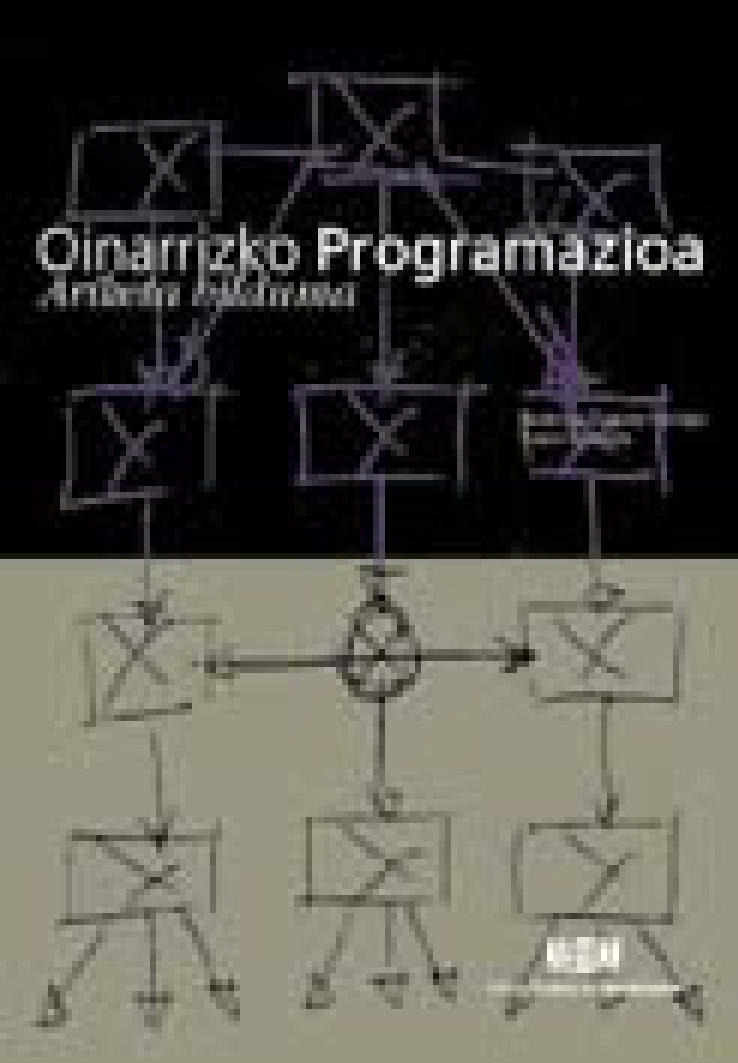
Oinarrizko Programazioa



OINARRIZKO PROGRAMAZIOA

ARIKETA BILDUMA

Arantza Diaz de Ilarraza
Kepa Sarasola

Udako **Euskal Unibertsitatea**
Bilbo, 1999

ISBN: 84-8438-002-5
Lege-gordailua: SS-1052-99

Inprimategia: ANTZA, Lasarte Oria
Azaleko diseinua: Iñigo Ordozgoiti

Banatzaileak: UEU. Concha Jenerala 25, 4. BILBO telf. 94-4217145
e-mail: bulegoa@ueu.org www.ueu.org

Zabaltzen: Igerabide, 88 DONOSTIA

Aurkibidea

Sarrera

Konputagailuen programazioan lehenengo pausoak egin nahi dituenarentzat sortu dugu liburu hau. Gehienetan unibertsitatean erabiliko da; informatika fakultateetan ez ezik, informatikaren sarrera egiten duten ikasgai guztietan ere. Adibidez: informatikan, enpresa ikasketetan, ingeniaritza ikasketetan, ingeniaritza teknikoetan, zientzi fakultateetan eta abar.

Konputagailuen programazioaren oinarrizko kontzeptuak landu nahi dira ariketen bidez, azalpen teoriko labur baten ostean: programazio agintzailearen kontrol-egiturak, oinarrizko datu-egituren diseinua, eta azpiprogramen erazagupena eta erabilpena. Mundu honetara lehenengo aldiz hurbiltzen denarentzat ez da egokiena zuzenean programazio-lengoaia bat ikastea. Askoz egokiagoa da programazio-lengoaia agintzaile guztiek konpartitzen dituzten kontzeptuak ezagutzea, eta abiapuntu horretatik gero programa idaztea lengoaia bat erabiliz. Euskaraz erraz irakurtzen diren algoritmoak asmatu beharko ditu ikasleak kontzeptu komun horiek lantzeko, beti ere, programazio-lengoaien sintaxi-arau estuak alboratuz.

Kapitulu bakoitza programazioko kontzeptu bati dagokio, eta zailtasun txikienetik handienera ordenatuta azaltzen dira liburuan zehar. Algoritmoen zehaztapenak eta azpiprogramak, funtsezko kontzeptuak direnez, salbuespenak izan dira; bina kapitulu eskaini dizkiegu bi zailtasun-maila bereiziz. Kapituluz kapitulu landu ditugun kontzeptuak hauexek izan dira:

1. Algoritmoen oinarrizko osagaiak. Algoritmo sinpleak dira.
   Datu motak: osokoa, erreala, boolearra eta karakterea
   Ekintzak: esleipena, irakurketa eta idazketa.
   Kontrol-egiturak: baldin eta bitartean
2. Algoritmo sinpleen zehaztapena.
   Aurrebaldintza eta post-baldintzaren bidez.
3. Sekuentzien tratamendua.
   Sekuentzia baten korritze-prozesu desberdinak aztertzen dira; adibidez, elementu nabarmen baten (edo batzuen) bilaketa, edo sekuentzia baten elementu guztiekin eragiketa bat egitea proposatzen da. Hasierako ariketetan sekuentziako elementuak irakurriz lortuko dira; bukaerako ariketetan, ordea, sortu egin beharko dira tratatu beharreko elementuak.
4. Azpiprogramen definizioa eta erabilera.
   Azpiprograma kontzeptua lantzea izan da helburua, problemen ebazpena errazteko baliagarria dela erakutsiz. Enuntziatu multzo honetan batetik azpiprogramen erabilera lantzen da, eta bestetik, azpiprogramen definizioa.

5. Azpiprogramen zehaztapena.
6. Azpiprogramen definizioa eta erabilera.
   Aurretik egindako algoritmo batzuk birplanteatu egiten dira hemen azpiprograma bezala egiteko.
7. *Bektorea* datu-egitura. Zenbait ariketa Ada programazio-lengoaian lantzen hasten da.
8. *Matrizea* datu egitura
9. Programa aurreratuak.
   Bektore eta *erregistro* datu-motekin datu-egitura berriak sortuz, sinpleak ez diren programak planteatzen dira.

Hasierako kapituluetako ebazpenetan notazio algoritmikoa erabili dugu, baina azkeneko kapituluetan Ada programazio-lengoaiara jo dugu pixkanaka. Bi arrazoi eman behar ditugu horrela jokatzeko: alde batetik, ikasketa-prozesuaren barruan programazio-lengoaia bat apurka sartzea metodologikoki egokia iruditzen zaigulako, eta bestetik, ikasleari, gehiegi luzatu gabe, bere lehenengo programak inplementatzeko aukera ematen zaiolako.

Ada ez den beste lengoaia bat aukeratzea bazegoen; adibidez: C, Java, Pascal, Prolog edo Miranda. Hori eztabaidagarria da, jakina. Beste aukerak ere onak ziren, baina programazio sendo eta metodologikorako tresna egokienak eskaintzen dituelakoan aukeratu dugu Ada; beti ere eredu agintzailearen barruan.

Adaren tresna ahaltsuenak ez dira agertuko hasi berrientzako liburu honetan. Berau ezin da hartu Ada ikasteko helburuarekin; Adaren erabilera oso murritza eginez, lengoaiaren baliabide oinarri-oinarrizkoak baino ez dira erabiltzen gure ariketetan. Azalpen teorikoak labur-labur eman dira.

Liburu honek orain dela 15 urte argitaratu zen beste liburu baten eguneraketa, moldaketa eta hobekuntza izan nahi du. Orduko liburuak *"Programatzeko algoritmoak. Ariketa Bilduma"* izenburua zuen eta UEUk berak atera zuen. Hamabost urtetan euskarak eta informatikak izan dituzten aldaketek behartu gaituzte liburu berri hau argitaratzera.

Etorkizunean Interneteko edo paperezko bertsio berriei begira, erroreak edo oharrak jasotzea interesatzen zaigu. Liburu honetan errorerik aurkitzen baduzu edo aholkurik luzatu nahi badiguzu jo ezazu jipsagak@si,ehu,es posta-helbidera. Eskerrik asko.

Donostia, 1999ko uztailaren 16an

Arantza Díaz de Ilarraza
Kepa Sarasola

Esker emanez

Ariketa gehienak geuk asmatuak izan dira, baina ezin utzi aipatu gabe inspirazio-iturri izan direnak: batetik UPV-EHUko Informatika Fakultateko *Oinarrizko Programazioa* irakasgaiko azterketa-ariketak (hainbat urtetakoak eta hainbat irakaslerenak), eta bestetik, bide honetan aspaldiko laguna dugun ondoko liburuaren egileei.

Peyrin, Jean-Pierre eta Scholl, Pierre-Claude

ALGORITHMIQUE ET PROGRAMMATION

Laboratoire IMAG, bp53 x, 38041 GRENOBLE-CEDEX, 1982

Ariketa-bilduma honen zirriborroak sufritu dituzten ikasleek zenbait akats zuzentzeko iradokizunak helarazi dizkigute.

Lehenengo hiru kapituluetako azalpen teorikoan hainbat adibide eta deskribapen Xabier Artolak idatzitako apunteetatik jaso ditugu, baita Elhuyarren euskaratutako liburu honetatik ere:

Watt, D., Wichmann, B., Findlay, W

"ADA Lengoaia eta Metodologia"

EHUko Argitalpen Zerbitzua Leioa 1996.

Bukatzeko, eskerrak Jose Ramon Etxebarriari testuari egindako zuzenketengatik, eta Nekane Intxaurtzari formatua egokitzeagatik.

A zatia:
Ariketa bildumarako sarrera

I. Konputagailuen programazioa: algoritmoak

I.1. KONPUTAGAILUEN PROGRAMAZIORAKO ALGORITMOAK

Konputagailua luze, zail eta errepikakorrak diren agindu-sekuentziak burutzen dituen makina da. Agindu horiek eragiketa logikoak edo kalkulu matematikoak izaten dira. Agindu-sekuentzia egikarituz erabiltzaileak ematen dituen datuak (informazioa) hartu, eta lortu nahi diren emaitzak (beste informazio bat) kalkulatzen dituzte. Berau lortzeko asmatu behar den agindu-sekuentziari *programa* deitzen zaio.

Nahiz eta konputagailuak berez ez diren adimendunak, ematen zaizkien agindu-sekuentziak exekutatzen badakite, eta datuak irakurtzen eta idazten ere bai; baina ez dakite problema baterako programa asmatzen. Datuekin zer nolako aginduak egikaritu behar diren zehatz-mehatz eta zuzen definitzen ez badiogu (programa egokia ematen ez bazaio, hain zuzen), konputagailuak ez du ezer egingo, edo emaitza okerrak eskainiko ditu programa hori zuzena ez bada.

Programak idazteko, programazio-lengoaiez baliatzen gara. Programatzeko lengoaia bakoitzak bere arauak ditu, batetik datuak memoria barruan adierazteko eta bestetik exekutatu ahal diren aginduak definitzeko. Beraz, programa bat idazteko, programatzeko lengoaia baten arauak bete behar dira.

Goi-mailako lengoaia asko daude gaur egunean, bakoitzari bere erabilpen-eremu egokiena dagokiolarik. Zenbait lengoaiak adierazpide algebraiko eta zenbakizkoetarako erraztasuna eskaintzen dute eta, beraz, kalkulu zientifikoetarako erabiltzen dira; beste zenbaitek hitzak eta zenbaki sinpleak erabiltzen dituzte, baina datu-kopuru handietarako balio dute eta ondorioz, enpresa-arazoetarako erabiltzen dira.

Liburu honetan problema-bilduma bat aurkezten da, problema bakoitzerako algoritmo bat eraikiz. Algoritmoak idazteko lengoaia berezia erabiliko dugu; ez da programazio-lengoaia bat, baina edozein programazio-lengoaia hautatuz gero, erraza izango da programa idaztea algoritmotik abiatuta. Azken kapituluko ariketetan Ada programazio-lengoaia ere erabili dugu.

I.2. ALGORITMOAK ETA PROGRAMAK

Problema bat konputagailuaren bidez ebatzi gura denean, ondoko hiru pausoak bete behar dira: zehaztapena, algoritmoa, programa eta proba. Ikus ditzagun banan-banan.

I.2.1. Zehaztapena.

Oso ondo definituta egon behar du problemaren enuntziatuak; zalantza-izpirik gabekoa izan behar du, bestela nahasteak azalduko baitira gero. Sarritan, problema daukan pertsona ez da programa asmatuko duena, eta horrelakoetan oso garrantzitsua da zehaztapena, zeren pauso horretan zehatz-mehatz ezartzen baita programak zer egin beharko duen eta zer egin behar ez duen.

Zehaztapenean, alde batetik, jasoko diren sarrera-datuak, eta beste aldetik, lortu nahi diren irteera-datuak zehaztu behar dira; hau da, sarrera-datuek eta irteera-datuek bete beharko dituzten propietateak zein diren.

Pauso hau funtsezkoa da. Zalantza guztiak argitu behar dira, zeren bestela, problema duenak eta programa asmatuko duenak gauza bera ulertzen ez badute, sortuko den programak ez baitu ezertarako balio izango. Programa ondo eginda egongo da, agian, baina pertsona eta egun askoren fruitua izan arren, programak ez dio problema ebatziko erabiltzaileari. Beste problema bat ebazteko balioko du, programa asmatu duenak ulertu zuena ebazteko, hain zuzen, baina ez konputagailua erabili nahi duenaren problemari soluzioa emateko.

I.2.2. Algoritmoa.

Problemaren zehaztapena egin ondoren, eta erabiliko den programazio-lengoaia ondo ezagutuz gero, programa idazten besterik gabe has daitekeela dirudi. Nahiz eta kasu errazenetan noizbait horrela jokatzen den, hori ez da ohiko prozedura. Zuzenean idatzi gabe, tarteko pauso bat bete behar da: algoritmo bat diseinatzea.

Programan bezala algoritmoan ere agindu-sekuentziak definitzen dira, baina ez programazio-lengoaia jakin baten aginduak erabiliz, programazio-lengoaia guztietan erabiltzen diren aginduen abstrakzioak (lengoaia guztiek konpartitzen dutena edo) erabiliz baizik. Ondorioz, algoritmoan ez dira gorde behar programazio-lengoaia batez idazteko gorde behar diren arau sintaktiko estuak. Zailtasun-maila hori geroago egin beharko den kodeketa-faserako lagatzen da.

I.2.3. Programa.

Pauso honetan, hautatutako programazio-lengoaiaren arauak ondo ezagututa, programagileak programazio-lengoaiara itzultzen ditu algoritmoko aginduak. Ondoren, programa-itzultzaile bat (konpiladorea edo interpretzailea) beharko da goi-mailako programa hori ordenadorearen makina-lengoaiara itzul dezan. Kasu gehienetan konpilatzaileak programako errore sintaktikoak nabarituko ditu. Horrelako errore guztiak zuzendu arte, konpilatzaileak ez du sortuko programa itzulia. Orduan prest egongo gara programa lehenengo aldiz egikaritzeko.

I.2.4. Proba.

Eraikitako programak erroreren bat izan dezake. Frogatu egin behar da programaren zuzentasuna. Horregatik, programaren zuzentasuna frogatzeko, proba-kasuak egikaritzen dira. Horretarako, programaren zehaztapenaren arabera esanguratsu diren sarrera-datuen multzoak definitu eta bakoitzari dagokion emaitza ezarri behar da aldez aurretik. Errorerik aurkitzen bada, zuzendu egin beharko da programa, edo algoritmoa, edo zehaztapena. Pauso honi *errore-arazketa* deritzo. Errore-arazketa ez da inoiz ere amaitzen programa handiekin, ezinezkoa izaten baita zuzentasunerako proba guztiak egikaritzea, oso errepresentagarriak direnak soilik hartzen dira-eta; probetan errorerik agertzen ez bada, zuzentzat ematen da programa eta erabiltzen hasteko prest dago; baina erabiltzerakoan kontuan hartu ez diren posibilitateak gertatuz gero, errore berririk ager daiteke, eta orduan ere zuzenketa berri bat egin beharko da.

I.2.5. Estiloa.

Programak ahalik eta modu ulergarrienean idatzi behar dira, beren zuzentasunaz edo errorerik eza ziurtasuna emateko eta errore-zuzenketa errazagoa izan dadin. Programazio-estiloari begira, ondoko puntuak hartu behar ditugu kontuan:

- Programa erabilgarri gehienak, egileez gain, beste programatzaile batzuek ere irakurtzen dituzte.

- Irakurle bakoitzak programak nola lan egiten duen erraz ikusi behar du, erroreak aurkitu ahal izateko eta zuzentzeko edo aldaketak egiteko.

- Estilo txarrez eginiko programa bat irakurtzea ez da batere atsegina, ezta egilearentzat berarentzat ere.

II. Eredu algoritmikoa. Algoritmoen osagaiak

Bost dira algoritmoak idaztean menperatu behar diren oinarrizko kontzeptuak: **objektuak** (edo datuak), objektuak manipulatzeko **aginduak,** datu berriak kalkulatzeko balio duten **adierazpenak,** aginduak betetzeko ordena alda dezaketen **kontrol-egiturak** eta algoritmo zailak azpiproblema sinpleagotan banatu eta azpiproblema bakoitza bere aldetik ebazteko laguntza eskaintzen duten **moduluak (funtzioak eta prozedurak)**.

II.1. OBJEKTUAK: ALDAGAIAK, KONSTANTEAK ETA LITERALAK

Problema baten datuak objektuen bidez adierazten ditugu algoritmoetan. Objektu horiek hiru ezaugarri dituzte beti:

- IZENA: identifikadore bat.
- MOTA: mota batek definitzen du:
  - bere balio posibleen multzoa
  - eta balio horiekin egin daitezkeen eragiketak.

  Adibidez: osokoa, erreala, karakterea
- BALIOA: une zehatz batean daukana

Objektuen artean konstanteak eta aldagaiak bereizten dira. Kasu bietan izena eta mota aldaezinak dira algoritmoa egikaritzen denean; konstanteen kasuan berdin gertatzen da balioarekin, baina aldagaien balioa aldatu egin daiteke.

Konstantea deritzo, behin bere balioa finkatu denetik aurrera, balio hori inoiz aldatzen ez zaion objektuari (adibidez: *Pi* konstantea 3.14159 balio erreala adierazteko defini daiteke, eta *Segundoak_Orduko* konstantea, ordu batek dituen 3600 segundoren kopurua adierazteko).

Aldagaia berriz, alda daitekeen balioa duen objektua da (adibidez: algoritmoa egikaritzen hasten denean, Kontagailua izeneko aldagai batek 0 osoko balioa du eta bukaeran 21 balioa).

Literalak deritzen objektuak ere balio konstanteak adierazteko erabiltzen dira, baina zuzenean balioa idatziz, identifikadorerik erabili gabe; adibidez:

0 1 60 1_000_000 (osoko literalak)

0.0 3.14158 (literal errealak)

'H' ':' ' ' (karaktere motako literalak)

"Ordua: " "???" (kate motako literalak)

Objektu bat datu-mota batekoa dela esaten dugunean, bi gauza zehazten dugu: batetik objektuak har ditzakeen **balio posibleen multzoa** (*domeinu* deritzona) zein den, eta bestetik balio horiekin erabil daitezkeen **eragiketak** zein diren.

Objektu batzuk **sinpleak** dira, datu bakar batekin errepresentatzen baitira (*bakunak* edo *eskalarrak* ere esaten zaie); beste batzuk, ordea, konposatuak dira eta datu-mota **egituratuak** erabili behar dira haiek errepresentatu ahal izateko. Azter ditzagun oinarrizko algoritmoetan erabiltzen diren datu-mota arruntenak! Lehenengoz, datu-mota sinpleak, eta gero, egituratuak.

II.1.1. Datu-mota sinpleak

Hauexek dira balio bakar bat dakarten datu-motak:

Datu-mota	**Balioak**
Osoko	osoko zenbakiak
Erreal	zenbaki errealak
Karaktere	karaktereak
Boolear	balio boolearrak
Kate	karaktere-kateak

II.1.1.1. Osokoak

Zenbaki osoak adierazteko datu-mota

DOMEINUA: …, –3, –2, –1, 0, +1, +2, +3, …

ERAGIKETAK:

Eragigaiak eta emaitzak osokoak dira. Adi zatiketarekin!

Eragile diadikoak		*Eragile monadikoak*	
+	batuketa	-	ukapena
-	kenketa	abs	balio absolutua
*	biderketa		
/	zatiketa moztua		
rem	zatiketaren hondarra		
mod	modulua		
**	berreketa		

Eragiketa erlazionalak: <, <=, >, >=, /=, =

II.1.1.2. Errealak

Zenbaki errealak adierazteko datu-mota.

DOMEINUA:

0.0	1.5	386_473.0	3.141_592_65
3.86473e5		3.0e+8	0.1234E-20

ERAGIKETAK:

Eragigaiak eta emaitzak errealak dira. Berreketako berretzailea izan ezik, osoko motakoa izan behar baita

Eragile diadikoak		*Eragile monadikoak*	
+	batuketa	abs	balio absolutua
-	kenketa	-	ukapena
*	biderketa		
/	zatiketa moztua		

Eragiketa erlazionalak: <, <=, >, >=, /=, =

II.1.1.3. Boolearrak

Balio logikoak (egiazkoa ala faltsua) adierazteko datu-mota.

DOMEINUA:

{egiazkoa , faltsua }

ERAGIKETAK:

Eragile erlazionalak.

Eragigai biak mota berekoak dira eta, emaitza, boolearra.

=	berdin	<=	txikiago edo berdin
/=	desberdin	>=	handiago edo berdin
<	txikiago	>	handiago

Eragiketa logikoak.

Eragigai biak eta emaitza ere boolearrak dira.

and	*eta* eragiketa logikoa
or	*edo* eragiketa logikoa
xor	*ala* eragiketa logikoa
not	*ez* eragiketa logikoa

and, or, xor eta not eragiketa boolearren egia-taulak:

A	**B**	**A and B**	**A or B**	**A xor B**	**not B**
Faltsua	Faltsua	Faltsua	Faltsua	Faltsua	Egiazkoa
Faltsua	Egiazkoa	Faltsua	Egiazkoa	Egiazkoa	Faltsua
Egiazkoa	Faltsua	Faltsua	Egiazkoa	Egiazkoa	
Egiazkoa	Egiazkoa	Egiazkoa	Egiazkoa	Faltsua	

II.1.1.4. Karaktereak

Karaktereak adierazteko datu-mota.

DOMEINUA:

‘a’ ‘A’ ‘1’ ‘?’ ...

Kontuz ! 3 eta ‘3’ balio desberdinak dira.

ERAGIKETAK:

Eragiketa erlazionalak: <, <=, >, >=, /=, = (ordena alfabetikoaren arabera)

II.1.1.5. Kateak

Karaktere-kateak adierazteko datu-mota.

DOMEINUA:

Karaktere-kateak.

“Maite”, “Kale Nagusia 12, 3.C”

Kontuz ! “3”, 3 eta ‘3’ balio desberdinak dira.

ERAGIKETAK:

Eragiketa erlazionalak: <, <=, >, >=, /=, =

II.1.2. Datu-mota egituratuak

Orain arte definitutako datu-motak *sinpleak* izan dira: balio bakarra biltzen dute. Datu-mota *egituratuek* balio bat baino gehiago biltzen dute. Datu-mota egituratu arruntenak hiru dira:

- **Bektorea:** Osagaiak mota berekoak dira eta indize bakar batekin erreferentziatzen dira.
- **Matrizea:** Osagaiak mota berekoak dira eta indize birekin erreferentziatzen dira.
- **Erregistroa:** osagai guztiak ez dira mota berekoak

II.1.2.1. Bektoreak

Bektore motako aldagai batek bere barruan mota berekoak diren hainbat aldagai dauzka. Osagai horiek indize bat erabiliz erreferentziatzen dira.

Hasieran ariketa-bilduma honetan lau datu-mota berri erabili ahal izango ditugu: *Osokoen_Bektore* (osagai bezala zenbaki osoak dituen bektorea), *Errealen_Bektore, Boolearren_Bektore* eta *Karaktereen_Bektore;* baina edozein motatako osagaiak dituen bektorea definitzea ere posiblea da, aurrerago azalduko dugunez.

Bektore motako aldagai (edo objektu) bat erazagutzeko hiru gauza zehaztu behar dira: bektorearen izena, osagaien kopurua (indizeetarako tartea definituz) eta osagaien mota. Adibidez:

```
Kalifikazioak: Osokoen_Bektore(1..10)
Batezbestekoak: Errealen_Bektore(1..66)
```

Bektorearen osagai bat adierazteko, bektorearen izena eta osagaiari dagokion indizea idatzi behar dira.

Kalifikazioak(6) Kalifikazioak deritzon bektorearen seigarren osagaia da

Batezbestekoak(2) Batezbestekoak bektorearen bigarren osagaia da

Indizea adierazpen bat ere izan daiteke:

Kalifikazioak(I) Kalifikazioak izeneko bektorearen I-garren osagaia da.

I aldagaiak exekuzio-unean daukan balioa da indizea.

*Kalifikazioak(2*I -3)* Une batean I-ren balioa 4 bada, 5. osagaia da aipatzen dena; geroago I-ren balioa 5 bada, 7. osagaia da orduan aipatzen dena.

KONTUZ! *Kalifikazioak(2*I -3)* erabiliz gero, une batean I-ren balioa 8 bada, 13. osagaia aipatu nahi da eta *Kalifikazioak* bektoreak hainbeste osagai ez duenez, errore bat sortuko da.

Hona hemen nola definitzen diren bektoreen datu-motak:

BALIO POSIBLEAK:

Osagai bakoitzak bere motako balio bat dauka.

Bektorearen balio posibleak bere osagaien balio-konbinazio guztiak dira.

ERAGIKETAK:

Osagai bakoitzak bere motako aldagai bakun modura jokatzen du

Osagai-kopuru berdineko bektoreen artean bi eragiketa hauek egin daitezke:

- Bektore osoaren asignazioa
- Bektore osoen arteko konparazioa (=, /=)

Definitu ditugun lau datu-motak (*Osokoen_Bektore*, *Errealen_Bektore*, …) azpiprogrametan erabil daitezke parametro baten mota adierazteko, osagaien kopurua zehaztu gabe. Parametro errealak zehaztuta eduki beharko du osagaien kopurua, baina parametro formala orokorragoa izan daiteke; horrela eginez, azpiprogramak edozein tamainatako bektoreen tratamendua definitzeko aukera eskainiko du.

Parametro erreal eta formalak zer diren ulertzeko jo II.5 atalera.

1. erabilera-adibidea: *Gehienez 100 karaktere dituen eta puntuz bukatzen den karaktere-sekuentzia bat emanda, karaktere guztiak atzetik aurrera idatziko dituen algoritmoa espezifikatu eta egin.*

Honelako bektore bat behar dugu

```
Bektorea: Karaktereen_Bektore (1..100)

algoritmo Alderantziz
Aurrebaldintza: S (karaktere-sekuentzia).
           Puntuz bukatzen da eta ez dago beste
           punturik.
Postbaldintza: S' (karaktere-sekuentzia).
     S sekuentziako karaktereak dauzka baina
     alderantzizko ordenan
hasiera
     Testua_Gorde(Bektorea, Kar_Kopurua)
     Idatzi_Atzekoz Aurrera (Bektorea, Kar_Kopurua)
amaia

algoritmo  Testua_Gorde
               (B: emaitza Karaktereen_Bektore ;
               Kop: emaitza Osokoa)
Aurrebaldintza: S (karaktere-sekuentzia).
           Puntuz bukatzen da eta ez dago beste
           punturik.
Postbaldintza:  Kop-ek adierazten du S
           sekuentziaren karaktere-kopurua puntua
           kontatu gabe.
           S sekuentziako karaktereak B bektorean
           gorde dira lehenengo posiziotik Kop-garren
           posizioraino
```

```
hasiera
     Irakurri_Karakterea  (Kar)
     I:= 1
     bitartean  Kar /= ´ ´  egin
          B(I) := Kar
          I := I + 1
          Irakurri_Karakterea (Kar)
     ambitartean
     Kop := I – 1
amaia

algoritmo  Idatzi_Atzekoz Aurrera
              (B: datu Karaktereen_Bektore;
              Kop: datu Osokoa)
Aurrebaldintza:  Kop-ek adierazten du B
          bektoreko indize bat
Postbaldintza:  S´ (karaktere-sekuentzia)
          B bektoreko osagaien karaktereak idatzi dira
          lehenengo posiziotik Kop-garren posizioraino
hasiera
     I:= Kop
     bitartean  I > 0  egin
          Idatzi_Karakterea (B(I))
          I := I – 1
     ambitartean
amaia
```

2. erabilera-adibidea: *N elementu duen osokoen bektore bat emanda, bektoreko zenbakien batezbesteko aritmetikoa kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma bezala inplementatu.*

```
funtzio Batezbestekoa (B: Osokoen_Bektore; N: Osokoa)
               itzuli  Erreala
Aurrebaldintza: N>0
Postbaldintza: Itzuli da zenbaki erreal bat, B
          bektoreko N osagaien batezbesteko aritmetikoa
          adierazten duena
hasiera
     Batura := 0
     egin I guztietarako 1 tik N raino
          Batura := Batura + B(I)
     amguztietarako
     itzuli (Erreal_Bihurtu (Batura) / Erreal_Bihurtu(N))
amaia
```

Erreal_Bihurtu funtzioak osoko bat hartu eta zenbaki erreal bihurtzen du.

II.1.2.2. Matrizeak

Azaldu dugun bezala, bektore motakoa den aldagai bat osatzen duten aldagai guztiak mota berekoak dira, eta indize bakar baten arabera erreferentziatzen dira. Matrizeetan, berriz, bi indize erabiltzen dira.

Lau datu-mota berri erabili ahal izango ditugu: Osokoen_Matrize (osagai bezala zenbaki osoak dituen matrizea), Errealen_Matrize, Boolearren_Matrize eta Karaktereen_Matrize. Edozein motatako osagaiak dituen bektorea definitzea ere posiblea da.

Matrize motako aldagai bat erazagutzeko hiru gauza zehaztu behar dira: matrizearen izena, osagaien kopurua (bi indizeetarako tarteak definituz) eta osagaien mota. Adibidez:

```
Itsasontziak: Boolearren_Matrize (1 .. 10, 1 .. 10)
Pantaila: Karaktereen_Matrize (1 .. 24, 1 .. 80)
```

Matrizeko osagai bat adierazteko, bektorearen izena eta osagaiari dagozkion indizeak idatzi behar dira. Adibidez:

Pantaila(6, 25) Pantaila izeneko matrizeko 6. errenkadako 25. osagaia da

Itsasontziak(I, J+1) Itsasontziak izeneko matrizeko I. errenkadako (J+1)garren osagaia da

II.1.2.3. Erregistroak

Datu-mota *egituratuek* balio bat baino gehiago biltzen dute. Datu-mota egituratu arruntenen arteko aldeak hauexek dira:

Bektorea:

osagai guztiak mota berekoak dira
osagaiak indexazio bidez erreferentziatuko dira

Erregistroa:

osagai guztiak ez dira mota berekoak
osagaiak identifikadore bidez erreferentziatuko dira.

