# 2013ko urtarrileko azterketa globalaren ebazpena
# Oinarrizko Programazioa
**(Ebaluazio globala, 2013-01-15)**

1. **GALDERA (2,5 puntu).** Demagun inplementatuta dagoela honako azpiprograma hau:

```
function Lehena_Da (N : Natural) return Boolean;
--Post: Emaitza=True N zenbaki lehena denean
--      Emaitza=False  bestela
```

Azpiprograma hori eta honako datu mota hau erabiliz

```
type Osoko_Bektore is array (1..10) of Natural;
```

1.1. **Inplementa** ezazu azpiprograma hau:

```
function Lehen_Lehenaren_Pos (B : Osoko_Bektore) return Natural;
--Post: Emaitza = bektoreko lehenengo zenbaki lehenaren posizioa
--      Emaitza=0 zenbaki lehenik ez badago.
```

```
function Lehen_Lehenaren_Pos (B : Osoko_Bektore) return Natural is
   --Post: Emaitza = bektoreko lehenengo zenbaki lehenaren posizioa
   --      Emaitza=0 zenbaki lehenik ez badago.
   Pos, I : Natural;
   I : Natural;
   Aurkitua : Boolean;
begin
   Pos := 0;
   I := 1;
   Aurkitua := False;
   while I <= B'Last and not Aurkitua loop
      if Lehena_Da(B(I)) then Aurkitua := True;
         Pos := I;
      else
         I := I + 1;
      end if;
   end loop;
   return Pos;
end Lehen_Lehenaren_Pos;
```

1.2. Osoko-bektore bat emanda, **espezifika** eta **inplementa** ezazu azpiprograma bat bi gauza itzuliko dituena:

- bektore bera baina zenbaki lehenak ordeztuta 1 zenbakiarekin (gainontzeko elementuak ez dira aldatuko)
- zenbaki bat, zenbat aldaketa egin diren adierazten duena.

```
procedure Aldatu (B: in out Osoko_Bektore; Aldaketak: out Natural) is
   -- post: B bektorean zenbaki lehenen ordez 1 zenbakia jarri da.
   --       Aldaketak: zenbat ordezkapen egin diren
begin
   Aldaketak:= 0;
   for I in B'First .. B'Last loop
      if Lehena_Da ( B(I)) then
         B(I) := 1;
         Aldaketak:= Aldaketak+1;
      end if;
   end loop;
end Aldatu;
```

2. **GALDERA (4,5 puntu)** Gehienez 15 karaktere dituzten kateak erazagutzeko balio du honako datu mota honek:

```
type Hitz is record
   Letrak: string(1..15);
   Kont: Natural;
end record;
```

2.1. **Inplementa** ezazu honako funtzio hau:

```
function Palindromoa (H: Hitz) return Boolean;
--Post: Emaitza=True H hitza palindromoa bada. Adibidez: “amama”
--      Emaitza=False bestela.
```

```
function Palindromoa (H: Hitz) return Boolean is
--Post: Emaitza=True H hitza palindromoa bada. Adibidez: “amama”
--         Emaitza=False bestela.
   I : Natural := H.Letrak'First;
   J : Natural := H.Kont;
begin
   while I < J and H.Letrak(I)=H.Letrak(J) loop
      I:= I+1;
      J:= J-1;
   end loop;
   return (I>=J);
end Palindromoa;
```

2.2. Hitz bat eta letra bat emanda, ea hitzean letra dagoen aztertuko duen `Dago` azpiprograma **espezifika** eta **implementa** ezazu.

```
function Dago (H: Hitz; Kar: Character) return Boolean is
   -- post: Aurkitua = True Kar karakterea H hitzean badago
   --       Aurkitua = False bestela
   I : Natural := H.Letrak'First;
   Aurkitua : Boolean;
begin
   I := H.Letrak'First;
   Aurkitua := False;
   while I <= H.Kont and not Aurkitua loop
      if H.Letrak(I) = Kar then
         Aurkitua := True;
      else
         I := I + 1;
      end if;
   end loop;
   return Aurkitua;
end Dago;
```

2.3. `H1` eta `H2` hitzak emanda, hitz batean eta bestean (bietan aldiberean) dauden letrekin beste hitz bat sortuko duen `Komunak` azpiprograma **espezifika** eta **implementa** ezazu**. Proba-kasuak ere definitu itzazu.**

*Adibidea:*

Hitzak “kalapita” eta “utopiko” badira hauek dira letra komunak: 'k', 'p', 'i' eta 't'.

