Oinarrizko Programazioa (ebaluazio globala)
2010/11 ikasturtea
2011-I-10

Telefono mugikor batean telefono-lista batean antolatuko ditugu gure kontaktuak. Kontaktu bakoitzerako bere izena eta telefono-zenbakia sartuko ditugu. Kontaktu bakoitzean zenbaki bakarra sar dezakegunez, posible izango da izen berarekin kontaktu bat baino gehiago sartzea bakoitzean telefono desberdin bat sartuta. Kontaktuak alfabetikoki ordenatuta egongo dira izenaren arabera.

Kontaktu bakoitzeko zenbaki bakarra onartzen duen horrelako telefono-lista bat honela errepresentatuko dugu:

datuak.ads

```
   ...
   Datuak_Max : constant Integer := 25;
   subtype Indize0_Datuak_Max is Integer range 0 .. Datuak_Max;
   type Datu is
      record
         Info : String (1 .. Datuak_Max);
         Kop  : Indize0_Datuak_Max;
      end record;
```

telefono_zenbakiak.ads

```
   ...
   Digitu_Max : constant Natural := 9;
   subtype Indize is Natural range 1.. Digitu_Max;
   subtype Digitu is Natural range 0..9;
   type Telefono_zenbaki is array (Indize) of Digitu;
```

kontaktuak.ads

```
   ...
   type Kontaktu is
      record
         Norena   : Datuak.Datu;
         Telefonoa: Telefono_Zenbakiak.Telefono_zenbaki;
      end record;
```

kontaktu_listak.ads

```
   ...
      Kont_Max : constant Integer := 50;
      subtype Indize0_Kont_Max is Integer range 0 .. Kont_Max;
      subtype Indize1_Kont_Max is Integer range 1 .. Kont_Max;
      type Kont_Taula is array (1 .. Kont_Max)
                              of Kontaktuak.Kontaktu;
      type Kont_Lista is
         record
            Info : Kont_Taula;
            Kop  : Indize0_Kont_Max;
         end record;
```

Zehaztapena definitu eta inplementatu itzazu ondoko azpiprogramak:

1. **(3 puntu)**:
   a. Telefono-zenbaki bat emanda, ea telefono finkoa den zehaztuko du (kontuan hartu finkoen lehen digitoa 9 dela).(0,25)
   b. Bi telefono-zenbaki emanda, aztertuko du ea posizioren batean digitu desberdinak dituzten. Adibidez, 948234567 eta 943234667 zenbakiak horrelakoak dira, gutxienez beren hirugarren digituak diferenteak baitira. (0,5)
   c. Bi telefono-zenbaki emanda, aztertuko du ea zenbat posiziotan dituzten digitu desberdinak. Adibidez, 94**8**234**5**67 eta 94**3**234**6**67 zenbakiek bi digitu desberdin dituzte, 3. eta 7. posiziokoak. (0,5)
   d. Bi telefono-zenbaki emanda, aztertuko du ea berdinak diren. (0,25)
   e. Bi izen emanda (`Datuak.Datu` motakoak), aztertuko du ea berdinak diren. (0,5)
   f. Bi izen emanda (`Datuak.Datu` motakoak), aztertuko du ea lehenengoa alfabetikoki txikiago ote den bigarrena baino. (0,5)
   g. Bi kontaktu emanda, aztertuko du ea berdinak diren. Esango dugu berdinak direla izena eta zenbaki berdinak badituzte. (0,5)
2. **(2,25 puntu)** Kontaktu-lista bat eta telefono-zenbaki bat emanda, listan zenbaki hori duen pertsonaren izena itzuliko du. Listan ez badago horrelako zenbakirik "Zenbaki ezezaguna" itzuliko du.
3. **(2,25 puntu)** Kontaktu-lista bat eta kontaktu bat emanda, kontaktu hori lehenago listan ez bazegoen listan dagokion tokian txertatuko du.

