

Ariketa 1:

Zerrenda bateko elementu errepikakorrak ezabatzen dituen programa garatu ezazu.

- 1) Sortu datu egitura;
- 2) Dago: Zerrenda bat eta posizio bat emanda, posizio horretan dagoen elementua posizio hori baina lehenagoko beste posizio batean agertzen bada TRUE itzuliko du, FALSE bestela.
- 3) Ezkerretara_desplazatu: Zerrenda eta posizio bat emanda, posizio horretatik aurrerako elementu guztiak ezkerretara desplazatuko ditu.
- 4) Zerrenda_pantailaratu: Zerrenda emanda, zerrenda hori pantailaratuko du.
- 5) Errepikatuak_ezabatu: Zerrenda bat emanda, zerrenda horretako elementu errepikakorrak ezabatuko ditu.

ADIB:

Zerrenda: 1 1 3

Zerrenda irteera: 1 3

Zerrenda: 1 3 3 5 7 7 7 8

Zerrenda irteera: 1 3 5 7 8

```
package datuak is
```

```
Max_Elem : constant Integer := 1000;
```

```
type osokoen_bektorea is array (1..Max_Elem) of integer;
```

```
type osokoen_zerrenda is record  
    zenbakiak : osokoen_bektorea;  
    zenbat    : Integer;
```

```
end record;
```

```
end datuak;
```

```
with datuak; use datuak;
```

```
function dago (L: in osokoen_zerrenda; ind: in integer) return boolean is
```

```
    i:integer:=1;
```

```
    topatua:boolean:=false;
```

```
begin
```

```
    loop exit when i>=ind or topatua;
```

```
    if L.zenbakiak(i)=L.zenbakiak(ind) then
```

```
        topatua:=true;
```

```
    end if;
```

```
    i:=i+1;
```

```
end loop;
```

```
return topatua;
```

```
end dago;
```

```

with datuak;
use datuak;
procedure desplazatu_ekzerrera(L: in out osokoen_zerrenda; ind: in
integer) is
    i:integer:=ind;
begin
    loop exit when i=L.zenbat;
        L.zenbakiak(i):=L.zenbakiak(i+1);
        i:=i+1;
    end loop;
    L.zenbat:=L.zenbat-1;
end desplazatu_ekzerrera;

```

```

with Ada.Integer_Text_Io;
use Ada.Integer_Text_Io;
with datuak; use datuak;

procedure zerrenda_idatzi (L : in      osokoen_zerrenda ) is
    I:integer;
begin
    for I in 1..L.zenbat loop
        put (L.zenbakiak(I));
    end loop;
end zerrenda_idatzi;

```

```

with datuak, dago, desplazatu_ekzerrera;
use datuak;

procedure errepikatuak_ezabatu(L: in out osokoen_zerrenda) is
    i:integer:=2;
begin
    loop exit when i>L.zenbat;
        if dago(L, i) then
            desplazatu_ekzerrera(L,i);
        else
            i:=i+1;
        end if;
    end loop;
end errepikatuak_ezabatu;

```

```

with Ada.Text_IO;
use Ada.Text_IO;
with Datuak;
use Datuak;
with errepikatuak_ezabatu, zerrenda_idatzi;

procedure proba_errepikatuak_ezabatu is

    V1 : osokoen_zerrenda;

begin

    Put_Line("Aurreneko proba.
errepikatuak_ezabatu(1,2,3,4,5,6,7,8,9,10)");
    V1.zenbakiak := (1,2,3,4,5,6,7,8,9,10, OTHERS => 0);
    V1.zenbat := 10;
    zerrenda_idatzi(V1);
    New_Line;
    Put_Line("Errepikatuak ezabatu ondoren:");
    errepikatuak_ezabatu(V1);
    zerrenda_idatzi(V1);
    New_Line(3);
    Put_Line("Sakatu return jarraitzeko");
    Skip_Line;

    Put_Line("Bigarren proba.
errepikatuak_ezabatu(1,1,1,3,4,3,5,5,5,6)");
    V1.zenbakiak := (1,1,1,3,4,3,5,5,5,6, OTHERS => 0);
    V1.zenbat := 10;
    zerrenda_idatzi(V1);
    New_Line;
    Put_Line("Errepikatuak ezabatu ondoren:");
    errepikatuak_ezabatu(V1);
    zerrenda_idatzi(V1);
    New_Line(3);
    Put_Line("Sakatu return jarraitzeko");
    Skip_Line;

    Put_Line("Hirugarren proba.
errepikatuak_ezabatu(1,3,4,3,5,5,5,6,6,6)");
    V1.zenbakiak := (1,3,4,3,5,5,5,6,6,6, OTHERS => 0);
    V1.zenbat := 10;
    zerrenda_idatzi(V1);
    New_Line;
    Put_Line("Errepikatuak ezabatu ondoren:");
    errepikatuak_ezabatu(V1);
    zerrenda_idatzi(V1);
    New_Line(3);
    Put_Line("Sakatu return jarraitzeko");
    Skip_Line;

end proba_errepikatuak_ezabatu;