Objektu-bilduma bat erregistro batean biltzeak abantailak ditu: batetik, objektuek batak bestearekin duten erlazioa esplizituki adierazten da, eta bestetik,

erregistroa objektu bakun modura maneiatu edo bere osagaiak indibidualki maneiatzearen artean aukera dezakegu, uneko beharren arabera.

Erregistro motako objektu bat erazagutzeko bi gauza definitu behar dira: objektuaren izena eta erregistro-motaren definizioa, hau da, erregistro-osagai bakoitzaren identifikadorea eta mota. Adibidez:

```
Hitza: erregistro
          Karaktereak: katea(1..20) ;
          Luzera:      osoko ;
amerregistro;
```

Erazagupen bera lortu liteke, lehenago erregistroarekin mota berri bat definituz, eta ondoren Hitza aldagaia mota horrekin erazagutuz:

```
mota Hitz da erregistro
              Karaktereak: katea(1..20) ;
              Luzera:      osoko ;
amerregistro;

Hitza: Hitz ;
```

Modu batera edo bestera eginda, *Hitza* izeneko aldagaiaren erazagupena berdina da. Hala ere, komeni da *Hitz* izeneko mota definitzea; horrela eginez, azpiprogrametako parametroetan edo programako beste toki batean ere erabili ahal izango da gero.

Erregistro baten osagai bat erreferentziatzeko erregistro motako aldagaiaren izena, puntu bat eta osagaiari dagokion identifikadorea idatzi behar dira. Adibidez:

Hitza.Luzera	Hitza aldagaiko Luzera osagaia.
Hitza.Karaktereak	Hitza aldagaiko karaktere-bektorea.
Hitza.Karaktereak(2)	Hitza aldagaiko karaktere-bektorearen 2. osagaia.
Hitza	Erregistro osoa duen Hitza aldagaia.

Hona hemen nola definitzen diren erregistroen datu-motak:

BALIO POSIBLEAK:

Osagai bakoitzak bere motako balio bat dauka.

Erregistro osoak balio guztiek osaturiko egitura dauka baliotzat.

ERAGIKETAK:

Osagai bakoitzak bere motako aldagai bakun modura jokatzen du.

Mota bereko erregistroen artean bi eragiketa hauek egin daitezke:

- erregistro osoaren asignazioa
- eerregistro osoen arteko konparazioa (=, /=)

II.1.2.4. Datu-egitura mistoak

Algoritmo batean erabili behar den objektu bat oso konplexua bada, datu--egitura bat definitu beharko da hura adierazteko. Datu-egitura gisa, osagaien arteko erlazioa erakusteko moduan antolatuta dagoen datu-elementuen bilduma ulertzen dugu.

Datu-egiturak definitzeko oinarrizko eraikuntza-blokeak bektoreak eta erregistroak dira. Bektoreen eta erregistroen osagaiak berak ere bektoreak edo erregistroak izan daitezke. Honek nahi adinako konplexutasuneko datu-egiturak definitzeko aukera ematen du. Hona hemen datu-egitura mistoen zenbait adibide:

Datu-egitura mistoa (adibidea)

```
mota Hiztegi da Hitz bektore(1..1000)
mota Hitz da erregistro
              Karaktereak: Katea(1..20)
              Luzera: Osoko
amerregistro
```

Datu-egitura mistoa (adibidea)

```
mota Talde da Ikasle bektore(1..100)
mota Hitz da erregistro
              Karaktereak: Katea(1..20)
              Luzera: Osoko
mota Ikasle da erregistro
              Kodea: Osoko
              Izena, Deitura1, Deitura2: katea (1..20)
              Kalifikazioak: Osokoen_Bektore (1..10)
amerregistro
```

Datu-egitura mistoa (adibidea)

```
mota Talde da erregistro
              Ikasleak: Ikasle bektore(1..100)
              Batezbestekoak: Errealen_Bektore(1..10)
amerregistro
mota Ikasle da erregistro
              Kodea: osoko
              Izena, Deitura1, Deitura2: katea(1..20)
              Kalifikazioak: Osokoen_Bektore(1..10)
amerregistro
```

II.2. ADIERAZPENAK

Balio berri bat kalkulatzeko formulak dira. Oro har, balio bat erabil daitekeen tokian adierazpen bat ere jar daiteke, adierazpena kalkulatuz lortzen den balioa adierazten duelarik. Adierazpen bat ebaluatzean, bertan dauden aldagaien ordez une horretan aldagaiek duten balioak erabiltzen dira kalkuluak egiteko. Adibidez:

Une batean I aldagaiaren balioa 5 baldin bada, *2*I+3* adierazpena ebaluatuz 13 lortuko da. Geroxeago I aldagaiaren balioa 6 baldin bada orduan *2*I+3* adierazpen bera ebaluatuz 15 lortuko da.

Adierazpen berean zenbait eragiketa biltzen direnean, ondo zehaztu beharko da eragiketak burutzeko ordena. Ebaluatzeko ordena zein izango den jakiteko, eragiketen arteko lehentasunak ezagutu behar dira. Hala ere, beti izango da posible parentesiak erabiltzea, lehentasun esplizitua ezartzeko.

Adibidez, *I+J/K* adierazpena honako adierazpen hauetako zeinen arabera ebaluatuko da: *I+(J/K)* erara? edo *(I+J)/K* erara? Maila berean azalduz gero, zatiketak batuketak baino lehentasun handiagoa duenez, *I+(J/K)* erara ebaluatuko da.

Lehentasun-maila bereko bi eragile biltzen direnean, ezkerraldetik hasiko da ebaluatzen. Adibidez, *I/J*K* adierazpena *(I/J)*K* erara ebaluatuko da. Ondoko taulan eragiketa arrunten arteko lehentasunak erakusten dira:

Eragile diadikoak	**Eragile monadikoak**	**Lehentasuna**
**	abs not	handiena
* / mod rem		↕
+ - &	+ -	↕
= /= < <= >= >		↕
eta edo ala		txikiena

II.3. AGINDUAK

Hiru dira datuak erabiltzeko oinarizko aginduak[1]: balio baten **asignazioa** edo esleipena, datu baten **irakurketa** eta datu baten **idazketa**..

II.3.1. Asignazioa

aldagaia := *adierazpena*

1. Aginduei ekintza ere esaten diegu algoritmoez ari garenean.

Asignazioak adierazpena ebaluatuz lortzen den balioa ezartzen dio aldagaiari, balio berri modura. Aldagaiak galdu egiten du aurreko balioa. Adierazpenean aldagairik azaltzen bada, adierazpena ebaluatzean aldagaiak duen balioa erabiliko da, baina aldagaiaren balio hori ez da aldatzen ikusia izateagatik. Adibidea:

Hasierako egoera:

N: 3 P: 4 X: 1.0 Y: 4.5

Asignazioak

```
M := (N + P) * 6
Z := Y - X
P := P + 1
```

Asignazioak egikaritu ondoko egoera hauxe izango da:

N: 3 **P: 5** X: 1.0 Y: 4.5 **M: 42** **Z: 3.5**

II.3.2. Datuak irakurri

Datuak sekuentzia batetik irakurtzen dira. Sekuentziako elementu guztiak mota berekoak direla suposatzen da. Gehienetan sekuentzia hori teklatutik idatziko diren balio-sekuentzia modura ikus daiteke. Oinarrizko mota bakoitzerako, agindu bat dago mota horretako balioak irakurtzeko.

Irakurri_Osokoa (*ald1*)

Sekuentzian oraindik irakurri gabe dagoen lehenengo osokoa ald1 aldagaiari asignatzen dio.

Antzekoak dira honako hauek:

Irakurri_Erreala (*ald2*)

Irakurri_Karakterea (*ald3*)

Sekuentziako elementuen motak eta aldagaiarenak berdinak izan behar dute. Sekuentziako elementu bat irakurri ahal izateko, aldez aurretik irakurri behar izan dira aurretik zeuden guztiak.

Adibidea. Demagun datuak irakurtzeko sekuentzia hau dela:

37 9 4 17 -78 0

Honako irakurketak egikaritzen badira:

```
Irakurri_Osokoa (X)
Irakurri_Osokoa (Z)
Irakurri_Osokoa (Y)
```

Bukaeran aldagaien balioak hauek izango dira:

X: 37 Y: 4 Z: 9

Eta hurrengo irakurketan hartuko den zenbakia 17 izango da.

II.3.3. Datuak idatzi

Emaitzak beste sekuentzia batean idazten dira.

Idatzi_Osokoa (*adierazpen1*)

Agindu honek adierazpena ebaluatuz lortzen den balioa (osokoa) idatzi egiten du irteera-sekuentzian. Ordenadoreko pantailan idazten du balioa. Adibidez:

Hasierako egoera:

X : 3 Y : 45

Agindua:

```
Idatzi_Osokoa ((X+Y)*2)
```

Pantailan idazten dena:

96

Antzekoak dira honako hauek:

Idatzi _Erreala (*adierazpen2*)

Idatzi_Karakterea (*adierazpen3*)

II.4. KONTROL-EGITURAK

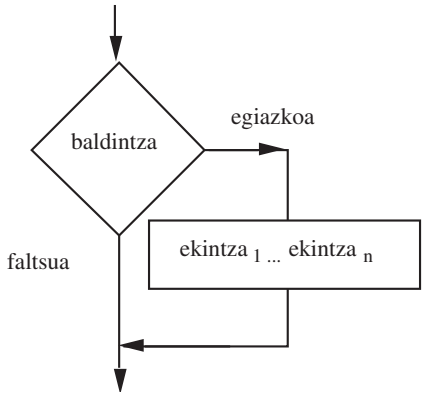
II.4.1. Agindu-sekuentzia

Algoritmoko agindu-sekuentzia pausoz pauso eta ordenan egikaritzen da. Ordena sekuentzial eta lineal hori aldatzeko bi aukera daude: baldintzazko egiturak edo iterazio-egiturak erabiltzea.

II.4.2. Baldintzazko egiturak

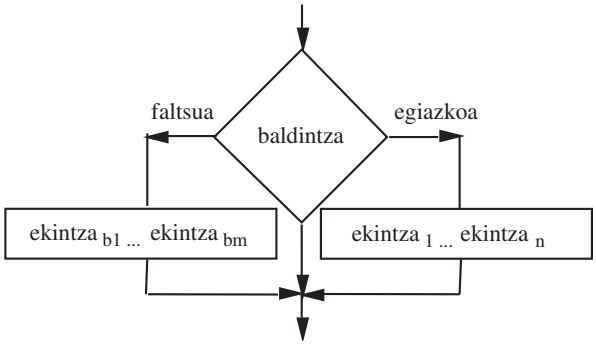
Agindu bat (edo gehiago) baldintza bat betetzen denean bakarrik egin behar bada, *baldin* izeneko kontrol-egitura erabili beharko da. Emaitza boolearra itzuliko duen adierazpena izan behar du baldintzak.

```
baldin  baldintza orduan
     ekintza1... ekintzan
ambaldin
```



Baldintza betetzen ez denean beste agindu bat (edo gehiago) egin behar bada, *bestela* motako atal bat ere sar daiteke:

```
baldin baldintza orduan
     ekintza1... ekintzan
bestela
     ekintzab1... ekintzabm
ambaldin
```

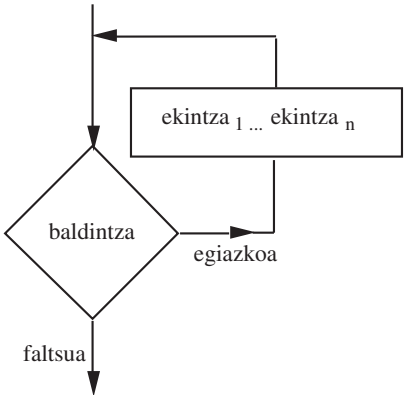


II.4.3. Iterazio-egiturak

II.4.3.1. Iterazio arrunta

Agindu bat (edo gehiago) errepikatu behar direla adierazteko, *bitartean* izeneko kontrol-egitura erabiltzen dugu: Aginduak errepikatuko dira baldintza betetzen den bitartean

```
bitartean  baldintza egin
     ekintza1
     ekintza2
     ...
     ekintzan
ambitartean
```

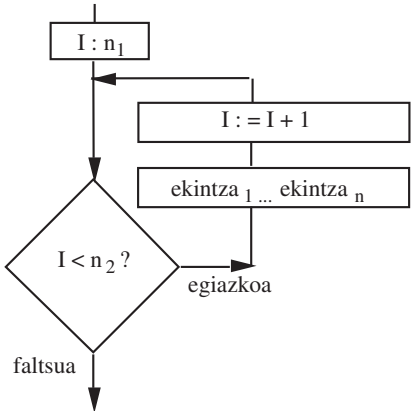


II.4.3.2. Aldi-kopuru jakineko iterazioa

Iteratzen hasi baino lehenago aginduak zenbat aldiz errepikatu behar diren jakiterik badago, egokiago izaten da ***guztietarako*** motako kontrol-egitura erabiltzea. Definizioko n_1 eta n_2 horiek adierazpenak dira, alegia, errepikaketak kontrolatzeko *I* aldagaiak hartu behar dituen lehenengo eta azken balioak kalkulatzeko erabiliko direnak. Adierazpen horiek iterazioa hasi aurretik bakarrik

kalkulatzen dira, baina, kontuz!, ez geroago. *I* kontrol-aldagaiaren balioa erabil daiteke iterazioko adierazpenetan; baina I kontrol-aldagai horri ezin zaio balio berri bat asignatu begiztaren barruko ekintzaren batean.

```
egin I guztietarako n1 tik n2 raino
     ekintza1 ... ekintzan
amguztietarako
```



Adibidea:

```
algoritmo Kuboak_1_30
hasiera
     egin I guztietarako 1 tik 30 raino
          Idatzi_Osokoa (I**3)
     amguztietarako
amaia
```

Emaitza:

1 8 27 64 125 216 … 27000

Ondoko algoritmoak ere emaitza berdinak ematen ditu baina bitartean izeneko kontrol-egitura erabiliz definitu da:

```
algoritmo Kuboak_1_30
hasiera
     I := 1
     bitartean I <=30 egin
          Idatzi_Osokoa (I**3)
          I := I + 1
     ambitartean
amaia
```

II.5. MODULUAK: FUNTZIOAK ETA PROZEDURAK

Algoritmo zailak azpiproblema sinpleagotan banatu eta azpiproblema bakoitza bere aldetik ebazteko laguntza eskaintzen dute moduluek. Moduluak bi eratakoak izan daitezke:

Funtzioa

- Emaitza bat kalkulatzeko balio du.
- Balio modura erabiltzen da.
- Emaitza hau adierazpen baten barruan erabili behar da nahitaez. Adibidez, Pred funtzioa zenbaki oso baten aurrekoa itzultzeko definitu bada, *Pred(8)*2* adierazpenak 14 balioa itzuliko du.

Prozedura

- Aldagai baten balioa aldatu edo sarrera-irteera bat egikaritzeko balio du.
- Agindu modura erabiltzen da, beste algoritmo edo modulu baten barruan. Adibidez, *Idatzi_Osokoa* prozedura osoko bat idazteko prozedura bada, *Idatzi_Osokoa(N-4)* agindu posible bat da algoritmo batean.

Modulu batek parametroak eduki ditzake. Parametroei esker moduluaren definizioa orokorragoa izango da, eta eragin berdina lortu ahal izango da datu edo aldagai desberdinekin. Parametroen bitartez datuak pasatzen zaizkio eta emaitzak jasotzen dira.

Moduluak algoritmoak bezala definitzen dira agindu-multzo batekin; eta multzo honi moduluaren *gorputza* deritzo. Gorputz hau moduluari deitzen zaionean egikaritzen da; ondoren algoritmoaren exekuzioak deiaren ondoko puntutik jarraituko du. Moduluko barruko aldagaiak eta konstanteak zeintzuk diren hobeto zehaztearren, hobe izaten da agindu-sekuentzia baino lehenago aipatzea. Horrela, objektu horiek modulu horretatik kanpo ezin erabil daitezkeela adierazi nahi dugu.

Programazio-lengoaiek oso erabilgarriak diren azpiprograma edo modulu estandarrak eskainiko dituzte. Programategi bakoitzak programatzaile-talde berezientzat erabilgarriak diren azpiprogramak ere izango ditu. Adibidez, programatzaile ‘zientifikoek’ sinuak, kosinuak, erro karratuak eta horrelakoak kalkulatzeko gai diren funtzio matematikoak beharko dituzte. Horrela, programategiak esfortzu-bikoizketa ekiditen du eta adituek eginiko algoritmo sofistikatuak erabilgarri izango dira esperientziarik ez duten programatzaileentzako ere.

II.5.1. Funtzioak

Imajina ezazu karaktere bat hartu eta letra bat den ala ez aztertzen duen *Alfabetikoa* izeneko funtzio bat. *Alfabetikoa('K')* eta *Alfabetikoa(Iniziala)* funtzio--deien adibideak dira (*Iniziala* delakoa karaktere motako aldagaia izanik). Parentesi artean dagoen adierazpenari *parametro erreal* deitzen zaio, eta funtzioak parametro erreal horrekin egingo du kalkulua. Funtzio-dei hauetan parametroa karaktere motakoa da, eta emaitza boolear motakoa.

Adibidez, *ZKH* funtzioak bi osoko positiboen zatitzaile komunetatik handiena kalkulatzen badu, *Y := 1+ ZKH(N+1, 72)* asignazioan erabili den modura erabil daiteke funtzio-deietan. Funtzio honen bi parametroak eta bere emaitza *Osoko* motakoak izan beharko dira.

Funtzioaren definizioan izena, parametroak (bakoitzerako izena eta mota), eta kalkulatuko duen emaitzaren mota zehaztu behar dira. Funtzio-etan nahi adina parametro erreal defini daitezke. Hemen dituzu, aurrez aipaturiko *Alfabetikoa* eta *ZKH* funtzioen erazagupenak:

```
funtzio Alfabetikoa (Kar : Karaktere) itzuli Boolear
funtzio ZKH (Zenb1, Zenb2 : Osoko)    itzuli Osoko
```

Funtzioaren izena *funtzio* hitz erreserbatuaren jarraian dago. Bere emaitzaren mota *itzuli* hitz erreserbatuaren ondoren dago zehaztuta. Hauen artean eta parentesien bidez bildurik, edozein *parametro-espezifikazio* idazten dugu. Hauetako bakoitzak parametro bat edo gehiagoren izena eta hauen mota zehazten ditu.

Alfabetikoa izeneko funtzioak parametro bakarra du, *Karaktere* motakoa da eta bere izena *Kar* da. *Kar* izenari esker, posible izango zaigu funtzio-gorputzaren barrutik parametroa erreferentziatzea, deietan erabiliko den parametro erreala aipatu gabe. Horregatik *Kar* delakoari funtzioaren *parametro formal* deitzen zaio.

ZKH funtzioak bi parametro formal ditu, *Zenb1* eta *Zenb2*, biak *Osoko* motakoak.

Hala ere, funtzio-erazagupen batek ez du zehazten funtzioaren emaitza nola kalkulatzen den. Horregatik, funtzioaren gorputza ere idatzi behar dugu, honek baititu parametroen balioetatik abiatuz funtzioaren emaitza kalkulatuko duten aginduak.

Ondoko hau *Alfabetikoa* deritzon funtzioaren definizio posible bat duzu:

```
funtzio Alfabetikoa (Kar : Karaketere) itzuli Bollearra
hasiera
     baldin ((Kar >= 'A') eta (Kar <= 'Z'))
               edo  ((Kar >= 'a') eta (Kar <= 'z'))
     orduan itzuli egiazkoa
     bestela itzuli faltsua
amaia
```

Edo baliokidea eta laburragoa den beste gorputz batekin:

```
funtzio Alfabetikoa (Kar : Karaketere) itzuli Bollearra
hasiera
     itzuli ((Kar >= 'A') eta (Kar <= 'Z'))
               edo ((Kar >= 'a') eta (Kar <= 'z'))
amaia
```

Funtzio-gorputz batean *itzuli* izeneko aginduak bi xederi erantzuteko balio du: (a) funtzio-gorputzaren egikaritzapenari bukaera ematen dio; eta (b) *itzuli* hitz erreserbatuaren ondoren datorren adierazpena ebaluatzen du, eta hori izango da funtzioaren emaitza.

Jo dezagun, adibidez, *Alfabetikoa(Iniziala)* delako funtzio-deia ebaluatzen ari garela. Lehenik, *Kar* parametro formalari dagokion parametro errealaren balioa, *Iniziala*-ren balioa, ematen zaio. Suposa dezagun bere balioa '?' dela. Orduan *hasiera* eta *amaia* hitzen arteko aginduak egikaritzen dira. Hain zuzen, *itzuli* aginduaren adierazpena *Faltsua* da, eta beraz, funtzio-deiak *Faltsua* emaitza itzuliko du. Parametro errealaren balioa *'K'* izango balitz, emaitza *Egiazkoa* izango litzateke.

Orokorki esanda, funtzio-gorputz batek agindu asko eduki ditzake eta, again, barne-erazagupen batzuk ere bai.

ZKH funtzioa era askotara inplementa daiteke. Hemen duzu beste aukeretako bat:

```
funtzio ZKH (Zenb1, Zenb2 : Osoko) itzuli Osoko
     M, N, R   : Osoko;
hasiera
     M := Zenb1
     N := Zenb2
     R := M mod N
     bitartean R/=0 egin
          M := N
          N := R
          R := M mod N
     ambitartean
     itzuli N
amaia
```

Funtzio-gorputza *M*, *N* eta *R* aldagaiak aurkezten dituen erazagupenarekin hasten da, funtzio-gorputzaren barnean aldagai moduan erabiliko direnak dira horiek. Funtzioaren emaitza, algoritmo euklidearra inplementatzeko *M*, *N* eta *R* erabiltzen dituen begizta baten bidez kalkulatzen da. Halako batean, itzuli sententzia egikarituko da, *M*, *N*-ren multiplo zehatza denean, funtzioaren emaitza gisa *N*-ren azken balioa itzuliz.

Funtzio-gorputz batean *itzuli* agindu bat edo gehiago egon daitezke. Horrelakoetan ere, *itzuli* agindu bakoitzak funtzioaren emaitzaren motako adierazpen bat izan behar du. Beharrezkoa da *itzuli* agindu hauetako bat momenturen batean egikaritzea. Hau gertatzen denean, *itzuli*-ren ondoren dagoen adierazpena ebaluatzen da funtzioaren emaitza zehazteko, eta funtzio-gorputzaren egikaritzapena bukatu egiten da. Balioak emaitzaren motakoa behar du izan.

Zenb1 eta *Zenb2* parametro formalak ez dira aldagai modura erabili behar funtzio barruan, konstante bezala baizik; eta beraz, ezin dira eguneratu. Algoritmo euklidearrak bi zenbakiak beste zenbaki batzuez ordezkatu behar ditu behin eta berriz, zenbaki bat bestearen multiplo dela ikusi arte, beraz, *ZKH* funtzio-gorputzak nahi denean egunera daitezkeen aldagai lokaletan kopiatzen ditu bi zenbakiak.

II.5.2. Prozedurak

Funtzioak emaitza bakarra itzuli behar du. Emaitza asko dituen (edo emaitzarik ez duen) modulu bat idatzi nahi badugu, *prozedura* batera joko dugu.

Prozedura-dei bat agindu modura erabiltzen da.

Funtzioetan bezala, prozeduren definizioan ere, erazagupena eta gorputza bereizten ditugu. Erazagupenak prozeduraren izena eta bere parametro formalen izena eta mota zehazten ditu. Gorputzak prozedura inplementatzen duten aginduak eta aldagai lokalak ditu.

Hemen duzu zuriune-kopuru jakin bat idatzi behar duen prozedura baten erazagupen egokia:

```
algoritmo Zuriuneak_Idatzi (Zuriune_Kop : Natural)
```

Prozedura honi horrelako sententzien bidez dei dakioke:

```
Zuriuneak_Idatzi (10);   Zuriuneak_Idatzi (N-1);
```

Hona hemen gorputz posible bat prozedura honentzat:

```
algoritmo Zuriuneak_Idatzi (Zuriune_Kop : Natural)
hasiera
     egin Kont guztietarako 1 tik  Zuriune_Kop  raino
          Idatzi_Karakterea (' ')
     amguztietarako
amaia
```

Funtzio-gorputz bateko *itzuli* aginduak funtzioaren emaitza zehaztu eta funtzio-gorputzaren egikaritzapena bukatzeko balio du. Prozedura-gorputz batean lehenengo xedea ez da beharrezkoa; beraz, ez da egongo jarraian adierazpenik duen *itzuli* agindurik. Gainera, itzulketa inplizitu bat dago prozedura-gorputza bakoitzaren azken *amaia* baino lehen.

Modulu bat exekutatu ondoren, modulu barruan erazagututako parametro eta objektu formal guztiak desagertzen dira. (Horrela, hauek betetzen zuten memoria libre uzten da beste xede batzuetarako)

II.5.3. Moduluen parametroak

Hiru parametro klase daude: sarrerakoak, irteerakoak eta sarrera-irteerakoak. Diferentzia, datuak modulutik atera ala sartzeko erabiliko diren zehaztean datza.

II.5.3.1. Sarrera-parametroak

Orain arte ikusi ditugun moduluetan parametroak moduluari balioak emateko erabiltzen direla ikusi dugu. Balioa moduluan **sartzen** dutenez, horrelako parametroei sarrera-parametroak deritzegu.

Sarrera-parametroen mekanismoa ondorengo hau da. Sarrera-parametro formal bakoitzak, bere balioa dagokion parametro errealetik hartzen duen *konstante lokal* gisa jokatzen du (beraz, parametro errealak parametro formalaren mota berekoa behar du izan).

Parametro formalak esplizituki zehaztu behar dira sarrera-parametro gisa, parametro-espezifikazioan bi puntuen ondoren *datu* hitza ipiniz. Hala ere, hitz hau aukerakoa da; beraz, ondoko bi prozedura-erazagupen hauek baliokideak dira:

```
algoritmo Zuriuneak_Idatzi (Zuriune_Kop : Osoko)
algoritmo Zuriuneak_Idatzi (Zuriune_Kop : datu Osoko)
```

Parametro errealak adierazpen baten bidez erabiliko dira. Adibidez:

```
Zuriuneak_Idatzi (7)
Zuriuneak_Idatzi (2*N+4)
```

II.5.3.2. Irteera-parametroak

Balioak kalkulatzea ere posiblea da prozeduretan; hau da, posiblea da parametroak zehaztea balioak prozeduraz kanpo erabili ahal izateko edo gerora erabiliko diren aldagaietan gordetzeko. Horrelako parametroei *irteera-parametro* deritzegu, eta beren parametro-espezifikazioan, bi puntuen ondoren emaitza hitza ipiniz zehazten dira. Horrelako parametroekin, emaitza bat baino gehiago duten moduluak idatz ditzakegu (gogoratu funtzioek balio bakarra itzultzen dutela).

```
algoritmo Zatiketa_Moztua (M, N: datu Osoko;
                           Zatidura, Hondarra: emaitza Osoko)
hasiera
     Zatidura := M / N
     Hondarra := M mod N
amaia
```

Zatiketa_Moztua prozedurak bi datu jasotzen ditu eta bi emaitza itzuliko ditu.

Parametro erreala aldagai bat izan beharko da. Adibidez:

```
Zatiketa_Moztua (107, 7, N1, N2)
Zatiketa_Moztua (107*33, 7, Z, H)
```

Irteera-parametroen mekanismoak horrela jokatzen du: irteera-parametro formal bakoitzak *aldagai lokal* baten gisa jokatzen du. Prozedura-gorputzak balioa emango dio aldagai honi. Moduluaren exekuzioa bukatutakoan, parametro formalak duen balioa parametro errealari pasako zaio (beraz, parametroak aldagaiaren mota bera izan behar du). Kontuan har ezazu, prozedura-gorputzak ezin duela irteera-parametro formal baten balioa erabili.

II.5.3.3. Sarrera-irteera parametroak

Sarrera-parametro batek balio bat prozedura batera sartzea eta irteera-parametro batek bertatik balio bat ateratzea ahalbidetzen dute. Batzuetan, prozedura batek parametro erreal moduan pasatutako aldagai bat *eguneratzea* nahiko dugu; beste modu batera esanda, bere balioa prozedurari pasatu, eguneratu eta, hau egin ondoren, balioa prozeduratik kanpo bueltatzea nahiko dugu. Xede hau betetzeko, sarrera-irteera erako parametroa erabiliko dugu. Parametro-espezifikazioan, bi puntuen ondoren *datu-emaitza* hitza ipiniz zehazten da hau.

Hemen duzu sarrera-irteera parametroa duen prozedura sinple bat:

```
algoritmo Gehitu (Kont : datu emaitza Osokoa)
hasiera
     Kont := Kont + 1;
amaia Gehitu;
```

Horrela dei diezaiokegu prozedura honi:

```
Gehitu (N)
```

Sarrera-irteera erako parametroen funtzionamendua honako hau da: sarrera--irteera erako parametro formal bakoitzak prozedurako *aldagai lokal* baten moduan jokatzen du. Prozedura-gorputzean sartzean, aldagai lokalaren balioa berari dagokion parametro errealaren balioa da. Prozedura-gorputza egikaritzen ari den bitartean honi esleitzen zaion edozein balio, dagokion parametro errealari pasatuko zaio.

Irteera-parametroekin bezala, parametro errealak parametro formalaren mota bereko aldagaia behar du izan.

Funtzioek sarrera-parametroak bakarrik eduki ditzakete. Honek arrazoi on bat du: irteera- edo sarrera-irteera erako parametroa duen funtzio batek, deitzen zaionean, bere barrukoa ez den aldagai baten balioa alda dezake. Horrelako fenomenoari *albo-ondorio* deritzo. Funtzioetan, albo-ondorio hauek ez dira gomendagarriak, algoritmoen irakurgarritasuna zailtzen baitute.

II.6. ALGORITMOEN IDAZKERA

II.6.1. Formatu orokorra

Hau da algoritmoak idazteko jarraituko dugun eredua:

```
algoritmo <identifikadorea>
hasiera
    <agindua>*
amaia
```

non agindu bakoitza ondoko bat den:

- Asignazioa
- Idazketa
- Irakurketa
- *Baldin* kontrol-egitura
- *Bitartean* kontrol-egitura
- *Guztietarako* kontrol-egitura
- Prozedura-deia

Batzuek nahiago dute aldagaiak esplizituki erazagutzea (gure ariketabilduman ez dugu horrela egingo):

```
algoritmo <identifikadorea>
     [<identifikadorea> : <datu-mota>]*
hasiera
     <agindua>*
amaia
```

II.6.2. Programazio-estiloa

Programak irakurterrazak izan daitezen :

- Erabil itzazu iruzkinak lasai, programaren funtzioa adierazi eta zati bakoitzaren funtzionamendua azaltzeko. Iruzkinak "-" gidoi-parearen ondoan idatziko ditugu. Lerro bukaeraraino dauden gainontzeko karaktereak, ohar gisa hartuko dira.

- Aukera itzazu ahalik eta identifikadore deskriptiboenak.

