**Trabajo Práctico N° 2:**

**Módulo Imperativo (Recursión).**

**Ejercicio 1.**

*Implementar un programa que invoque a los siguientes módulos:*

**(a)** *Un módulo recursivo que permita leer una secuencia de caracteres terminada en punto, los almacene en un vector con dimensión física igual a 10 y retorne el vector.*

**(b)** *Un módulo que reciba el vector generado en (a) e imprima el contenido del vector.*

**(c)** *Un módulo recursivo que reciba el vector generado en (a) e imprima el contenido del vector.*

**(d)** *Un módulo recursivo que permita leer una secuencia de caracteres terminada en punto y retorne la cantidad de caracteres leídos. El programa debe informar el valor retornado.*

**(e)** *Un módulo recursivo que permita leer una secuencia de caracteres terminada en punto y retorne una lista con los caracteres leídos.*

**(f)** *Un módulo recursivo que reciba la lista generada en (e) e imprima los valores de la lista en el mismo orden que están almacenados.*

**(g)** *Un módulo recursivo que reciba la lista generada en (e) e imprima los valores de la lista en orden inverso al que están almacenados.*

program TP2\_E1;

*{$codepage UTF8}*

uses crt;

const

  char\_salida='.';

  dimF=10;

type

  t\_vector\_chars=array[1..dimF] of char;

  t\_lista\_chars=^t\_nodo\_chars;

  t\_nodo\_chars=record

    ele: char;

    sig: t\_lista\_chars;

  end;

procedure leer\_char(var c: char);

begin

  c:=chr(ord('.')+random(dimF));

end;

procedure cargar\_vector\_chars(var vector\_chars: t\_vector\_chars; var dimL: int8);

var

  c: char;

begin

  leer\_char(c);

  if ((dimL<dimF) and (c<>char\_salida)) then

  begin

    dimL:=dimL+1;

    vector\_chars[dimL]:=c;

    cargar\_vector\_chars(vector\_chars,dimL);

  end;

end;

procedure imprimir\_secuencial\_vector\_chars(vector\_chars: t\_vector\_chars; dimL: int8);

var

  i: int8;

begin

  for i:= 1 to dimL do

  begin

    textcolor(green); write('Elemento ',i,' del vector: '); textcolor(red); writeln(vector\_chars[i]);

  end;

end;

procedure imprimir\_recursivo\_vector\_chars(vector\_chars: t\_vector\_chars; dimL: int8);

begin

  if (dimL>0) then

  begin

    imprimir\_recursivo\_vector\_chars(vector\_chars,dimL-1);

    textcolor(green); write('Elemento ',dimL,' del vector: '); textcolor(red); writeln(vector\_chars[dimL]);

  end;

end;

function contar\_chars(): int16;

var

  c: char;

begin

  leer\_char(c);

  if (c=char\_salida) then

    contar\_chars:=0

  else

    contar\_chars:=contar\_chars()+1

end;

procedure agregar\_adelante\_lista\_chars(var lista\_chars: t\_lista\_chars; c: char);

var

  nuevo: t\_lista\_chars;

begin

  new(nuevo);

  nuevo^.ele:=c;

  nuevo^.sig:=lista\_chars;

  lista\_chars:=nuevo;

end;

procedure cargar\_lista\_chars(var lista\_chars: t\_lista\_chars);

var

  c: char;

begin

  leer\_char(c);

  if (c<>char\_salida) then

  begin

    agregar\_adelante\_lista\_chars(lista\_chars,c);

    cargar\_lista\_chars(lista\_chars);

  end;

end;

procedure imprimir1\_lista\_chars(lista\_chars: t\_lista\_chars; i: int8);

begin

  if (lista\_chars<>nil) then

  begin

    i:=i+1;

    textcolor(green); write('Elemento ',i,' de la lista: '); textcolor(red); writeln(lista\_chars^.ele);

    imprimir1\_lista\_chars(lista\_chars^.sig,i);