```

Ariketa 2: Mamu bokalak

Karakterez osatutako zerrenda bat emanik, pantailaratu itzazu zerrendako hitzak (lerro bakoitzean bat) non bokalak ez diren agertuko:

Adibidez:

| | | | | | | | | | |
|--|---|---|---|---|---|--|--|---|---|
| | k | a | i | x | o | | | n | i |
|--|---|---|---|---|---|--|--|---|---|

Irteera:

kx
n

Erabili beharreko datu egitura:

type karaktereen_zerrenda is array (Integer range <>) of Character;

Garatu beharreko azpiprogramak:

```
function/procedure bokala_da (...) ... is
--aurre: karaktere bat izango du sarrera bezala
--post: karakterea bokala bada orduan TRUE, bestela FALSE

function/procedure hitza_salto (...) is
-- Aurre: Sarrera zerrenda bat eta indize bat. Zerrendak hizkiak eta
hutsuneak ditu -- eta 1 <= indizea <= Z'Last + 1
-- Indizea <= Z'Last bada orduan posizio horretan hizki bat egongo da.
-- Post: Zerrendako hizkiak salto egingo ditu hutsune bat bilatu arte
eta posizio hori itzuliko du. Ez bada hutsunerik bilatzen orduan indizea
= Z'Last + 1 izango da

function/procedure hutsuneak_salto (...) is
-- Aurre: Sarrera zerrenda bat eta indize bat. Zerrendak hizkiak eta
hutsuneak ditu eta 1 <= indizea <= Z'Last + 1
-- Post: Zerrendako hutsuneak salto egingo ditu hizki bat bilatu arte
eta posizio hori itzuliko du.
-- Ez bada hizkirik bilatzen orduan indizea = Z'Last + 1 izango
da

function/procedure idatzi_zerrenda (...) is
-- aurre: sarrera zerrenda bat izango da.
-- post: zerrendako hizki guztiak pantailaratzen dira, letik Zren
luzera bitarte

function/procedure bokalik_gabe_idatzi (...) is
-- Aurre: Sarrera zerrenda bat izango da. Zerrendak hizkiak eta
hutsuneak ditu
-- Post: Zerrendako hitzak idatzi dira bokalak kenduta
```

```

package datuak is
    type karaktereen_zerrenda is array (Integer range <>) of Character;
end datuak;

```

```

function bokala_da(t: in character) return boolean is
    --aurre: karaktere bat izango du sarrera bezala
    --post: karakterea bokala bada orduan TRUE, bestela FALSE

    emaitza:boolean:=false;
begin
    if (t='a' or t='e' or t='i' or t='o' or t='u' or
        t='A' or t='E' or t='I' or t='O' or t='U') then
        emaitza:=true;
    end if;
    return emaitza;
end bokala_da;

```

```

with datuak; use datuak;
with ada.text_io; use ada.text_io;
with bokala_da;
procedure hitza_salto (Z: in karaktereen_zerrenda;
                      indizea: in out Integer) is
    -- Aurre: Sarrera zerrenda bat eta indize bat. Zerrendak hizkiak eta
    hutsuneak ditu      -- eta 1 <= indizea <= Z'Last + 1
    -- Indizea <= Z'Last bada orduan posizio horretan hizki bat egongo da.
    -- Post: Zerrendako hizkiak salto egingo ditu hutsune bat bilatu arte
    eta posizio hori itzuliko du. Ez bada hutsunerik bilatzen orduan indizea
    = Z'Last + 1 izango da

    hutsune_bat_topatua: Boolean;
begin
    hutsune_bat_topatua := False;
    loop exit when indizea > Z'Last or hutsune_bat_topatua;
    if Z(indizea) = ' ' then
        hutsune_bat_topatua := True;
    else
        if not bokala_da(Z(indizea)) then
            Put(Z(indizea));
        end if;
        indizea := indizea + 1;
    end if;
end loop;
New_Line;
end hitza_salto;