- Erabil maiuskulak edo minuskulak identifikadoreetan, baina beti era berean! Liburu honetan minuskulaz idatzi ditugu baina lehenengo letra beti maiuskulaz. Objektuen identifikadoreak izenak izaten dira; prozedurenak, aditzak.

- Ez idatzi algoritmo luzeegiak. Saia zaitez problema nagusia azpiproblema errazagotan banatzen eta, geroago, hauek aparte definitzen.

Programatzaile askok sententziak bikoiztea ekiditeko tresna erabilgarri huts gisa ikusten dituzte moduluak. Ikuspegi honek, ordea, moduluak erabiliz lortzen diren onurak gutxiesten ditu. Hemen duzu horien laburpen bat:

- Modulu bat, *nola* funtzionatzen duen jakin gabe izan daiteke deitua; moduluaren erabiltzaileak moduluak *zer* egiten duen bakarrik jakin behar du. Modulu bat zertan erabiliko den jakin gabe irakur eta uler daiteke. Gai hauek bereizteari *abstrakzio* deritzo eta tresna intelektual ahaltsua da problema konplexuak ebazteko.

- Azpiprogramek zati askeez osatutako programa bat eraikitzeko balio dute, non modulu bakoitzak xede bakun bat izango duen. Horrela programa luze baten eraikuntza asko errazten da modulu bakoitza bereizita idatz eta azter baitaiteke.

- Azpiprogramek izenak ematen dizkiete programen zatiei. Azpiprogramentzako (eta objektu eta datu-motak moduko hainbat entitateentzako) izen zentzudunak aukeratuz gero, programa "testu" gisa irakur daiteke. Modu-

luarentzat izen egokia aukeratzea errazagoa da, moduluak xede bakun eta sinpleren bat baldin badu.

- Modulu bati toki desberdin askotatik dei dakioke, nahi izanez gero parametro erreal desberdinekin. Honek programa-testuaren bikoizte aspergarria (eta errore-sortzailea) ekiditen du.

- Erraza da modulu-gorputzaren bertsio bat beste batekin ordezkatzea. Moduluaren *erazagupena* aldatzen ez den bitartean, programaren gainerako zatiek ere ez dute aldatu beharrik.

III. Algoritmoen osagaiak Ada programazio-lengoaian

III.1. PROGRAMETAKO HITZAK ETA IRUZKINAK

Ada-programa batean idatz daitezkeen ikurrak (~hitzak) honelaxe sailka daitezke:

- Literalak.
- Identifikadoreak.
- Hitz erreserbatuak.
- Banatzaileak.

Beraz,programa bat literal, identifikadore eta hitz erreserbatuekin idazten da, beti ere haiek bereizteko banatzaileak erabiliz. Gainera, programak irakurgarriago bihurtzeko, iruzkinak ere tarteka daitezke.

III.1.1. Literalak

Balio konstanteak adierazteko erabiltzen dira (mota desberdinetakoak), adibidez:

0 1 60 1_000_000 (osoko literalak)

0.0 3.14158 (literal errealak)

'H' ':' ' ' (karaktereak)

"Ordua: " "???" (kateak)

III.1.2. Identifikadoreak

Konstante, aldagai, mota, azpiprograma, pakete eta Ada programetako beste entitate batzuei ematen zaizkien izenak dira.

Letra beraren maiuskulak eta minuskulak baliokideak dira:

seg_minutuko = Seg_Minutuko = SEG_MINUTUKO /= SegMinutuko

Aukera itzazu identifikadore ahalik eta deskriptiboenak. Erabil itzazu maiuskulak edo minuskulak (baina beti era berean!).

III.1.3. Hitz erreserbatuak

Ada lengoaian xede jakinetarako erabiltzen diren ingelesezko hitzak dira. Ez dira aukeratu behar identifikadore modura. Horregatixe esaten zaie 'hitz erreserbatuak'.

Esate baterako : **if, then, else, while, loop, procedure, begin, end, constant, …**

III.1.4. Banatzaileak

Ondoz ondoko zenbaki, identifikadore edo hitz erreserbatuen artean, gutxienez ondoko banatzaileetako bat jarri behar da:

, ; : . '

( )

**** * / + - &**

= /= < <= >= >

:= .. | => <>

baita zuriunea ere.

III.1.5. Iruzkinak

Erabil itzazu lasai iruzkinak, programaren funtzioa adierazi eta zati bakoitzaren funtzionamendua azaltzeko..

Iruzkinak Ada lengoaian: "– –" gidoi pare baten ondoan eta lerro bukaeraraino.

III.2. OBJEKTUAK ADA LENGOAIAN

III.2.1. Oinarrizko DATU-MOTAK Ada lengoaian

Datu-mota	Balioak	ADA-n
Osokoa	osoko zenbakiak	Integer
Erreala	zenbaki errealak	Float
Karakterea	karaktereak	Character
Boolearra	true eta false	Boolean
Katea	karaktere-kateak	String

III.2.2. Objektu konstanteak eta aldagaiak

Prozedura hasieran erazagutu behar dira.

Konstanteei balioa ezartzen zaie:

```
Pi : constant Float := 3.14159 ;
Zuriunea : constant Character  := ´ ´  ;
```

Aldagaiei hasierako balioa ezar dakieke:

```
Zabalera : Float  ;
Kontagailua : Integer := 0 ;
```

III.3. OINARRIZKO AGINDUAK ADA LENGOAIAN

Agindu guztiek “;” karakterearekin bukatu behar dute.

III.3.1. Datu-irakurketa (teklatutik)

Prozedura hauek erabiliko ditugu gure programak ulergarriago izan daitezen:

Irakurri_Osokoa (*aldagai*) ;

Irakurri_Erreala ((*aldagai*) ;

Irakurri_Karakterea (*aldagai*) ;

Baina prozedura horiek ez dira estandarrak, honela definitu behar ditugu:

```
with Ada.Integer_Text_IO;
procedure Irakurri_Osokoa (I: out Integer) is
begin
   Ada.Integer_Text_IO.Get (I);
end Irakurri_Osokoa;


with Ada.Float_Text_IO;
procedure Irakurri_Erreala (X: out Float) is
begin
   Ada.Float_Text_IO.Get (X);
end Irakurri_Osokoa;


with Ada.Text_IO;
procedure Irakurri_Karakterea (Kar: out Character) is
begin
   Ada.Text_IO.Get (Kar);
end Irakurri_Osokoa;
```

III.3.2. Datu-idazketa (pantailan)

Prozedura hauek erabiliko ditugu gure programak ulergarriago izan daitezen:

Idatzi_Osokoa (*adierazpen*) ;

Idatzi_Erreala (*adierazpen*) ;

Idatzi_Karakterea (*adierazpen*) ;

Idatzi_Katea (*adierazpen*) ;

Baina prozedura horiek ez dira estandarrak, honela definitu behar ditugu:

```
with Ada.Integer_Text_IO;
procedure Idatzi_Osokoa (I: in Integer) is
begin
   Ada.Integer_Text_IO.Put (I);
end Idatzi_Osokoa;


with Ada.Float_Text_IO;
procedure Idatzi_Erreala (X: in Float) is
begin
   Ada.Float_Text_IO.Put (X);
end Idatzi_Osokoa;
```

```
with Ada.Text_IO;
procedure Idatzi_Karakterea (Kar: in Character) is
begin
   Ada.Text_IO.Put (Kar);
end Idatzi_Osokoa;
```

```
with Ada.Text_IO;
procedure Idatzi_Katea (Katea: in String) is
begin
   Ada.Text_IO.Put (Katea);
end Idatzi_Katea;
```

III.3.3. Esleipenak (asignazioak)

Algoritmoetan bezalaxe

aldagaia := adierazpena ;

Adierazpena ebaluatuz lortzen den balioa, aldagaiaren balio berri gisa ezartzen du. Aldagaiak aurretik zeukan balioa galtzen du.

Adierazpenean aldagairik azaltzen bada, berau ebaluatzen denean daukan balioa lortzen da, baina balio hori ez da aldatzen.

III.4. KONTROL-EGITURAK ADAN

III.4.1. Baldintzazko egitura

Hona hemen Ada lengoaian baldintzazko egiturak adierazteko dauden aukerak:

```
baldin  baldintza orduan
    ekintza1...ekintzan
ambaldin
```

```
if  baldintza then
    ekintza1...ekintzan
end if;
```

```
baldin  baldintza orduan
    ekintza1...ekintzan
bestela ekintza21...ekintza2n
ambaldin
```

```
if  baldintza then
    ekintza1... ekintzan
else ekintza21... ekintza2n
end if;
```

```
baldin  b1 orduan
    ekintza11...ekintza1n
bestela
    baldin b2 orduan
      ekintza21... ekintza2n
    bestela
      baldin b3 orduan
        ekintza31...ekintza3n
      bestela
        ekintza41... ekintza4n
      ambaldin
    ambaldin
ambaldin
```

```
if  b1 then
    ekintza11... ekintza1n
elsif b2 then
    ekintza21... ekintza2n
elsif b3 then
    ekintza31... ekintza3n
else
    ekintza41... ekintza4n
end if ;
```

III.4.2. Iterazioa

III.4.2.1. Iterazio arrunta

```
bitartean baldintza egin
   ekintza1
   ...
   ekintzan
ambitartean

errepikatu
   ekintza1
   ...
   ekintzan
amerrepikatu
```

```
while  baldintza loop
    ekintza1
    ...
    ekintzan
end loop ;

loop
    ekintza1
    ...
    ekintzan
exit when baldintza ;
```

III.4.2.2. Aldi kopuru jakineko iterazioa

```
egin I guztietarako n1 tik n2 raino
    ekintza1 ..ekintzan
amguztietarako
```

```
for I in  n1 .. n2  loop
    ekintza1 ..ekintzan
end loop;
```

III.5. MODULUAK EDO AZPIPROGRAMAK ADA LENGOAIAN (SINPLIFIKATUA)

```
procedure ident_prozedura atal_formala is
     erazagupen_atala

begin
     sententzia_sekuentzia
end ident_prozedura ;
```

Atal formalean parametroak eta beren motak definitu behar dira, eta horiek datu (*in*), emaitza (*out*) edo datu-emaitza (*in out*) erakoak diren adierazi behar da.

Azpiprograma-adibidea (I)

```
-- Ordua_Erakutsi programak, gauerdiaz gero pasatu diren
-- segundoak irakurtzen ditu eta 24 orduko adierazpidea
-- erabiliz idazten du O:M:S formatuan.
-- 54450 sarrera-datuarekin programaren irteera hau da:
-- Ordua:   15:    7:    30
with Irakurri_Osokoa, Idatzi_Osokoa, Idatzi_Katea ;
procedure Ordua_Erakutsi is
     Seg_Minutuko: constant Integer:= 60;
     Min_Ordubeteko: constant Integer:= 60;
     Seg_Ordubeteko: constant Integer
          := Seg_Minutuko * Min_Ordubeteko;
     Ordua : Integer;
     O, M, S   : Integer;
begin
     Irakurri_Osokoa (Ordua);  -- gauerdiaz geroko segundoak
     H:= Ordua / Seg_Ordubeteko; -- gauerdiaz geroko orduak
     Ordua:= Ordua rem  Seg_Ordubeteko;
          -- azken orduaz geroko segundoak
     M := Ordua / Seg_Minutuko;
          -- azken orduaz geroko minutuak
     S := Ordua  rem  Seg_Minutuko;
          -- azken minutuaz geroko segundoak
     Idatzi_Katea ("Ordua:   ");
     Idatzi_Osokoa (O);
     Idatzi_Katea (":");
     Idatzi_Osokoa (M);
     Idatzi_Katea (":");
     Idatzi_Osokoa (S);
end Ordua_Erakutsi;
```

Azpiprograma-adibidea (II)

```
function Faktoriala (N : in Integer) return Integer is
-- Aurrebaldintza: N osoko zenbakia, N > 0
-- Postbaldintza: Fakt zenbaki osokoa, Fakt N zenbakiaren
--                faktoriala da
     Fakt : Integer := 1;
     N, I : Integer;
begin
     I := 1;
     while  I <= N  loop
          Fakt := Fakt * I ;
          I := I + 1 ;
     end loop ;
     return Fakt ;
end Faktoriala ;
```

Azpiprograma baten barruan beste azpiprograma bat erabili nahi denean, bi aukera daude hori egiteko.

1) Azpiprograma lagungarria barruan definitzea. Kasu honetan azpiprograma lagungarri hori ezin da erabili azpiprograma nagusi horretatik kanpo, azpiprograma lokala izango baita. Adibidea:

a.adb fitxategia:

```
procedure  A … is
     …
     procedure  B … is
     begin
          …
     end B;
begin
     …
     B(…)
     …
end A;
```

2) Azpiprograma lagungarria beste fitxategi batean definitzea, prozeduraren izen berarekin, eta hura erabili nahi duen azpiprogramaren definizioa baino lehen *with* erako sententzia bat erabiltzea beste prozeduraren izenarekin. Kasu honetan, azpiprograma lagungarria beste edozein programatatik ere erabili ahal izango da.

b.adb fitxategia:

```
procedure B … is
begin
     …
end B;
```

a.adb fitxategia:

```
with B;
procedure A … is
begin
     …
     B (…)
     …
end A;
```

B zatia:
Ariketa-bilduma

IV. Algoritmoen oinarrizko osagaiak

Helburuak: Algoritmo sinpleak sortzea oinarrizko osagaiak erabiliz.

ENUNTZIATUAK

Adierazpenak ebaluatzeko ordena

Zein izango da adierazpen hauek ebaluatzeko ordena?

```
a)  Not Eguzkitsua or Euritsua
b)  X>4.0 eta Y>0.0
c)  -4.0 * A**2
d)  abs (1 + A) + B
e)  A / B * C
f)  A / (B * C)
g)  (- 4) * ( A ** (5 + 1))
h)  (- 4) *  A ** (5 + 1)
i)  (A / (B * C))
j)  A *B / C
k)  abs (X-Y**2) > 2.0*X*0.001
l)  (A / B) * C
m)  A + B * C
```

Balio absolutua

Osoko zenbaki bat irakurri eta bere balio absolutua idatzi.

Ordenatu bi zenbaki

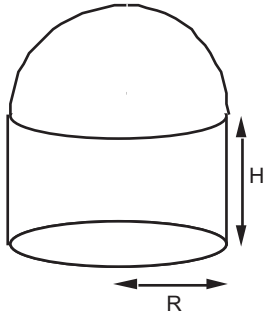
Osoko zenbaki bi irakurri eta ordenatuta idatzi (lehenengoz handiena eta txikiena gero).

Orduak beste formatuan

Gauerdiaz gero pasatu diren segundoak irakurri eta 24 orduko adierazpidera pasa. Adibidez: 4005 irakurrita, 1:6:45 idatzi behar da, zeren gauerdiaz gero 4005 segundo pasa direnean, ordu bat 6 minutu eta 45 segundo pasa baitira.

Bolumenaren kalkulua

Ondoko irudiaren bolumena idatzi *R* erradioaren neurria eta *H* altuerarena (biak zenbaki errealak) irakurri ondoren.



Zatitzaileen kalkulua

Osoko zenbaki positibo bat irakurri eta bere zatitzaile guztiak idatzi.

EBAZPENAK

IV.1. Adierazpenak ebaluatzeko ordena

Zein izango da adierazpen hauek ebaluatzeko ordena?

a) `Not Eguzkitsua or Euritsua`
`((Not Eguzkitsua) or Euritsua)`

b) `X>4.0 eta Y>0.0`
`((X>4.0) eta (Y>0.0))`

c) `-4.0 * A**2`
`((-4.0) * (A**2))`

d) `abs (1 + A) + B`
`((abs (1 + A)) + B)`
Ez dago zalantzarik

e) `A / B * C`
`((A / B) * C)`

f) `A / (B * C)`
`( A / (B * C))`
Ez dago zalantzarik

g) `(- 4) * ( A ** (5 + 1))`
`((- 4) * ( A ** (5 + 1)))`
Ez dago zalantzarik

h) `(- 4) *  A ** (5 + 1)`
`((- 4) *  (A ** (5 + 1)))`
Aurrekoaren berdina

i) `(A / (B * C))`
`(A / (B * C))`
Ez dago zalantzarik

j) `A *B / C`
`((A *B) / C)`

k) `abs (X-Y**2) > 2.0*X*0.001`
`((abs (X-(Y**2))) > ((2.0*X)*0.001))`

l) `(A / B) * C`
`((A / B) * C)`
Ez dago zalantzarik

m) `A + B * C`
`(A + (B * C))`

IV.2. Balio absolutua

Osoko zenbaki bat irakurri eta bere balio absolutua idatzi.

```
algoritmoa Absolutua
hasiera
     Irakurri_Osokoa(N)
     baldin N < 0  orduan  N := -N
     ambaldin
     Idatzi_Osokoa (N)
amaia
```

IV.3. Ordenatu bi zenbaki

Osoko zenbaki bi irakurri eta ordenatuta idatzi (lehenengoz handiena eta txikiena gero).

```
algoritmo Bi_Ordenatuta
hasiera
     Irakurri_Osokoa(N1)
     Irakurri_Osokoa(N2)
     baldin N1 < N2  orduan
          Lagun := N1
          N1 := N2
          N2 := Lagun
     ambaldin
     Idatzi_Osokoa (N1)
     Idatzi_Osokoa (N2)
amaia
```

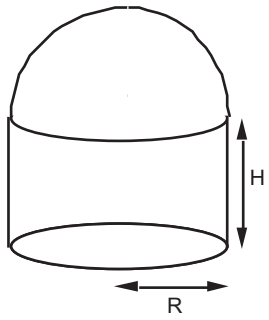
IV.4. Orduak beste formatoan

Gauerdiaz gero pasatu diren segundoak irakurri eta 24 orduko adierazpidera pasa. Adibidez: 4005 irakurrita, 1:6:45 idatzi behar da, zeren gauerdiaz gero 4005 segundo pasa direnean, ordu bat 6 minutu eta 45 segundo pasa baitira.

```
algoritmo Lortu_Ordua
hasiera
     Irakurri_Osokoa(Denbora)
     Orduak := Denbora / 3600
     Denbora := Denbora mod 3600
     Minutuak := Denbora / 60
     Segundoak := Denbora mod 60
     Idatzi_Osokoa (Orduak)
     Idatzi_Osokoa (Minutuak)
     Idatzi_Osokoa (Segundoak)
amaia
```

IV.5. Bolumenaren kalkulua

Ondoko irudiaren bolumena idatzi *R* erradioaren neurria eta *H* altuerarena




```
algoritmo Lortu_Bolumena
hasiera
     Irakurri_Erreala(R)
     Irakurri_Erreala(H)
     Bolumena := (1/2) * (4/3) * 3.1416 * R**3
           + H * 3.1416 * R**2
     Idatzi_Erreala (Bolumena)
amaia.
```

IV.6. Zatitzaileen kalkulua

Osoko zenbaki positibo bat irakurri eta bere zatitzaile guztiak idatzi.

```
algoritmo Zatitzaileak
hasiera
     Irakurri_Osokoa(N)
     I := 1
     bitartean I<= N egin
           Hondarra := N mod I
           baldin Hondarra = 0 orduan Idatzi_Osokoa(I)
           ambaldin
           I := I+1
     ambitartean
amaia.
```

V. Algoritmoen zehaztapenak I.

Helburuak: Zehaztapenak egiten ikastea eta ohitzea algoritmoak egiten osagai sinpleekin: irakurketak eta idazketak, asignazioak, eta abar.

ENUNTZIATUAK

Bi aldagaien arteko balio trukaketa

A eta B aldagaien balioak elkarren artean trukatzeko algoritmoa espezifikatu eta idatzi.

Hiru zenbaki ordenatu

Hiru osoko zenbaki irakurri eta hirurak handienetik txikienera ordenatuta idatziko dituen algoritmoa espezifikatu eta idatzi.

Txanponak

Jo dezagun 25, 10, 5 eta 1 pezetako txanponak onartzen dituen makina bat dugula. Sarrera bezala (1) makinan sartutako guztizkoa eta (2) aukeratutakoaren prezioa ematen zaizu (biak pezetatan). Idatz ezazu algoritmo bat makinak itzuli behar duen txanpon-kopuru txikiena zehaztuko duena.

Triangelua

A, B eta C triangelu baten aldeen luzera emanik, triangeluaren azalera kalkulatuko duen algoritmoa idatz ezazu. (S triangeluaren perimetroaren erdia bada, bere azalera horrela kalkulatzen da $\sqrt{[S(S-A)(S-B)(S-C)]}$. Ez dago A, B eta C aldeak dituen triangelurik biderkaketa hau negatiboa bada). X zenbakiaren erro karratua Erro_Karratua (X) adierazpena ebaluatuz lortzen dela suposa ezazu.

Bigarren mailako ekuazioak

A, B eta C emanda Ax2 + Bx + C = 0 bigarren mailako ekuazioko emaitzak kalkulatuko duen algoritmoa idatz ezazu. Suposatu bi soluzioak errealak direla; hau da, ez dagoela soluzio irudikaririk eta A /= 0 dela beti.

EBAZPENAK

V.1. Bi aldagaien arteko balio trukaketa

A eta B aldagaien balioak elkarren artean trukatzeko algoritmoa espezifikatu eta idatzi.

Zehaztapena
Aurrebaldintza: A=a eta B=b
Postbalbintza: A=b eta B=a

```
algoritmo Balioak_Trukatu (A, B: datu emaitza Osokoak)
hasiera
     C := A
—— A-ren balioa ez galtzeko beste aldagai batean utzi behar da
     A := B
     B := A
amaia
```

V.2. Hiru zenbaki ordenatu

Hiru osoko zenbaki irakurri eta hirurak handienetik txikienera ordenatuta idatziko dituen algoritmoa espezifikatu eta idatzi.

Zehaztapena
Aurrebaldintza: A, B eta C (osokoak)
Postbaldintza: A >= B >= C

```
algoritmo Ordenatu3
hasiera
     Irakurri_Osokoa (A)
     Irakurri_Osokoa (B)
     Irakurri_Osokoa (C)
     baldin  A < B orduan      —— A ?>= B,  B ?>= C ,  A ?>= C
          Lagun := A
          A := B
          B := Lagun
     ambaldin
     baldin  B < C orduan      —— A >= B,  B ?>= C,   A ?>= C
          Lagun := B
          B := C
          C := Lagun
     ambaldin
     baldin  A < B orduan      —— A >= C,  B >= C,  A ?>= B
          Lagun := A
          A := B
          B := Lagun
     ambaldin
     —— A >= B >= C
     Idatzi_Osokoa (A)
     Idatzi _Osokoa (B)
     Idatzi _Osokoa (C)
amaia
```

Baina aldez aurretik bi aldagaien balioak trukatzen dituen Balioak_Trukatu algoritmoa definituta badugu, berau erabiliz errazago idatz dezakegu gure azken algoritmoa:

```
algoritmo Ordenatu3
hasiera
     Irakurri_Osokoa (A)
     Irakurri_Osokoa (B)
     Irakurri_Osokoa (C)
     baldin  A < B orduan      —— A ?>= B,  B ?>= C ,  A ?>= C
          Balioak_Trukatu (A, B)
     ambaldin
     baldin  B < C orduan      —— A >= B,  B ?>= C,   A ?>= C
          Balioak_Trukatu (B, C)
     ambaldin
     baldin  A < B orduan      —— A >= C,  B >= C,  A ?>= D
          Balioak_Trukatu (A, B)
     ambaldin
     —— A >= B >= C
     Idatzi_Osokoa (A)
     Idatzi_Osokoa (B)
     Idatzi_Osokoa (C)
amaia.
```

Egokitu beharko genuke *Balioak_Trukatu* algoritmoa. Erabili nahi izango denean adierazi beharko da zeintzuk diren balioak trukatu behar duten aldagaiak. Kasu honetan ez du balio adierazpen bat ematea, aldagaiak izan behar dute parametroek. Beraien hasierako balioa datu izango da eta azkeneko balioak emaitza, aldagai horiek datu eta emaitzaren papera egiten dute. Beraz, honela adieraziko dugu:

```
algoritmo Balioak_Trukatu (A, B: datu emaitza Osokoak)
hasiera
     C := A
—— A-ren balioa ez galtzeko beste aldagai batean utzi behar da
     A := B
     B := A
amaia
```

V.3. Txanponak

Jo dezagun 25, 10, 5 eta 1 pezetako txanponak onartzen dituen makina bat dugula. Sarrera bezala (1) makinan sartutako guztizkoa eta (2) aukeratutakoaren prezioa ematen zaizu (biak pezetatan). Idatz ezazu algoritmo bat makinak itzuli behar duen txanpon-kopuru txikiena zehaztuko duena.

Zehaztapena
Aurrebaldintza: Sartutakoa, Prezioa (Osokoa)
Postbaldintza: Hogeitabostekoa, Hamarrekoa, Bostekoa eta Batekoa (osokoak),
Hogeigetabostekoa*25+Hamarrekoa*10 + Bostekoa*5 + Batekoa
= Sartutakoa – Prezioa

```
algoritmo Itzuli_Txanponak
hasiera
     Irakurri_Osokoa (Sartutakoa)
     Irakurri_Osokoa (Prezioa)
     Itzultzekoa := Sartutakoa - Prezioa
     bitartean   Itzultzekoa  >=  25 egin
     —— 25eko txanponak eman
          Itzultzekoa := Itzultzekoa -25
          Idatzi_Osokoa (25)
     ambitartean
     bitartean   Itzultzekoa  >= 10 egin — 10eko txanponak eman
          Itzultzekoa := Itzultzekoa -10
          Idatzi_Osokoa (10)
     ambitartean
     bitartean   Itzultzekoa  >=  5 egin — 5eko txanponak eman
          Itzultzekoa := Itzultzekoa - 5
          Idatzi_Osokoa (5)
     ambitartean
     bitartean   Itzultzekoa  >=  1 egin — bateko txanponak eman
          Itzultzekoa := Itzultzekoa - 1
          Idatzi_Osokoa (1)
     ambitartean
amaia
```

Beste era batera:

```
algoritmo Itzuli_Txanponak
hasiera
     Irakurri_Osokoa (Sartutakoa)
     Irakurri_Osokoa (Prezioa)
     Itzultzekoa := Sartutakoa - Prezioa
     Idatzi_Osokoa (Itzultzekoa / 25) — 25eko txanponak eman
     Idatzi_Katea (“  txanpon 25ekoak”)
     Itzultzekoa := Itzultzekoa mod 25 — 10eko txanponak eman
     Idatzi_Osokoa (Itzultzekoa / 10)
     Idatzi_Katea (“  txanpon 10ekoak”)
     Itzultzekoa := Itzultzekoa mod 10 — 5eko txanponak eman
     Idatzi_Osokoa (Itzultzekoa / 5)
     Idatzi_Katea (“  txanpon 5ekoak”)
     Itzultzekoa := Itzultzekoa mod 5 — 10eko txanponak eman
     Idatzi_Osokoa (Itzultzekoa)
     Idatzi_Katea (“  txanpon 1ekoak”)
amaia
```

V.4. Triangelua

A, B eta C triangelu baten aldeen luzera emanik, triangeluaren azalera kalkulatuko duen algoritmoa idatz ezazu. (S triangeluaren perimetroaren erdia bada, bere azalera horrela kalkulatzen da $\sqrt{[S(S-A)(S-B)(S-C)]}$. Ez dago A, B eta C aldeak dituen triangelurik biderkaketa hau negatiboa bada). X zenbaki erreal baten erro karratua *Erro_Karratua(X)* adierazpena ebaluatuz lortzen dela suposa ezazu.

Zehaztapena
Aurrebaldintza: A, B eta C (zenbaki errealak), A>0 eta B>0 eta C > 0
Postbalbintza: K (zenbaki erreala) A, B eta C aldeak dituen triangeluaren azalera

```
algoritmo Azalera
hasiera
     Irakurri_Erreala (A)
     Irakurri_Erreala (B)
     Irakurri_Erreala (C)
     S :=  (A + B + C) / 2.0
     Biderketa :=  S * (S - A) * (S - B) * (S - C)
     baldin Biderketa  <= 0 orduan
          Idatzi_Katea (" Ezin da osatu triangelurik")
     bestela
          Azalera := Erro_Karratua (Biderketa)
          Idatzi_Erreala (Azalera)
     ambaldin
amaia
```

VI. Sekuentzien tratamendua

Helburuak: Sekuentzia baten korritze-prozesu desberdinak aztertzen dira; adibidez, elementu nabarmen baten (edo batzuen) bilaketa edo sekuentzia baten elementu guztiekin eragiketa bat egitea proposatzen da. Hasierako ariketetan sekuentziako elementuak irakurriz lortuko dira, bukaerako ariketetan, ordea, sortu egin beharko dira tratatzeko elementuak.

ENUNTZIATUAK

'A' karakterearen kontaketa sekuentzian

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta ezazu 'A' letra zenbat aldiz agertzen den.

Bokalen agerpena sekuentzian

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta ezazu karaktere horien guztien artean zenbat bokal agertzen diren.

Bokal ez direnen agerpen-kopurua sekuentzian

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta itzazu karaktere horien guztien artean bokalak ez direnak (puntua ez kontatu).

Bokalen, ez-bokalen eta karaktereen kontaketa

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta ezazu karaktere guzti horien artean zenbat bokal, bokal ez diren zenbat karaktereak eta guztira zenbat karaktere agertzen diren.

Sekuentziako osagaien batezbestekoaren kalkulua.

Zero zenbakiaz amaitzen den eta gutxienez beste zenbaki bat duen osoko-sekuentzia bat emanda, kalkula ezazu sekuentziako zenbakien batezbesteko aritmetikoa (zeroa kontatu gabe).

Sekuentziako osagai positiboen batezbestekoaren kalkulua

Zero zenbakiaz amaitzen den eta gutxienez beste zenbaki bat duen osoko-sekuentzia bat emanda, kalkula ezazu sekuentziako zenbaki positiboen batezbesteko aritmetikoa.

Elementu baten bilaketa sekuentzia ez-ordenatuan

Zero zenbakiaz amaitzen den osoko-sekuentzia ez-ordenatu batean zenbaki bat bilatzeko algoritmoa espezifikatu eta idatzi. Bilatu behar den zenbakia, sekuentziako lehenengoa izango da. Zenbakia sekuentziako gainontzeko guztien artean badago, sekuentzia barruko posizioa idatzi beharko da (lehenengoa kontuan hartu gabe, noski); eta bestela, ez dagoenean, zero idatzi beharko da. Adibidez:

Sekuentzia:	7	9	8	-23	147	7	58	71	0
Posizioak:		1	2	3	4	5	6	7	

7 zenbakia sekuentziako 5. posizioan dagoenez, 5 izan beharko da emaitza.

Elementu baten bilaketa sekuentzia ordenatuan

Aurreko ariketan bezala, baina kasu honetan txikienetik handienera ordenatuta dagoen sekuentzia batean.

Elementu maximoaren bilaketa sekuentzian

Zero zenbakiaz amaitzen den osoko-sekuentzia batean zenbaki maximoa bilatzeko algoritmoa espezifikatu eta idatzi.

Elementu maximoaren posizioa sekuentzian

Zero zenbakiaz amaitzen den osoko-sekuentzia batean zenbaki maximoaren posizioa bilatzeko algoritmoa espezifikatu eta idatzi.

Elementu minimoaren bilaketa sekuentzian

Zero zenbakiaz amaitzen den osoko-sekuentzia batean zenbaki minimoa bilatzeko algoritmoa espezifikatu eta idatzi.