  end;

end;

procedure imprimir2\_lista\_chars(lista\_chars: t\_lista\_chars; i: int8);

begin

  if (lista\_chars<>nil) then

  begin

    i:=i+1;

    imprimir2\_lista\_chars(lista\_chars^.sig,i);

    textcolor(green); write('Elemento ',i,' de la lista: '); textcolor(red); writeln(lista\_chars^.ele);

  end;

end;

var

  vector\_chars: t\_vector\_chars;

  lista\_chars: t\_lista\_chars;

  dimL, i: int8;

begin

  randomize;

  dimL:=0;

  lista\_chars:=nil;

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_vector\_chars(vector\_chars,dimL);

  writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

  imprimir\_secuencial\_vector\_chars(vector\_chars,dimL);

  writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

  imprimir\_recursivo\_vector\_chars(vector\_chars,dimL);

  writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

  textcolor(green); write('La cantidad de caracteres leídos es '); textcolor(red); writeln(contar\_chars());

  writeln(); textcolor(red); writeln('INCISO (e):'); writeln();

  cargar\_lista\_chars(lista\_chars);

  if (lista\_chars<>nil) then

  begin

    writeln(); textcolor(red); writeln('INCISO (f):'); writeln();

    i:=0;

    imprimir1\_lista\_chars(lista\_chars,i);

    writeln(); textcolor(red); writeln('INCISO (g):'); writeln();

    i:=0;

    imprimir2\_lista\_chars(lista\_chars,i);

  end;

end.

**Ejercicio 2.**

*Realizar un programa que lea números hasta leer el valor 0 e imprima, para cada número leído, sus dígitos en el orden en que aparecen en el número. Debe implementarse un módulo recursivo que reciba el número e imprima lo pedido. Ejemplo, si se lee el valor 256, se debe imprimir 2 5 6.*

program TP2\_E2;

*{$codepage UTF8}*

uses crt;

const

  num\_salida=0;

procedure leer\_numero(var num: int8);

begin

  num:=num\_salida+random(high(int8));

end;

procedure descomponer\_numero(var digito: int8; var num: int16);

begin

  digito:=num mod 10;

  num:=num div 10;

end;

procedure imprimir\_digitos(num: int16);

var

  digito: int8;

begin

  if (num<>num\_salida) then

  begin

    descomponer\_numero(digito,num);

    imprimir\_digitos(num);

    textcolor(red); write(digito,' ');

  end;

end;

procedure leer\_numeros();

var

  num: int8;

begin

  leer\_numero(num);

  if (num<>num\_salida) then

  begin

    textcolor(green); writeln(); write('Número entero: '); textcolor(red); writeln(num);

    textcolor(green); write('Número entero (dígito por dígito): ');

    imprimir\_digitos(num);

    writeln();

    leer\_numeros();

  end;

end;

begin

  leer\_numeros();

end.

**Ejercicio 3.**

*Escribir un programa que:*

**(a)** *Implemente un módulo recursivo que genere una lista de números enteros “random” mayores a 0 y menores a 100. Finalizar con el número 0.*

**(b)** *Implemente un módulo recursivo que devuelva el mínimo valor de la lista.*

**(c)** *Implemente un módulo recursivo que devuelva el máximo valor de la lista.*

**(d)** *Implemente un módulo recursivo que devuelva verdadero si un valor determinado se encuentra en la lista o falso en caso contrario.*

program TP2\_E3;

