VII.1.1. Número de elementos de una lista

Dada una lista de enteros, especificar e implementar un subprograma que calcule el número de elementos de la lista. (usar el fichero *longitud.adb* y el programa de prueba *prueba_longitud.adb*)

longitud.adb

```
with Datos;
use Datos;

function Longitud (L : Lista ) return Natural is
   L_kop: Natural;
   egungoaren_apuntadorea: Lista;
begin
   L_kop := 0;
   egungoaren_apuntadorea := L;
   loop exit when egungoaren_apuntadorea = null;
         egungoaren_apuntadorea := egungoaren_apuntadorea.sig;
         L_kop := L_kop + 1;
   end loop;
   return L_kop;
end Longitud;
```

VII.1.2. Media aritmética de los valores de una lista

Dada una lista de enteros, especificar e implementar un subprograma que calcule la media aritmética de los elementos de la lista. (media.adb)

media.adb

```
with Datos;
use Datos;

function Media (L : Lista ) return Float is
  -- pre: ??? Escribe la precondicion!!!
  -- post: ??? Escribe la postcondicion!!!
akum: Float := 0.0;
kont: Integer:= 0;
media_lista: Lista;
begin
   media_lista := L;
   loop exit when media_lista = null;
      akum := akum + float(media_lista.Info);
      media_lista := media_lista.sig;
      kont := kont + 1;
   end loop;
   if kont /= 0 then
      akum := akum / float(kont);
   end if;
   return akum;
end Media;
```

VII.1.3. Escribir lista

Dada una lista de enteros, especificar e implementar un subprograma que escriba en pantalla los elementos de la lista. (escribir.adb)

escribir.adb

```
with Ada.Text_IO, Ada.Integer_Text_IO, Datos;
use Datos;

procedure Escribir ( L : in     Lista ) is

   Actual : Lista;

begin

    Actual := L;

   Ada.Text_Io.New_Line;
   Ada.Text_Io.New_Line;
   Ada.Text_Io.Put("el contenido de la lista es: ");
   Ada.Text_Io.New_Line;
   Ada.Text_Io.Put("       <");
   Ada.Text_Io.New_Line;

    loop exit when(Actual = null);
       Ada.Integer_Text_IO.Put(Actual.info);

       Actual := Actual.Sig;
    end loop;


   Ada.Text_Io.Put("       >");
   Ada.Text_Io.New_Line;
end Escribir;
```

VII.1.4. Máximo de la lista y posición

Dada una lista de enteros, especificar e implementar un subprograma que obtenga el elemento máximo de la lista y su posición. (calcular_máximo_y_posicion.adb)

calcular_máximo_y_posicion.adb

```
with Datos;
use Datos;

procedure Calcular_Maximo_Y_Posicion (L: in Lista;  Max, Pos_Max : out Integer ) is
   -- pre:
   -- post: Max contendra el mayor valor de L y Pos_max su posicion
   --       Si L es vacia entonces Pos_Max vale cero

    Actual: Lista;
```

```
   kont: Integer;

begin

   Actual := L;
   Pos_Max := 0;
   Max := 0;
   kont := 0;

   loop exit when(Actual = null);

      kont := kont + 1;

      if (Actual.Info > Max) then
         Max := Actual.Info;
         Pos_Max := kont;
      end if;

      Actual := Actual.Sig;

   end loop;


end Calcular_Maximo_Y_Posicion;
```

VII.1.5. Insertar al comienzo

Dada una lista de enteros y un valor entero, especificar e implementar un subprograma que inserte ese valor como primer elemento de la lista. (insertar_al_comienzo.adb)

insertar_al_comienzo.adb

```
with Datos;
use Datos;

procedure Insertar_Al_Comienzo (L: in out Lista; Num : in Integer ) is
  -- pre:
  -- post: se ha insertado el nuevo valor al comienzo de L

  Nuevo : Lista;
begin
   Nuevo := new Nodo;
   Nuevo.info := num;
   Nuevo.sig := L;
   L := Nuevo;
end Insertar_Al_Comienzo;
```

VII.1.6. Insertar al final

Dada una lista de enteros y un valor entero, especificar e implementar un subprograma que inserte ese valor detrás del último elemento de la lista. (insertar_al_final.adb)

insertar_al_final.adb

```
with Datos, azkena;
use Datos;

procedure Insertar_Al_Final (L: in out Lista; Num : in Integer ) is
   -- pre:
   -- post: se ha insertado el nuevo valor al final de L

    aurreko: Lista;
    berria: Lista;

begin

    berria := new Nodo;
    berria.Info := Num;
    aurreko := azkena(L);

    if aurreko = null then
        L:= berria;
    else
        aurreko.Sig := berria;
    end if;


end Insertar_Al_Final;
```