Elementu minimoaren posizioa sekuentzian

Zero zenbakiaz amaitzen den osoko-sekuentzia batean zenbaki minimoaren posizioa bilatzeko algoritmoa espezifikatu eta idatzi.

"TA" karaktere-bikotearen kontaketa sekuentzian

Puntu karaktereaz amaitzen den karaktere-sekuentzia batean, konta ezazu zenbat aldiz agertzen den 'A' karakterea 'T' karaktere baten atzetik.

Lehena ote den aztertu

N osoko zenbaki bat emanda, zenbaki lehena denentz aztertuko duen algoritmoa espezifikatu eta idatzi.

M baino txikiagoak diren zenbaki lehenak kalkulatu

Aurreko ariketako algoritmoa erabiliz, *M* osokoa baino txikiagoak diren zenbaki lehenak idatziko dituen algoritmoa espezifikatu eta idatzi.

Erro karratuaren kalkulua, Newtonen metodoaren arabera

Idatz ezazu *X* zenbaki erreal positiboaren erro karratua kalkulatu eta idatziko duen algoritmoa. Erabil ezazu Newtonen metodoa: *E* balioa *X*-ren erro karraturako estimatutako balioa bada, orduan *E* eta *X/E*-ren batezbesteko aritmetikoa estimazio hobea izango da. Eman hasieran *E*-ri balio positibo bat, ondoren aurreko formula erabiliz *E* estimazio hobe batez ordezkatu behin eta berriz, *E*-ren balioa *X*-ren baliora nahikoa hurbiltzen denean geldituz.

Exekutatu algoritmoa

Ondoko algoritmoa emanda, esan ezazu zer egiten duen.

```
hasiera
     K := 3
     bitartean K >= 1 egin
          J := K
          bitartean J >= 1 egin
              Idatzi_Osokoa (J)
              J := J - 1
          ambitartean
          Idatzi_Osokoa (K)
          K := K-1
     ambitartean
amaia
```

Exekutatu algoritmoa

Zein da ondoko algoritmoak emango duen emaitza, ondoko osoko-sekuentzia emanez gero: <5 6 -3 7 -4 0 5 8 9>

```
hasiera
     Batura := 0
     K:= 1
     B := False
     bitartean K <= 8 and B= False egin
           Irakurri _Osokoa (N)
           baldin N > 0 orduan
               Batura := Batura + N
     bestela
           baldin N=0 orduan
               B := True
           ambaldin
     ambaldin
     K := K+1
   ambitartean
   Idatzi _Osokoa (Batura)
amaia
```

EBAZPENAK

VI.1. 'A' karakterearen kontaketa sekuentzian

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta ezazu 'A' letra zenbat aldiz agertzen den.

Zehaztapena
Aurrebaldintza: S (karaktere-sekuentzia). Sekuentziaren azken karakterea puntua da eta beste punturik ez dago.
Postbaldintza: Kontagailua (osokoa). Kontagailua-k S sekuentziaren A' letraren Agerpen-kopurua adierazten du.

```
algoritmo A_Letrak_Kontatu
hasiera
      Irakurri_Karakterea (Kar)
      Kontagailua := 0
      bitartean Kar /= '.' egin
            baldin Kar = 'A' orduan
                Kontagailua := Kontagailua + 1
            ambaldin
            Irakurri_Karakterea (Kar)
      ambitartean
      Idatzi_Osokoa (Kontagailua )
amaia.
```

VI.2. Bokalen agerpena sekuentzian

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta ezazu karaktere horien guztien artean zenbat bokal agertzen diren.

Aurreko ariketan bezala, baina aldaketa batekin: Karakterea edozein bokal denean kontatu behar da.

Zehaztapena
Aurrebaldintza: S (karaktere-sekuentzia), Sekuentziaren azken karakterea puntua da eta beste punturik ez dago.
Postbaldintza: Kontagailua (osokoa), Kontagailua-k S sekuentzian guztira zenbat bokal dauden adierazten du.

```
algoritmo Bokalak_Kontatu
hasiera
     Irakurri_Karakterea (Kar)
     Kontagailua := 0
     bitartean Kar /= '.' egin
          baldin Kar = 'A'  edo Kar = 'E'  edo Kar = 'I'
              edo Kar = 'O'  edo Kar = 'U' orduan
              Kontagailua := Kontagailua + 1
          ambaldin
          Irakurri_Karakterea (Kar)
     ambitartean
     Idatzi_Osokoa (Kontagailua )
amaia
```

VI.3. Bokal ez direnen agerpen-kopurua sekuentzian

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta itzazu karaktere horien guztien artean bokalak ez direnak (puntua ez kontatu).

Aurreko ariketan bezala, baina aldaketa batekin. Karakterea bokala ez denean kontatu behar da.

Zehaztapena
Aurrebaldintza: S (karaktere-sekuentzia). Sekuentziaren azken karakterea puntua da eta beste punturik ez dago.
Postbaldintza: Kontagailua (osokoa). Kontagailua-k S sekuentzian bokalak ez diren karaktereen kopurua adierazten du.

```
algoritmo Ez_Bokalak_Kontatu
hasiera
     Irakurri_Karakterea (Kar)
     Kontagailua := 0
     bitartean Kar /= '.' egin
          baldin   ez   ( Kar = 'A'  edo Kar = 'E'  edo Kar = 'I'
               edo Kar = 'O'  edo Kar = 'U'  ) orduan
               Kontagailua := Kontagailua + 1
          ambaldin
          Irakurri_Karakterea (Kar)
     ambitartean
     Idatzi_Osokoa (Kontagailua )
amaia
```

VI.4. Bokalen, ez-bokalen eta karaktereen kontaketa

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta ezazu karaktere horien guztien artean zenbat bokal, bokal ez diren zenbat karaktereak eta guztira zenbat karaktere agertzen diren.

Aurreko ariketetan bezala baina aldaketa batekin: Karakterea bokala bada alde batetik kontatu behar da, eta bokala ez bada beste alde batetik; eta bukaeran biak idatzi eta beraien batura ere bai.

Zehaztapena
Aurrebaldintza: S (karaktere-sekuentzia), Sekuentziaren azken karakterea puntua da eta beste punturik ez dago.
Postbaldintza: Bokalak, Ez_Bokalak eta Karaktereak (osokoa) direlakoak, hurrenez hurren, S sekuentzian zenbat bokal, zenbat ez-bokal eta zenbat karaktere dauden adierazten dute.

```
algoritmo Bokalak_eta_Ez_Bokalak_Kontatu
hasiera
     Irakurri_Karakterea (Kar)
     Bokalak := 0
     Ez_Bokalak := 0
     bitartean Kar /= '.' egin
          baldin Kar = 'A'  edo Kar = 'E'  edo Kar = 'I'
               edo Kar = 'O'  edo Kar = 'U'
          orduan  Bokalak := Bokalak + 1
          bestela Ez_Bokalak := Ez_Bokalak + 1
          ambaldin
          Irakurri_Karakterea (Kar)
      ambitartean
     Karaktereak := Bokalak + Ez_Bokalak
     Idatzi_Osokoa (Bokalak )
     Idatzi_Osokoa (Ez_Bokalak )
     Idatzi_Osokoa (Karaktereak )
amaia
```

VI.5. Sekuentziako osagaien batezbestekoaren kalkulua.

Zero zenbakiaz amaitzen den eta gutxienez beste zenbaki bat duen osoko-sekuentzia bat emanda, kalkula ezazu sekuentziako zenbakien batezbesteko aritmetikoa (zeroa kontatu gabe).

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken digitua zeroa da eta beste zerorik ez dago. Gutxienez beste zenbaki bat dago.
Postbaldintza: Batezbestekoa (erreala) delakoak S sekuentziako zenbakien batezbesteko aritmetikoa adierazten du.

```
algoritmo Batezbestekoa
hasiera
     Kontagailua := 0
     Batura := 0
     Irakurri_Osokoa (Zenb)
     bitartean  Zenb /= 0  egin
          Kontagailua := Kontagailua + 1
          Batura := Batura + Zenb
          Irakurri_Osokoa (Zenb)
     ambitartean
     Idatzi_Erreala ( Batura / Kontagailua )
amaia
```

Oharra: Batura/Kontagailua adierazpenaren emaitza osokoa denez, Batura eta Kontagailua aldagaien balioak erreal bihurtu behar dira. Beraz, zehazki, azken ekintza honako hau izan beharko litzateke:

Idatzi_Erreala (Erreal(Batura)/Erreal(Kontagailua))

VI.6. Sekuentziako osagai positiboen batezbestekoaren kalkulua

Zero zenbakiaz amaitzen den eta gutxienez beste zenbaki bat duen osoko-sekuentzia bat emanda, kalkula ezazu sekuentziako zenbaki positiboen batezbesteko aritmetikoa.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia), Sekuentziaren azken digitua zeroa da eta beste zerorik ez dago. Gutxienez beste zenbaki bat dago.
Postbaldintza: Positiboen_Batezbestekoa (erreala) delakoak S sekuentziako zenbaki positiboen batezbesteko aritmetikoa adierazten dute.

```
algoritmo Positiboen_Batezbestekoa
hasiera
     Kontagailua := 0
     Batura := 0
     Irakurri_Osokoa (Zenb)
     bitartean  Zenb /= 0  egin
          baldin  Zenb > 0 orduan
              Kontagailua := Kontagailua + 1
              Batura := Batura + Zenb
          ambaldin
          Irakurri_Osokoa (Zenb)
     ambitartean
     Idatzi_Erreala ( Batura / Kontagailua )
amaia
```

Oharra: *Batura/Kontagailua* adierazpenaren emaitza osokoa denez, *Batura* eta *Kontagailua* aldagaien balioak erreal bihurtu behar dira. Beraz, zehazki, azken ekintza honako hau izan beharko litzateke:

Idatzi_Erreala (Erreal(Batura)/Erreal(Kontagailua))

VI.7. Elementu baten bilaketa sekuentzia ez ordenatuan

Zero zenbakiaz amaitzen den osoko-sekuentzia ez-ordenatu batean zenbaki bat bilatzeko algoritmoa espezifikatu eta idatzi. Bilatu behar den zenbakia, sekuentziako lehenengoa izango da. Zenbakia sekuentziako gainontzeko guztien artean badago, sekuentzia barruko posizioa idatzi beharko da (lehenengoa kontuan hartu gabe, noski); eta bestela, ez dagoenean, zero idatzi beharko da. Adibidez:

Sekuentzia:	7	9	8	-23	147	7	58	71	0
Posizioak:			1	2	3	4	5	6	7

7 zenbakia sekuentziako 5. posizioan dagoenez, 5 izan beharko da emaitza.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken digitua zeroa da eta beste zerorik ez dago. Beti daude gutxienez zeroa eta beste zenbaki bat.
Postbaldintza: Posizioa (osokoa). Posizioa-k S sekuentziako lehenengo zenbakiaren bigarren agerpenaren posizioa (lehenengoa kontatu gabe) adierazten du. Edo zero zenbakia S sekuentzian berriz ez badago.

```
algoritmo Bilatu
hasiera
     Irakurri_Osokoa( Bilatzekoa)
     Emaitza := 0
     Irakurri_Osokoa (Zenb)
     Posizioa := 1
     bitartean  Zenb /= 0  egin
          baldin  Zenb = Bilatzekoa  orduan
              Emaitza:= Posizioa
          ambaldin
          Irakurri_Osokoa (Zenb)
          Posizioa:= Posizioa + 1
     ambitartean
     Idatzi_Osokoa (Emaitza)
amaia
```

Baina algoritmo horrekin sekuentziako elementu guztiak begiratzen dira, nahiz eta zenbakia lehenago aurkitu. Gainera, zenbakia behin baino gehiagotan badago, zer egiten du algoritmoak? Hona hemen beste bertsio eraginkorrago bi.

```
algoritmo Bilatu2
hasiera
     Irakurri_Osokoa( Bilatzekoa)
     Emaitza := 0
     Irakurri_Osokoa (Zenb)
     Posizioa := 1
     bitartean  Zenb /= 0  eta   Emaitza /= 0 egin
          baldin  Zenb = Bilatzekoa  orduan
              Emaitza:= Posizioa
          ambaldin
          Irakurri_Osokoa (Zenb)
          Posizioa:= Posizioa + 1
     ambitartean
     Idatzi_Osokoa (Emaitza)
amaia

algoritmo Bilatu3
hasiera
     Irakurri_Osokoa( Bilatzekoa)
     Irakurri_Osokoa (Zenb)
     Posizioa := 1
     bitartean  Zenb /= 0  eta   Zenb /= Bilatzekoa egin
          Irakurri_Osokoa (Zenb)
          Posizioa:= Posizioa + 1
     ambitartean
     baldin Zenb = Bilatzekoa orduan
          Emaitza:= Posizioa
     bestela
          Emaitza:= 0
     ambaldin
     Idatzi_Osokoa ( Posizioa)
amaia
```

VI.8. Elementu baten bilaketa sekuentzia ordenatuan

Aurreko ariketan bezala, baina kasu honetan txikienetik handienera ordenatuta dagoen sekuentzia batean.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia),.Sekuentzia ordenatuta dago txikienetik handienera. Sekuentziaren azken digitua zeroa da eta beste zerorik ez dago. Beti daude gutxienez zero zenbakia eta beste zenbaki bat.
Postbaldintza: Posizioa (osokoa). Posizioa-k S sekuentziako lehenengo zenbakiaren bigarren agerpenaren posizioa (lehenengoa kontatu gabe) adierazten du. Edo zero zenbakia S sekuentzian berriz ez badago.

Aldatu behar da sekuentzian aurrera elementuak pasatzeko baldintza da.

Lehen:

```
bitartean  Zenb /= 0  eta   Zenb /= Bilatua  egin
     hurrengora pasatu eta kontatu
```

Orain:

```
bitartean Zenb /= 0  eta Zenb < Bilatua  egin
     hurrengora pasatu eta kontatu
```

Zenbaki handiago bat aurkitzean badakigu jakin bilatua ez dugula aurkituko

```
algoritmo Bilatu_Ordenatuan
hasiera
     Irakurri_Osokoa( Bilatzekoa)
     Irakurri_Osokoa (Zenb)
     Posizioa := 1
     bitartean  Zenb /= 0  eta   Zenb < Bilatzekoa egin
           Irakurri_Osokoa (Zenb)
           Posizioa:= Posizioa + 1
     ambitartean
     baldin Zenb = Bilatzekoa orduan
           Emaitza:= Posizioa
     bestela
           Emaitza:= 0
     ambaldin
     Idatzi_Osokoa ( Posizioa)
amaia
```

VI.9. Elementu maximoaren bilaketa sekuentzian

Zero zenbakiaz amaitzen den osoko-sekuentzia batean zenbaki maximoa bilatzeko algoritmoa espezifikatu eta idatzi.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken digitua zeroa da eta beste zerorik ez dago. Beti daude gutxienez zeroa eta beste zenbaki bat.
Postbaldintza: Maximoa (osokoa). Maximoa-k S sekuentziako balio handiena adierazten du.
Azken zeroa ez da kontuan hartzen. Sekuentzian beste elementurik ez badago, Maximoa zero da.

```
algoritmo Bilatu_Maximoa
hasiera
     Irakurri_Osokoa (Zenb)
     Maximoa := Zenb
     bitartean  Zenb /= 0  egin
          baldin  Zenb > Maximoa  orduan
              Maximoa := Zenb
          ambaldin
          I:= I + 1
          Irakurri_Osokoa (Zenb)
     ambitartean
     Idatzi_Osokoa ( Maximoa )
amaia.
```

VI.10. Elementu maximoaren posizioa sekuentzian

Zero zenbakiaz amaitzen den osoko-sekuentzia batean zenbaki maximoaren posizioa bilatzeko algoritmoa espezifikatu eta idatzi.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken digitua zeroa da eta beste zerorik ez dago.
Postbaldintza: Posizioa (osokoa). Posizioa-k S sekuentziako balio handienaren posizioa adierazten du..
Azken zeroa ez da kontuan hartzen. Sekuentzian beste elementurik ez badago, Posizioaren balioa zero da.

```
algoritmo Bilatu_Maximoaren_Posizioa
hasiera
     Irakurri_Osokoa (Zenb)
     Maximoa := Zenb
     I := 1
     bitartean  Zenb /= 0  egin
          baldin  Zenb > Maximoa  orduan
              Maximoa := Zenb
              Posizioa := I
          ambaldin
          I:= I + 1
          Irakurri_Osokoa (Zenb)
     ambitartean
     Idatzi_Osokoa (Posizioa)
amaia
```

VI.11. Elementu minimoaren bilaketa sekuentzian

Zero zenbakiaz amaitzen den osoko-sekuentzia batean zenbaki minimoa bilatzeko algoritmoa espezifikatu eta idatzi.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken digitua zeroa da eta beste zerorik ez dago.
Postbaldintza: Minimoa (osokoa). Minimoa-k S sekuentziako balio txikiena adierazten du. Azken zeroa ez da kontuan hartzen. Sekuentzian beste elementurik ez badago, Minimoa zero da.

```
algoritmo Bilatu_Minimoa
hasiera
     Irakurri_Osokoa (Zenb)
     Minimoa := Zenb
     bitartean  Zenb /= 0  egin
          baldin  Zenb < Minimoa  orduan
              Minimoa := Zenb
          ambaldin
          I:= I + 1
          Irakurri_Osokoa (Zenb)
     ambitartean
     Idatzi_Osokoa (Minimoa)
amaia.
```

VI.12. Elementu minimoaren posizioa sekuentzian

Zero zenbakiaz amaitzen den osoko-sekuentzia batean zenbaki minimoaren posizioa bilatzeko algoritmoa espezifikatu eta idatzi.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken digitua zeroa da eta beste zerorik ez dago.
Postbaldintza: Posizioa (osokoa). Posizioa-k S sekuentziako balio txikienaren posizioa. Azken zeroa ez da kontuan hartzen. Sekuentzian beste elementurik ez badago, Posizioa-ren balioa zero da.

```
algoritmo Bilatu_Minimoaren_Posizioa
hasiera
     Irakurri_Osokoa (Zenb)
     Minimoa := Zenb
     I := 1
     bitartean  Zenb /= 0  egin
          baldin  Zenb < Minimoa orduan
              Minimoa := Zenb
              Posizioa := I
          ambaldin
          I:= I + 1
          Irakurri_Osokoa (Zenb)
     ambitartean
     Idatzi_Osokoa (Posizioa)
amaia
```

VI.13. "T" karaktere-bikotearen kontaketa sekuentzian

Puntu karaktereaz amaitzen den karaktere-sekuentzia batean, konta ezazu zenbat aldiz agertzen den 'A' karakterea 'T' karaktere baten atzetik.

Zehaztapena
Aurrebaldintza: S (karaktere-sekuentzia), Sekuentziaren azken karakterea puntua da eta beste punturik ez dago.
Postbaldintza: Kontagailua (osokoa). Kontagailua-k adierazten du zenbat aldiz agertzen den 'A' karakterea 'T' karaktere baten atzetik

```
algoritmo TA_Kontatu
hasiera
     Kontagailua := 0
     Irakurri_Karakterea (Kar)
     baldin   Kar /= '.' orduan
           Aurrekoa := Kar
           Irakurri_Karakterea (Kar)
           bitartean Kar /= '.' egin
                baldin Aurrekoa = 'T'  eta  Kar = 'A'  orduan
                     Kontagailua := Kontagailua + 1
               ambaldin
           Aurrekoa := Kar
           Irakurri_Karakterea (Kar)
           ambitartean
     ambaldin
     Idatzi_Osokoa (Kontagailua)
amaia
```

VI.14. Lehena ote den aztertu

N osoko positiboa emanda, zenbaki lehena denentz aztertuko duen algoritmoa espezifikatu eta idatzi.

Zehaztapena
Aurrebaldintza: N (osokoa), N > 0.
Postbalbintza: B (Boolearra). B=egiazkoa N zenbaki lehena bada, bestela B=faltsua

```
algoritmo Lehena_Da
hasiera
     Irakurri_Osokoa (N)
     Kontagailua := 0
     egin I guztietarako 1 tik N raino
           baldin  N mod I  = 0  orduan
              Kontagailua:= Kontagailua + 1
           ambaldin
     amguztietarako
     baldin  Kontagailua  > 2  orduan
           B := faltsua
     bestela
           B := egiazkoa
     ambaldin
     idatzi_Boolearra (B)
amaia.
```

Modu eraginkorrago batez, zenbakia bera eta 1 zenbakia zatitzaileak direnentz aztertu gabe:

```
algoritmo Lehena_Da
hasiera
     Irakurri_Osokoa (N)
     Kontagailua := 0
     egin I guztietarako 2 tik N - 1 raino
          baldin  N mod I  = 0  orduan
              Kontagailua:= Kontagailua + 1
          ambaldin
     amguztietarako
     baldin  Kontagailua  > 0  orduan
          B := faltsua
     bestela
          B := egiazkoa
     ambaldin
idatzi_Boolearra (B)
amaia
```

```
algoritmo Lehena_Da
hasiera
     Irakurri_Osokoa (N)
     B := egiazkoa
     I :=  2
     bitartean  B    and    I <= N - 1  egin
          baldin  N mod I  = 0  orduan
              B := faltsua
          ambaldin
     ambitartean
     idatzi_Boolearra (B)
amaia
```

Errepikatzen segitzeko baldintzan, *(I<=N-1)* adierazpenaren ordez *(I <= N/2)* baldintza erabil liteke N hiru zenbakia baino handiago bada, horrela eraginkortasuna handiagotuz.

Oharra: egiazta ezazu ondoko bi ekintza hauek baliokideak direla, bigarrena sinpleagoa delarik.

```
baldin  Kontagailua = 0
orduan
      B := egiazkoa
bestela
      B :=faltsua
ambaldin
```

```
B :=  Kontagailua = 0
```

VI.15. M baino txikiagoak diren zenbaki lehenak kalkulatu

Aurreko ariketako algoritmoa erabiliz, *M* osokoa baino txikiagoak diren zenbaki lehenak idatziko dituen algoritmoa espezifikatu eta idatzi.

Zehaztapena
Aurrebaldintza: M (osokoa), M>0.
Postbalbintza: S (osoko-sekuentzia). Elementuak zenbaki lehenak dira eta M baino txikiagoak

```
algoritmo Lehenak
hasiera
     Irakurri_Osokoa (M)
     egin N guztietarako 1 tik M-1 raino
          B := egiazkoa
          bitartean B eta (I <= (N / 2)) egin
              baldin  N mod I  = 0  orduan
                    B := faltsua
              ambaldin
          ambitartean
          baldin  B  orduan
              Idatzi_Osokoa (N)
          ambaldin
     amguztietarako
amaia
```

Baina aldez aurretik Lehena_Da algoritmoa definituta badugu, berau erabiliz errazago idatz dezakegu geure azken algoritmoa:

```
algoritmo Lehenak
hasiera
     Irakurri_Osokoa (M)
     egin N guztietarako 1 tik M-1 raino
          Lehena_Da (N,B)
          baldin  B  orduan
              Idatzi_Osokoa (N)
          ambaldin
     amguztietarako
amaia
```

Lehena_Da algoritmoa aldatu beharko genuke: A) Jaso behar duen datua ez da irakurriko sekuentzia batetik; aitzitik, erabili nahi izango denean adierazi beharko da, balioa zein den azalduz.. B) Itzuli behar duen emaitza ez da idatzi behar; *Lehena_Da* algoritmoa erabili ondoren nongo aldagaian utzi behar den zehaztu beharko da.

```
algoritmo Lehena_Da ( N: datu Osokoa;
                      B: emaitza Boolearra)
hasiera
     Kontagailua := 0
     egin  I guztietarako 2 tik N-1 raino
           baldin  N mod I  = 0  orduan
               Kontagailua:= Kontagailua + 1
           ambaldin
     amguztietarako
     baldin  Kontagailua >0  orduan
           B := faltsua
     bestela
           B := egiazkoa
     ambaldin
amaia
```

VI.16. Erro karratuaren kalkulua, Newtonen metodoaren arabera

Idatz ezazu *X* zenbaki erreal positiboaren erro karratua kalkulatu eta idatziko duen algoritmoa. Erabil ezazu Newtonen metodoa: *E* balioa *X*-ren erro karraturako estimatutako balioa bada, orduan *E* eta *X/E*-ren batezbesteko aritmetikoa estimazio hobea izango da. Eman hasieran *E*-ri balio positibo bat, ondoren aurreko formula erabiliz *E* estimazio hobe batez ordezkatu behin eta berriz, *E*-ren balioa *X*-ren baliora nahikoa hurbiltzen denean geldituz.

Zehaztapena
Aurrebaldintza: X (zenbaki erreala), X > 0
Postbalbintza: Y (zenbaki erreala), abs $((Y^2-X) / X) <$ 0. 0000001

```
algoritmo Erro_Karratua
hasiera
     Irakurri_Erreala (X)
     Erroa := X /2.0
     bitartean abs (X - Erroa**2) >  X * 0.0000001   egin
           Erroa := (Erroa + X / Erroa) / 2.0
     ambitartean
     Idatzi_Erreala (Erroa)
amaia
```

Watt-en liburuko 282. orrialdekoa:

```
function Sqrt (X : Float) return Float is
-- Newton-en metodoaren bidezko
-- gutxi gorabeherako erro karratua

Erroa : Float :=  X / 2.0;
begin
     while abs (X - Erroa**2) > 2.0 * X * Float'Epsilon loop
           Erroa := (Erroa + X / Erroa) / 2.0;
     end loop;
     return Erroa;
end Sqrt;
```

VI.17. Exekutatu algoritmoa

```
hasiera
     K := 3
     bitartean K >= 1 egin
          J := K
          bitartean J >= 1 egin
              Idatzi_Osokoa (J)
              J := J - 1
          ambitartean
          Idatzi_Osokoa (K)
          K := K-1
     ambitartean
amaia
```

Ondoko zerrenda idazten du: 3 2 1 3 2 1 2 1 1

VI.18. Exekutatu algoritmoa

Zein da ondoko algoritmoak emango duen emaitza, ondoko osoko-sekuentzia emanez gero: <5 6 -3 7 -4 0 5 8 9>

```
hasiera
     Batura := 0
     K:= 1
     B := False
     bitartean K <= 8 and B= False egin
          Irakurri _Osokoa (N)
          baldin N > 0 orduan
              Batura := Batura + N
          bestela
              baldin N=0 orduan
                   B := True
              ambaldin
          ambaldin
          K := K+1
     ambitartean
     Idatzi _Osokoa (Batura)
amaia
```

Emaitza: 18 (Hau da: 5 + 6 + 7)

Zeroa baino lehenago dauden lehenengo 8 zenbaki positiboen batura (edo gutxiagorena, besterik ez badago).

VII. Azpiprogramen definizioa eta erabilera I

Helburuak: Azpiprograma deritzon kontzeptua lantzea, problemen ebazpena errazteko baliagarria dela erakutsiz. Enuntziatu-multzo honetan batetik azpiprogramen erabilera lantzen da, eta bestetik, azpiprogramen definizioa.

ENUNTZIATUAK

Zenbaki perfektua

N osokoa emanda, perfektua den ala ez esango duen algoritmoa espezifikatu eta egin. Oharra: *N* osokoari perfektua esaten zaio, bere zatitzaileen (*N* ezik) batura *N* bera denean.

Sekuentziako zenbaki perfektuak

Zero zenbakiaz amaitzen den osoko-sekuentzia bat emanda, sekuentziako zenbaki perfektuak idatziko dituen algoritmoa espezifikatu eta egin. Egokitu aurreko ariketako algoritmoa, oraingo honetan azpiprograma modura erabili ahal izateko.

Sekuentziako zenbaki bikoitiak

Zero zenbakiaz amaitzen den osoko-sekuentzia bat emanda, sekuentziako bikoitiak diren zenbakiak idatziko dituen algoritmoa espezifikatu eta egin. Definitu eta erabili zenbaki bat bikoitia den ala ez aztertzen duen funtzio bat.

Sekuentziako posizio bakoitiko osagaiak idatzi.

Zero zenbakiaz amaitzen den sekuentzia bat emanda, posizio bakoitietan dauden elementuak idatziko dituen algoritmoa espezifikatu eta egin. Erabil ezazu ea zenbaki bat bikoitia den ala ez aztertzen duen funtzioa.

Zenbaki positibo baten atzekoz aurreko zenbakia

N osoko zenbaki positibo bat emanda, bere atzekoz aurreko zenbakia kalkulatzen duen algoritmoa espezifikatu eta egin. Adibidez, 485 zenbakiaren atzekoz aurreko zenbakia 584 da; hau da, digitu berberak baina atzekoz aurrera jarrita ditu.

Kapikua ote da

N osoko bat emanda, kapikua den ala ez kalkulatzen duen algoritmoa espezifikatu eta egin. Oharra: *N* zenbakia kapikua da, bera eta bere atzekoz aurreko zenbakia berdinak badira.

Sekuentziako kapikuen kopurua

Zeroz amaitzen den osoko-sekuentzia bat emanda, kapikuen kopurua kalkulatzen duen algoritmoa espezifikatu eta egin.

Sekuentziako monotonia gorakorren kopurua

Zeroz amaitzen den osoko-sekuentzia bat emanda, monotonia gorakorren kopurua kalkulatzen duen algoritmoa espezifikatu eta egin.

Sekuentziako hitz-kopurua kontatu

Puntu karakterez bukatzen den karaktere-sekuentzia bat emanda, hitz-kopurua kontatzen duen algoritmoa espezifikatu eta egin. Oharra: Demagun sekuentziako karaktere guztiak letrak, zuriuneak eta azkeneko puntua direla, eta ez dagoela beste puntuazio-markarik.

Sekuentziako karaktere guztiak atzekoz aurrera

Puntuz bukatzen den karaktere-sekuentzia bat emanda, karaktere guztiak baina atzetik aurrera idatziko dituen algoritmoa espezifikatu eta egin.

Zenbat aldiz sekuentziako lehenengo hitza?

Zuriune bat eta puntu batekin bukatzen den karaktere-sekuentzia bat emanda, lehenengo hitza zenbat aldiz azaltzen den kontatzen duen algoritmoa espezifikatu eta egin.

EBAZPENAK

VII.1 Zenbaki perfektua

N osokoa emanda, perfektua den ala ez esango duen algoritmoa espezifikatu eta egin. Oharra: *N* osokoari perfektua esaten zaio, bere zatitzaileen (*N* ezik) batura *N* bera denean.

Zehaztapena
Aurrebaldintza: N (osokoa), N >= 0.
Postbaldintza: B = Egiazkoa, baldin N perfektua bada eta
B = Faltsua, baldin N ez bada perfektua

```
algoritmo Perfektua_Da
hasiera
     Batura := 0
     egin  I guztietarako 1 tik N-1 raino
           baldin  N mod I = 0  orduan
              Batura:= Batura + I
           ambaldin
     amguztietarako
     baldin Batura = N orduan
           Idatzi_Boolearra (Egiazkoa)
     bestela
           Idatzi_Boolearra (Faltsua)
     ambaldin
amaia
```

Edo beste modu batera, adierazpide boolearraren emaitza zuzenean asignatuz:

```
algoritmo Perfektua_Da
hasiera
     Batura := 0
     egin  I guztietarako 1 tik N-1 raino
        baldin  N mod I = 0  orduan
           Batura:= Batura + I
        ambaldin
     amguztietarako
     B:= Batura = N
     Idatzi_Boolearra ( B )
amaia
```

VII.2. Sekuentziako zenbaki perfektuak

Zero zenbakiaz amaitzen den osoko-sekuentzia bat emanda, sekuentziako zenbaki perfektuak idatziko dituen algoritmoa espezifikatu eta egin. Egokitu aurreko ariketako algoritmoa, oraingo honetan azpiprograma modura erabili ahal izateko.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken karakterea zeroa da eta beste zerorik ez dago.
Postbaldintza: S_Emaitza (osoko_sekuentzia). S sekuentzian dauden zenbaki perfektu guztiak.