*{$codepage UTF8}*

uses crt;

const

  num\_ini=0; num\_fin=100;

  num\_salida=0;

type

  t\_numero=num\_ini..num\_fin;

  t\_lista\_numeros=^t\_nodo\_numeros;

  t\_nodo\_numeros=record

    ele: int16;

    sig: t\_lista\_numeros;

  end;

procedure leer\_numero(var num: t\_numero);

begin

  num:=num\_salida+random(num\_fin);

end;

procedure agregar\_adelante\_lista\_numeros(var lista\_numeros: t\_lista\_numeros; num: t\_numero);

var

  nuevo: t\_lista\_numeros;

begin

  new(nuevo);

  nuevo^.ele:=num;

  nuevo^.sig:=lista\_numeros;

  lista\_numeros:=nuevo;

end;

procedure cargar\_lista\_numeros(var lista\_numeros: t\_lista\_numeros);

var

  num: t\_numero;

begin

  leer\_numero(num);

  if (num<>num\_salida) then

  begin

    agregar\_adelante\_lista\_numeros(lista\_numeros,num);

    cargar\_lista\_numeros(lista\_numeros);

  end;

end;

procedure imprimir\_lista\_numeros(lista\_numeros: t\_lista\_numeros; i: int16);

begin

  if (lista\_numeros<>nil) then

  begin

    i:=i+1;

    textcolor(green); write('Elemento ',i,' de la lista: '); textcolor(red); writeln(lista\_numeros^.ele);

    imprimir\_lista\_numeros(lista\_numeros^.sig,i);

  end;

end;

procedure calcular\_minimo\_lista\_numeros(lista\_numeros: t\_lista\_numeros; var num\_min: t\_numero);

begin

  if (lista\_numeros<>nil) then

  begin

    if (lista\_numeros^.ele<num\_min) then

      num\_min:=lista\_numeros^.ele;

    calcular\_minimo\_lista\_numeros(lista\_numeros^.sig,num\_min);

  end;

end;

procedure calcular\_maximo\_lista\_numeros(lista\_numeros: t\_lista\_numeros; var num\_max: t\_numero);

begin

  if (lista\_numeros<>nil) then

  begin

    if (lista\_numeros^.ele>num\_max) then

      num\_max:=lista\_numeros^.ele;

    calcular\_maximo\_lista\_numeros(lista\_numeros^.sig,num\_max);

  end;

end;

function buscar\_lista\_numeros(lista\_numeros: t\_lista\_numeros; num: int16): boolean;

begin

  if (lista\_numeros=nil) then

    buscar\_lista\_numeros:=false

  else

    if (lista\_numeros^.ele=num) then

      buscar\_lista\_numeros:=true

    else

      buscar\_lista\_numeros:=buscar\_lista\_numeros(lista\_numeros^.sig,num);

end;

var

  lista\_numeros: t\_lista\_numeros;

  num\_min, num\_max: t\_numero;

  i, num: int16;

begin

  randomize;

  lista\_numeros:=nil;

  num\_min:=high(t\_numero);

  num\_max:=low(t\_numero);

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_lista\_numeros(lista\_numeros);

  if (lista\_numeros<>nil) then

  begin

    i:=0;

    imprimir\_lista\_numeros(lista\_numeros,i);

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    calcular\_minimo\_lista\_numeros(lista\_numeros,num\_min);

    textcolor(green); write('El mínimo valor de la lista es '); textcolor(red); writeln(num\_min);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    calcular\_maximo\_lista\_numeros(lista\_numeros,num\_max);

    textcolor(green); write('El máximo valor de la lista es '); textcolor(red); writeln(num\_max);

    writeln(); textcolor(red); writeln('INCISO (d):'); writeln();

    num:=(num\_ini+1)+random(num\_fin-(num\_ini+1));

    textcolor(green); write('¿El número '); textcolor(yellow); write(num); textcolor(green); write(' se encuentra en la lista?: '); textcolor(red); write(buscar\_lista\_numeros(lista\_numeros,num));

  end;

end.