```

```

with datuak; use datuak;
procedure hutsuneak_salto (Z: in karaktereen_zerrenda;
                           indizea: in out Integer) is
    -- Aurre: Sarrera zerrenda bat eta indize bat. Zerrendak hizkiak eta
    hutsuneak ditu eta 1 <= indizea <= Z'Last + 1
    -- Post: Zerrendako hutsuneak salto egingo ditu hizki bat bilatu arte
    eta posizio hori itzuliko du.
    -- Ez bada hizkirik bilatzen orduan indizea = Z'Last + 1 izango da

    hitz_bat_bilatua: Boolean;
begin
    hitz_bat_bilatua := False;
    loop exit when indizea > Z'Last or hitz_bat_bilatua;
        if Z(indizea) /= ' ' then
            hitz_bat_bilatua := True;
        else
            indizea := indizea + 1;
        end if;
    end loop;
end hutsuneak_salto;

```

```

with Ada.Text_IO, datuak;
use Ada.Text_IO, datuak;
procedure idatzi_zerrenda (Z : in karaktereen_zerrenda ) is
    -- aurre: sarrera zerrenda bat izango da.
    -- post: zerrendako hizki guztiak pantailaratzen dira, letik Zren
    luzera bitarte
begin
    for I in 1..Z'Last loop
        Put(Z(I));
    end loop;
end idatzi_zerrenda;

```

```

with datuak; use datuak;
with hutsuneak_salto, hitza_salto;
procedure bokalik_gabe_idatzi (Z: in karaktereen_zerrenda) is
    -- Aurre: Zerrendak hizkiak eta hutsuneak ditu
    -- Post: Zerrendako hitzak idatzi dira bokalak kenduta
    i: Integer;
begin
    i:= 1;
    loop exit when i > Z'Last;
        hutsuneak_salto(Z, i);
        hitza_salto(Z, i);
    end loop;
end bokalik_gabe_idatzi;

```

```

with Ada.Text_IO;
use Ada.Text_IO;
with datuak;
use datuak;
with bokalik_gabe_idatzi, idatzi_zerrenda;
procedure nagusia is
    Z1: karaktereen_zerrenda(1..30);
begin
    Put_Line("Aurreneko proba: ( Kaixo ni_ )");
    Z1 := ( " kaixo ni_ " );

```

```

idatzi_zerrenda(Z1);
New_Line;
Put_Line("Bokalak ezabatu ondoren:");
bokalik_gabe_idatzi(Z1);
New_Line(3);
Put_Line("Sakatu return jarraitzeko");
Skip_Line;
end nagusia;

```

Ariketa 3: Roland Garros

Roland Garroseko partidu bat errepresentatzeko datu egitura bat sortzeko eskatzen digute. Tennis partidu batetan 2 jokalaria daude, eta jokalaria bakoitzeko jolastu diren set-etan zenbat joku egin dituen gorde beharko dugu. Roland Garros-en partidu bakoitzak gehienez 5 set izan ditzake, eta partidu bat irabazteko 3 set irabazi behar dira. Jokalari batek lehenengo hiru set-ak irabaziko balitu, orduan ez dira jolastuko set gehiago jada garaile bat dagoelako. Bestela 4 set jolastuko dira, eta oraindik ez bata ez besteak ez balitu 3 set irabaziko 5. Bat jolastu beharko litzateke.

Datu egitura beteta egongo da eta irabazlea zein izan den bilatu nahi dugu.

Lehenik pentsatu datu egitura bat informazio hori gordetzeko.

Behin datu egitura pentsatu ondoren, bete datuak programa nagusian eta honako azpiprogramak garatu:

```

procedure/function bat_inprimatu(...) is
  --pre: Gutxienez 3 sets. Sarrera bezala Partiduak eta jokalaria
  zerbakia izango dira.
  --post: jokalaria baten set-ak inprimatuko dira

procedure/function biak_inprimatu(...) is
  --pre:Gutxienez 3 sets. Sarrera bezala partiduak jasoko dira
  --post:Jokalari bakoitzerako irabazitako set bakoitza inprimatuko da
  --- jokalaria_baten_jokoak_inprimatu azpiprogramari deituko dio

procedure/function topatu_irabazlea (...) is
  --pre: jokalaria batek 3 set irabazi ditu
  --post: irabazi duen jokalariairen zerbakia (1 edo 2) itzuliko dugu

```

1. Bertsioa:

zenbat_sets

3

sets

| | | | | | |
|---|---|---|---|--|--|
| 1 | 2 | 1 | 1 | | |
| 2 | 6 | 6 | 6 | | |