Perfektua_da funtzio modura definituz:

```
algoritmo Sekuentziako_Zenbaki_Perfektuak
hasiera
     Irakurri_Osokoa (Zenb)
     bitartean  Zenb /= 0  egin
          baldin Perfektua_Da (Zenb) orduan
              Idatzi_Osokoa (Zenb)
          ambaldin
          Irakurri_Osokoa (Zenb)
     ambitartean
amaia

funtzio Perfektua_Da ( N: datu Osokoa)
     itzuli Boolearra
Aurrebaldintza: N (osokoa), N >= 0.
Postbaldintza:  Egiazkoa, baldin N perfektua bada eta
                Faltsua, baldin N ez bada perfektua
hasiera
     Batura := 0
     egin  I  guztietarako 1 tik N-1 raino
          baldin  N mod I = 0  orduan
              Batura:= Batura + I
          ambaldin
     amguztietarako
     itzuli (N = Batura)
amaia
```

Perfektua_Da algoritmo modura definituz:

```
algoritmo Sekuentziako_Zenbaki_Perfektuak
hasiera
     Irakurri_Osokoa (Zenb)
     bitartean  Zenb /= 0  egin
        Perfektua_Da (Zenb, B)
        baldin B  orduan
           Idatzi_Osokoa (Zenb)
        ambaldin
        Irakurri_Osokoa (Zenb)
     ambitartean
amaia
```

```
algoritmo Perfektua_Da ( N: datu Osokoa; B: Emaitza Boolearra)
Aurrebaldintza: N (osokoa), N >= 0.
Postbaldintza:  B = Egiazkoa, baldin N perfektua bada eta
                B = Faltsua, baldin N ez bada perfektua
hasiera
     Batura := 0
     egin  I  guztietarako 1 tik N-1 raino
           baldin  N mod I = 0  orduan
               Batura:= Batura + I
           ambaldin
     amguztietarako
     B:= N = Batura
amaia
```

VII.3. Sekuentziako zenbaki bikoitiak

Zero zenbakiaz amaitzen den osoko-sekuentzia bat emanda, sekuentziako bikoitiak diren zenbakiak idatziko dituen algoritmoa espezifikatu eta egin. Definitu eta erabili zenbaki bat bikoitia den ala ez aztertzen duen funtzio bat.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken elementua zeroa da eta beste zerorik ez dago.
Postbaldintza: S_Emaitza (osoko_sekuentzia). S sekuentzian posizio bakoitietan dauden zenbaki guztiak.

```
algoritmo Sekuentziako_Zenbaki_Bikoitiak
hasiera
     Irakurri_Osokoa (Zenb)
     bitartean  Zenb /= 0  egin
           baldin Bikoitia_Da (Zenb) orduan
               Idatzi_Osokoa (Zenb)
           ambaldin
           Irakurri_Osokoa (Zenb)
     ambitartean
amaia

funtzio Bikoitia _Da ( N: datu Osokoa)
     itzuli Boolearra
Aurrebaldintza: N (osokoa), N >= 0.
Postbaldintza:  Egiazkoa, baldin N bikoitia bada
                eta Faltsua bestela
hasiera
     itzuli ( N mod 2 = 0)
amaia
```

VII.4. Sekuentziako posizio bakoitiko osagaiak idatzi.

Zero zenbakiaz amaitzen den sekuentzia bat emanda, posizio bakoitietan dauden elementuak idatziko dituen algoritmoa espezifikatu eta egin. Erabil ezazu ea zenbaki bat bikoitia den ala ez aztertzen duen funtzioa.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken karakterea zeroa da eta beste zerorik ez dago.
Postbaldintza: S_Emaitza (osoko_sekuentzia). S sekuentziako dauden zenbaki bikoiti guztiak.

```
algoritmo Posizio_Bakoitikoak
hasiera
      Irakurri_Osokoa (Zenb)
      Posizioa := 1
      bitartean  Zenb /= 0  egin
         baldin ez Bikoitia_Da( Posizioa) orduan
            Idatzi_Osokoa (Zenb)
         ambaldin
         Irakurri_Osokoa (Zenb)
         Posizioa := Posizioa  + 1
      ambitartean
amaia
```

VII.5. Zenbaki positibo baten atzekoz aurreko zenbakia

N osoko zenbaki positibo bat emanda, bere atzekoz aurreko zenbakia kalkulatzen duen algoritmoa espezifikatu eta egin. Adibidez, 485 zenbakiaren atzekoz aurreko zenbakia 584 da; hau da, digitu berberak baina atzekoz aurrera jarrita ditu.

Zehaztapena
Aurrebaldintza: N (osokoa), N>0 .
Postbaldintza: I (osokoa), N-ren digituak ditu, baina alderantzizko ordenan.

```
algoritmo Atzekoz_Aurreko_Zenbakia
hasiera
      Irakurri_Osokoa (N)
      I := 0
      bitartean  N /= 0  egin
         Hondarra := N mod 10
         N :=N / 10
         I :=  I*10 + Hondarra
      ambitartean
      Idatzi_Osokoa (I)
amaia
```

VII.6. Kapikua ote da

N osoko bat emanda, kapikua den ala ez kalkulatzen duen algoritmoa espezifikatu eta egin. Oharra: *N* zenbakia kapikua da, bera eta bere atzekoz aurreko zenbakia berdinak badira.

Zehaztapena
Aurrebaldintza: N (osokoa), N>0
Postbaldintza: Kapikua (osokoa), Kapikua= Egiazkoa N zenbakia kapikua bada, hau da, N eta bere atzekoz aurreko zenbakia berdinak badira

```
algoritmoa Kapikua_Da
hasiera
     Kapikua:= N = Atzekoz_Aurreko_Zenbakia (N)
     Idatzi_Boolearra( Kapikua)
amaia

funtzio Atzekoz_Aurreko_Zenbakia (X: datu osokoa) itzuli osokoa
hasiera
     I := 0
     bitartean  X /= 0  egin
           Hondarra := X mod 10
           X :=X / 10
           I :=  I*10 + Hondarra
     ambitartean
     itzuli (I)
amaia
```

Edo beste modu batera:

```
algoritmo Kapikua_Da
hasiera
     Alderantzizkoa (N, Ald)
     Kapikua:= N = Ald
     Idatzi_Boolearra( Kapikua)
amaia

algoritmo Atzekoz_Aurreko_Zenbakia
           (X: datu osokoa; I: emaitza osokoa)
hasiera
     I := 0
     bitartean  X /= 0  egin
           Hondarra := X mod 10
           X :=X / 10
           I :=  I*10 + Hondarra
     ambitartean
amaia.
```

VII.7. Sekuentziako kapikuen kopurua

Zeroz amaitzen den osoko-sekuentzia bat emanda, kapikuen kopurua kalkulatzen duen algoritmoa espezifikatu eta egin.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken karakterea zeroa da eta beste zerorik ez dago.
Postbaldintza: Kontagailua (osokoa). S sekuentziako zenbaki kapikuen kopurua.

```
algoritmo Posizio_Bakoitikoak
hasiera
     Irakurri_Osokoa (Zenb)
     Kontagailua := 0
     bitartean  Zenb /= 0  egin
           baldin Kapikua_Da ( Zenb) orduan
               Kontagailua := Kontagailua + 1
           ambaldin
           Irakurri_Osokoa (Zenb)
     ambitartean
     Idatzi_Osokoa (Kontagailua)
amaia.

funtzio Kapikua_Da (N : datu osokoa) itzuli boolearra
hasiera
     Kapikua:=     N = Alderantzizkoa (N)
     itzuli  ( Kapikua)
amaia

funtzio Alderantzizkoa (X: datu osokoa) itzuli osokoa
hasiera
     I := 0
     bitartean  X /= 0  egin
           Hondarra := X mod 10
           X :=X / 10
           I :=  I*10 + Hondarra
     ambitartean
     itzuli (I)
amaia.
```

VII.8. Sekuentziako monotonia gorakorren kopurua

Zeroz amaitzen den osoko-sekuentzia bat emanda, monotonia gorakorren kopurua kalkulatzen duen algoritmoa espezifikatu eta egin.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia), Sekuentziaren azken karakterea zeroa da eta beste zerorik ez dago.
Postbaldintza: N (osokoa), S sekuentzian zenbat monotonia gorakor dagoen.

```
algoritmo Zenbat_Monotonia
hasieraIrakurri_Osokoa (Zenb)
      baldin Zenb = 0 orduan
         Kontagailua := 0
      bestela
         Kontagailua := 1
         Aurrekoa:= Zenb
         bitartean  Zenb /= 0  egin
            baldin Zenb < Aurrekoa  orduan
               Kontagailua := Kontagailua + 1
            ambaldin
            Irakurri_Osokoa (Zenb)
         ambitartean
      ambaldin
      Idatzi_Osokoa (Kontagailua)
amaia
```

VII.9. Sekuentziako hitz-kopurua kontatu

Puntu karakterez bukatzen den karaktere-sekuentzia bat emanda, hitz-kopurua kontatzen duen algoritmoa espezifikatu eta egin. Oharra: Demagun sekuentziako karaktere guztiak letrak, zuriuneak eta azkeneko puntua direla, eta ez dagoela beste puntuazio-markarik.

Zehaztapena
Aurrebaldintza: S (osoko-sekuentzia). Sekuentziaren azken karakterea zeroa da eta beste zerorik ez dago.
Postbaldintza: N (osokoa). S sekuentzian zenbat hitz dagoen adierazten du. Hitzak zuriunez bereizten dira

Hitz-bukaerak kontatu beharko dira, hau da, (letra, zuriunea) edo (letra, puntua) erako bikoteak.

```
algoritmo Zenbat_Hitz
hasiera
     Irakurri_Karakterea (Kar)
     Kontagailua := 0
     Aurrekoa:= Kar
     bitartean  Kar /= '.'  egin
          baldin Aurrekoa  /= ' '   eta   Kar =' '  orduan
             Kontagailua := Kontagailua + 1
          ambaldin
          Irakurri_Karakterea (Kar)
     ambitartean
     baldin Aurrekoa  /= ' '  orduan
          Kontagailua := Kontagailua + 1
     ambaldin
     Idatzi_Osokoa (Kontagailua)
amaia.
```

VII.10. Sekuentziako karaktere guztiak atzekoz aurrera

Puntuz bukatzen den karaktere-sekuentzia bat emanda, karaktere guztiak baina atzetik aurrera idatziko dituen algoritmoa espezifikatu eta egin.

Ariketa hau ezin da egin orain arte ikusi ditugun tresnekin.

VII.11. Zenbat aldiz sekuentziako lehenengo hitza?

Zuriune bat eta puntu batekin bukatzen den karaktere-sekuentzia bat emanda, lehenengo hitza zenbat aldiz azaltzen den kontatzen duen algoritmoa espezifikatu eta egin.

Ariketa hau ezin da egin orain arte ikusi ditugun tresnekin.

VIII. Azpiprogramen zehaztapenak II

Helburuak: Zailtasun ertaineko azpiprogramen zehaztapenak definitzea. Prozedura eta funtzioen arteko diferentziak bereiztea. Ariketa guztietan aztertuko da ea azpiprograma funtzio edo prozedura izan daitekeen.

ENUNTZIATUAK

N zenbakia [N1,N2] tartean al dago?

N, *N1* eta *N2* osokoak emanda, *N* zenbakia *[N1,N2]* tartean dagoen ala ez esango duen algoritmoaren zehaztapena idatzi. Funtzio edota prozedura modura defini daiteke?

Urte bisustukoa ote da?

Data baten hila eta eguna adierazten duten bi zenbaki emanda, urte bisustu bateko egun posible bat ote den esango duen algoritmoaren espezifikazioa egin.

Zenbaki bitarra hamartar bihurtu

Zenbaki bitar bat emanda, bere balioa adierazpide hamartarrean emango duen algoritmoaren espezifikazioa egin.

Hiru zenbakiren arteko maximoa

Hiru zenbaki oso emanda, hiru zenbaki horien maximoa emango duen algoritmoaren espezifikazioa egin.

Hiru zenbaki ordenatu

N1, *N2* eta *N3* zenbaki osoak emanda, hiru zenbaki horiek ordenatuko dituen algoritmoa espezifikatu. Txikiena *N1*ean eta handiena *N3*an itzuli beharko dira.

Eman hurrengo segundoa

Ordu bat <orduak, minutuak, segundoak> formatuan emanda, hurrengo segundoari dagokion ordua formatu berean ematen duen algoritmoa espezifikatu.

N baino txikiagoa edo berdina den handiena bilatu

Handienetik txikienera ordenatuta dagoen eta zero zenbakiaz amaitzen den osoko-sekuentzia bat eta N osokoa emanda, N baino txikiago edo berdina den balio handiena emango duen algoritmoa espezifikatu. **Oharra:** Sekuentzian zenbakiak errepikatuta ager daitezke.

Esaldiko hitz guztiak letra berarekin hasten dira?

Esaldi bat adierazten duen eta puntu karaktereaz bukatzen den karaktere-sekuentzia bat emanda, esaldiko hitz guztiak letra berarekin hasten diren ala ez aztertuko duen algoritmoa espezifikatu.

Bi sekuentzia kateatu

Puntu karaktereaz amaitzen den *S1* eta *S2* bi karaktere-sekuentzia emanda, *S1*-ekoen elementu guztiak eta atzetik *S2*-koak izango dituen sekuentzia bakarra lortuko duen algoritmoa espezifikatu.

Positiboak eta negatiboak bereizi

Zero zenbakiaz amaitzen den osoko-sekuentzia bat emanda, bertako zenbaki positiboez osatutako sekuentzia bat eta zenbaki negatiboez osatutako bigarren sekuentzia bat emango dituen algoritmoa espezifikatu.

Bi sekuentzia ordenatutako osagaiak sekuentzia batean bildu

Handienetik txikienera ordenatuta dagoen eta zero zenbakiaz amaitzen diren osoko-sekuentzia bi emanda, sekuentzia bietako elementuak bilduta, hirugarren sekuentzia ordenatu bat sortuko duen algoritmoa espezifikatu. Oharra: Sekuentzian zenbakiak errepikatuta ager daitezke

EBAZPENAK

VIII.1. N zenbakia [N1,N2] tartean al dago?

N, *N1* eta *N2* osokoak emanda, *N* zenbakia *[N1,N2]* tartean dagoen ala ez esango duen algoritmoaren zehaztapena idatzi. Funtzio edota prozedura modura defini daiteke?

```
algoritmo Tartean_Dago (N, N1, N2 : datu Osokoa;
                        B_E : emaitza Boolearra)
Aurrebaldintza: N1<= N2
Postbaldintza:
     B = Egiazkoa, baldin N1 <= N <= N2  eta
     B = Faltsua,  baldin N < N1  edo  N > N2

funtzio Tartean _Dago (N1, N2 : datu Osokoa;
                       N : datu Osokoa)
           itzuli Boolearra
Aurrebaldintza: N1<= N2
Postbaldintza:
     Egiazkoa balioa itzultzen da, baldin N1 <= N <= N2;
     Faltsua balioa itzultzen da, baldin N < N1 edo N > N2
```

VIII.2. Urte bisustukoa ote da?

Data baten hila eta eguna adierazten duten bi zenbaki emanda, urte bisustu bateko egun posible bat ote den esango duen algoritmoaren espezifikazioa egin.

```
algoritmo Bisustuko_Eguna  (Hila, Eguna : datu Osokoa;
                            Bisustukoa : emaitza Boolearra)
Aurrebaldintza:
Postbaldintza:
 Bisustukoa = Egiazkoa, baldin Data_ondo_dago (Hila, Eguna)
 Bisustukoa = Faltsua   baldin ez Data_ondo_dago (Hila, Eguna)
 non:
  Data_ondo_Dago (Hila, Eguna) ==
       (1<= Eguna <= 29)  eta  (1<=Hila<=12)
       edo (Eguna =30    eta  ((Hila=1) edo (3<=Hila<=12)))
       edo (Eguna =31    eta  (Hila=1 edo Hila=3 edo Hila=5
                                 edo Hila=7 edo Hila=8
                                 edo Hila=10 edo Hila=12))

funtzioa Bisustuko_Eguna (Hila, Eguna : datu Osokoa)
     itzuli Boolearra
Aurrebaldintza:
Postbaldintza:
    Egiazkoa itzultzen du, baldin Data_ondo_dago(Hila, Eguna);
    Faltsua itzultzen du, baldin ez Data_ondo_dago(Hila,Eguna)
```

VIII.3. Zenbaki bitarra hamartar bihurtu

Zenbaki bitar bat emanda, bere balioa adierazpide hamartarrean emango duen algoritmoaren espezifikazioa egin.

```
algoritmo Hamartarrean_Jarri (N_Bitarra: datu Osokoa;
                              N_Hamartarra : emaitza Osokoa)
Aurrebaldintza:  N_Bitarra zenbakiko digitu guztiak 0 edo 1.
Postbaldintza:
```

$$N\_Hamartarra = \sum_{I=1}^{n} Digitu(I, N\_Bitarra) \times 2^{\,n-I}$$

```
   non Digitu(X,Y) == Y zenbakiaren Xgarren digitua
   eta  n  N_Bitarra zenbakiaren digitu kopurua

funtzio   Hamartarra_Bihurtu (Zenb_Bitarra: datu Osokoa)
   itzuli Osokoa
Aurrebaldintza:
Postbaldintza:   itzuli N_Hamartarra
```

$$N\_Hamartarra = \sum_{I=1}^{n} Digitu(I, N\_Bitarra) \times 2^{\,n-I}$$

```
   non Digitu(X,Y) == Y zenbakiaren Xgarren digitua den,
   eta  n  N_Bitarra zenbakiaren digitu kopurua den.
```

VIII.4. Hiru zenbakiren arteko maximoa

Hiru zenbaki oso emanda, hiru zenbaki horien maximoa emango duen algoritmoaren espezifikazioa egin.

```
algoritmo Maximoa3 (N1, N2,N3 : datu Osokoa;
                    Maximoa : emaitza Osokoa)
Aurrebaldintza: N1=n1, N2=n2, N3=n3
Postbaldintza:
     Maximoa = n1 baldin n2 <= n1 eta n3 <= n1, edo
     Maximoa = n2 baldin n1 <= n2 eta n3 <= n2, edo
     Maximoa = n3 baldin n1 <= n3 eta n2 <= n3

funtzio Maximoa3 (N1, N2, N3: datu Osokoa) itzuli Osokoa
Aurrebaldintza: N1=n1, N2=n2, N3=n3
Postbaldintza:
     n1 itzultzen du, baldin n2 <= n1 eta n3 <= n1, edo
     n2 itzultzen du, baldin n1 <= n2 eta n3 <= n2, edo
     n3 itzultzen du, baldin n1 <= n3 eta n2 <= n3
```

VIII.5. Hiru zenbaki ordenatu

N1, *N2* eta *N3* zenbaki osoak emanda, hiru zenbaki horiek ordenatuko dituen algoritmoa espezifikatu. Txikiena *N1*ean eta handiena *N3*an itzuli beharko dira.

Ezin da definitu funtzio modura, hiru emaitza itzuli behar ditu-eta.

```
algoritmo Ordenatu3 (N1, N2, N3: datu emaitza Osokoa)
Aurrebaldintza:  N1 = n1, N2 = n2, N3 = n3
Postbaldintza:
     N1=n1, N2=n2, N3=n3   baldin n1 <= n2 <= n3;
     N1=n1, N2=n3, N3=n2   baldin n1 <= n3 <= n2;
     N1=n1, N2=n3, N3=n2   baldin n1 <= n3 <= n2;
     N1=n1, N2=n3, N3=n2   baldin n1 <= n3 <= n2;
     N1=n1, N2=n3, N3=n2   baldin n1 <= n3 <= n2;
     N1=n1, N2=n3, N3=n2   baldin n1 <= n3 <= n2;
```

VIII.6. Eman hurrengo segundoa

Ordu bat <orduak, minutuak, segundoak> formatuan emanda, hurrengo segundoari dagokion ordua formatu berean ematen duen algoritmoa espezifikatu.

Ezin da definitu funtzio modura, hiru emaitza itzuli behar ditu-eta.

```
algoritmo Hurrengo_Segundoa(Ordua, Minutua,
                            Segundoa: datu emaitza Osokoa)
Aurrebaldintza:
   Ordua = o,        Minutuak = m,   Segundoak = s,
   0<= o <= 23,      0<= m <= 59,    0<= s <= 59
Postbaldintza:
   Ordua = o, Minutua = m, Segundoa = s + 1
      baldin Segundoa <= 58;

   Ordua = o, Minutua = m + 1, Segundoa = 0
      baldin  Minutua <= 58  eta  Segundoa = 59;

   Ordua = o + 1, Minutua = 0, Segundoa = 0
      baldin Ordua<=22 eta Minutua=59 eta Segundoa=59;

   Ordua = 0, Minutua = 0, Segundoa = 0
      baldin Ordua=23 eta Minutua=59 eta Segundoa=59
```

VIII.7. N baino txikiagoa edo berdina den handiena bilatu

Handienetik txikienera ordenatuta dagoen eta zero zenbakiaz amaitzen den osoko-sekuentzia bat eta N osokoa emanda, N baino txikiago edo berdina den balio handiena emango duen algoritmoa espezifikatu. **Oharra:** Sekuentzian zenbakiak errepikatuta ager daitezke.

```
algoritmo Ordenatu (S : datu Osoko-sekuentzia;
                    N : datu Osokoa;
                    N_baino_Minimoa : emaitza Osokoa)
Aurrebaldintza: S, Sekuentziaren azken osagaia zeroa da
  eta beste zerorik ez dago eta S ordenatuta dago.
Postbaldintza: M (osokoa), N baino txikiagoa den sekuentziako
  balio handiena.

funtzio Ordenatu  (S : datu Osoko-sekuentzia;
                   N : datu Osokoa) itzuli Osokoa
Aurrebaldintza: Sekuentziaren azken karakterea zeroa da
  eta beste zerorik ez dago eta S ordenatuta dago.
Postbaldintza:N baino txikiagoa den sekuentziako balio
  handiena itzultzen du.
```

VIII.8. Esaldiko hitz guztiak letra berarekin hasten dira?

Esaldi bat adierazten duen eta puntu karaktereaz bukatzen den karaktere-sekuentzia bat emanda, esaldiko hitz guztiak letra berarekin hasten diren ala ez aztertuko duen algoritmoa espezifikatu.

```
algoritmo Hasten_Berdin (S : datu Karaktere-sekuentzia;
                         Berdin : emaitza Boolearra)
Aurrebaldintza: S sekuentziaren azken karakterea puntua da eta
   beste punturik ez dago. S sekuentzia hitzez osatuta dago.
Postbaldintza:
     Berdin = Egiazkoa, baldin S sekuentziako hitz guztiak
      letra berarekin hasten badira,
     Berdin = Faltsua,  baldin S sekuentziako hitz guztiak
      letra berarekin hasten ez badira.
```

VIII.9. Bi sekuentzia kateatu

Puntu karaktereaz amaitzen den *S1* eta *S2* bi karaktere-sekuentzia emanda, *S1*-ekoen elementu guztiak eta atzetik *S2*-koak izango dituen sekuentzia bakarra lortuko duen algoritmoa espezifikatu.

```
algoritmo Kateatu (S1, S2 : datu Karaktere-Sekuentzia;
                   S : emaitza Karaktere-Sekuentzia)
Aurrebaldintza: S1 eta S2 sekuentzien azken karakterea puntua
     da eta beste punturik ez dute.
Postbaldintza: S = kateatu (S1, S2) non
     Kateatu (S, S´) == S eta S´ sekuentzien kateaketa
```

VIII.10. Positiboak eta negatiboak bereizi

Zero zenbakiaz amaitzen den osoko-sekuentzia bat emanda, bertako zenbaki positiboez osatutako sekuentzia bat eta zenbaki negatiboez osatutako bigarren sekuentzia bat emango dituen algoritmoa espezifikatu.

```
algoritmo Positiboak_Negatiboak
             (S: datu Osoko-sekuentzia;
              S1, S2: emaitza Osoko-sekuentzia)
Aurrebaldintza:S Sekuentzien azken osagaia zeroa da eta beste
     zerorik ez du.
Postbaldintza:
     S1 = (S sekuentziako osagai positibo guztiak);
     S2 = (S sekuentziako osagai negatibo guztiak).
```

VIII.11. Bi sekuentzia ordenatutako osagaiak sekuentzia batean bildu

Handienetik txikienera ordenatuta dagoen eta zero zenbakiaz amaitzen diren osoko-sekuentzia bi emanda, sekuentzia bietako elementuak bilduta, hirugarren sekuentzia ordenatu bat sortuko duen algoritmoa espezifikatu. Oharra: Sekuentzian zenbakiak errepikatuta ager daitezke

```
algoritmo Bildu_Ordenatuta (S1, S2 : datu Osoko-sekuentzia;
                            S: emaitza Osoko-sekuentzia)
Aurrebaldintza:S1, S2 sekuentzien azken osagaia zeroa da eta
   beste zerorik ez dute; S1 eta S2 ordenatuta daude handienetik
   txikienera.
Postbaldintza: S sekuentziak S1 eta S2 sekuentzietako osagai
   guztiak ditu. S ordenatuta dago handienetik txikienera.
   Ez dago S1 sekuentzian S sekuentzian ez dagoen elementurik,
   Ez dago S2 sekuentzian S sekuentzian ez dagoen elementurik.
   S sekuentziako zenbaki bakoitzaren agerpen-kopurua hau da:
       zenbaki hori zenbatetan azaldu den S1-ean
       gehi zenbatetan S2an.
```

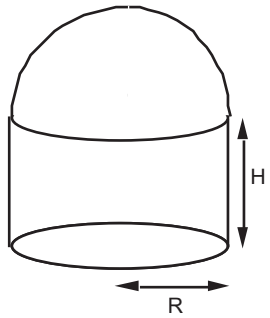
IX. Azpiprogramen definizioa eta erabilera II.

Helburuak: Azpiprogramak definitzea. Aurretik egindako ariketa batzuk berriro egitea baina orain azpiprograma modura planteatuta.

ENUNTZIATUAK

Bolumena kalkulatu

Ondoko irudiaren bolumena kalkulatu R erradioaren neurria eta H altuera (biak zenbaki errealak) emanda.



Zatitzaileak

Osoko zenbaki positibo bat emanda, bere zatitzaile guztiak idatzi

A eta B aldagaien arteko balio-trukaketa

A eta B aldagaien balioak elkarren artean trukatzeko algoritmoa espezifikatu eta idatzi.

Hiru zenbaki ordenatu

Hiru osoko zenbaki emanda, hirurak handienetik txikienera ordenatuta idatziko dituen algoritmoa espezifikatu eta idatzi.

Triangelua

Triangelu baten *A*, *B* eta *C* aldeen luzerak emanik, idatz ezazu triangeluaren azalera kalkulatuko duen algoritmoa. (*S* delakoa triangeluaren perimetroaren erdia bada, bere azalera honelaxe kalkulatzen da: $\sqrt{(S(S - A)(S - B)(S - C))}$.

Biderkadura hori negatiboa bada, ez dago *A*, *B* eta *C* aldeak dituen triangelurik). Suposa ezazu *X* zenbakiaren erro karratua *Erro_Karratua (X)* adierazpena ebaluatuz lortzen dela.

'A' karakterea zenbat aldiz?

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, 'A' letra zenbat aldiz agertzen den kontatu.

K1 karakterea zenbat aldiz?

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, *K1* letra zenbat aldiz agertzen den kontatu.

Bilatu zenbaki bat

Zero zenbakiaz amaitzen den osoko-sekuentzia ez-ordenatu batean zenbaki bat bilatzeko algoritmoa espezifikatu eta idatzi. Zenbakia sekuentziako gainontzeko guztien artean badago, sekuentziaren barruko posizioa idatzi beharko da; eta bestela, ez dagoenean, zero idatzi beharko da. Adibidez:

Bilatzeko zenbakia: 7

Sekuentzia	**9**	**8**	**-23**	**147**	**7**	**58**	**71**	**0**
Posizioak	**1**	**2**	**3**	**4**	**5**	**6**	**7**	

7 zenbakia sekuentziako 5. posizioan dagoenez, 5 izan beharko da emaitza.

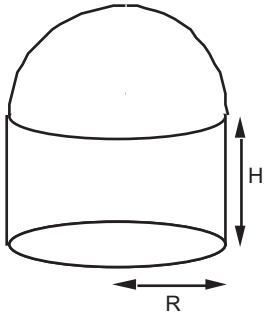
K1-K2 karaktere-bikotea zenbat aldiz segidan?

K1 eta *K2* karaktereak emanda puntu karaktereaz amaitzen den karaktere-sekuentzia batean *K2* karakterea *K1* karaktere baten atzetik zenbat aldiz agertzen den kontatu.

EBAZPENAK

IX.1. Bolumena kalkulatu

Ondoko irudiaren bolumena kalkulatu, R erradioaren neurria eta H altuera (biak zenbaki errealak) emanda.




```
algoritmo Bolumena (R, H: datu Erreala;
                    Bolumena: emaitza Erreala)
Aurrebaldintza: R, H >0
Postbaldintza: Bolumena = Pi * R2 * H + 4/6 Pi * R3
hasiera
     itzuli (1/2) * (4/3) * 3.1416 * R**3
          + H * 3.1416 * R**2
amaia.
```

IX.2. Zatitzaileak

Osoko zenbaki positibo bat emanda, bere zatitzaile guztiak idatzi.

```
algoritmo Zatitzaileak (N: datu Osokoa;
                        S: emaitza Osoko_Sekuentzia)
Aurrebaldintza: N >0
Postbaldintza: S = <S1, …, Sm> non
     i guztietarako Hondarra(N, Si) = 0
     eta Hondarra(N, j) = 0 betetzen duten j guztietarako
     existitzen da k, ondokoa betetzen duena Sk= j
hasiera
     egin  I guztietarako 1 tik N raino
           baldin  N mod I = 0  orduan
              Idatzi_Osokoa (S, I)
           ambaldin
     amguztietarako
amaia
```

IX.3. A eta B aldagaien arteko balio-trukaketa

A eta B aldagaien balioak elkarren artean trukatzeko algoritmoa espezifikatu eta idatzi.

```
algoritmo Trukatu_Balioak (A, B: datu emaitza Osokoa)
Aurrebaldintza:     A = a eta B = b
Postbaldintza: A = b eta B = a
hasiera
     C := A
     -- A-ren balioa ez galtzeko,
     -- beste aldagai batean utzi behar da
     A := B
     B := A
amaia
```

IX.4. Hiru zenbaki ordenatu

Hiru osoko zenbaki emanda, hirurak handienetik txikienera ordenatuta idatziko dituen algoritmoa espezifikatu eta idatzi.

```
algoritmo Ordenatu3 (N1, N2, N3: datu emaitza Osokoa)
     Aurrebaldintza: N1 = n1, N2 = n2, N3 = n3
     Postbaldintza:
          N1=n1, N2=n2, N3=n3   baldin n1 <= n2 <= n3
          N1=n1, N2=n3, N3=n2   baldin n1 <= n3 <= n2
          N1=n2, N2=n1, N3=n3   baldin n2 <= n1 <= n3
          N1=n2, N2=n3, N3=n1   baldin n2 <= n3 <= n1
          N1=n3, N2=n1, N3=n2   baldin n3 <= n1 <= n2
          N1=n3, N2=n2, N3=n1   baldin n3 <= n2 <= n1
hasiera
     baldin  N1 < N2 orduan
          Trukatu_Balioak (N1, N2)
     ambaldin
     baldin  N2 < N3 orduan
          Trukatu_Balioak (N2, N3)
     ambaldin
     baldin  N1 < N2 orduan
          Trukatu_Balioak (N1, N2)
     ambaldin
amaia
```

IX.5. Triangelua

Triangelu baten *A*, *B* eta *C* aldeen luzerak emanik, idatz ezazu triangeluaren azalera kalkulatuko duen algoritmoa. (*S* delakoa triangeluaren perimetroaren erdia bada, bere azalera honelaxe kalkulatzen da: . $\sqrt{(S(S - A)(S - B)(S - C))}$.