Beraz, H hitza lau letra horiek bakarrik dituen edozein hitz izan daiteke, “kpit” esate baterako.

```
procedure Komunak (H1, H2: in Hitz; H: out Hitz) is
   -- Post: H hitzeko letrak H1ean eta H2an (bietan aldiberean) dauden letrak dira
   --       Adibidez H1: “kalapita”, H2: “utopiko” eta H:“kpit”
   begin
   P.Kont:= 0;
   for I in H1.Letrak'First .. H1.Kont loop
      if (not Dago(H, H1.Letrak(I))) and Dago(H2, H1.Letrak(I)) then
         H.Kont := H.Kont+1;
         H.Letrak(H.Kont) := H1.Letrak(I);
      end if;
   end loop;
end Komunak;
```

3. **GALDERA (3 puntu).** Urte osoko jarduerak biltzen dituen lista bat emanda, **diseina** eta **inplementa** ezazu azpiprograma bat hilabete bakoitzean zenbat jarduera hasten diren inprimatuko duena. Hilabetearen zenbakia eta jarduera kopurua azalduko dira lerro bakoitzean eta jarduera kopuruaren arabera ordenatuta idatziko dira, hau da, hasieran jarduera gehien dituzten hilabeteak eta bukaeran gutxien dituztenak.
Praktikan erabili ditugun paketeak erabili ahal izango dira ariketa hau egiteko. (ikus definizioak)

```
with Ada.Text_IO, Ada.Integer_Text_IO,J arduera_Listak, Jarduerak, Uneak;

procedure Hilebeteak_Kargaren_Arbera_inprimatu
            (L : in Jarduera_Listak.Jarduera_Lista) is
      -- Aurre: L lista urte oso batean egin diren jarduera-lista da
      -- Post: Inprimatu dira hileak eta bakoitzaren jarduera kopurua
      --       jarduera kopuruaren ordenatuta handienetik txikira

      type Hile_Taula_Mota is array (1..12) of Integer;
        -- Behar den datu-mota


   procedure Kontatu_Hileetako_Jarduerak (L : in Jarduera_Listak.Jarduera_Lista;
                                          HT: out Hile_Taula_Mota) is
    -- Post: HT taulak hile bakoitzean L listako zenbat jarduera egin diren dauka
    J : Jarduera_Mota;
    Ur, Hi, Eg : Integer;
      begin
    for I in HT'First .. HT'Last loop
       HT(I) := 0;
    end loop;
    for I in 1 .. Jarduera_Listak.Zenbat_Jarduera(L) loop
       Jarduera_Listak.Jarduera (L, I, J);
       Uneak.Banatu_Data (Jarduerak.Data(J), Ur, Hi, Eg);
       HT(Hi) := HT(Hi) + 1;
    end loop;
  end Kontatu_Hileetako_Jarduerak;

  function Maximoaren_Posizioa (T: Hile_Taula_Mota) return Natural is
    -- Post: T(Max_Pos) balioa T taulako balio maximoa da
    Max_Pos : Natural := 1;
  begin
    Max_Pos := 1 ;
    for I in T'First+1 ..T'Last loop
       if T(I) > T(Max_Pos) then
          Max_Pos := I;
       end if;
    end loop;
```

```
   return Max_Pos;
end Maximoaren_Posizioa;


   Pos: Natural;
   T: Hile_Taula_Mota;
begin
   Kontatu_Hileetako_Jarduerak (L, T);
   for I in T'First .. T'Last loop
      Pos:= Maximoaren_Posizioa(T);
      -- Idatzi Posgarren hilabetearen jarduerak
      Ada.Integer_Text_IO.Put(Pos,3);
      Ada.Text_IO.Put(". hilabeteak ");
      Ada.Integer_Text_IO.Put(T(Pos), 3);
      Ada.Text_IO.Put(" jarduera ditu.");
      -- Kendu maximo hori,
      -- jarri -1 Posgarren hilabetearen jarduera kopuruan
      T(Pos) := -1;
   end loop;
end Kontatu_Hileko_Jarduerak;
```

4. **HAUTAZKO GALDERA GEHIGARRIA** (Nota igotzeko, gainontzekoa gaindituz gero)

B osoko-bektore bat emanda (osagai kopurua bakoitia da eta osagai guztiak desberdinak dira), **diseina** eta **inplementa** ezazu azpiprograma bat beste bektore bat sortuko duena B-ko osagai berdinekin baina “ordena piramidalean” kokatuta.