1 bertsioa:

```
package datu_mota is
  type T_sets is array(1..2,1..5) of Natural;
  type T_Partida is record
    Zenbat_sets: Natural;
    Sets: T_Sets;
  end record;
end datu_mota;
```

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with datu_mota; Use datu_mota;
with biak_inprimatu;
with topatu_irabazlea;
```

```
procedure nagusia is
  partida: T_partida;
  irabazlea: Positive;
```

```
begin
  partida.zenbat_Sets:=3;
  partida.Sets:=(2,1,1,0,0),(6,6,6,0,0));
  biak_inprimatu(partida);
  irabazlea:=topatu_irabazlea(partida);
  Put_Line("Irabazlea: ");
  Put(irabazlea);New_Line;
  New_Line;
  New_Line;
  Put_Line("Agur.");
end nagusia;
```

```
with Ada.Text_IO; use Ada.Text_IO;
with ada.Integer_Text_IO; use ada.Integer_Text_IO;
with datu_mota; use datu_mota;
procedure bat_inprimatu(Partida: in T_Partida; jokalari_zenb: in
positive) is
```

```
  ---pre: Gutxienez 3 sets
  ---post: jokalari baten set-ak inprimatuko dira
```

```
begin
  Put("Lortutako jokoak");
  Put(jokalari_zenb);New_Line;
  for I in 1..Partida.Zenbat_Sets loop
    Put(Partida.Sets(jokalari_zenb,I));
  end loop;
  New_Line;
end bat_inprimatu;
```

```
with datu_mota; use datu_mota;
with bat_inprimatu;
procedure biak_inprimatu(Partida: in T_Partida) is
  ---pre:Gutxienez 3 sets
  ---post:Jokalari bakoitzerako irabazitako set bakoitza inprimatuko da
  --- jokalari_baten_jokoak_inprimatu azpiprogramari deituko dio
```

```
begin
  bat_inprimatu(Partida,1);
  bat_inprimatu(Partida,2);
end biak_inprimatu;
```



```

with datu_mota; use datu_mota;
function topatu_irabazlea (Partida: in T_Partida) return Positive is
  ---pre: jokalaria batek 3 set irabazi ditu
  ---post: irabazi duen jokalaria itzuliko dugu
  irabazlea:Positive;
  kontagailua:Natural:=0;
begin
  for I in 1..Partida.zenbat_Sets loop
    if Partida.Sets(1,I) > Partida.Sets(2,I) then
      kontagailua:=kontagailua+1;
    end if;
  end loop;
  if kontagailua >= 3 then
    irabazlea:=1;
  else
    irabazlea:=2;
  end if;
  return(irabazlea);
end topatu_irabazlea;

```

2. Betsioa:

| | | | | | | | | | | | | | |
|------------|---|---|---|---|---|--|--|---|--|---|---|--|--|
| zenbat_set | 3 | | | | | | | | | | | | |
| jokalariak | | | | | | | | | | | | | |
| | <table><tr><td>2</td><td></td><td>1</td><td>1</td><td></td><td></td></tr><tr><td>6</td><td></td><td>6</td><td>6</td><td></td><td></td></tr></table> | 2 | | 1 | 1 | | | 6 | | 6 | 6 | | |
| 2 | | 1 | 1 | | | | | | | | | | |
| 6 | | 6 | 6 | | | | | | | | | | |

3. Aukera:

| | | | | | | |
|------------|--|---|---|---|--|--|
| zenbat_set | 3 | | | | | |
| Jokalari1 | <table><tr><td>2</td><td>1</td><td>1</td><td></td><td></td></tr></table> | 2 | 1 | 1 | | |
| 2 | 1 | 1 | | | | |
| Jokalari2 | <table><tr><td>6</td><td>6</td><td>6</td><td></td><td></td></tr></table> | 6 | 6 | 6 | | |
| 6 | 6 | 6 | | | | |

4. Aukera:

| | | | | | | |
|------------|--|---|---|---|--|--|
| zenbat_set | 3 | | | | | |
| set-ak | <table><tr><td>2</td><td>1</td><td>1</td><td></td><td></td></tr></table> | 2 | 1 | 1 | | |
| 2 | 1 | 1 | | | | |
| zenbat_set | 3 | | | | | |
| set-ak | <table><tr><td>6</td><td>6</td><td>1</td><td></td><td></td></tr></table> | 6 | 6 | 1 | | |
| 6 | 6 | 1 | | | | |