Biderkadura hori negatiboa bada, ez dago *A, B* eta *C* aldeak dituen triangelurik). Suposa ezazu X zenbakiaren erro karratua *Erro_Karratua (X)* adierazpena ebaluatuz lortzen dela.

```
algoritmo Triangeluaren_Azalera (A, B, C: datu Osokoa;
                                 Azalera: emaitza Erreala)
Aurrebaldintza: A, B, C >0
Postbaldintza: Azalera = A, B eta C aldeak dituen
     triangeluaren azalera:
        Azalera = √(S(S - A)(S - B)(S - C)) , non S = A + B + C den

hasiera
     S :=  (A + B + C) / 2.0
     Biderketa :=  S * (S – A) * (S – B) * (S – C)
     baldin Biderketa  <= 0 orduan
           Idatzi_Katea (" Ezin da osatu triangelurik")
     bestela  Azalera := Erro_Karratua (Biderketa)
     ambaldin
amaia

funtzio Triangeluaren_Azalera (A, B, C: datu Osokoa)
                        itzuli Erreala
Aurrebaldintza: A, B, C >0
Postbaldintza: √(S(S - A)(S - B)(S - C))  kalkuluaren balioa
     itzultzen du, non S = A + B + C den
hasiera
     S :=  (A + B + C) / 2.0
     Biderketa :=  S * (S – A) * (S – B) * (S – C)
     baldin Biderketa  <= 0 orduan
           Idatzi_Katea (" Ezin da osatu triangelurik")
     bestela
           itzuli Erro_Karratua (Biderketa)
     ambaldin
amaia
```

IX.6. 'A' karakterea zenbat aldiz?

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, 'A' letra zenbat aldiz agertzen den kontatu.

```
algoritmo Zenbat_A (S: datu Karaktere-sekuentzia;
                    Kontagailua: emaitza Osokoa)
Aurrebaldintza: S sekuentziaren azken karakterea puntua da eta
     beste punturik ez dago.
Postbaldintza: Zenbat = S sekuentziako 'A' letraren agerpen
     kopurua adierazten du
hasiera
     Irakurri_Karakterea (S, Kar)
     Kontagailua := 0
     bitartean Kar /= '.' egin
          baldin Kar = 'A'  orduan
              Kontagailua := Kontagailua + 1
          ambaldin
          Irakurri_Karakterea (S, Kar)
     ambitartean
amaia
```

IX.7. K1 karakterea zenbat aldiz?

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, *K1* letra zenbat aldiz agertzen den kontatu.

```
algoritmo Zenbat_K1 (S: datu Karaktere-sekuentzia;
                     K1: datu Karakterea;
                     Kontagailua: emaitza Osokoa)
Aurrebaldintza: S sekuentziaren azken karakterea puntua da eta
     beste punturik ez dago.
Postbaldintza: Zenbat = S sekuentziako K1 letraren agerpen
     kopurua adierazten du
hasiera
     Irakurri_Karakterea (S, Kar)
     Kontagailua := 0
     bitartean Kar /= '.' egin
          baldin Kar = K1  orduan
              Kontagailua := Kontagailua + 1
          ambaldin
          Irakurri_Karakterea (S, Kar)
     ambitartean
amaia
```

IX.8. Bilatu zenbaki bat

Zero zenbakiaz amaitzen den osoko-sekuentzia ez-ordenatu batean zenbaki bat bilatzeko algoritmoa espezifikatu eta idatzi. Zenbakia sekuentziako gainontzeko guztien artean badago, sekuentziaren barruko posizioa idatzi beharko da; eta bestela, ez dagoenean, zero idatzi beharko da. Adibidez:

Bilatzeko zenbakia: 7

Sekuentzia	**9**	**8**	**-23**	**147**	**7**	**58**	**71**	**0**
Posizioak	**1**	**2**	**3**	**4**	**5**	**6**	**7**	

7 zenbakia sekuentziako 5. posizioan dagoenez, 5 izan beharko da emaitza.

```
algoritmo Bilatu (S: datu Osoko-sekuentzia;
                  Bilatzekoa: datu Osokoa;
                  Posizioa: emaitza Osokoa)
Aurrebaldintza: S Sekuentzien azken osagaia zeroa da eta
     beste zerorik ez dago.
Postbaldintza:
     Posizioa = 0, ez badago Bilatzekoa S sekuentzian
     Posizioa = I, baldin badago Bilatzekoa S sekuentzian
                 eta I. posiziokoa bada.
hasiera
     Irakurri_Osokoa (S, Zenb)
     Posizioa := 1
     bitartean  Zenb /= 0  eta   Zenb /= Bilatzekoa egin
           Irakurri_Osokoa (S, Zenb)
           Posizioa:= Posizioa + 1
     ambitartean
     baldin Zenb /= Bilatzekoa orduan
           Posizioa:= 0
     ambaldin
amaia
```

IX.9. K1-K2 karaktere-bikotea zenbat aldiz segidan?

K1 eta *K2* karaktereak emanda puntu karaktereaz amaitzen den karaktere-sekuentzia batean *K2* karakterea *K1* karaktere baten atzetik zenbat aldiz agertzen den kontatu.

```
algoritmo Zenbat_K1_K2 (S: datu Karaktere-sekuentzia;
                        K1, K2: datu Karakterea;
                        Kontagailua: emaitza Osokoa)
Aurrebaldintza: Sekuentziaren azken karakterea puntua da eta
     beste punturik ez dago.
Postbaldintza: Kontagailua-k adierazten du zenbat aldiz
     agertzen den K2 karakterea K1 karaktere baten atzetik
hasiera
     Kontagailua := 0
     Irakurri_Karakterea (S, Kar)
     baldin   Kar /= '.' orduan
           Aurrekoa := Kar
           Irakurri_Karakterea (S, Kar)
           bitartean Kar /= '.' egin
               baldin Aurrekoa = K1  eta  Kar = K2  orduan
                     Kontagailua := Kontagailua + 1
               ambaldin
               Aurrekoa := Kar
               Irakurri_Karakterea (S, Kar)
           ambitartean
     ambaldin
amaia
```

X. *Bektorea* datu-egitura

Helburuak: *Bektorea* datu-egitura lantzea. Ada programazio--lengoaia erabiltzea oinarrizko algoritmoak inplementatzeko.

ENUNTZIATUAK

Bektoreko batezbesteko aritmetikoa

N elementu dituen osoko-bektore bat emanda, bektoreko zenbakien batezbesteko aritmetikoa kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Bektorea idatzi

N elementu dituen osoko-bektore bateko elementuak inprimatuko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Bektoreko elementu maximoa eta bere posizioa

N elementu dituen osoko-bektore batean zenbaki maximoa eta bere posizioa bilatzeko algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

'A' karakterea zenbat aldiz karaktere-bektorean

N elementu dituen karaktere-bektore bat emanda, 'A' letra zenbat aldiz agertzen den kontatu. Azpiprograma modura inplementatu.

Zenbat bokal karaktere-bektorean

N elementu dituen karaktere-bektore bat emanda, karaktere horien guztien artean zenbat bokal agertzen diren kontatu.

Zenbat ez-bokal karaktere-bektorean

N elementu dituen karaktere-bektore bat emanda, karaktere horien guztien artean bokalak ez direnak kontatu. Azpiprograma modura inplementatu. Karaktere bat bokala den ala ez aztertzen duen funtzio bat definitu eta erabili.

Zenbat aldiz errepikatzen den "TA" karaktere-bikotea

N elementu dituen karaktere-bektore batean 'A' karakterea 'T' karaktere baten atzetik zenbat aldiz agertzen den. kontatu Azpiprograma modura inplementatu.

Zenbat zenbaki lehen osoko-bektorean

N elementu dituen osoko-bektore bat emanda, zenbaki lehenen kopurua kalkulatuko duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Zenbakia bilatu osoko-bektore ez-ordenatuan

N elementu dituen osoko-bektore **ez-ordenatu** batean zenbaki bat bilatzeko algoritmoa espezifikatu eta egin. Zenbakia bektorean badago, lehenengo agerpenaren posizioa itzuli beharko da; eta bestela, ez dagoenean, zero itzuli beharko da. Azpiprograma modura inplementatu.

Zenbakia bilatu osoko-bektore ordenatuan

N elementu dituen osoko-bektore **ordenatu** batean (txikienetik handienera) zenbaki bat bilatzeko algoritmoa espezifikatu eta egin. Zenbakia bektorean badago, posizioa itzuli beharko da; eta bestela, ez dagoenean, zer posiziotan kokatu beharko genukeen eman. Zenbakia bektoreko guztiak baino handiagoa bada, N+1 itzuli beharko da. N zero bada, 1 itzuli beharko da. Azpiprograma modura inplementatu.

Posizio bat eskuinera mugitu

N elementuz osatutako osoko-bektore bat emanda, osagai guztiak posizio bat eskuinera mugitzen dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Sekuentzia irakurri eta bektorean ordenaturik sartu

Zero zenbakiaz bukatzen den osokoen sekuentzia bat emanda, zenbakiak osoko-bektore batean ordenatuta sartzen dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Sekuentzia irakurri eta N osagaiko bektorean ordenaturik sartu

Aurreko ariketan bezala, baina kasu honetan txikienetik handienera ordenatuta dagoen N osagai dituen osoko-bektore batean sartuz. Azpiprograma modura inplementatu.

Bektoreko ordenazioa, Burbuila deritzon algoritmoa jarraituz

N elementu dituen osoko-bektore bat emanda, bektoreko osagaiak *Burbuila* deritzon algoritmoa jarraituz ordenatuko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Bektoreko ordenazioa, Txertaketa deritzon algoritmoa jarraituz

N elementu dituen osoko-bektore bat emanda, bektoreko osagaiak Txertaketa deritzon algoritmoa jarraituz ordenatuko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Bektoreko zenbaki perfektuak eman

N elementu dituen osoko-bektore bat emanda, bektoreko zenbaki perfektuak beste bektore batean itzuliko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Sarrerako sekuentziako palindromoak idatzi

Zuriune eta puntu batez bukatzen den karaktere-sekuentzia batean dauden hitz palindromoak idazten dituen algoritmoa espezifikatu eta egin.

Lehenengo hitza zenbat aldiz?

Zuriune eta puntu batez bukatzen den karaktere-sekuentzia batean lehenengo hitza zenbat aldiz agertzen den kontatuko duen algoritmoa espezifikatu eta egin.

EBAZPENAK

X.1. Bektoreko batezbesteko aritmetikoa

N elementu dituen osoko-bektore bat emanda, bektoreko zenbakien batez besteko aritmetikoa kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
funtzio Batezbestekoa (B: datu Osokoen_Bektore;
                       N: datu Integer)
                  itzuli  Erreala
Aurrebaldintza: N>0
Postbaldintza: Itzuli da zenbaki erreal bat, B bektoreko N
     osagaien batezbesteko aritmetikoa adierazten duena
hasiera
     Batura := 0
     egin I guztietarako 1 tik N raino
               Batura := Batura + B(I)
     amguztietarako
     itzuli (Erreal_Bihurtu (Batura) / Erreal_Bihurtu (N))
amaia
```

X.2. Bektorea idatzi

N elementu dituen osoko-bektore bateko elementuak inprimatuko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
algoritmo Bektorea_Idatzi (B1: datu Osokoen_Bektorea;
                           Osagai_Kop : datu Integer) is
Aurrebaldintza: B1 bektorean Osagai_kop osagai daude
Postbaldintza: Sek (Osokoen Sekuentzia)
               non B1 bektorearen osagaiak dauden.
hasiera
     egin I guztietarako 1 tik Osagai_Kop raino
        Idatzi_Osokoa (B1(I))
     amguztietarako
amaia

procedure Bektorea_Idatzi (B1: in Osokoen_Bektorea;
                           Osagai_Kop : in Integer) is
-- Aurrebaldintza: B1 bektorean Osagai_kop osagai daude
-- Postbaldintza: Sek (Osokoen Sekuentzia
                  non B1 bektorearen osagaiak dauden.
begin
     for I in 1 .. Osagai_Kop loop
           Idatzi_Osokoa (B1(I)) ;
end loop;
end Bektorea_Idatzi;
```

X.3. Bektoreko elementu maximoa eta bere posizioa

N elementu dituen osoko-bektore batean zenbaki maximoa eta bere posizioa bilatzeko algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
algoritmoa Bilatu_Maximoa (B : datu Osokoen_Bektore ;
                           N: datu Osokoa ;
                           Maximoa, Pos : emaitza Osokoa)
Aurrebaldintza: N>0
Postbaldintza: Maximoa delakoak B bektoreko N osagaien arteko
   maximoa adierazten du.
   Pos delakoak B bektoreko N osagaien arteko maximoaren
   posizioa adierazten du.
hasiera
   I := 1
   Maximoa := B(I)
   bitartean  I <= N egin
      baldin B(I)> Maximoa  orduan
         Maximoa := B(I)
         Pos := I
      ambaldin
      I:= I + 1
   ambitartean
amaia
```

X.4. 'A' karakterea zenbat aldiz karaktere-bektorean

N elementu dituen karaktere-bektore bat emanda, 'A' letra zenbat aldiz agertzen den kontatu. Azpiprograma modura inplementatu.

```
funtzio A_Letrak_Kontatu (B: datu Karaktereen_Bektore;
                          N: datu Osokoa)
                      itzuli  Osokoa
Aurrebaldintza: N>0
Postbaldintza: Itzuli da zenbaki erreal bat, B bektoreko N
   osagaien artean 'A' letraren agerpen-kopurua adierazten
   duena
hasiera
   Kont := 0
   I := 1
   bitartean  I <= N egin
      baldin B(I)= 'A'orduan
         Kont := Kont + 1
      ambaldin
      I:= I + 1
   ambitartean
   itzuli  Kont
amaia
```

X.5. Zenbat bokal karaktere-bektorean

N elementu dituen karaktere-bektore bat emanda, karaktere horien guztien artean zenbat bokal agertzen diren kontatu.

```
funtzio Bokalak_Kontatu (B: datu Karaktereen_Bektore;
                         N: datu Osokoa)
                      itzuli  Osokoa
Aurrebaldintza: N>0
Postbaldintza: Itzuli da zenbaki erreal bat, B bektoreko N
   osagaien artean zenbat bokal dauden adierazten duena
hasiera
   Kont := 0
   I := 1
   bitartean  I <= N egin
      baldin B(I) = 'A'  edo B(I) = 'E'  edo B(I) = 'I' edo
             B(I) = 'O'  edo B(I) = 'U'
      orduan
         Kont := Kont + 1
      ambaldin
      I:= I + 1
   ambitartean
   itzuli  Kont
amaia
```

X.6. Zenbat ez-bokal karaktere-bektorean

N elementu dituen karaktere-bektore bat emanda, karaktere horien guztien artean bokalak ez direnak kontatu. Azpiprograma modura inplementatu. Karaktere bat bokala den ala ez aztertzen duen funtzio bat definitu eta erabili.

```
funtzio Bokala_izan (Kar : Karakterea)
                 itzuli  Boolearra
Aurrebaldintza:
Postbaldintza: egiazkoa itzuliko da, karakterea bokala bada;
               faltsua bestela
hasiera
 itzuli (B(I) = 'A'  edo B(I) = 'E'  edo B(I) = 'I' edo
         B(I) = 'O'  edo B(I) = 'U' )
amaia
```

```
funtzio Ez_Bokalak_Kontatu (B: datu Karaktereen_Bektore;
                            N: datu Osokoa) itzuli  Osokoa
Aurrebaldintza: N>0
Postbaldintza: Itzuliko da zenbaki erreal bat, B bektoreko N
   osagaien artean bokalak ez diren karaktereak zenbat diren
   adierazten duena
hasiera
   Kont := 0
   I := 1
   bitartean  I <= N egin
      baldin ez Bokala_Izan (B(I))
      orduan
         Kont := Kont + 1
      ambaldin
      I:= I + 1
   ambitartean
   itzuli  Kont
amaia
```

X.7. Zenbat aldiz errepikatzen den "TA" karaktere-bikotea

N elementu dituen karaktere-bektore batean kontatu zenbat aldiz agertzen den 'A' karakterea 'T' karaktere baten atzetik. Azpiprograma modura inplementatu.

```
funtzio TA_Kontatu (B: datu Karaktereen_Bektore;
                    N: datu Osokoa)
                  itzuli Osokoa
Aurrebaldintza: N>0
Postbaldintza: Itzuliko da zenbaki erreal bat, B bektoreko N
  osagaien artean zenbat aldiz agertzen den 'A' karakterea
  'T' karaktere baten atzetik adierazten duena
hasiera
   Kont := 0
   I := 2
   bitartean  I <= N  egin
      baldin  B(I-1) = 'T'  eta  B(I) = 'A'
      orduan
         Kont := Kont + 1
      ambaldin
      I:= I + 1
   ambitartean
   itzuli  Kont
amaia
```

X.8. Zenbat zenbaki lehen osoko-bektorean

N elementu dituen osoko-bektore bat emanda, zenbaki lehenen kopurua kalkulatuko duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
funtzio Zenbat_Lehenak (B: datu Osokoen_Bektore;
                        N: datu Osokoa) itzuli  Osokoa
Aurrebaldintza: N>0
Postbaldintza: Itzuliko da zenbaki erreal bat, B bektoreko N
     osagaien artean zenbat diren lehenak adierazten duena
hasiera
     Kont := 0
     I := 1
     bitartean  I <= N  egin
          baldin Lehena_Da (B(I))
          orduan
              Kont := Kont + 1
          ambaldin
          I:= I + 1
     ambitartean
     itzuli  Kont
amaia
```

X.9. Zenbakia bilatu osoko-bektore ez-ordenatuan

N elementu dituen osoko-bektore **ez-ordenatu** batean zenbaki bat bilatzeko algoritmoa espezifikatu eta egin. Zenbakia bektorean badago, lehenengo agerpenaren posizioa itzuli beharko da; eta bestela, ez dagoenean, zero itzuli beharko da. Azpiprograma modura inplementatu.

```
funtzio Bilatu_Posizioa (B : datu Osokoen_Bektore ;
                         N, X :datu Osokoa)
                    itzuli  Osokoa
Aurrebaldintza: N>0
Postbaldintza: Itzuliko da osoko bat, X zenbakia B bektoreko
     zenbatgarren posizioan dagoen adierazten duena;
     X zenbakia B bektoreko N osagaien artean ez badago,
          0 itzuliko da
hasiera
     I := 1
     bitartean  I< N eta X /= B(I) egin
          I := I + 1
     ambitartean
     -- I=N edo X=B(I)
     baldin X=B(I) orduan
          itzuli  I
     bestela  itzuli 0
     ambaldin
amaia
```

kontuz ibili! zer gertatzen da honela definituz gero?

```
hasiera
     I := 1
     bitartean  I<=N eta X /= B(I) egin
          I := I + 1
     ambitartean
     baldin I<=N orduan
          itzuli  I
     bestela  itzuli 0
     ambaldin
amaia
```

Errorea gerta liteke zenbakia ez badago bektorean, B(N+1) ebaluatu nahi izango baita denean horrelakorik existitzen ez bada. Beste aukera bat:

```
funtzio Bilatu_Posizioa (B : datu Osokoen_Bektore ;
                         N, X :datu Osokoa) itzuli  Osokoa
Aurrebaldintza: N>0
Postbaldintza: Itzuliko da osoko bat, X zenbakia B bektoreko
     zenbatgarren posizioan dagoen adierazten duena;
     X zenbakia B bektoreko N osagaien artean ez badago,
     0 itzuliko da
hasiera
     Pos := 0
     I := 1
     bitartean  I<=N  eta  Pos/=0 egin
          baldin X=B(I) orduan
              Pos := I
          ambaldin
          I := I + 1
     ambitartean
     -- I=N+1 edo X=B(I)
     itzuli Pos
amaia
```

X.10. Zenbakia bilatu osoko-bektore ordenatuan

N elementu dituen osoko-bektore **ordenatu** batean (txikienetik handienera) zenbaki bat bilatzeko algoritmoa espezifikatu eta egin. Zenbakia bektorean badago, posizioa itzuli beharko da; eta bestela, ez dagoenean, zer posiziotan kokatu beharko genukeen eman. Zenbakia bektoreko guztiak baino handiagoa bada, N+1 itzuli beharko da. N zero bada, 1 itzuli beharko da. Azpiprograma modura inplementatu.

Aurreko ariketako soluzio bera erabil daiteke, baina kasu honetan elementua bektorean ez dagoela jakin daiteke, bera baino handiago den zenbaki bat aurkitzen dugunean. Beraz, begiztako baldintza aldatu beharko da, kasu horietan bilaketa askoz lehenago bukatzeko.

Lehen :

```
bitartean  I< N eta X /= B(I) egin
```

Orain :

```
bitartean  I< N eta X < B(I) egin
```

N=0 kasua bereiztu egin beharko da.

```
funtzio Bilatu_Posizioa_Ord (B: Osokoen_Bektore; N, X: Osokoa)
     itzuli  Osokoa
Aurrebaldintza: N>=0
     B bektoreko N elementuak ordenatuta daude
     txikienetik handienera
Postbaldintza: Itzuliko da osoko bat,
     B bektoreko zenbatgarren posiziotan kokatu beharko
     litzateke adierazten duena.
     Zenbakia bektoreko guztiak baino handiagoa bada,
     N+1 itzuli beharko da
hasiera
     baldin N=0 orduan
          itzuli 1
     bestela
          I := 1
          bitartean  I< N eta X >B(I) egin
              I := I + 1
          ambitartean -- I=N   edo   X<= B (I)
          baldin X<=B(I) orduan
              itzuli  I
          bestela  -- I=N eta X>B(I)
              itzuli N+1
          ambaldin
     ambaldin
amaia
```

X.11. Posizio bat eskuinera mugitu

N elementuz osatutako osoko-bektore bat emanda, osagai guztiak posizio bat eskuinera mugitzen dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
procedure Mugitu1 (B1: in out Osokoen_Bektorea;
                   I: in Integer ;
                   Osagai_Kop: in out Integer) is
-- Aurrebaldintza: B1 bektorean Osagai_Kop elementu daude
--   I<=Osagai_Kop
-- Postbaldintza: B1 bektorean Igarren posiziotik aurrera
-- dauden osagaiak posizio bat eskuinera mugitu dira.
-- Osagai_Kop eguneratu egin da
begin
     for N in reverse I..Osagai_Kop loop
          B1(N+1) := B1(N);
     end loop;
     Osagai_Kop := Osagai_Kop + 1;
end Mugitu1;
```

X.12. Sekuentzia irakurri eta bektorean ordenaturik sartu

Zero zenbakiaz bukatzen den osokoen sekuentzia bat emanda, zenbakiak osoko-bektore batean ordenatuta sartzen dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
procedure Sortu_Bektore_Ordenatua (Btaula: out Osokoen_Bektorea;
                                   Kop: out Integer) is
-- Aurrebaldintza: Sek (Osokoen_Sekuentzia), Sek zeroz bukatzen
--   da
-- Postbaldintza: Btaula bektoreak Sek sekuentziako zenbakiak
--   dauzka eta era ordenatuan kokatuta
--   Kop osokoak taulak zenbat zenbaki sartu diren adierazten du.
N, Pos: Integer;
begin
     Irakurri_Osokoa (N);
     BTaula(1):= N;
     Kop := 1;
     Irakurri_Osokoa (N);
     while N /= 0 loop
           Pos := Bilatu_Posizioa_Ord (BTaula, Kop, N);
           Mugitu1 (BTaula, Pos, Kop);
           BTaula (Pos) := N ;
           Irakurri_Osokoa (N);
     end loop;
end Sortu_Bektore_Ordenatua;
```

X.13. Sekuentzia irakurri eta N osagaiko bektorean ordenaturik sartu

Aurreko ariketan bezala, baina kasu honetan txikienetik handienera ordenatuta dagoen N osagai dituen osoko-bektore batean sartuz. Azpiprograma modura inplementatu.

```
procedure Osatu_Bektore_Ordenatua
            (Btaula: in out Osokoen_Bektorea;
             Kop: in out Integer) is
-- Aurrebaldintza: BTaula (Osokoen_Sekuentzia);
--   Btaula bektoreko osagaiak ordenatuta daude
-- Postbaldintza: Btaula bektoreak berak hasieran zeuzkan
--   osagaiak dauzka
--   Btaula bektoreko osagaiak ordenatuta daude
--   Kop osokoak taulak zenbat zenbaki dauzkan adierazten du.
N, Pos : Integer;
begin
     Irakurri_Osokoa (N);
     while N /= 0 loop
           Pos := Bilatu_Posizioa_Ord (BTaula, Kop, N);
           Mugitu1 (BTaula, Pos, Kop);
           BTaula (Pos) := N ;
           Irakurri_Osokoa (N);
           N := N - 1;
     end loop;
end Osatu_Bektore_Ordenatua;
```

X.14. Bektoreko ordenazioa, Burbuila deritzon algoritmoari jarraituz

N elementu dituen osoko-bektore bat emanda, bektoreko osagaiak *Burbuila* deritzon algoritmoa jarraituz ordenatuko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
procedure Ordenatu_Burbuila (B: in Osokoen_Bektore;
                             N: in Integer) is
-- Aurrebaldintza: B bektoreak gutxienez N osagai ditu
-- Postbaldintza:  Bko lehenengo N elementuak ordenatuta
-- txikienetik handienera
begin
   for I in reverse 2 .. N loop
      Korritu_Elementuak_Handienak_Jasotzen (B, 1, I);
      -- B(I)..B(N)  bektoreko osagaiak ordenatuta daude
      -- eta B bektoreko handienak dira
   end loop ;
end Ordenatu_Burbuila ;

procedure Korritu_Elementuak_Handienak_Jasotzen
                        (B: in out Osokoen_Bektore;
                         I_Hasiera, I_Bukaera: in Integer) is
begin
   for I in I_Hasiera .. I_Bukaera - 1  loop
      if B(I) > B(I+1) then Balioak_Trukatu (B(I), B(I+1))
      end if ;
   end loop;
end Korritu_Elementuak_Handienak_Jasotzen ;
```

X.15. Bektoreko ordenazioa, Txertaketa algoritmoari jarraituz

N elementu dituen osoko-bektore bat emanda, bektoreko osagaiak Txertaketa deritzon algoritmoa jarraituz ordenatuko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
procedure Ordenatu (B: in Osokoen_Bektore; N: in Integer) is
-- Aurrebaldintza: B bektoreak gutxienez N osagai ditu
-- Postbaldintza:  B bektoreko lehenengo N elementuak
--    ordenatuta daude txikienetik handienera
begin
   for I in  1 .. N loop
      Bilatu_Minimoaren_Posizioa (B, I, N, Pos);
      Balioak_Trukatu (B(I), B(Pos)) ;
      -- B(1) ... B(I)  bektoreko osagaiak ordenatuta daude
      -- eta B bektoreko  txikienak dira
   end loop ;
end Ordenatu ;

procedure Bilatu_Minimoaren_Posizioa
                (B: in out Osokoen_Bektore;
                 I1, I2: in Integer;
                 Minimoaren_Posizioa: out Integer) is
-- Aurrebaldintza:
--   I1 eta I2 B bektorearen indize posibleak dira.
-- Postbaldintza:  B(I1) ... B(I2) balioen arteko txikiena
--   B(Minimoaren_Posizioa) da.
begin
   Minimoa :=Integer'last ;
   for I in I1 .. I2 - 1  loop
      if B(I) < Minimoa  then
              Minimoa := B(I);
              Minimoaren_Posizioa := I;
      end if ;
   end loop;
end Bilatu_Minimoaren_Posizioa ;
```

X.16. Bektoreko zenbaki perfektuak eman

N elementu dituen osoko-bektore bat emanda, bektoreko zenbaki perfektuak beste bektore batean itzuliko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
procedure Perfektuak (B1: in Osokoen_Bektorea;
                      Osagai_Kop1: in Integer ;
                      B2: in Osokoen_Bektorea;
                      Osagai_Kop2: in Integer) is
-- Aurrebaldintza: B1 (Osokoen_Bektorea)
--  Osagai_kop1 (osokoa), B1 bektoreak dauzkan osagai-kopurua
-- Postbaldintza: B2 (Osokoen_Bektorea)
--  B2 osagaiak B1 bektoreko zenbaki perfektuak dira
--  Osagai_kop2 (osokoa), B2 bektoreak dauzkan osagai-kopurua
begin
     Osagai_Kop2 := 0;
     for I in 1.. Osagai_Kop1 loop
           if Perfektua_Da (B1 (I)) then
               Osagai_Kop2 := Osagai_Kop2  + 1;
               B2 (Osagai_Kop2) := B1 (I);
           end if ;
     end loop;
end Taula_Idatzi;
```

X.17. Sarrerako sekuentziako palindromoak idatzi

Zuriune eta puntu batez bukatzen den karaktere-sekuentzia batean dauden hitz palindromoak idazten dituen algoritmoa espezifikatu eta egin.