4. **(2,5 puntu)**
   Beste mota berri bateko telefono-lista definitu behar duzu (*Lista_Zenbakianitz*): lehengoaren antzekoa da baina kontaktu bakoitzean telefono-zenbaki bat baino gehiago sartzea onartuko du. Beraz, kontaktu bakoitzean telefono-zenbaki lista bat egongo da, gehienez bost telefono-zenbakirekin. Lehengoan bezala kontaktuak alfabetikoki ordenatuta egongo dira izenaren arabera. Lehengo listari *zenbakibakar* esango diogu, berriari *zenbakianitz*.

   a. Definitu itzazu beharrezko diren datu-egiturak (motak) zenbaki anitzeko lista berri hori erabili ahal izateko. Aurreko datu motak erabil ditzakezu definizio berrian, noski,
   b. Kontatu-lista *zenbakibakar* eta kontaktu-lista *zenbakianitz* bana emanda, lista *zenbakibakarreko* kontaktuak bigarrenean sartzen dituen azpiprograma bat egin. Gogora ezazu lehenengo listan osagai bat baino gehiago egon daitezkeela izen berarekin (baina zenbaki desberdinekin). Eta gogoratu ere lista biak ordena alfabetikoaren arabera ordenatuta daudela. Ondoko azpiprogramak erabili ahal izango dituzu:

```
procedure Izen_Zbkia (T: Kontaktu;
                      Iz: out Datuak.Datu;
                      Z: out Telefono_Xenbaki);
      -- Post: Iz eta Z dira T kontaktuaren izena eta zenbakia
function Luzera(KL : Kontaktu_Listak.Kont_Lista) return Natural;
   -- pos: Emaitza= KL listaren osagai kopurua
function Igarrena (KL : Kontaktu_Listak.Kont_Lista; I: Natural)
                return Kontaktu;
   -- Aurre: 1<=I<= Luzera(KL).
   -- Pos: Emaitza= KL listako I. Osagaia.
```

   c. Demagun orain zenbaki bakarreko hainbat kontaktu-lista biltzen dituen "kontaktu-listazko lista bat" honela definitu dugula:

```
Kont_Lista_Max : constant Integer := 10;
subtype I_Kont_Lista0_Max is Integer range 0 .. Kont_Lista_Max;
subtype I_Kont_Lista1_Max is Integer range 1 .. Kont_Lista_Max;
 type Kont_Lista_Taula is array (1 .. Kont_Lista_Max)
```

```
                                of Kontaktu_Listak.Kontaktu_Lista;
        type Kont_Lista_Lista is
           record
              Info : Kont_Lista_Taula;
              Kop  : I_Kont_Lista0_Max;
        end record;
```

Hainbat lista zenbakibakar batzen dituen lista bat emanda, berak dituen lista guztietako informazioa lista zenbakianitz bakar batean bilduko duen azpiprograma bat diseinatu eta inplementatu. Nahi baduzu erabili azpiprograma hau:

```
   procedure Lista_Hutsa_Sortu (KLP: out Zanitzeko_Kontaktu_Lista);
--Post: KLP lista hutsa da.
```

datuak.ads

```
package Datuak is

type Datu is private;
procedure Sortu (S : String; D : out Datu);
-- Aurre:
-- Post: D datuak S karaktere-katea errepresentatzen du,

function Info (D : Datu) return String;
-- Aurre:
-- Post:  Kat (String), Kat = D datuan dagoen katea

function Kop (D : Datu) return Natural;
-- Aurre:
-- Post: Zenbat (Natural), Zenbat = D datuan dagoen katearen luzera

function Berdin (D1, D2 : Datu) return Boolean;
-- Aurre:
-- Post: B (Boolean),
--       B = True D1 eta D2 datuek kate bera errepresentatzen badute,
--       bestela  B = False.

function Txikiagoa_Lehenengoa (D1, D2 : Datu)
                               return Boolean;
-- Aurre:
-- Post: B (Boolean), B = true baldin D1ren katea < D2ren katea,
--       bestela B = false

private
  ...
end Datuak;
```

telefono_zenbakiak.ads

```
package  Telefono_Zenbakiak is
type Telefono_Zenbaki is private;

procedure Sortu (S  : String; Tz : out Telefono_Zenbaki );
-- aurre:
-- Post: Tz = S, Tz'luzera = S luzera,
--       edozein izanda 1<i<s'luzera, tz(i) = S(I)

function Telefonoa (Tz : Telefono_Zenbaki ) return String;
-- aurre:
-- Post: Tz = S, S'luzera = Tz'luzera,
--       edozein i zanda 1<i<Tz'luzera, S(i) = Tz(I)

function Berdin (Tz1, Tz2 : Telefono_Zenbaki )
             return Boolean ;
-- aurre:
-- Post: B (Boolean), B = true baldin Tz1 = Tz2, bestela
--       B = false
private
 ...
end  Telefono_Zenbakiak;
```