Esaten dugu zenbaki batzuk ordena piramidalean daudela handiena erdiko posizioan badago, hurrengo bi zenbaki handienak erdikoaren alde bietan badaude, hurrengo biak aurreko bien alboetan, eta horrela gainontzekoak ere.

Adibidea:
B bektoreko osagaiak hauek badira: <**8, 15, 7, 2, 4, 9, 10**>,
Bektore berrirako aukera bat hau izan daiteke:<**2, 7, 9, 15, 10, 8, 4**>.
Beste aukera batzuk dira ondokoak: <4, 8, 10, 15, 9, 7, 2>, <2, 8, 9, 15, 10, 7, 4>...
Zein da zure algoritmoak lortzen duena?

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

procedure Ordena_Piramidala_Proba  is
   N: constant Natural := 7;
   type Osoko_Bektore is array(1..N) of Integer;
   -- Zenbaki bakoitia izan behar da N hori

   procedure Trukatu (B : in out Osoko_Bektore; I, J: in Integer) is
      -- post: B bektoreko I eta J posizioko balioak elkarrekin trukatu dira
      Lag : Integer;
   begin
      Lag  := B(I);
      B(I) := B(J);
      B(J) := Lag;
   end Trukatu;

     function  Tarteko_Minimoaren_Posizioa  (B:  Osoko_Bektore;  I,J:  Natural)
return Natural is
      -- pre: I, J zenbakiak B bektoreko indizeak dira, I<=J.
      -- post: B(I..J) bektore-tarteko balio minimoaren posizioa
      Min_Pos: Natural;
   begin
      Min_Pos:= I;
      for K in I..J loop
         if B(K) < B(Min_Pos) then
            Min_Pos := K;
         end if;
      end loop;
      return Min_Pos;
   end Tarteko_Minimoaren_Posizioa;




   procedure Ordena_Piramidala (B1: in Osoko_Bektore;
                                B2: out Osoko_Bektore) is
      -- pre: Los elementos de A son naturales y todos distintos
      -- post: B formado con los elementos de A ordenados en forma piramidal
      Ezkerretik: Boolean;
      I, J, Pos : Natural;
```

```
   begin
      B2:= B1;
      Ezkerretik := True;
      I:= B2'First;
      J:= B2'Last;
      while I < J loop
         Pos := Tarteko_Minimoaren_Posizioa (B2, I, J) ;
         if Ezkerretik then
            Trukatu (B2, I, Pos);
            I := I+1;
            Ezkerretik := False;
         else
            Trukatu (B2, J, Pos);
            J := J-1;
            Ezkerretik := True;
         end if;
      end loop;
   end Ordena_Piramidala;


   B1: Osoko_Bektore := (8,15,7,2,4,9,10);
   B2: Osoko_Bektore;
begin
   Put_Line("Bektore honekin egingo dugu aproba (8,15,7,2,4,9,10) ");
   Put_Line("Ordena piramidalean honela utziko genuke: ");
   Ordena_Piramidala (B1, B2);
   for I in B2'Range loop
      Ada.Integer_Text_IO.Put( B2(I), 3);
   end loop;
end Ordena_Piramidala_Proba;


 --  Bektore honekin egingo dugu aproba (8,15,7,2,4,9,10)
 --  Ordena piramidalean honela utziko genuke:
 --    2  7  9 15 10  8  4
```

**jarduera_listak.ads**

```
-- Listan gehienez 100 jarduera egongo dira.
-- Jarduerak dataren arabera ordenatuta daude.
-- Jarduera baten iraupena zenbait egunetakoa izan daiteke.
-- Egun bakoitzean gehienez jarduera bat dago.

procedure Irakurri_Jarduera_Lista (F_Izena: String;
                                   J_Lista: out Jarduera_Lista);
--Aurre: F_izena testu-fitxategian gehienez 20 jarduera deskribatzen dira.
--       Jarduera bakoitzaren datuak hiru lerrotan deskribatzen dira:
--       Kodea, Deskribapena, Data (modu trinkoan), ordua segunduetan eta iraupena segundutan.
--Post: J_Lista listan fitxategiko jarduera guztiak daude kodearen
--       arabera ordenatuta txikitik handira.

