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.
```