kontaktuak.ads

```
package Kontaktuak is
type Kontaktu is private;


procedure Sortu
     (Nor     : in     Datuak.Datu;
      Tel_Zenb: in     Telefono_Zenbakiak.Telefono_Zenbaki;
      K       :    out Kontaktu);
-- aurre:
-- Post: K.Norena = Nor eta K.Telefonoa = Tel_Zenb

procedure Izen_Zbkia (K: in     Kontaktu;
                      Iz:   out Datuak.Datu;
                      Z:    out Telefono_Zenbaki);
      -- Post: Iz eta Z dira K kontaktuaren izena eta zenbakia

function Berdin (K1, K2 : Kontaktu ) return Boolean;
-- aurre:
-- Post: B (Boolean),
-- B = true K1en eta K2ren izenak eta telefonoak berdinak
--       baldin badira bestela B= false

private
   ...
end Kontaktuak;
```

kontaktu_listak.ads

```
package Kontaktu_Listak is

type Kontaktu_Lista is private;

function Zenbat (KL : Kontaktu_Lista )
           return Natural;
--Aurre:
--Post: Zenbat (Natural), Zenbat = KL listaren elementu kopurua

function Eman_Igarrena (KL : Kontaktu_Lista;
                        I  : Integer)
                    return Kontaktuak.Kontaktu ;
--Aurre: KL listan gutxienez I elementu daude
--Post: K (Kontaktua), K = KL listako I. elementua
...

private
  ...
end Kontaktu_Listak;
```

Ebazpena

Zehaztapena definitu eta inplementatu itzazu ondoko azpiprogramak:

1. **(3 puntu)**: `Kont_zenbakianitz_Lista`
   a. Telefono-zenbaki bat emanda, ea telefono finkoa den zehaztuko du (kontuan hartu finkoen lehen digitua 9 dela).(0,25)

```
function Finkoa_Da (Tel: Telefono_Zenbakiak.Telefono_Zenbaki)
             return Boolean is
begin
   return Tel(1) = 9 ;
end  Finkoa_Da:
```

   b. Bi telefono-zenbaki emanda, aztertuko du ea posizioren batean digitu desberdinak dituzten. Adibidez, 948234567 eta 943234667 zenbakiak horrelakoak dira, gutxienez beren hirugarren digituak diferenteak baitira. (0,5)

Telefono_Zenbakiak paketean definitu beharko zen.

```
function Digitu_Desberdinik_Posizioren_Batean
               (Tel1, Tel2:  Telefono_Zenbaki)
            return Boolean is
   Berdinak : Boolean ;
   I: Indize ;
begin
   Berdinak := True;
   I:= 1;
   while Berdinak and I<= Digitu_Max loop
      Berdinak := Tel1(I) = Tel2(I) ;
      I:= I + 1 ;
   end loop;
   return Berdinak ;
end  Digitu_Desberdinik_Posizioren_Batean ;
```

   c. Bi telefono-zenbaki emanda, aztertuko du ea zenbat posiziotan dituzten digitu desberdinak. Adibidez, 94**8**234**5**67 eta 94**3**234**6**67 zenbakiek bi digitu desberdin dituzte, 3. eta 7. posiziokoak. (0,5)

Telefono_Zenbakiak paketean definitu beharko zen.

```
function Zenbat_Posizio_Digitu_Desberdinekin
            (Tel1, Tel2:  Telefono_Zenbaki)
          return Natural is
   Zenbat : Natural ;
   I: Indize ;
begin
   Zenbat := 0;
   for I in 1..Digitu_Max loop
     if Tel1(I) /= Tel2(I) then
        Zenbat := Zenbat + 1;
     end if;
   end loop;
   return Zenbat ;
end Zenbat_Posizio_Digitu_Desberdinekin;
```

d. Bi telefono-zenbaki emanda, aztertuko du ea berdinak diren. (0,25)

Telefono_Zenbakiak paketean definitu beharko zen.

```
function Berdin (Tel1, Tel2: Telelefono_zenbaki)
             return Boolean is
begin
   return Tel1 = Tel2 ;
end Berdinak;
```

e. Bi izen emanda (`Datuak.Datu` motakoak), aztertuko du ea berdinak diren. (0,5)