VII.1.7. Buscar en lista no ordenada

Dada una lista de enteros **no ordenada** y un valor entero, especificar e implementar un subprograma que diga si el valor pertenece o no a la lista. En caso de que el valor pertenezca a la lista devolverá su posición y si no devolverá cero. (posicion.adb)

posicion.adb

```
with Datos;
use Datos;

function Posicion (L: Lista; Num : Integer ) return Natural is
   -- pre:
   -- post: el resultado es la posicion de la primera aparicion de Num,
   --       caso de que Num pertenezca a L, y cero en otro caso

Lista_posicion: Lista;
kont: Integer;
aurkitua: boolean;
```

```
begin
   aurkitua := false;
   kont := 0;
   Lista_posicion := L;
   loop exit when (Lista_posicion = null) or aurkitua = true;
      if (Lista_posicion.Info = num) then
         aurkitua := true;
      end if;
      Lista_posicion := Lista_posicion.sig;
      kont := kont + 1;
   end loop;
   if aurkitua /= false then
      return kont;
   else
      return 0;
   end if;
end Posicion;
```

VII.1.8. Buscar en lista ordenada

Dada una lista de enteros **ordenada** (de menor a mayor) y un valor entero, especificar e implementar un subprograma que diga si el valor pertenece o no a la lista. En caso de que el valor pertenezca a la lista devolverá su posición y si no devolverá la posición en que debería colocarse. (posición_lista_ordenada.adb)

posición_lista_ordenada.adb

```
with Datos;
use Datos;

procedure Posicion_Lista_Ordenada (
      L    :       Lista;
      Num  :       Integer;
      Esta :    out Boolean;
      Pos  :    out Natural  ) is
   -- pre: L esta ordenada, con sus valores de menor a mayor
   -- post: Esta valdra true si Num pertenece a L y false si no
   --       Pos es la posicion de la primera aparicion de Num,
   --       en caso de que Num pertenezca a L, y si no
   --       devolverá la posición en que debería colocarse

    Actual: Lista;
    kont : Integer;

begin

    Actual := L;
    Esta := False;
    Pos := 1;
    kont := 0;
```

```
   loop exit when( (Actual = null) or else (Actual.Info > num) or else (Esta = True) );

      kont := kont + 1;

      if (Actual.Info = num) then
         Esta := True;
      else
         Pos := kont + 1;
      end if;

      Actual := Actual.Sig;

   end loop;


end Posicion_Lista_Ordenada;
```

VII.1.9. Insertar en lista ordenada *

Dada una lista de enteros **ordenada** y un valor entero, especificar e implementar un subprograma que inserte el valor en el lugar apropiado. (insertar_en_lista_ordenada.adb)

insertar_en_lista_ordenada.adb

```
with Datos;
use Datos;

with longitud;

procedure Insertar_En_Lista_Ordenada (L: in out Lista; Num: in Integer ) is
   -- pre: L esta ordenada de menor a mayor
   -- post: se ha insertado el nuevo valor en L de manera ordenada

    aurrekoa: Lista;
    egungoa: Lista;

    atera: Boolean;

begin

    atera := False;

    -- L zerrendan elementurik ez badago
    if (L = null) then

       L:= new Nodo;
       L.info := Num;

    -- Hasieran txertatu
    elsif (L.Info > Num) then
```

```
      egungoa := new Nodo;
      egungoa.Info := Num;
      egungoa.sig := L;
      L := egungoa;

   else
      egungoa := L;

      loop exit when( (atera = True) or (egungoa = null) );

         if (egungoa.Info > Num) then

            egungoa := New Nodo;
            egungoa.Info := Num;
            egungoa.Sig := aurrekoa.Sig;
            aurrekoa.Sig := egungoa;
            atera := True;

         else

            aurrekoa := egungoa;
            egungoa := egungoa.Sig;

         end if;

      end loop;

      if (atera = False) then
         egungoa := new Nodo;
         egungoa.Info := Num;
         aurrekoa.Sig := egungoa;
      end if;
   end if;

end Insertar_En_Lista_Ordenada;
```

VII.1.10. Insertar en la posición N-ésima *

Dada una lista de enteros, un valor positivo N y un valor entero, especificar e implementar un subprograma que inserte el entero en la posición N de la lista. (no hay programa de prueba)

insertar_en_la_posicion_nsima.adb

```
with Ada.Text_Io, Datos;
use Datos;
procedure insertar_en_la_posicion_nsima (L: in out Lista ; Num , Pos : in Integer) is

k:integer;
aux1: Lista;
aux2: Lista;
Begin
```

```
k:=1;
aux1:=L;
   loop exit when k=pos;
      aux2:=aux1;
      aux1:=aux1.Sig;
      k:=k+1;
   end loop;
   if (aux2=null) then
      aux1:=new Nodo;
      aux1.Info:=Num;
      aux1.Sig:=L;
      L:=aux1;
   else
      aux1:=new Nodo;
      aux1.info:=Num;
      aux1.Sig:=aux2.Sig;
      aux2.Sig:=aux1;
   end if;

end insertar_en_la_posicion_nsima;
```