procedure Jarduera (J_Lista: in Jarduera_Lista; I: in Natural;
                    J: out Jarduerak.Jarduera_mota);
--Aurre: : 1<= I <=Zenbat_Jarduera(J_Lista)
--Post: J_lista-ko Igarren jarduera J da.

procedure Sortu_Lista_Hutsa (J_Lista: out Jarduera_Lista);
--Post: J_Lista = lista hutsa

procedure Txertatu_Ordenatuan (J_Lista: in out Jarduera_Lista;
                               J      : in     Jarduerak.Jarduera_mota) ;
   --Aurre: J_Lista-ko osagaiak ordenatuta daude kodearen arabera.
   --Post: J_lista-n J jarduera txeratu da.
   --      J_Lista-ko osagaiak ordenatuta daude kodearen arabera.

function Zenbat_Jarduera (J_Lista: Jarduera_Lista ) return Natural;
--Post: Emaitza = listako osagai kopurua

function Pos_Jarduera (J_Lista: Jarduera_Lista; K: natural) return natural;
 --Aurre: K kodea duen jarduera bat dago J_Lista listan
 --Post: K kodea duen jarduerareb posizioa = Emaitza

procedure Idatzi_Jarduera_Lista (J_Lista: Jarduera_Lista);
--Post: Irteera estandarrean J_lista-ko jarduerak idatzi dira.
```

**jarduerak.ads**

```
   -- Jarduera bat bere lau ezaugarrirekin definitzen dugu:
   --   Kodea: Jarduera bakoitzak berea dauka.
   --              Zenbaki oso bat da 100 eta 399 artekoa.
   --          Esplorazio-lanen kodeak 100 eta 199 artekoak dira.
   --          Material-bilketako lanen kodeak 200 eta 299 artekoak dira.
   --          Kimika-azterketenak 300 eta 399 artekoak dira.
   --   Data: jarduera abian jartzeko eguna
   --             (Urtea*10000+Hilea*100+Eguna formatuan)
   --   Segundoak: jarduera abian jartzeko unea
   --             (segundo kopurua gauerditik kontatuta)
   --   Iraupena: Zenbat segundotan bukatu behar den.

  procedure Irakurri_Jarduera(F : in  Ada.Text_Io.File_Type; J: out Jarduera_mota);
  -- Aurre: F fitxategi-identifikadorea zabalik dagoen testu-fitxategi batena da.
  --           hurrengoirakurtzeko hurrengo lerroan lau zenbaki dauzkana
  --           jarduera baten kodea, data, segundoak eta iraupena direnak
  -- Post: J jarduerak F fitxategiaren hurrengo lerro horren datuak ditu.

  procedure Sortu_Jarduera(Kodea, Data, Seg, Ir: in integer; J: out Jarduera_mota);
  -- Aurre: Kodea, Data, Seg eta Ir zenbakiak jarduera baten ezaugarrien balioak
  -- Post: J jardueraren eremuen balioak Kodea, Data, Seg eta Ir dira.

  procedure Aldatu_Data(Data: in integer; J: in out Jarduera_mota);
  -- Post: J jardueraren data Data-ren balioa da.

  procedure Aldatu_Seg(Seg: in integer; J: in out Jarduera_mota);
  -- Post: J jarduera abian jartzeko unea Seg-ren balioa da.

  procedure Aldatu_Iraupena(Ir: in integer; J: in out Jarduera_mota);
  -- Post: J jardueraren iraupena Ir-aren balioa da.