**Ejercicio 4.**

*Escribir un programa con:*

**(a)** *Un módulo recursivo que retorne un vector de 20 números enteros “random” mayores a 0 y menores a 100.*

**(b)** *Un módulo recursivo que devuelva el máximo valor del vector.*

**(c)** *Un módulo recursivo que devuelva la suma de los valores contenidos en el vector.*

program TP2\_E4;

*{$codepage UTF8}*

uses crt;

const

  dimF=20;

  num\_ini=0; num\_fin=100;

type

  t\_numero=num\_ini..num\_fin;

  t\_vector\_numeros=array[1..dimF] of t\_numero;

procedure cargar\_vector\_numeros(var vector\_numeros: t\_vector\_numeros; var dimL: int8);

begin

  if (dimL<dimF) then

  begin

    dimL:=dimL+1;

    vector\_numeros[dimL]:=(num\_ini+1)+random(num\_fin-(num\_ini+1));

    cargar\_vector\_numeros(vector\_numeros,dimL);

  end;

end;

procedure imprimir\_vector\_numeros(vector\_numeros: t\_vector\_numeros; dimL: int8);

begin

  if (dimL>0) then

  begin

    imprimir\_vector\_numeros(vector\_numeros,dimL-1);

    textcolor(green); write('Elemento ',dimL,' del vector: '); textcolor(red); writeln(vector\_numeros[dimL]);

  end;

end;

procedure calcular\_maximo\_vector\_numeros(vector\_numeros: t\_vector\_numeros; dimL: int8; var num\_max: t\_numero);

begin

  if (dimL>0) then

  begin

    if (vector\_numeros[dimL]>num\_max) then

      num\_max:=vector\_numeros[dimL];

    calcular\_maximo\_vector\_numeros(vector\_numeros,dimL-1,num\_max);

  end;

end;

function sumar\_vector\_numeros(vector\_numeros: t\_vector\_numeros; dimL: int8): int16;

begin

  if (dimL=1) then

    sumar\_vector\_numeros:=vector\_numeros[dimL]

  else

    sumar\_vector\_numeros:=sumar\_vector\_numeros(vector\_numeros,dimL-1)+vector\_numeros[dimL];

end;

var

  vector\_numeros: t\_vector\_numeros;

  num\_max: t\_numero;

  dimL: int8;

begin

  randomize;

  dimL:=0;

  num\_max:=low(t\_numero);

  writeln(); textcolor(red); writeln('INCISO (a):'); writeln();

  cargar\_vector\_numeros(vector\_numeros,dimL);

  if (dimL>0) then

  begin

    imprimir\_vector\_numeros(vector\_numeros,dimL);

    writeln(); textcolor(red); writeln('INCISO (b):'); writeln();

    calcular\_maximo\_vector\_numeros(vector\_numeros,dimL,num\_max);

    textcolor(green); write('El máximo valor del vector es '); textcolor(red); writeln(num\_max);

    writeln(); textcolor(red); writeln('INCISO (c):'); writeln();

    textcolor(green); write('La suma de los valores contenidos en el vector es '); textcolor(red); write(sumar\_vector\_numeros(vector\_numeros,dimL));

  end;

end.

**Ejercicio 5.**

*Implementar un módulo que realice una búsqueda dicotómica en un vector, utilizando el siguiente encabezado:*

*Procedure busquedaDicotomica(v: vector; ini,fin: indice; dato: integer; var pos: indice);*

*Nota: El parámetro “pos” debe retornar la posición del dato o -1 si el dato no se encuentra en el vector.*

program TP2\_E5;

*{$codepage UTF8}*

uses crt;

const

  dimF=10;

  num\_salida=0;

type

  t\_vector\_numeros=array[1..dimF] of int8;

procedure cargar\_vector\_numeros(var vector\_numeros: t\_vector\_numeros; var dimL: int8);

var

  num: int8;

begin

  if (dimL<dimF) then

  begin

    num:=num\_salida+random(high(int8));

    if (num<>num\_salida) then

    begin

      dimL:=dimL+1;

      vector\_numeros[dimL]:=num;

      cargar\_vector\_numeros(vector\_numeros,dimL);

    end;

  end;

end;

procedure imprimir\_vector\_numeros(vector\_numeros: t\_vector\_numeros; dimL: int8);