prueba_instertar_en_la_posicion_nsima.adb

```
with Datos, Ada.Text_Io;
use Datos, Ada.Text_Io;

with insertar_en_la_posicion_nsima, escribir, crear_lista_vacia, insertar_al_comienzo;


procedure Prueba_Insertar_en_posicion is

   Lis: Lista;

   procedure Pedir_Return is
   begin
      Put_Line("pulsa return para continuar ");
      Skip_Line;
   end Pedir_Return;

begin

   Put_Line("Programa de prueba: ");
   Put_Line("**********");

   Crear_Lista_Vacia(Lis);
   Put_Line("Caso de prueba 1: Insertar en lista vacia en la posicion 1 ");
   Insertar_en_la_posicion_nsima(Lis, 5,1);
   Put_Line("Ahora deberia escribir la lista <5> ");
   Escribir(Lis);
   New_Line;
   New_Line;
   Pedir_Return;
```

```
   Crear_Lista_Vacia(Lis);
   Insertar_Al_Comienzo(Lis, 4);
   Insertar_Al_Comienzo(Lis, 9);
   Insertar_Al_Comienzo(Lis, 7);
   Insertar_Al_Comienzo(Lis, 5);
   Put_Line("Caso de prueba 2: insertar en la mitad");
   Put_Line("La lista inicial contiene ");
   Escribir(Lis);
   Insertar_en_la_posicion_nsima(Lis,3,3);
   Put_Line("Ahora deberia escribir la lista <5, 7, 3, 9, 4> ");
   Escribir(Lis);
   New_Line;
   New_Line;
   Pedir_Return;


   Crear_Lista_Vacia(Lis);
   Insertar_Al_Comienzo(Lis, 4);
   Insertar_Al_Comienzo(Lis, 9);
   Insertar_Al_Comienzo(Lis, 7);
   Insertar_Al_Comienzo(Lis, 5);
   Put_Line("Caso de prueba 3: insertar al final");
   Put_Line("La lista inicial contiene ");
   Escribir(Lis);
   Insertar_en_la_posicion_nsima(Lis, 3,5);
   Put_Line("Ahora deberia escribir la lista <5, 7, 9, 4, 3> ");
   Escribir(Lis);
   New_Line;
   New_Line;
   Pedir_Return;



   Crear_Lista_Vacia(Lis);
   Insertar_Al_Comienzo(Lis, 4);
   Insertar_Al_Comienzo(Lis, 9);
   Insertar_Al_Comienzo(Lis, 7);
   Insertar_Al_Comienzo(Lis, 5);
   Put_Line("Caso de prueba 3: insertar en el principio");
   Put_Line("La lista inicial contiene ");
   Escribir(Lis);
   Insertar_en_la_posicion_nsima(Lis, 3,1);
   Put_Line("Ahora deberia escribir la lista <3, 5, 7, 9, 4> ");
   Escribir(Lis);
   New_Line;
   New_Line;
   Pedir_Return;




end;
```

VII.1.11. Borrar la primera aparición *

Dada una lista de enteros y un valor entero, especificar e implementar un subprograma que borre ese elemento de la lista. (eliminar_primera_aparicion.adb)

eliminar_primera_aparicion.adb

```
with Datos, Ada.Text_Io;
use Datos, Ada.Text_Io;

procedure Eliminar_Primera_Aparicion (
      L   : in out Lista;
      Num : in     Integer ) is
   -- Pre:
   -- Post: se ha eliminado de L la primera aparicion de Num
   --       en caso de que no aparezca se escribira un mensaje

          Aux: Lista;
          Aux1: Lista;
          agertua: boolean;
begin

          agertua := False;

    Berria := L;

          loop exit when ((agertua = True) or (Aux = null));

          if (Num = Aux.Info) then

             if (Aux1 /= null) then

                Aux1.sig := Aux.sig;
                agertua := True;

             else

                                          agertua := True;
                L := L.sig;

             end if;

          else

             Aux1 := Aux;
             Aux := Aux.sig;

          end if;
       end loop;
end Eliminar_Primera_Aparicion;
```

VII.1.12. Borrar todas las apariciones *

Dada una lista de enteros y un valor entero, especificar e implementar un subprograma que borre todas las apariciones de ese elemento de la lista. (eliminar_todas_las_apariciones.adb)

eliminar_todas_las_apariciones.adb

```
with Datos;
use Datos;

with Ada.Text_Io; use Ada.Text_Io;

with posicion;

procedure Eliminar_Todas_Las_Apariciones (L: in out Lista; Num: in Integer ) is
   -- Pre:
   -- Post: se han eliminado de L todas las apariciones de Num

    Pos : Natural;

    Aux: Lista;
    Aux1: Lista;

    -- 234
    -- 3
    -- 24


begin

    Pos := Posicion(L, Num);

    Aux := L;

    if ( (L /= null) and (Pos /= 0) ) then

        loop exit when(Aux = null);

            if (Aux.Info = Num) then

                if (Aux1 /= null) then
                    Aux1.Sig := Aux.Sig;
                    Aux := Aux.Sig;
                else
                    L := L.Sig;
                    Aux := L;

                end if;

            else
```

```
            Aux1 := Aux;
            Aux := Aux.Sig;

         end if;

      end loop;

   end if;


end Eliminar_Todas_Las_Apariciones;
```