  Function Kodea(J: Jarduera_mota) return natural;
  -- Post: Emaitza J jarduararen kodea da.

  procedure Unea(J: in Jarduera_mota; Data, Segundoak: out Integer);
  -- Post: Data-ren balioa J jarduera abian jartzeko eguna da,
  --          eta Segundoak-ena abian jartzeko unearena

  function Data(J: Jarduera_mota) return Integer;
  -- Post: Emaitza J jardueraren data da.

  function Segundoak(J: Jarduera_mota) return Integer;
  -- Post: Emaitza J jarduera abian jartzeko unea (segundoak) da.

  function Iraupena(J:Jarduera_mota) return Integer;
  -- Post: Emaitza J jardueraren data iraupena da.

  procedure Idatzi_Jarduera(J: in Jarduera_mota);
  -- Post:  Idatzi dira lerro batean J jardueraren kodea, data, unea eta iraupena
```

**uneak.ads**

```
-- UNE bat (edo denbora bat) bi zenbaki osorekin errepresentatzen dugu:
--    - Data bat (urtea, hilea eta eguna dauzkana)
--       Adibidez: 20120806 zenbakia 2012 abuztuaren 6a da.
--    - Segundoak (gauerdiaz gero pasatako segundo kopurua)
--       Adibidez: 3620 zenbakia 01:00:20 da, hots, ordubata eta 20 segundo
--       Noski, segundo horiek banatu daitezke ordu, minutu eta segundotan
--  Adibidez:  2012/11/02 datako 12:05:10 ordua hau litzateke:
--      Data=  20121102 (2012*10000+11*100+2)
--      Segundoak= 43510 (=12*3600+5*60+10)
--      Unea = (Data, Segundoak)

procedure Oraingo_Unea (Data, Segundoak :    out Integer);
--Post: Data eta Segundoak ordenagailuko erlojuak une honetan daukan
--       data eta segundoak (gauerdiaz gero pasa direnak) dira.

procedure Unea (Urt, Hil, Egn,
                H, M, S : Integer;
                Data, Segundoak : out Integer);
--Aurre: Urt, Hil, Egn, H, M eta S dira urtea, hilea, eguna, ordua,
--        minutua eta segundoa dira hurrenez hurren.
--       1<=Hil<=12, 1<=Egn<= días del mes(Hil), 0<=H<=23, 0<=M<=59, 0<=S<=59
--Post: Data = Urt*10000+Hil*100+Egn.
--      Segundoak H orduak M minutuak eta S segunduak direnean
--      gauerdiaz gero pasatako segundo kopuru osoa da. Segundoak=H*3600+M*60+S.

procedure Irakurri_Unea (F : in Ada.Text_Io.File_Type;
                         Data, Segundoak: out Integer);
--Aurre: F fitxategia zabalik dago. irakurtzeko prest dauka bi osoko:
--        D data bat eta Seg segundo kopuru bat.
--Post: Data=D eta Segundoak=Seg.

procedure Banatu_Data (Data :  Integer;  Urt, Hil, Egn : out Integer);
--Post: Urtea (Urt), Hilabetea (Hil) eta eguna (Egn) Datari dagozkionak dira.

procedure Banatu_HMStan (Segundoak :  Integer; H, M, S:    out Integer);
--Post: H orduak, M minutuak eta S segundoak Segundoak kopuruari dagozkionak dira.
--       0<= Segundoak <=86.000 (egunbeteko segundoak)

function Kendura (Data1, S1, Data2, S2 : Integer) return Integer;
--Aurre: (D1, S1)<=(D2, S2); D1 eta D2 datak dira; S1 eta S2 segundoak..
--Post: (D2, S2) = (D1, S1) + Emaitza segundoak.

function Trinkotu_Data (Urt, Hil, Egn: Integer) return Integer;
--Post: Emaitza data trinkotu bat da.
--      Emaitza = Urt*10000 + Hil*100 + Egn.

function Trinkotu_Unea (H, M, S: Integer) return Integer;
--Aurre: 0<= H<= 23, 0<= M<=59, 0<= S<=59,
--Post: Emaitza = H*3600 + M*60 * S.

procedure Idatzi_Data (Data : Integer);
--Post: 3 osoko idatzi dira; dataren urtea, hilabetea eta eguna UUUU/HH/EE formatoan.

procedure Idatzi_Ordua (Segundoak: Integer);
--Post: 3 osoko idatzi dira,
--      Segundoak kopuruari dagozkion ordua, minutua eta segundoa, HH/MM/SS formatoan.

procedure Idatzi_Unea (Data, Segundoak: Integer);
--Post: 6 osoko idatzi dira,
--      dataren urtea, hilabetea eta eguna UUUU/HH/EE formatoan
--      eta Segundoak kopuruari dagozkion ordua, minutua eta segundoa, HH/MM/SS formatoan.
```