Datuak paketean definitu beharko zen.

```
function Berdin (Iz1, Iz2:  Datuak.Datu) return Boolean is
-- Post: Emaitza = True
--             Iz1 eta Iz2 datuek kate bera errepresentatzen badute,
--       bestela  Emaitza = False.
   Emaitza : Boolean := False;
begin
   if Iz1.Kop = Iz2.Kop then
        if  Iz1(1.. Iz1.Kop) = Iz2(1.. Iz2.Kop) then
           Emaitza := True ;
        end if;
   end if;
   return Emaitza ;
end Berdinak;
```

f. Bi izen emanda (`Datuak.Datu` motakoak), aztertuko du ea lehenengoa alfabetikoki txikiago ote den bigarrena baino. (0,5)

Datuak paketean definitu beharko zen.

```
function Txikiagoa_Lehenengoa (D1, D2 : Datu)
             return Boolean;
-- Post: B (Boolean), B = true baldin D1ren katea < D2ren katea,
--       bestela B = false
begin
   return D1(1..D1.Kop) < D2(1..D2.Kop);
end Txikiagoa_Lehenengoa;
```

g. Bi kontaktu emanda, aztertuko du ea berdinak diren. Esango dugu berdinak direla izena eta zenbaki berdinak badituzte. (0,5)

Kontaktuak paketean definitu beharko zen.

```
function Berdinak (Kont1, Kont2:  Kontaktu) return Boolean is
   Emaitza : Boolean;
   Iz1, Iz2: Datuak.Datu;
   Z1,Z2:    Telefono_Zenbakiak.Telefono_Zenbaki;

begin
   Kontaktuak.Izen_Zbkia(Kont1, Iz1, Z1);
   Kontaktuak.Izen_Zbkia(Kont2, Iz2, Z2);
   Emaitza := False;
   if Datuak.Berdin(Iz1, Iz2) then
     if Telelefono_zenbakiak.Berdin(Z1, Z2) then
           Emaitza := True ;
     end if;
   end if;
   return Emaitza ;
end Berdinak;
```

2. **(2,25 puntu)** Kontaktu-lista bat eta telefono-zenbaki bat emanda, listan zenbaki hori duen pertsonaren izena itzuliko du. Listan ez badago horrelako zenbakirik “Zenbaki ezezaguna” itzuliko du.

Kontaktu_listak paketean definitu beharko zen.

```
function Jabearen_izena (KL: Kontaktu_listak.Kont_Lista;
                         Tz: Telefono_Zenbakiak.Telefono_Zenbaki)
           return Datuak.Datu Boolean is
  --POST:  Aurkitua = True eta Emaitza =Tz zenbakia duen pertsona.
  --       EDO  Aurkitua = False eta Emaitza=“Zenbaki ezezaguna”
      Ize: Datuak.Datu;
      Z:   Telefono_Zenbakiak.Telefono_Zenbaki;
      Emaitza: Datuak.Datu;
      Aurkitua: Boolean := False,
      I: Integer;
      Kont: Kontaktuak.Kontaktu;
   begin
      Aurkitua := False;
      I:= 1;
      while I<=kontaktu_listak.Zenbat(KL) and not Aurkitua loop
         Kont:= Kontaktu_listak.Eman_Igarrena (KL, I);
         Kontaktuak.Izen_Zbkia (Kont, Ize, Z);
         if Telefono_Zenbakiak.Berdin (Z, Tz)
         then
            Aurkitua := True;
         else
            I:= I+1;
         end if;
      end loop;
      if Aurkitua then
         Emaitza := Ize;
      else
         Datuak.Sortu(“Zenbaki ezezaguna”, Emaitza);
      end if;
      return Emaitza;
   end Jabearen_izena;
```

3. **(2,25 puntu)** Kontaktu-lista bat eta kontaktu bat emanda, kontaktu hori lehenago listan ez bazegoen listan dagokion tokian txertatuko du.

   "Txertatu ordenatuan" prozedura egokitu behar dugu kontaktu listekin erabiltzeko.