begin

  if (dimL>0) then

  begin

    imprimir\_vector\_numeros(vector\_numeros,dimL-1);

    textcolor(green); write('Elemento ',dimL,' del vector: '); textcolor(red); writeln(vector\_numeros[dimL]);

  end;

end;

procedure ordenar\_vector\_numeros(var vector\_numeros: t\_vector\_numeros; dimL: int8);

var

  i, j, k, item: int8;

begin

  for i:= 1 to (dimL-1) do

  begin

    k:=i;

    for j:= (i+1) to dimL do

      if (vector\_numeros[j]<vector\_numeros[k]) then

        k:=j;

    item:=vector\_numeros[k];

    vector\_numeros[k]:=vector\_numeros[i];

    vector\_numeros[i]:=item;

  end;

end;

function buscar\_vector\_numeros(vector\_numeros: t\_vector\_numeros; num, pri, ult: int8): int8;

var

  medio: int8;

begin

  if (pri<=ult) then

  begin

    medio:=(pri+ult) div 2;

    if (num=vector\_numeros[medio]) then

      buscar\_vector\_numeros:=medio

    else if (num<vector\_numeros[medio]) then

      buscar\_vector\_numeros:=buscar\_vector\_numeros(vector\_numeros,num,pri,medio-1)

    else

      buscar\_vector\_numeros:=buscar\_vector\_numeros(vector\_numeros,num,medio+1,ult)

  end

  else

    buscar\_vector\_numeros:=-1;

end;

var

  vector\_numeros: t\_vector\_numeros;

  dimL, num, pri, ult, pos: int8;

begin

  randomize;

  dimL:=0;

  cargar\_vector\_numeros(vector\_numeros,dimL);

  if (dimL>0) then

  begin

    imprimir\_vector\_numeros(vector\_numeros,dimL);

    ordenar\_vector\_numeros(vector\_numeros,dimL);

    imprimir\_vector\_numeros(vector\_numeros,dimL);

    num:=1+random(high(int8));

    pri:=1; ult:=dimL;

    pos:=buscar\_vector\_numeros(vector\_numeros,num,pri,ult);

    if (pos<>-1) then

    begin

      textcolor(green); write('El número '); textcolor(yellow); write(num); textcolor(green); write(' se encontró en el vector, en la posición '); textcolor(red); write(pos);

    end

    else

    begin

      textcolor(green); write('El número '); textcolor(yellow); write(num); textcolor(green); write(' no se encontró en el vector');

    end;

  end;

end.

**Ejercicio 6.**

*Realizar un programa que lea números y que utilice un módulo recursivo que escriba el equivalente en binario de un número decimal. El programa termina cuando el usuario ingresa el número 0 (cero). Ayuda: Analizando las posibilidades, se encuentra que Binario (N) es N si el valor es menor a 2. ¿Cómo se obtienen los dígitos que componen al número? ¿Cómo se achica el número para la próxima llamada recursiva? Ejemplo: si se ingresa 23, el programa debe mostrar 10111.*

program TP2\_E6;

*{$codepage UTF8}*

uses crt;

const

  num\_salida=0;

procedure leer\_numero(var num: int8);

begin

  num:=num\_salida+random(high(int8));

end;

procedure convertir\_binario(num: int16);

var

  digito: int16;

begin

  if (num<>num\_salida) then

  begin

    digito:=num mod 2;

    convertir\_binario(num div 2);

    write(digito);

  end;

end;

var

  num: int8;

  i: int16;

begin

  randomize;

  i:=0;

  leer\_numero(num);

  while (num<>num\_salida) do

  begin

    i:=i+1;

    textcolor(green); write(i,'. Número en decimal: '); textcolor(red); writeln(num);

    textcolor(green); write(i,'. Número en binario: '); textcolor(red);

    convertir\_binario(num);

    leer\_numero(num);

    writeln();

  end;

end.