = true)) do
```

```
        if (listanumeros[numero1]=listanumeros[numero2]) then
```

```
            begin
```

```
                numero1:=numero1+1;
```

```
                numero2:=numero2-1;
```

```
            end
```

```
        else
```

```
            tof:=false;
```

```
        compara:=tof;
```

```
end;
```

```
var listavalores:Tnumeros;
```

```
begin
```

```
    cargar(listavalores);
```

```
    writeln(compara(listavalores));
```

```
end.
```

```
/*****/
```

```
Program e8p7;  
{invierte el arreglo ingresado por el usuario, y luego lo imprime por pantalla}
```

```
const
```

```
    min=1;  
    max=10;
```

```
Type Tnumeros=array[min..max] of integer;
```

```
Procedure cargar(var listado:Tnumeros);  
{ingresa Numeros hasta llenar el arreglo}
```

```
var indice,numero:integer;
```

```
begin
```

```
    for indice:=min to max do
```

```
        begin
```

```
            read(numero);
```

```
            listado[indice]:=numero;
```

```
        end;
```

```
end;
```

```
Procedure invertir(var listanumeros:Tnumeros);
```

```
{dado un arreglo, lo invierte y lo devuelve invertido}
```

```
var lugar1,lugar2,auxnum:integer;
```

```
begin
```

```
    lugar1:=min;
```

```
    lugar2:=max;
```

```
    while (lugar1 < lugar2) do
```

```
        begin
```

```
            auxnum:=listanumeros[lugar1];
```

```
            listanumeros[lugar1]:=listanumeros[lugar2];
```

```
            listanumeros[lugar2]:=auxnum;
```

```
            lugar1:=lugar1+1;
```

```
            lugar2:=lugar2-1;
```

```
        end;
```

```
end;
```

```
var listavalores:Tnumeros; espacio:integer;
```

```
begin
```

```
    cargar(listavalores);
```

```
    invertir(listavalores);
```

```
    for espacio:=min to max do
```

```
        write(listavalores[espacio]);
```

```
end.
```

```
/*****/
```

```
Program e9p7;
```

```
{ordena el arreglo ingresado por el usuario, y luego lo imprime por pantalla}
```

```
const
    min=1;
    max=10;

Type Tnumeros=array[min..max] of integer;

Procedure cargar(var listado:Tnumeros);
    {ingresa Numeros hasta llenar el arreglo}
var indice,numero:integer;
begin
    for indice:=min to max do
        begin
            read(numero);
            listado[indice]:=numero;
        end;
    end;

Procedure ordenar(var listanumeros:Tnumeros);
    {dado un arreglo, lo ordena y lo devuelve ordenado}
var lugar1,lugar2,auxnum,otronum:integer;
begin
    otronum:=min;
    lugar1:=min;
    lugar2:=lugar1+1;
    while (otronum < (max-1)) do
        begin
            while (lugar2 <= max) do
                begin
                    if(listanumeros[lugar1]>listanumeros[lugar2]) then
                        begin
                            auxnum:=listanumeros[lugar1];
                            listanumeros[lugar1]:=listanumeros[lugar2];
                            listanumeros[lugar2]:=auxnum;
                        end;
                    lugar1:=lugar2;
                    lugar2:=lugar2+1;
                end;
            lugar1:=min;
            lugar2:=lugar1+1;
            otronum:=otronum+1;
        end;
    end;

var listavalores:Tnumeros; espacio:integer;
begin
    cargar(listavalores);
    ordenar(listavalores);
    for espacio:=min to max do
        write(listavalores[espacio]);
    end.
```