   Kontaktu_listak paketean definitu beharko zen.

```
procedure Kontaktu_Berria_Sartu
              (KL:   in out Kont_Lista;
               Kont: in     Kontaktuak.Kontaktu) is
  --AURRE: KL kontaktuak ordenatuta daude izenaren arabera alfabetikoki
  --POST:  Kont kontaktua KL Kontaktu listan dago.
  --       KL ordenatuta dago.
  --       (Txertatu ordenatuan)

     Posizioa: Indize1_Kont_Max ;
   begin
     Bilatu_Posizioa_Ordenatuan (KL, Kont, Posizioa);
     Desplazatu_Eskuinaldera (KL, Posizioa, KL.Kop);
     KL.Info(Posizioa) := Kont;
     L.Kop := L.Kop +1;
   end Kontaktu_Berria_Sartu;

   procedure Bilatu_Posizioa_Ordenatuan
                (L   : in     Kont_Lista;
                 Kont: in     Kontaktuak.Kontaktu;
                 Pos :    out Integer) is
     --Post: K kontaktua L lista ordenatuan
     --      Pos posizioan sartu beharko litzateke
     --      L lista ordenatuta jarraitzeko
     I : Integer;
     Aurkitua: Boolean;
     Ize, Ize1 : Datuak.Datu;
     Z, Z1:   Telefono_Zenbakiak.Telefono_Zenbaki;
     Kont1: in     Kontaktuak.Kontaktu;
   begin
     I := 1;
     Aurkitua := False;
     Kontaktuak.Izen_Zbkia (Kont, Ize, Z);
     while I<= L.Kop and not Aurkitua loop
        Kont1:= Kontaktu_listak.Eman_Igarrena (KL, I);
        Kontaktuak.Izen_Zbkia (Kont1, Ize1, Z1);
        if Datuak.Txikiagoa_Lehenengoa (Ize, Ize1) then
           Aurkitua := True ;
        else
           I := I + 1;
        end if;
     end loop;
    --(Aurkitua =True
    --   eta listako Igarrenaren izena kontaktu berriarena
    --         baino handiago den lehengoa da lista ordenatuan)
    -- edo
    --(Aurkitua = False
    --      eta listako izen guztiak kontaktuarena baino txikiagoak dira)
     if Aurkitua then
        Pos := I;
     else
        Pos := L.Kop + 1;
     end if;
   end Bilatu_Posizioa_Ordenatuan;
```

```
procedure Desplazatu_Eskuinaldera (L     : in out Kont_Lista;
                                   I1,I2 : in     integer) is
  --Aurre:
  --Post: LL listako [I1..I2] indize-tarteko osagaiak posizio bat
  --     desplazatu dira eskuinaldera [I1+1..I2+1] tarteko osagaietara.
begin
   for I in reverse I1..I2 loop
      L.Info(I+1) := L.Info(I);
   end loop;
end Desplazatu_Eskuinaldera;
```

4. **(2,5 puntu)**
   Beste mota berri bateko telefono-lista definitu behar duzu (*Lista_Zenbakianitz*): lehengoaren antzekoa da baina kontaktu bakoitzean telefono-zenbaki bat baino gehiago sartzea onartuko du. Beraz, kontaktu bakoitzean telefono-zenbaki lista bat egongo da, gehienez bost telefono-zenbakirekin. Lehengoan bezala kontaktuak alfabetikoki ordenatuta egongo dira izenaren arabera. Lehengo listari *zenbakibakar* esango diogu, berriari *zenbakianitz*.

   a. Definitu itzazu beharrezko diren datu-egiturak (motak) zenbaki anitzeko lista berri hori erabili ahal izateko. Aurreko datu motak erabil ditzakezu definizio berrian, noski.

```
tel_listak.ads
   ...
      Tel_Kop_Max : constant Integer := 5;
      subtype Indize0_Tel_Max is Integer range 0 .. Tel_Kop_Max ;
      subtype Indize1_Tel_Max is Integer range 1 .. Tel_Kop_Max ;
      type Tel_Taula is array (1 .. Tel_Kop_Max)
                              of Telefono_Zenbakiak.Telefono_Zenbaki;
      type Tel_Lista is
         record
            Info : Tel_Taula;
            Kop  : Indize0_Tel_Max;
         end record;

kontaktu_zenbakianitzak.ads
   ...
      type Kontaktu_Zenbakianitz is
      record
         Norena    : Datuak.Datu;
         Telefonoak: Tel_Listak.Tel_Lista;
      end record;

kontaktu_zenbakianitz_listak.ads
   ...
      Kont_Max : constant Integer := 50;
      subtype Indize0_Kont_Max is Integer range 0 .. Kont_Max;
      subtype Indize1_Kont_Max is Integer range 1 .. Kont_Max;
      type Kont_Zenbakianitz_Taula is array (1 .. Kont_Max)
                              of Kontaktu_zenbakianitzak.Kontaktu_zenbakianitz;
      type Kont_zenbakianitz_Lista is
         record
            Info : Kont_Taula;
            Kop  : Indize0_Kont_Max;
         end record;
```