```
with Motak;
with Hitza_Irakurri;
with Idatzi_Osokoa;
with Palindromo;
with Azkeneko_Hitza;
with Hitza_idatzi;

procedure Hitz_Palindromoak is
   H : Motak.Hitz ;
begin
   Hitza_Irakurri (H);
   while not (Azkeneko_Hitza (H)) loop
      if Palindromo (H) then
         Hitza_Idatzi (H);
      end if;
      Hitza_Irakurri (H);
   end loop;
end Hitz_Palindromoak;
```

```
with Motak;
with Irakurri_Karakterea;
procedure Hitza_Irakurri (Hitza: out Motak.Hitz) is
   I: Integer ;
   K: Character;
begin
   Irakurri_Karakterea (K);
   while  K = ' ' loop
      Irakurri_Karakterea (K);
   end loop;
   I := 0;
   while K /= ' ' and K /= '.' loop
      I := I + 1;
      Hitza.Karaktereak (I) := K;
      Irakurri_Karakterea (K);
   end loop;
   if K = '.' then
      Hitza.Karaktereak (1) := '.';
      Hitza.Luzera := 1;
   else
      Hitza.Luzera := I;
   end if;
end Hitza_Irakurri;

with Motak;
function Palindromo (Hitza: in Motak.Hitz) return Boolean
is
   I, J: Integer;
   Desberdina: Boolean;
begin
   I := 1;
   J := Hitza.Luzera;
   Desberdina := False;
   while I <= J and Desberdina = False loop
      if Hitza.Karaktereak (I) /= Hitza.Karaktereak (J) then
         Desberdina := True;
      end if;
      I := I + 1;
      J := J - 1;
   end loop;
   return not Desberdina;
end Palindromo;

with Motak;
function Azkeneko_Hitza (H : in Motak.Hitz) return Boolean
is
   B: Boolean;
begin
   B := (H.Karaktereak(1) = '.');
   return B;
end Azkeneko_Hitza;
```

```
with Motak;
with Karaktereak_Idatzi;
with Ada.Text_Io ;
procedure Hitza_Idatzi (Hitza: in Motak.Hitz) is
begin
   for I in 1 .. Hitza.Luzera loop
      Karaktereak_Idatzi (Hitza.Karaktereak (I));
   end loop;
   Ada.Text_Io.New_Line ;
end Hitza_Idatzi;

package Motak is
type Osokoen_Bektorea is array (Integer range <>) of Integer;
type Karaktereen_Bektorea is
   array (Integer range <>) of Character;
type Hitz is
   record
      Karaktereak: Karaktereen_Bektorea (1 ..20) ;
      Luzera: Integer;
   end record;
end Motak;
```

X.18. Lehenengo hitza zenbat aldiz?

Zuriune eta puntu batez bukatzen den karaktere-sekuentzia batean lehenengo hitza zenbat aldiz agertzen den kontatuko duen algoritmoa espezifikatu eta egin.

```
with Motak;
with Hitza_Irakurri;
with Idatzi_Osokoa;
with Azkeneko_Hitza;
with Hitza_idatzi;
with Hitz_berdinak;
procedure Lehen_Hitza_Zenbat  is
   H, Lehena: Motak.Hitz ;
   Kop: Integer := 0;
begin
   Hitza_Irakurri (H);
   Lehena := H ;
   while not (Azkeneko_Hitza (H)) loop
      if Hitz_Berdinak (H, Lehena) then
         Kop := Kop + 1;
      end if;
      Hitza_Irakurri (H);
   end loop;
   Idatzi_Osokoa(Kop);
end Lehen_Hitza_Zenbat ;

with Motak;
function Hitz_Berdinak (H1, H2 : in Motak.Hitz) return Boolean
is
   Berdinak : Boolean := true;
   I : Integer;
begin
   if H1.Luzera /= H2.Luzera then
      Berdinak := False ;
   else
      I := 1;
      while H1.Karaktereak(I) = H2.karaktereak(I)
            and I<H1.Luzera
      loop
         I:= I + 1;
      end loop;
      Berdinak := H1.Karaktereak(I) = H2.karaktereak(I) ;
   end if ;
   return (Berdinak);
end Hitz_Berdinak ;
```

Ondokoak aurreko ariketan bezala definitzen dira:

```
procedure Hitza_Irakurri (Hitza : out Motak.Hitz)
function Azkeneko_Hitza (H : in Motak.Hitz) return Boolean
procedure Hitza_Idatzi (Hitza : in Motak.Hitz)
package Motak
```

XI. *Matrizea* datu-egitura

Helburuak: *Matrizea* datu-egitura lantzea

ENUNTZIATUAK

Bi matrizeren batura

M×N elementu dituen osoko-matrize bi emanda, beren batura kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Zenbaki baten eta matrize baten arteko biderkadura

M×N elementu dituen osoko-matrize bat, eta K zenbaki osoa emanda, beren arteko biderkadura kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Bi matrizeren biderkadura

M×N elementu dituen osokoen matrize bat, eta N×P elementu dituen osoko-matrize bat emanda, beren arteko biderkadura kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Matrizeak idatzi

M×N elementu dituen osoko-matrize bateko elementuak inprimatuko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

Matrizea irakurri

M×N elementu dituen osokoen matrize bateko elementuak irakurriko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu. Elementuak sekuentzia batetik irakurriko dira. Sekuentziako osagaiak lerroz lerro antolatuta daude eta lerro bakoitza matrizearen lerro bati dagokio. Sekuentzia zero zenbakiaz bukatzen da.

EBAZPENAK

XI.1. Bi matrizeren batura

M×N elementu dituen osoko-matrize bi emanda, beren batura kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
with Motak;
function Matrizeen_Batura (M, N : in Integer;
                           M1, M2: in Motak.Osoko_Matrize)
                    return Motak.Osoko_Matrize is
-- Aurrebaldintza: M1 eta M2 MxN dimentsioko matrizeak dira
-- Postbaldintza: emaitza M1 eta M2ren batura;
Emaitza : Motak.Osoko_Matrize (1 .. M,1 .. N);
begin
   for I in 1 .. M loop
      for J in 1 .. N loop
         Emaitza (I, J) := M1 (I, J) + M2 (I, J);
      end loop;
   end loop;
  return Emaitza;
end Matrizeen_Batura;
```

XI.2. Zenbaki baten eta matrize baten arteko biderkadura

M×N elementu dituen osoko-matrize bat, eta K zenbaki osoa emanda, beren arteko biderkadura kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
with Motak;
function Matrize_Osoko_Biderkadura(M, N, K : in Integer;
                                   M1: in Motak.Osoko_Matrize)
                           return Motak.Osoko_Matrize is
-- Aurrebaldintza: M1 MxN dimentsioko matrizea da
-- Postbaldintza: emaitza K zenbakiaren eta M1 matrizearen
--    arteko biderkadura da eta emaitza MxN dimentsiokoa
--    izango da;
Emaitza : Motak.Osoko_Matrize (1 .. M,1 .. N);
begin
     for I in 1 .. M loop
           for J in 1 .. N loop
              Emaitza (I, J) := M1 (I, J) * K;
           end loop;
     end loop;
     return Emaitza;
end Matrize_Osoko_Biderkadura;
```

XI.3. Bi matrizeren biderkadura

M×N elementu dituen osokoen matrize bat, eta N×P elementu dituen osoko-matrize bat emanda, beren arteko biderkadura kalkulatzen duen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
with Motak;
function Matrizeen_Biderkadura
              (M, N, P : in Integer;
               M1, M2  : in Motak.Osoko_Matrize)
      return Motak.Osoko_Matrize is
-- Aurrebaldintza: M1 MxP dimentsioko matrizea da eta
--   M2 PxN dimentsioko matrizea da
-- Postbaldintza: emaitza M1 eta M2ren biderkadura da
--   eta MxN dimentsiokoa;
Emaitza : Motak.Osoko_Matrize (1 .. M,1 .. N);
begin
   for I in 1 .. M loop
      for J in 1 .. N loop
         Emaitza (I, J) := 0;
         for K in 1 .. P loop
            Emaitza (I, J) :=
               Emaitza(I, J) +  M1(I, K) + M2(K, J);
         end loop;
      end loop;
   end loop;
   return Emaitza;
end Matrizeen_Biderkadura;
```

XI.4.Matrizeak idatzi

M×N elementu dituen osoko-matrize bateko elementuak inprimatuko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu.

```
with Motak;
with Idatzi_Osokoa;
with Hurrengo_Lerrora_Pasa;
procedure Matrizea_Idatzi (M, N : in Integer;
                           M1: in Motak.Osoko_Matrize) is
-- Aurrebaldintza: M1 MxN dimentsioko matrizea da
-- Postbaldintza: M1 matrizearen inpresioa;
begin
   for I in 1 .. M loop
      for J in 1 .. N loop
         Idatzi_Osokoa (M1 (I, J));
      end loop;
      Hurrengo_Lerrora_Pasa;
   end loop;
end Matrizea_Idatzi;
```

XI.5. Matrizea irakurri

M×N elementu dituen osokoen matrize bateko elementuak irakurriko dituen algoritmoa espezifikatu eta egin. Azpiprograma modura inplementatu. Elementuak sekuentzia batetik irakurriko dira. Sekuentziako osagaiak lerroz lerro antolatuta daude eta lerro bakoitza matrizearen lerro bati dagokio. Sekuentzia zero zenbakiaz bukatzen da.

```
with Motak;
with Irakurri_Osokoa;
with Salto_Egin;
procedure Matrizea_Irakurri (M, N : in Integer;
                             M1: out Motak.Osoko_Matrize) is
-- Aurrebaldintza: M eta N irakurri nahi den matrizearen
--    dimentsioa ematen dute.
-- Postbaldintza: M1 matrizea sarrerako datuekin eratuta;

begin
   for I in 1 .. M loop
      for J in 1 .. N loop
         Irakurri_Osokoa (M1(I, J));
      end loop;
      Salto_Egin;
   end loop;
end Matrizea_Irakurri;
```

XII. Ariketa aurreratuak

Helburuak: Datu-egitura mistoak erabiltzea eta problema luzetxoagoak ebaztea.

ENUNTZIATUAK

Asignazio zuzenak markatu

Ondoko datu-motak emanda:

• helbideei buruzko informazioa gordetzen duten erregistroak:

```
type Helbide is record
     Kalea : String (1 .. 80);
     Kode_Postala : Integer ;
     Herria : String (1 .. 80);
     Probintzia : String (1 .. 80);
     Nazioa : String (1 .. 15);
end record;
```

• datei buruzko informazioa gordetzen duten erregistroak:

```
type Data is record
     Eguna : Integer ;
     Hilabetea : Integer ;
     Urtea : Integer ;
end record;
```

• informazio pertsonalari buruzko informazioa gordetzen duten erregistroak:

```
type Datu_Pertsonalak is record
     Izena, Abizena1, Abizena2: String (1 .. 15);
     NAN : String (1 .. 8);
     Helbidea : Helbide;
end record;
```

• patologi kodeen taulak:

```
type Patologi_Kodeak is array (1 .. 12) of Integer;
```

• antibiotiko-kodeen taulak:

```
type Antibiotiko_Kodeak is array (1 .. 25) of Integer;
```

• tratamendu aplikatuei buruzko informazioa gordetzen duten erregistroak:

```
type Tratamendu_Datuak is record
     Pertsona : Datu_Pertsonalak;
     Sarrera_Data : Data;
     Irteera_Data : Data;
     Patologi_Kopurua : Integer ;
     Patologiak : Patologi_Kodeak;
     Antibiotiko_Kopurua : Integer ;
     Antibiotikoak : Antibiotiko_Kodeak;
     Arrakasta: Boolean ;
end record;
```

• gaixoen datuen taulak:

```
type Gaixoen_Tratamenduak  is array (1..100) of
     Tratamendu_Datuak;
```

Eta ondoko aldagaiak emanda:

```
V: Gaixoen_Tratamenduak;
P: Datu_Pertsonalak;
D: Data;
H: Helbide;
I,J: Integer ;
```

Esan ondoko asignazio bakoitzak konpilazio-errorea duen ala ez:

```
1) V(I) := P;  (BAI/EZ)
2) V(5).Patologiak (V(5).Patologi_Kopurua + 1) := 5; (BAI/EZ)
3) H.Kalea:= V(J).Pertsona.Helbidea.Kalea; (BAI/EZ)
4) D.Hilabetea:= V(100).Sarrera_Data; (BAI/EZ)
5) V(3).Pertsona.Helbidea.Nazioa (1) := 'P'; (BAI/EZ)
```

N txikienak lortu (1. bertsioa)

Ondoko datu-moten definizioak emanda:

```
Max: constant Integer := 100;
type Taula_Mota is array(1..Max) of Natural;
type Osokoen_Lista_Mota is record
     Osagai_Kop: Natural;    — 0 edo handiagoa
     Osagaiak: Taula_Mota;
end record;
```

Ondoko azpiprograma inplementatu:

```
procedure N_Txikienak_Lortu
     (L1: in Osokoen_Lista_Mota; N: in Natural;
     L2: out Osokoen_Lista_Mota)
-- Aurrebaldintza: N >= 0, L1.Osagai_Kop >= 0
--     N <= L1 listako osagai desberdinen kopurua
-- Postbaldintza: L2 listak L1-eko N elementu desberdin
--     txikienak ditu.
```

N txikienak lortu (2. bertsioa)

Sarrera estandarrean dugu luzera ez-ezaguneko osoko-sekuentzia bat (litekeena da oso luzea izatea eta, beraz, ez da posible array batean gordetzea). Sekuentziako N osoko txikienak lortuko dituen azpiprograma egin. Sekuentziaren luzera N baino askoz handiagoa da.

Permutazio izan

Permutazio_Izan funtzioaren inplementazioa egin:

```
subtype Letra is character range 'A'..'Z';
type Hitz is array (1..5) of Letra;
function Permutazio_Izan (X, Y: in Hitz) return Boolean;
-- Aurrebaldintza: Luzera bereko bi hitz (letra guztiak
--     maiuskulaz eta hitzen letra kopurua = 5)
-- Postbaldintza: true Y hitza X hitzaren permutazioa bada eta
--     false bestela (Y hitza X hitzaren permutazioa izango da,
--     baldin eta X-ren letrak tokiz aldatuz Y lortzen badugu.)
```

Adibidez:

```
Permutazio_Izan (X=> "ALUAN", Y => "LAUAN") = true
Permutazio_Izan (X=> "ETXEA", Y => "ATEXE") = true
Permutazio_Izan (X=> "ALAIA", Y => "ILAKI") = false
Permutazio_Izan (X=> "AMAMA", Y => "AMAIA") = false)
Permutazio_Izan (X=> "AMAMA", Y => "MAMAM") = false)
```

GOGOAN HARTZEKO:

Y hitza X hitzaren PERMUTAZIOA bada, zenbat aldiz agertzen da alfabetoaren letra bakoitza Y hitzean eta X hitzean?

Ordenatu bektorea kolorearen arabera

Ondoko erazagupenak emanda:

```
type Kolore is (Zuria, Beltza);
type Fitxa is record
                Kolorea: Kolore;
                Info: String (1..255);
             end record;
type Fitxen_Bektore is array (Positive range <>) of Fitxa;
```

Ondoko azpiprogramaren inplementazioa egin:

```
procedure Ordenatu (Fitxak: in out Fitxen_Bektore) is
-- Aurrebaldintza:
-- Postbaldintza: Fitxak parametroaren balioa sarrerako
--    balioaren permutazioa da, non kolore bereko fitxak segidan
--    dauden (hasieran kolore bateko fitxak daude eta segidan
--    bestekoak).
```

Oharra: Emaitza lortzeko ezin da beste taularik erabili.

Besteen bestekoa

Definitu azpiprograma hau:

```
procedure Besteen_Bestekoa_Dauka (B: in Osokoen_Bektore;
                                  Dauka: out Boolean;
                                  Pos: out Integer)
-- Aurrebaldintza:
-- Postbaldintza: B bektoreko osagai bat beste osagaien
--    batura bada Dauka = true
--    eta Pos horrelako osagai baten posizioa da,
--    bestela Dauka=false eta Pos-en balioa edozein da.
```

Eratostenesen bahea

Muga bat baino txikiagoak diren zenbaki lehenak kalkulatzeko araua. (Eratostenesen bahea, K.a. III. mendea)

Metodoak honela dio: finkatutako mugarainoko zenbaki arrunten segida idatzirik, 2 zenbaki lehenetik hasi eta 4tik aurrera daudenak binaka ezabatzen dira (4, 6, 8, …), mugaraino iritsi arte; segidan 3 zenbakiaren multiploak ezabatzen dira hirunaka, 9tik hasita —beti ere aldez aurretik ez badaude ezabaturik, jakina—; eta gauza bera 5enekin, 25etik hasita. Eta horrela jarraitzen da karratua muga baino handiagoa duen zenbaki lehenera iritsi arte. Ezabatu gabe gelditutakoak dira zenbaki lehenak.

Muga bat emanda, muga hori baino txikiago edo berdinak diren zenbaki lehenak irteera estandarrean idatziko dituen Ada azpiprograma egin. Azpiprogramaren emaitza azaldutako metodoaren arabera (Eratostenesen bahea) kalkulatu behar da.

Meseta luzeenaren luzera

Osoko-bektore baten meseta luzeenaren luzera itzuliko duen azpiprograma inplementatu. Adibidez, emanda ondoko bektorea: 3 3 5 5 5 2 8 7 7 7 7 3. Azpiprogramaren emaitza 4 izango da, 7 zenbakiaren meseta luzeena baita, eta meseta horretan lau osagai baitaude.

```
function Meseta_Luzeenaren_Luzera (B: in Osokoen_Bektore)
          return Integer
-- Aurrebaldintza:
-- Postbaldintza: B bektoreko meseta luzeenaren luzera itzuli
--    da. Meseta dagoela esaten da ondoz ondoko
--    osagaietan balio bera agertzen denean.
```

Multzoen ebakidura

Ondoko datu-moten definizioak emanda:

```
Max: constant Integer := 100;
type Taula_Mota is array(1..Max) of Natural;
type Multzo_Mota is record
     Osagai_Kop: Natural;    — 0 edo handiagoa
     Osagaiak: Taula_Mota;
end record;
```

Ondoko azpiprograma inplementatu:

```
function Ebakidura (M1, M2: in Multzo_Mota)
                return Multzo_Mota
-- Aurrebaldintza: M1.Osagai_Kop eta M2.Osagai_Kop >= 0
--    M1 eta M2-k osokoen multzoak adierazten dituzte
-- Postbaldintza: emaitza M1 eta M2-n dauden elementuen
-- multzoa
```

Multzoen Bildura

Ondoko datu-moten definizioak emanda:

```
Max: constant Integer := 100;
type Taula_Mota is array(1..Max) of Natural;
type Multzo_Mota is record
   Osagai_Kop: Natural;    -- 0 edo handiagoa
   Osagaiak: Taula_Mota;
end record;
```

Ondoko azpiprograma inplementatu:

```
function Bildura (M1, M2: in Multzo_Mota) return Multzo_Mota
-- Aurrebaldintza: M1.Osagai_Kop eta M2.Osagai_Kop >= 0
--    M1 eta M2-k osokoen multzoak adierazten dituzte
-- Postbaldintza: emaitza M1 edota M2-n dauden elementuen
--    multzoa da
```

Multzo guztien ebakidura

Ondoko datu-moten definizioak emanda:

```
Max: constant Integer := 100;
type Taula_Mota is array(1..Max) of Natural;
type Multzo_Mota    is record
     Osagai_Kop: Natural;    -- 0 edo handiagoa
     Osagaiak: Taula_Mota;
end record;
type Multzoen_Bektorea_Mota is array (Integer range  <>)
     of Multzo_Mota;
```

Ondoko azpiprograma inplementatu:

```
function Guztien_Ebakidura (B: Multzoen_Bektorea_Mota)
return Multzo_Mota
-- Postbaldintza: emaitza B bektoreko multzo guztien
--    ebakidura da
```

Zatikien definizioa

Zatikien definizioa: p/q zatikia da, p eta q osokoak badira eta q?0. Defini ezazu *Zatiki* izeneko datu-mota bat, zatikiak adierazteko balioko duena.

Zatidura-lista datu-mota definitu

M zatiki dituen L lista bat emanda, L/≡ zatidura-lista esango diogu L-ren osagaiekin osaturiko listak osagai dituen listari, honako baldintzak beteko dituelarik:

L/≡ lista osatzen duten listak binaka hartuta disjuntuak dira.

L/≡ lista osatzen duten listetako osagai guztien artean L listako osagai guztiak agertzen dira.

L/≡ listako lista bakoitzeko elementu guztiak bata bestearen baliokideak dira. (p/q≡r/s ⇔ p.s=q.r).

Adibidea: L = [1/3, 2/4, 5/3, 3/9, 6/18, 10/6]

L/≡ = [[1/3, 3/9, 6/18], [2/4], [5/3, 10/6]]

Defini itzazu, alde batetik, Zatiki_Lista izeneko datu-mota bat, zatiki-listak adierazteko balioko duena; eta bestetik, Zatidura_Lista datu-mota, zatikizko zatidura-listak adieraztekoa.

Zatidura-lista lortu

Idatz ezazu zatiki-lista baten zatidura-lista lortuko duen funtzioa:

```
function Lortu_Zatidura_Lista (ZL: Zatiki_Lista)
         return Zatidura_Lista;
```

Zatiki ez-negatiboen adierazle kanonikoa

p/q zatiki ez-negatibo baten adierazle kanonikoa r/s zatikia da, non r eta s ez-negatiboak diren eta duten zatitzaile komun bakarra 1 den (hau da, elkarrekiko lehenak dira), eta p/q≡r/s

Adibidea: 18/24 zatikiaren adierazle kanonikoa 3/4 da.

Idatz ezazu zatiki ez-negatibo baten adierazle kanonikoa lortuko duen funtzioa:

```
function Kanonikoa (Z: Zatiki) return Zatiki;
-- Aurrebaldintza:
--    Z-ren izendatzailea eta zenbakitzailea ez-
--    negatiboak dira.
```

Zatiki-listaren adierazle kanonikoen batura eta maiztasun handieneko kanonikoa.

L zatiki-lista bat emanda, izendatzailea eta zenbakitzailea ez-negatibo dituzten zatikiak dituena, idatz itzazu, batetik, L-ren osagai guztien adierazle kanonikoen batura itzuliko duen azpiprograma, eta, bestetik, L-ren osagai gehien biltzen dituen baliokidetza-klasearen adierazle kanonikoa itzuliko duena. Horretarako bi zatikien batura kalkulatzen duen ondoko funtzioa erabil daiteke:

```
function Batura (R1, R2: Zatiki) return Zatiki;
```

Adibidea: Izanik L = [1/3, 2/4, 5/3, 3/9, 6/18, 10/6], listako zatiki guztien batura: 29/6

Osagai gehien biltzen dituen baliokidetza-klasearen adierazle kanonikoa: 1/3

Ezkutatu laukia

Honako datu-motak erabili dira pantaila batean erakutsi nahi den irudi bat definitzeko

```
type Irudi_Puntu is record
     Horia, Urdina,  Gorria : Boolean;
     Intentsitatea : Integer ;
end record;

type Irudi is array (Integer range <>, Integer range <>)
                    of Irudi_puntu;
subtype Kolore is integer range 1..8;
```

Pantailako puntu bakoitzean hiru sentsore egongo dira, bakoitza oinarrizko kolore batekoa. Ikusleak puntu horretan ikusten duen kolorea piztuta dauden oinarrizko koloreen konbinazioa da.

Demagun azpiprograma hauek eginda daudela:

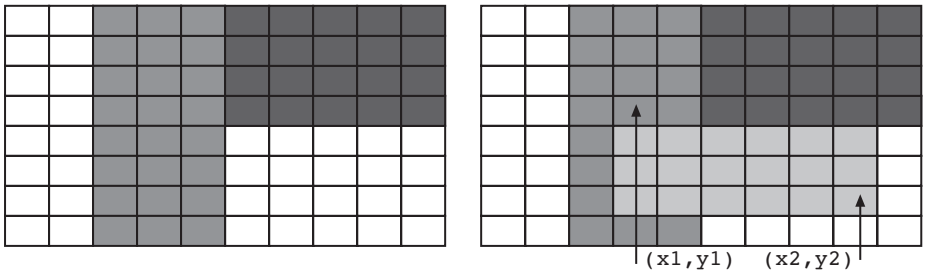
```
function Konbinazioa (Horia, Urdina, Gorria : Boolean)
    return Kolore

--Aurrebaldintza:
--Postbaldintza: true balio duten oinarrizko kolore
--    konbinatuz lortzen den kolorea
--    itzultzen du.
procedure Analizatu (Kolorea : in Kolore;
          Horia, Urdina, Gorria : out Boolean)
--Aurrebaldintza:
--Postbaldintza: Kolorea lortzeko behar diren oinarrizko
     koloreek true balio dute, besteek false
```

Honako prozedura diseinatu eta egin:

```
procedure Ezkutatu_Laukia (Irudia: in out Irudi;
                           X1, Y1, X2, Y2 : Integer);
--Aurrebaldintza:(X1,Y1) eta (X2,Y2) irudiko bi puntuen
--    koordenatuak dira
--    X1<X2   eta   Y1<Y2
--Postbaldintza: irudia hasierakoa bezalakoa da baina
--    barruko lauki bat lausotuta azaltzen da.
--    A) (X1,Y1) eta (X2,Y2) puntuak lauki
--    lausotuaren diagonalaren muturrak dira
--    B) Lauki lausotuko puntu guztien kolorea
--    berdina da: hasierako irudiko laukian
--    maizago azaltzen zen kolorea
--    (oinarrizko koloreen konbinazio bat).
--    V) Lauki lausotuaren puntu guztietako
--    intentsitateak berdinak dira:
--    hasierako irudiko laukiko puntuen
--    intentsitateen batezbestekoa.
```

Adibidea: Hasieran Bukaeran



Karrera bukaerako proiektuen asignazioa I

Karrera-bukaerako 50 proiektu definitu dira fakultate batean ikasleen artean banatuak izateko. Ikasle bakoitzak proiektu bat eskatu du. Proiektu bakoitzerako eskatzaile bat egon daiteke, edo bat baino gehiago, edo bat ere ez. Proiektuak asignatzeko laguntza informatikoa inplementatzeko ondoko erazagupenak definitu dira:

```
Proiektu_Kopurua: constant Integer:= 50;
Ikasle_Kopuru_Maximoa: constant Integer:= 100;

type Eskatzaile is record
   Espediente_Zbkia: string(1..8);
   Batezbestekoa: Float ;
end record;
subtype Ikasle_Kopuru is Integer range 0
..Ikasle_Kopuru_Maximoa;
type Eskatzaile_Bektore is array
(1..Ikasle_Kopuru_Maximoa) of
   Eskatzaile;
type Eskatzaile_Zerrenda is record
   Zerrenda: Eskatzaile_Bektore ;
   Eskatzaile_Kopurua: Ikasle_Kopuru := 0 ;
end record;
type Proiektu is record
   Kodea: Integer ;  — 1000tik 2000ra bitarteko
zenbakia
   Titulua: string (1..100);
   Zuzendaria: string (1..30);
   Eskatzaileak: Eskatzaile_Zerrenda ;
end record;
type Proiektu_Bektore is array(1..
Proiektu_Kopurua) of Proiektu;
type Ikasle is record
   Espediente_Zbkia: string (1..8);
   Nota: Float;
   Proiektu_Eskatua: Integer; — 1000tik 2000ra
bitarteko
   -- zenbakia;Definituta dagoen proiektu baten kodea da
end record;
type Ikasle_Bektore is array (1..Ikasle_Kopuru_Maximoa) of
   Ikasle;
type Ikasle_Zerrenda is record
   Zerrenda: Ikasle_Bektore;
   Zenbat: Ikasle_Kopuru;
end record;
```

Ondoko azpiprograma inplementatu:

```
procedure Hasieratu_Proiektuak
    (Proiektuak: in out Proiektu_Bektore;
     Ikasleak: in Ikasle_Zerrenda) is
-- Aurrebaldintza:
--    Proiektuak bektoreko eskatzaile-zerrenda hutsik dago;
--    proiektu guztietan eskatzaile-kopurua zeroa da.
-- Postbaldintza:Proiektuak parametroko proiektu bakoitzean
--    Eskatzaileak eremuan proiektu hori eskatu duten
--    ikasleen zerrenda dago
```

Karrera bukaerako proiektuen asignazioa II

Aurreko prozedura erabiliz eta Eskatzaile_Hoberena funtzioa erabiliz (ez inplementatu!), *Proiektuak_Asignatu* azpiprograma egin:

```
function Eskatzaile_Hoberena (Proiektua: Proiektu)
           return string is
-- Aurrebaldintza:  Gutxienez ikasle batek eskatu du Proiektu
--    Hau da, gutxienez eskatzaile bat dago
-- Postbaldintza:   Proiektua eskatu duen eskatzaile
--    hoberenaren espediente-zenbakia itzuli da

procedure Proiektuak_Asignatu
   (Proiektuak: in out Proiektu_Bektore
    Ikasleak: in Ikasle) is
-- Aurrebaldintza:
-- Postbaldintza: Honako listatuak idatzi dira pantailan:
--    1) Proiektuen zerrenda, bakoitzerako bere zenbakia eta
--    nori eman zaion adierazten da (ikaslearen
--    espediente-zenbakia).
--    Proiekturen bat inori ere eman ez bazaio, EMAN GABE
--    hitzak azalduko dira espediente-zenbakiaren tokian.
--    2) Proiekturik gabe geratu diren ikasleen zerrenda:
```

Adibidez:

Lehenengo listatua:

Proiektu-Kodea	**Espediente-Zenbakia**
1003	15151151
1999	14141141
1004	*EMAN GABE*
1111	13333333
2000	*EMAN GABE*

Bigarren listatua:

Proiekturik gabeko ikasleak
15555555
14444444
12333333

EBAZPENAK

XII.1. Asignazio zuzenak markatu

```
1) V(I) := P; BAI, konpilazio-errorea azalduko da
   V(I) Tratamendu_Datuak motakoa da,
   P Datu_Pertsonalak motakoa da,
   eta datu-mota biak ez dira bateragarriak
2) V(5).Patologiak (V(5).Patologi_Kopurua + 1) := 5;
   EZ du sortuko konpilazio-errorerik
3) H.Kalea:= V(J).Pertsona.Helbidea.Kalea;
   EZ du sortuko konpilazio-errorerik
4) D.Hilabetea:= V(100).Sarrera_Data;
   BAI, konpilazio-errorea azalduko da
   D.Hilabetea integer motakoa da,
   V(100).Sarrera_Data  data motakoa da,
   eta datu-mota biak ez dira bateragarriak
5) V(3).Pertsona.Helbidea.Nazioa (1) := 'P';
   EZ du sortuko konpilazio-errorerik
```

XII.2. N txikienak lortu (1. bertsioa)

Ondoko datu-moten definizioak emanda:

```
Max: constant Integer := 100;
type Taula_Mota is array(1..Max) of Natural;
type Osokoen_Lista_Mota is record
  Osagai_Kop: Natural;    -- 0 edo handiagoa
  Osagaiak: Taula_Mota;
end record;
```

Ondoko azpiprograma inplementatu:

```
procedure N_Txikienak_Lortu (L1: in Osokoen_Lista_Mota;
                             N: in Natural;
                             L2: out
Osokoen_Lista_Mota)
-- Aurrebaldintza: N >= 0, L1.Osagai_Kop >= 0
--     N <= L1 listako osagai desberdinen kopurua
-- Postbaldintza: L2 listak L1-eko N elementu desberdin
--    txikienak
```

Hau da ebazpena:

```
package Motak is
   Max : constant Integer := 10;
   type Taula_Mota is array (1 .. Max) of Natural;
   type Osokoen_Lista is record
      Osagai_Kop : Natural;
      Osagaiak : Taula_Mota;
   end record;
end Motak;

with Motak;
procedure Bilatu_M_Baino_Handiagoa_Den_Txikiena
  (Lista1 : in Motak.Osokoen_Lista;
   X : in Integer;
   Pos : out Natural) is
-- Aurrebaldintza:
-- Postbaldintza: Pos-ek emango du X baino handiagoa den
--    Lista1 listako elementu txikienaren posizioa
   Min : Integer := Integer'last;
begin
   for I in Lista1.Osagaiak'First .. Lista1.Osagai_Kop loop
      if Lista1.Osagaiak (I) < Min and
         Lista1.Osagaiak (I) > X
      then
         Pos := I;
         Min := Lista1.Osagaiak (I);
      end if;
   end loop;
end Bilatu_M_Baino_Handiagoa_Den_Txikiena;

with Motak;
with Bilatu_M_Baino_Handiagoa_Den_Txikiena;
procedure N_Txikienak_Lortu (L1 : in Motak.Osokoen_Lista;
                             N : in Natural;
                             L2 : out Motak.Osokoen_Lista) is
-- Aurrebaldintza: N >= 0, L1.Osagai_Kop >= 0
--    N <= L1.Osagai_Kop listako osagai desberdinen kopurua
-- Postbaldintza: L2 listak L1-eko N elementu desberdin
--    txikienak ditu
    M : Integer := Integer'First;
    J : Natural;
begin
    for I in 1 .. N loop
--      L1 listako txikiena baino M baino handiagoa den
--      osagaiaren posizioa ematen du.
        Bilatu_M_Baino_Handiagoa_Den_Txikiena (L1, M, J);
        L2.Osagaiak (I) := L1.Osagaiak (J);
        M := L1.Osagaiak (J);
    end loop;
    L2.Osagai_Kop := N;
end N_Txikienak_Lortu;
```

XII.3. N txikienak lortu (2. bertsioa)

Sarrera estandarrean dugu luzera ez-ezaguneko osoko-sekuentzia bat (litekeena da oso luzea izatea eta, beraz, ez da posible array batean gordetzea). Sekuentziako N osoko txikienak lortuko dituen azpiprograma egin. Sekuentziaren luzera N baino askoz handiagoa da.