   b. Kontatu-lista *zenbakibakar* eta kontaktu-lista *zenbakianitz* bana emanda, lista *zenbakibakarreko* kontaktuak bigarrenean sartzen dituen azpiprograma bat egin. Gogora ezazu lehenengo listan osagai bat baino gehiago egon daitezkeela izen berarekin (baina zenbaki desberdinekin). Eta gogoratu ere lista biak ordena alfabetikoaren arabera ordenatuta daudela. Ondoko azpiprogramak erabili ahal izango dituzu:

```
      procedure Izen_Zenbakia (T: Kontaktu;
                               Iz: out Datuak.Datu;
                               Z: out Telefono_Xenbaki);
            -- Post: Iz eta Z dira T kontaktuaren izena eta zenbakia
      function Luzera(KL : Kontaktu_Listak.Kont_Lista) return Natural;
         -- pos: Emaitza= KL listaren osagai kopurua
```

```
function Igarrena (KL : Kontaktu_Listak.Kont_Lista; I: Natural)
                  return Kontaktu;
   -- Aurre: 1<=I<= Luzera(KL).
   -- Pos: Emaitza= KL listako I. Osagaia.
```

Kontaktu_Zenbakianitz_Listak paketean definitu beharko zen.

```
procedure Gehitu (L : in out Kont_Lista;
                  KL: in out Kont_Zenbakianitz_Lista) is
   --Post: L listako telefonoak gehitu dira KL listan (ez zeudenak)
begin
   for I in 1.. Kontaktu_Listak.Zenbat(L) loop
      Kontaktu_Berria_Sartu(KL,
                            Kontaktu_Listak.Eman_Igarrena(L, I);
   end loop;
end Desplazatu_Eskuinaldera;

procedure Kontaktu_Berria_Sartu
            (KL:   in out Kont_Zenbakianitz_Lista;
             Kont: in     Kontaktuak.Kontaktu) is
  --AURRE: KL kontaktuak ordenatuta daude izenaren arabera alfabetikoki
  --POST: Kont kontaktuko telefonoa zenbakianitzeko KL listan dago.
  --       KL ordenatuta dago.
      Posizioa: Indize1_Kont_Max ;
begin
   Bilatu_Posizioa_Ordenatuan (KL, Kont, Posizioa);
   Kont1:= Kontaktu_Zenbakianitz_listak.Eman_Igarrena (KL, Posizioa);
   Kontaktu_Zenbakianitzak.Izen_Zbki_lista (Kont1, Ize1, ZL);
   Kontaktuak.Izen_Zbkia (Kont, Ize, Z);
   if Posizioa=KL.Kop+1  or (not Datuak.Berdin (Ize, Ize1)) then
      Desplazatu_Eskuinaldera (KL, Posizioa, KL.Kop);
      KL.Info(Posizioa) := Bihurtu_Zenbakianitzeko (Kont);
      KL.Kop := KL.Kop +1;
   else -- Posizioa<=KL.Kop eta Datuak.Berdin (Ize, Ize1)
      Kontaktu_Zenbakianitzak.Gehitu_Zenbakia (KL.Info(Posizioa), Z);
   end if;
end Kontaktu_Berria_Sartu;

procedure Bilatu_Posizioa_Ordenatuan
                 (L   : in     Kont_Zenbakianitz_Lista;
                  Kont: in     Kontaktuak.Kontaktu;
                  Pos :    out Integer) is
   --Post: K kontaktua L lista ordenatuan
   --      Pos posizioan sartu beharko litzateke
   --      L lista ordenatuta jarraitzeko
   I : Integer;
   Aurkitua: Boolean;
   Ize, Ize1 : Datuak.Datu;
   Z, Z1: Telefono_Zenbakiak.Telefono_Zenbaki;
   Kont1: Kontaktuak.Kontaktu;
begin
   I := 1;
   Kontaktuak.Izen_Zbkia (Kont, Ize, Z);
   Aurkitua := False;
   while I<= L.Kop and not Aurkitua loop
      Kont1:= Kontaktu_listak.Eman_Igarrena (KL, I);
      Kontaktu_Zenbakianitzak.Izen_Zbki_lista (Kont1, Ize1, ZL);
      if Datuak.Berdin (Ize, Ize1)
         or Datuak.Txikiagoa_Lehenengoa (Ize, Ize1)
      then
         Aurkitua := True ;
```

```
         else
            I := I + 1;
         end if;
      end loop;
      --(Aurkitua =True
      --   eta listako Igarrenaren izena kontaktu berriarena
      --         baino handiago den lehengoa da lista ordenatuan)
      -- edo
      --(Aurkitua = False
      --      eta listako izen guztiak kontaktuarena baino txikiagoak dira)
      if Aurkitua then
         Pos := I;
      else
         Pos := L.Kop + 1;
      end if;
   end Bilatu_Posizioa_Ordenatuan;


   procedure Desplazatu_Eskuinaldera (L     : in out Kont_Zenbakianitz_Lista;
                                      I1,I2 : in     integer) is
     --Aurre:
     --Post: LL listako [I1..I2] indize-tarteko osagaiak posizio bat
     --     desplazatu dira eskuinaldera [I1+1..I2+1] tarteko osagaietara.
   begin
      for I in reverse I1..I2 loop
         L.Info(I+1) := L.Info(I);
      end loop;
   end Desplazatu_Eskuinaldera;
```

Kontaktu_Zenbakianitzak paketean

```
   function Bihurtu_Zenbakianitzeko (Kont: Kontaktuak.Kontaktu)
                        return Kontaktu_Zenbakianitzeko is
     K:  Kontaktu_Zenbakianitzeko;
   begin
      Kontaktuak.Izen_Zbkia (Kont, Ize, Z);
      Tel_Listak.Sortu_lista_hutsa (ZL);
      Tel_Listak.Txertatu_Bukaeran (ZL, Z);
      K.Norena := Ize;
      K.Telefonoak := ZL;
      return K;
   end Bihurtu_Zenbakianitzeko;

   procedure Izen_Zbki_lista (K : in     Kontaktu_Zenbakianitzeko;
                              I :    out Datuak.Datu;
                              ZL:    out Tel_Listak.Tel_Lista) is
   begin
      I  := K.Norena;
      ZL := K. Telefonoak;
   end Izen_Zbki_lista;

   procedure Gehitu_Zenbakia  (K : in out Kontaktu_Zenbakianitzeko;
                               Z : in     Telefono_Zenbaki) is
   begin
      Tel_Listak.Txertatu_Bukaeran (K. Telefonoak, Z);
   end Gehitu_Zenbakia;
```

Tel_Listak paketean

```
   procedure Txertatu_Bukaeran (ZL: in out Tel_Lista;
                                Z : in     Telefono_Zenbaki);
   procedure Sortu_Lista_Hutsa (ZL: out Tel_Lista)
```

c. Demagun orain zenbaki bakarreko hainbat kontaktu-lista biltzen dituen “kontaktu-listazko lista bat” honela definitu dugula:

```
Kont_Lista_Max : constant Integer := 10;
subtype I_Kont_Lista0_Max is Integer range 0 .. Kont_Lista_Max;
subtype I_Kont_Lista1_Max is Integer range 1 .. Kont_Lista_Max;
 type Kont_Lista_Taula is array (1 .. Kont_Lista_Max)
                          of Kontaktu_Listak.Kontaktu_Lista;
type Kont_Lista_Lista is
   record
      Info : Kont_Lista_Taula;
      Kop  : I_Kont_Lista0_Max;
end record;
```

Hainbat lista zenbakibakar batzen dituen lista bat emanda, berak dituen lista guztietako informazioa lista zenbakianitz bakar batean bilduko duen azpiprograma bat diseinatu eta inplementatu. Nahi baduzu erabili azpiprograma hau:

```
procedure Lista_Hutsa_Sortu (KLP: out Kont_zenbakianitz_Lista );
   --Post: KLP lista hutsa da.
```

```
procedure Batu_Listak (KLL:  in Kont_Lista_Lista;
                       KLP: out Kont_zenbakianitz_Lista) is
  --Aurre:
  --Post:  KLL listako kontaktu guztiekin osatu da KLP lista
begin
  Lista_Hutsa_Sortu (KLP);
   for I in 1.. Kont_Lista_Listak.Zenbat(KLL) loop
      Gehitu (Kont_Lista_Listak.Eman_Igarrena(KLL),
              KLP);
   end loop;
end Batu_Listak;
```