Oharra: Aurreko ariketaren antzekoa da baina kasu honetan aztertzeko osagai guztiak ezin dira sartu taula batean.

Hau da ebazpena:

```
package Motak is
    type Lista is array (Integer range <>) of Integer;
end Motak;

with Motak;
procedure Korritu (Non : in out Motak.Lista;
                   Nondik : in Integer;
                   Noraino : in Integer) is
begin
    for I in reverse Nondik + 1 .. Noraino + 1 loop
       Non(I) := Non(I - 1)
    end loop;
end Korritu;

with Motak;
with Korritu;
procedure Sartu_Kokatu (I : in Integer;
                        L : in out Motak.Lista;
                        Topea : in Integer) is

    Kokatua : Boolean := False;
    J : Integer := 1;
begin
    while J <= Topea and not Kokatua loop
        if L(J) > I
        then Korritu (L, J, Topea);
             L(J) := I;
             Kokatua := True;
        else J := J+1;
        end if;
    end loop;
    if J > Topea
    then L(Topea + 1) := I;
    end if;
end Sartu_Kokatu;
```

```
with Ada.Integer_Text_IO;
procedure Idatzi_Osokoa (I : in Integer) is
begin
    Ada.Integer_Text_IO.Put (I);
end Idatzi_Osokoa;

with Motak;
with Ada.Text_IO;
with Ada.Integer_Text_IO;
with Idatzi_Osokoa;
with Sartu_Kokatu;
procedure N_Txikienak (Izena : in String;
                       N : in Positive;
                       L : out Motak.Lista) is
    I : Integer;
    Handiena :Integer := 1;
    F : Ada.Text_IO.File_Type;
begin
    Ada.Text_IO.Open (File => F,
                      Mode => Ada.Text_IO.In_File,
                      Name => Izena);
    Ada.Integer_Text_IO.Get (F, I);
    L(1) := I;
while not (Ada.Text_IO.End_Of_File (F)) loop
        Ada.Integer_Text_IO.Get (F, I);
        if Handiena < N
        then  Sartu_Kokatu (I, L, Handiena)
              Handiena := Handiena + 1;
        else if I < L(Handiena)
             then Handiena := Handiena - 1;
                  Sartu_Kokatu (I, L, Handiena);
                  Handiena := Handiena + 1;
             end if;
        end if;
    end loop;
    Ada.Text_IO.Close (F);
    for I in 1 .. N loop
        Idatzi_Osokoa (L(I));
    end loop;
end N_Txikienak;
```

XII.4. Permutazio izan

Permutazio_Izan funtzioaren inplementazioa egin:

```
subtype Letra is character range 'A'..'Z';
type Hitz is array (1..5) of Letra;
function Permutazio_Izan (X, Y: in Hitz) return Boolean;
-- Aurrebaldintza: Luzera bereko bi hitz (letra guztiak
-- maiuskulaz eta hitzen letra kopurua = 5)
-- Postbaldintza: true Y hitza X hitzaren permutazioa bada,
--    false bestela (Y hitza X hitzaren permutazioa izango da
--    baldin eta X-ren letrak tokiz aldatuz Y lortzen badugu.)
```

Adibidez:

```
Permutazio_Izan (X=> “ALUAN”, Y => “LAUAN”) = true
Permutazio_Izan (X=> “ETXEA”, Y => “ATEXE”) = true
Permutazio_Izan (X=> “ALAIA”, Y => “ILAKI”) = false
Permutazio_Izan (X=> “AMAMA”, Y => “AMAIA”) = false)
Permutazio_Izan (X=> “AMAMA”, Y => “MAMAM”) = false)
```

GOGOAN HARTZEKO:

Y hitza X hitzaren PERMUTAZIOA bada, zenbat aldiz agertzen da alfabetoko letra bakoitza Y hitzean eta X hitzean?

Hau da ebazpena:

```
package Motak is
   subtype Letra is Character range  ‘A’ .. ‘Z’;
   type Hitz is array (1 .. 5) of Letra;
end Motak;

with Motak;
function Permutazio_Izan (X, Y: Motak.Hitz) return Boolean is
type T_Alfabeto is array (Motak.Letra) of Integer;
LetrakX, LetrakY: T_Alfabeto;
-- hitz bakoitzeko letra bakoitzaren
-- agerpen-kopuruak kontatzekoak
begin
-- Kontatu hitz bakoitzean letra bakoitza zenbat aldiz dagoen
-- Hasieratu kontagailuak
   for L in Motak.Letra’First .. Motak.Letra’last loop
      LetrakX(L) := 0;
      LetrakY(L) := 0;
   end loop;
-- kontatu letrak
   for I in X’First..X’last loop
      LetrakX (X(I)) := LetrakX (X(I)) + 1;
      LetrakY (Y(I)) := LetrakY (Y(I)) + 1;
   end loop;
-- Egiaztatu letra bakoitzak hitz bietan agerpen kopuru
-- berdinak dituela
   return LetrakX = LetrakY;
end Permutazio_Izan;
```

XII.5. Ordenatu bektorea kolorearen arabera

Ondoko erazagupenak emanda:

```
type Kolore is (Zuria, Beltza);
type Fitxa is record
   Kolorea: Kolore;
   Info: String (1..255);
end record;
type Fitxen_Bektore is array (Positive range <>) of Fitxa;
```

Ondoko azpiprogramaren inplementazioa egin:

```
procedure Ordenatu (Fitxak: in out Fitxen_Bektore) is
-- Aurrebaldintza:
-- Postbaldintza: Fitxak parametroaren balioa sarrerako
-- balioaren permutazioa da, non kolore bereko fitxak segidan
-- dauden.
```

Oharra: Emaitza lortzeko ezin da beste taularik erabili.

Hau da ebazpena:

```
package Motak is
   type Kolore is (Zuria, Beltza);
   type Fitxa is record
                    Kolorea: Kolore;
                    Info: String (1..255);
                 end record;
   type Fitxen_Bektore is array (Positive range <>) of Fitxa;
end Motak;
```

```
with Motak;
procedure Ordenatu (Fitxak: in out Fitxen_Bektore) is
-- Aurrebaldintza:
-- Postbaldintza: Fitxak parametroaren balioa sarrerako
--    balioaren permutazioa da, non kolore bereko fitxak segidan
--    dauden (hasieran kolore bateko fitxak daude eta segidan
--    bestekoak).
   Azken_Zuria, Lehen_Beltza: Integer;
   Lagun : Fitxa;
begin
   Azken_Zuria := 0;
   Lehen_Beltza := Fitxak'Last + 1 ;
   while Azken_Zuria < Lehen_Beltza - 1  loop
   -- 1.. Azken_Zuria tarteko fitxak zuriak dira
   -- Lehen_Beltza..Fitxak'Last tarteko fitxak beltzak dira
      if Fitxak (Azken_Zuria + 1).Kolorea = Zuria
      then -- Eguneratu Lehen_Beltza
           Azken_Zuria := Azken _Zuria + 1 ;
      else -- Trukatu azken zuriaren ondokoa
           -- eta lehen beltzaren aurrekoa
           Lagun := Fitxak (Azken_Zuria + 1) ;
           Fitxak (Azken_Zuria + 1):=
                         Fitxak (Lehen_Beltza - 1);
           Fitxak (Lehen_Beltza - 1) := Lagun ;
           -- Eguneratu Lehen_Beltza
           Lehen_Beltza := Lehen_Beltza - 1 ;
      end if ;
   end loop ;
end Ordenatu;
```

XII.6. Besteen bestekoa

Definitu azpiprograma hau:

```
procedure Besteen_Bestekoa_Dauka
    (B: in Osokoen_Bektore;
     Dauka: out Boolean;
     Pos: out Integer)
-- Aurrebaldintza:
-- Postbaldintza: B bektoreko osagai bat beste
--    osagaien baturabada Dauka = true
--    eta Pos horrelako osagai baten posizioa da,
--    bestela Dauka=false eta Pos-en balioa
--    edozein da.
```

Hau da ebazpena:

```
procedure Besteen_Batezbestekoa_Dauka
    (B : in Osokoen_Bektore;
     Dauka: out Boolean;
     Pos: out Integer) is

   function Batura
       (Bekt : in Osokoen_Bektore)
    return Integer is
      Batua : Integer := 0;
   begin
      for I in Bekt'First .. Bekt'Last loop
          Batua := Batua + Bekt(I) ;
      end loop ;
      return (Batua);
   end Batura;

begin
   Batura_Erdia := Batura(B)/2;
   I:= 1;
   while B(I) /= Batura_Erdia and I<B'Last  loop
      I := I+1;
   end loop;
   Dauka:=   B(I) = Batura_Erdia;
   if Dauka then
      Pos := I ;
   end if ;
end Besteen_Batezbestekoa_Dauka ;
```

XII.7. Eratostenesen bahea

Muga bat baino txikiagoak diren zenbaki lehenak kalkulatzeko araua. (Eratostenesen bahea, K.a. III. mendea)

Metodoak honela dio: finkatutako mugarainoko zenbaki arrunten segida idatzirik, 2 zenbaki lehenetik hasi eta 4tik aurrera daudenak binaka ezabatzen dira (4, 6, 8, …), mugaraino iritsi arte; segidan 3 zenbakiaren multiploak ezabatzen dira hirunaka, 9tik hasita —beti ere aldez aurretik ez badaude ezabaturik, jakina—; eta gauza bera 5enekin, 25etik hasita. Eta horrela jarraitzen da karratua muga baino handiagoa duen zenbaki lehenera iritsi arte. Ezabatu gabe gelditutakoak dira zenbaki lehenak.

Muga bat emanda, muga hori baino txikiago edo berdinak diren zenbaki lehenak irteera estandarrean idatziko dituen Ada azpiprograma egin. Azpiprogramaren emaitza azaldutako metodoaren arabera (Eratostenesen bahea) kalkulatu behar da.

```
package Motak is
   type ETaula is array (Positive range <>) of Boolean;
end Motak;

with Motak;
withIdatzi_Osdokoa;
procedure Eratostenes (M : in Integer) is
    Taula : Motak.Etaula (1 .. M) := True;
    J, K : Integer;
begin
    J := 2;
    while J ** 2 <= M loop
        B (J) := False;
        K := J ** 2;
        while K <= M loop
            B(K) := False;
            K := K + J;
        end loop;
        J := J + 1;
    end loop;
    for I in 1 .. M loop
        if Taula(I)
        then Idatzi_Osokoa (I);
        end if;
    end loop;
end Eratostenes;
```

XII.8. Meseta luzeenaren luzera

Osoko-bektore baten meseta luzeenaren luzera itzuliko duen azpiprograma inplementatu. Adibidez, emanda ondoko bektorea: 3 3 5 5 5 2 8 7 7 7 7 3.

Azpiprogramaren emaitza 4 izango da, 7 zenbakiaren meseta luzeena baita meseta horretan lau osagai baitaude.

```
function Meseta_Luzeenaren_Luzera (B: Osokoen_Bektore)
          return Integer
-- Aurrebaldintza:
-- Postbaldintza: B bektoreko meseta luzeenaren luzera itzuli
--    da. Meseta dagoela esaten da ondoz ondoko
--    osagaietan balio bera agertzen denean.
```

Hau da ebazpena:

```
function Meseta_Luzeenaren_Luzera (B: in Osokoen_Bektore)
    return Integer is
begin
   Luzeena := 0;
   if B'Last > 1 then
      Luzera := 1;
      for I in 2..B'Last loop
         if B(I) = B(I-1) then
            Luzera := Luzera + 1 ;
         else
            if Luzera > Luzeena then
              Luzeena := Luzera ;
            end if;
            Luzera :=1;
         end if;
      end loop ;
      if Luzera > Luzeena then
         Luzeena := Luzera ;
      end if;
   end if;
   return Luzeena;
end Meseta_Luzeenaren_Luzera;
```

Beste modu batera:

```
function Meseta_Luzeenaren_Luzera (B: in Osokoen_Bektore)
    return Integer is
begin
   Luzeena := 0;
   if B'Last > 1 then
      Luzera := 1;
      for I in 2..B'Last loop
         if B(I) = B(I-1) then
            Luzera := Luzera + 1 ;
            if Luzera > Luzeena then
              Luzeena := Luzera ;
            end if;
         else Luzera := 1;
         end if;
      end loop ;
      if Luzera > Luzeena then
         Luzeena := Luzera ;
      end if;
   end if;
   return Luzeena ;
end Meseta_Luzeenaren_Luzera;
```

XII.9. Multzoen ebakidura

Ondoko datu-moten definizioak emanda:

```
Max: constant Integer := 100;
type Taula_Mota is array(1..Max) of Natural;
type Multzo_Mota is record
   Osagai_Kop: Natural;    — 0 edo handiagoa
   Osagaiak: Taula_Mota;
end record;
```

Ondoko azpiprograma inplementatu:

```
function Ebakidura (M1, M2: in Multzo_Mota)
          return Multzo_Mota
-- Aurrebaldintza: M1.Osagai_Kop eta M2.Osagai_Kop >= 0
--    M1 eta M2-k osokoen multzoak adierazten dituzte
-- Postbaldintza: emaitza M1 eta M2-n dauden elementuen
--    multzoa
```

Hau da ebazpena:

```
package Motak is
Max: constant Integer := 5;
type Taula_Mota is array(1..Max) of Natural;
type Multzo_Mota is record
-- osoko positiboen multzoak adierazteko
   Osagai_Kop: Natural;    — 0 edo handiagoa
   Osagaiak: Taula_Mota;
end record;
-- 1..Osagai_Kop posizioetako osagaiak desberdinak dira
end Motak;
```

```
with Motak;
with Barne;
function Ebakidura (M1, M2: in Motak.Multzo_Mota)
              return Motak.Multzo_Mota is
-- Aurrebaldintza: M1.Osagai_Kop eta M2.Osagai_Kop >= 0
--   M1 eta M2-k osokoen multzoak adierazten dituzte
-- Postbaldintza: emaitza M1 eta M2-n dauden elementuen
-- multzoa
Emaitza: Motak.Multzo_Mota;
begin
-- Sortu emaitza multzo huts bezala
   Emaitza.Osagai_kop := 0;
-- M1 multzoko elementuak tratatu,
-- M2-n badaude, Emaitza-n sartu
   for I in 1..M1.Osagai_Kop loop
      if Barne(M1.Osagaiak(I), M2)
      then -- M1 multzoaren I. elementua  M2 multzoan ere
      --  dago, sartu Emaitza multzoan
         Emaitza.Osagai_kop := Emaitza.Osagai_Kop + 1;
         Emaitza.Osagaiak(Emaitza.Osagai_Kop) :=
            M1.Osagaiak(I);
      end if;
   end loop;
   return Emaitza;
end Ebakidura;

with Motak;
function Barne (Elem: Integer; M: Motak.Multzo_Mota)
             return Boolean is
-- Aurrebaldintza: M aldagaiak osoko zenbakien multzoa
-- Postbaldintza: Emaitza true da elem multzoko osagaia bad
-- false bestela
    J: Integer;
begin
   if M.Osagai_kop = 0
   then
      return False;
   else
      j := 1;
      while j < M.Osagai_Kop and M.Osagaiak(j) /= Elem loop
         j := j + 1;
      end loop;
      return M.Osagaiak(j) = Elem;
   end if;
end Barne;
```

XII.10. Multzoen bildura

Ondoko datu-moten definizioak emanda:

```
Max: constant Integer := 100;
type Taula_Mota is array(1..Max) of Natural;
type Multzo_Mota    is record
   Osagai_Kop: Natural;    — 0 edo handiagoa
   Osagaiak: Taula_Mota;
end record;
```

Ondoko azpiprograma inplementatu:

```
function Bildura (M1, M2: in Multzo_Mota) return Multzo_Mota
-- Aurrebaldintza: M1.Osagai_Kop eta M2.Osagai_Kop >= 0
--    M1 eta M2-k osokoen multzoak adierazten dituzte
-- Postbaldintza: emaitza M1 edota M2-n dauden elementuen
--    multzoa da
```

Hau da ebazpena:

```
with Motak;
with Barne;
function Bildura (M1, M2: in Motak.Multzo_Mota)
              return Motak.Multzo_Mota is
-- Aurrebaldintza: M1.Osagai_Kop eta M2.Osagai_Kop >= 0
--   M1 eta M2-k osokoen multzoak adierazten dituzte
-- Postbaldintza: emaitza M1 edota M2-n dauden elementue
--   multzoa
    Emaitza: Motak.Multzo_Mota;
begin
-- M1 multzoko elementu guztiak sartu emaitzan
-- Emaitza := M1;
-- M2 multzoko elementuak tratatu, ez bazeuden M1-n sart
-- Emaitza-n
   for I in 1..M2.Osagai_Kop loop
      if not Barne(M2.Osagaiak(I), M1)
      then -- M2 multzoaren I. elementua  M1 multzoan ez
           -- zegoen, sartu Emaitza multzoan
          Emaitza.Osagai_kop := Emaitza.Osagai_Kop + 1;
          Emaitza.Osagaiak(Emaitza.Osagai_Kop) :=
             M2.Osagaiak(I);
      end if;
   end loop;
   return Emaitza;
end Bildura;
```

XII.11. Multzo guztien ebakidura

Ondoko definizioa emanda:

```
type Multzoen_Bektorea_Mota is array (Integer range <>)
   of Multzo_Mota;
```

Ondoko azpiprograma inplementatu:

```
function Guztien_Ebakidura (B: Multzoen_Bektorea_Mota)
                  return Multzo_Mota
-- Postbaldintza: emaitza B bektoreko multzo guztien
-- ebakidura da
```

Hau da ebazpena:

```
package Motak is
Max: constant Integer := 5;
type Taula_Mota is array(1..Max) of Natural;
type Multzo_Mota is record -- osoko positiboen multzoa
                           -- adierazteko
   Osagai_Kop: Natural;    -- 0 edo handiagoa
   Osagaiak: Taula_Mota;
end record;

-- 1..Osagai_Kop posizioetako osagaiak desberdinak dira
type Multzoen_Bektorea_Mota is array(Integer range <>)
   of Multzo_Mota;
end Motak;

with Motak;
with Ebakidura;  --aurreko ariketakoa
function Guztien_Ebakidura(B: Motak.Multzoen_Bektorea_Mota)
            return Motak.Multzo_Mota is
-- Aurrebaldintza: B'First = 1
-- Postbaldintza: emaitza B bektoreko multzo guztien
--    ebakidura da
    Emaitza: Motak.Multzo_Mota;
begin
  if B'Last <= 1
  then
     -- Bektoreak ez du elementu bat ere ez
     Emaitza.Osagai_Kop := 0; -- emaitza multzo hutsa da
  else
     -- Bada multzo bat gutxienez
     Emaitza := B(1);
     for i in 2..B'Last loop
        Emaitza := Ebakidura (Emaitza, B(i));
     end loop;
  end if;
  return Emaitza;
end Guztien_Ebakidura ;
```

XII.12. Zatikien definizioa

Zatikien definizioa: p/q zatikia da, p eta q osokoak badira eta q?0. Defini ezazu *Zatiki* izeneko datu-mota bat, zatikiak adierazteko balioko duena.

Hau da ebazpena:

```
type Zatiki is record
    Zenbakitzailea : Integer;
    Izendatzailea :  Positive;
end record;
```

XII.13. Zatidura-lista datu-mota definitu

M zatiki dituen L lista bat emanda, L/≡ zatidura-lista esango diogu L-ren osagaiekin osaturiko listak osagai dituen listari, honako baldintzak beteko dituelarik:

L/≡ lista osatzen duten listak binaka hartuta disjuntuak dira.

L/≡ lista osatzen duten listetako osagai guztien artean L listako osagai guztiak agertzen dira.

L/≡ listako lista bakoitzeko elementu guztiak bata bestearen baliokideak dira. (p/q≡r/s ⇔ p.s=q.r).

Adibidea: L = [1/3, 2/4, 5/3, 3/9, 6/18, 10/6]

L/≡ = [[1/3, 3/9, 6/18], [2/4], [5/3, 10/6]]

Defini itzazu, alde batetik, Zatiki_Lista izeneko datu-mota bat, zatiki-listak adierazteko balioko duena; eta bestetik, Zatidura_Lista datu-mota, zatikizko zatidura-listak adieraztekoa.

Hau da ebazpena:

```
subtype Indize1_0 is Integer range 0 .. M;
subtype Indize1 is Integer range 1 .. M;
type Taula1 is array (Indize1) of Zatiki ;
type Zatiki_Lista is record
    Info : Taula1;
    Zenbat : Indize1_0;
end record;
```

```
subtype Indize2_0 is Integer range 0 .. N;
subtype Indize2 is Integer range 1 .. N;
type Taula2 is array (Indize2) of Zatiki_Lista;
type Zatidura_Lista is record
    Info : Taula2;   -- Baliokidetasun-klaseen lista.
                     -- Klase bakoitza zatiki-lista izanik.
    Zenbat : Indize2_0; -- Zenbat klase dagoen.
end record
```

XII.14. Zatidura-lista lortu

Idatz ezazu zatiki-lista baten zatidura-lista lortuko duen funtzioa:

```
function Lortu_Zatidura_Lista (ZL: Zatiki_Lista)
          return Zatidura_Lista;
```

Hau da ebazpena:

```
function Baliokideak (Zer1, Zer2 : in Zatiki)
           return Boolean is
begin
    return Zer1.Zenbakitzailea * Zer2 Izendatzailea =
           Zer1.Izendatzailea * Zer2.Zenbakitzailea;
end Baliokideak;

with Baliokideak;
procedure Kokatu (Zer : in Zatiki;
                  Non : in out Zatidura_Lista) is
-- Postbaldintza: Zer zatikia zatidura-listako
--    bere baliokideen klasean sartu da,
--    edo klase berri batean sartu da
--    zatidura-listan, lehenago bere
--    baliokiderik ez bazegoen.
    Baliokidea : Boolean;
    I : Indize2;
begin
    Baliokidea := False;
    I : 1;
    while I <= Non.Zenbat and not Baliokidea loop
        if Baliokideak (Zer, Non.Info(I).Info (1))
        then Baliokidea := True;
             Non.Info(I).Zenbat :=
             Non.Info(I).Zenbat + 1;
             Non.Info(I):Info(Non.Info(I).Zenbat) := Zer;
        else I := I + 1;
        end if;
    end loop;
    if not Baliokidea
    then Z.Zenbat := Z.Zenbat + 1;
         Z.Info(Z.Zenbat).Zenbat := 1:
         Z.Info(Z.Zenbat).Info (Z.Info(Z.Zenbat)) := Zer;
    end if;
end Kokatu;
```

```
with Kokatu;
function Lortu_Zatidura_Lista (ZL: in Zatiki_Lista)
           return Zatidura_Lista is
    Z : Zatidura_Lista;
begin
    for I in 1 .. ZL.Zenbat loop
        Kokatu (ZL.Info(I), Z);
    end loop;
    return Z;
end Lortu_Zatidura_Lista;
```

XII.15. Zatiki ez-negatiboen adierazle kanonikoa

p/q zatiki ez-negatibo baten adierazle kanonikoa r/s zatikia da, non r eta s ez-negatiboak diren eta duten zatitzaile komun bakarra 1 den (hau da, elkarrekiko lehenak dira), eta p/q≡r/s

Adibidea: 18/24 zatikiaren adierazle kanonikoa 3/4 da.

Idatz ezazu zatiki ez-negatibo baten adierazle kanonikoa lortuko duen funtzioa:

```
function Kanonikoa (Z: Zatiki) return Zatiki;
-- Aurrebaldintza:
--    Z-ren izendatzailea eta zenbakitzailea ez-
--    negatiboak dira.
```

Hau da ebazpena:

```
function KMZ (Z1, Z2 : in Integer)
         return Integer is
    Lag1, Lag, Lag2 : Integer;
-- Postbaldintza: Z1 eta Z2-ren zatitzaile komunetako
--                handiena itzuliko du
begin
    Lag1 := Z1;
    Lag2 := Z2;
    while Lag2 /= 0 loop
        Lag := Lag1 mod Lag2;
        Lag1 := Lag2;
        Lag2 := Lag;
    end loop;
    return Lag1;
end KMZ;
```

```
with KMZ;
function Kanonikoa (Z : in Zatiki) return Zatiki is
    Lag : Integer;
    Emaitza : Zatiki;
begin
    Lag := KMZ(Z.Zenbakitzailea, Z.Izendatzailea);
    Emaitza.Zenbakitzailea := Z.Zenbakitzailea/Lag ;
    Emaitza.Izendatzailea:= Z.Izendatzailea/Lag ;
    return (Emaitza);
end Kanonikoa;
```

XII.16. Zatiki-listaren adierazle kanonikoen batura eta maiztasun handieneko kanonikoa.

L zatiki-lista bat emanda, izendatzailea eta zenbakitzailea ez-negatibo dituzten zatikiak dituena, idatz itzazu, batetik, L-ren osagai guztien adierazle kanonikoen batura itzuliko duen azpiprograma, eta, bestetik, L-ren osagai gehien biltzen dituen baliokidetza-klasearen adierazle kanonikoa itzuliko duena. Horretarako bi zatikien batura kalkulatzen duen ondoko funtzioa erabil daiteke:

```
function Batura (R1, R2: Zatiki) return Zatiki;
```

Adibidea: Izanik L = [1/3, 2/4, 5/3, 3/9, 6/18, 10/6], listako zatiki guztien batura: 29/6

Osagai gehien biltzen dituen baliokidetza-klasearen adierazle kanonikoa: 1/3

```
with Lortu_Zatidura_Lista;
with Baliokideen_Batura;
with Batura;
with Kanonikoa;
procedure Batura_eta_Maiztasun_Handieneko_Kanonikoa
         (ZL : in Zatiki_Lista;
          Batura_osoa : out Zatiki;
          Kanonikoa1 : out Zatiki) is
    Batura_osoa: Zatiki := (0,0);
    Z : Zatidura_Lista;
    Max : Integer := 0;
begin
    Z := Lortu_Zatidura_Lista (ZL);
    for I in 1 .. Z.Zenbat loop
        Batura_osoa:= Batura (Batura_osoa,
                              Baliokideen_Batura (Z.Info(I));
        if Z.Info(I).Zenbat > Max
        then Max  = Z.Info(I).Zenbat;
             Kanonikoa1 := Kanonikoa (Z.Info(I).Info(1));
        end if;
    end loop;
end Batura_eta_Maiztasun_Handieneko_Kanonikoa;
```

```
with Kanonikoa;
function Baliokideen_Batura
         (Baliokideak : in Zatiki_Lista)
        return Zatiki is
    ZLag : Zatiki;
begin
   ZLag.Zenbakitzailea :=
        Baliokideak.Zenbat *
        Baliokideak.Info(1).Zenbakitzailea;
   ZLag.Izendatzailea :=
        Baliokideak.Info(1).Izendatzailea;
   return (Kanonikoa (ZLag));
end Baliokideen_Batura;

function Batura (Z1, Z2 : Zatiki) return Zatiki is
    ZLag : Zatiki;
begin
    ZLag.Zenbakitzailea :=
        Z1.Zenbakitzailea * Z2.Izendatzailea +
        Z2.Zenbakitzailea * Z1.Izendatzailea;
    ZLag.Izendatzailea :=
        Z1.Izendatzailea * Z2.Izendatzailea;
   return (Kanonikoa (ZLag));
end Batura;
```

XII.17. Ezkutatu laukia

Honako datu-motak erabili dira pantaila batean erakutsi nahi den irudi bat definitzeko

```
type Irudi_Puntu is record
     Horia, Urdina,  Gorria : Boolean;
     Intentsitatea : Integer ;
end record;

type Irudi is array (Integer range <>, Integer range <>)
                    of Irudi_puntu;
subtype Kolore is integer range 1..8;
```

Pantailako puntu bakoitzean hiru sentsore egongo dira, bakoitza oinarrizko kolore batekoa. Ikusleak puntu horretan ikusten duen kolorea piztuta dauden oinarrizko koloreen konbinazioa da.

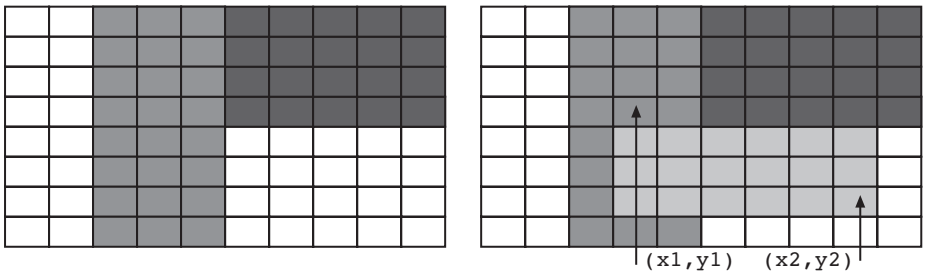
Demagun azpiprograma hauek eginda daudela:

```
function Konbinazioa (Horia, Urdina, Gorria : Boolean)
    return Kolore
--Aurrebaldintza:
--Postbaldintza: true balio duten oinarrizko kolore
--    onbinatuz lortzen den kolorea
--    itzultzen du.
procedure Analizatu (Kolorea : in Kolore;
      Horia, Urdina, Gorria : out Boolean)
--Aurrebaldintza:
--Postbaldintza: Kolorea lortzeko behar diren oinarrizko
      koloreek true balio dute, besteek false
```

Honako prozedura diseinatu eta egin:

```
procedure Ezkutatu_Laukia (Irudia: in out Irudi;
                           X1, Y1, X2, Y2 : Integer)
--Aurrebaldintza:(X1,Y1) eta (X2,Y2) irudiko bi puntuen
--    koordenatuak dira
--    X1<X2   eta   Y1<Y2
--Postbaldintza: irudia hasierakoa bezalakoa da baina
--    barruko lauki bat lausotuta azaltzen da.
--    A) (X1,Y1) eta (X2,Y2) puntuak lauki
--    lausotuaren diagonalaren muturrak dira
--    B) Lauki lausotuko puntu guztien kolorea
--    berdina da: hasierako irudiko laukian
--    maizago azaltzen zen kolorea
--    (oinarrizko koloreen konbinazio bat).
--    V) Lauki lausotuaren puntu guztietako
--    intentsitateak berdinak dira:
--    hasierako irudiko laukiko puntuen
--    intentsitateen batezbestekoa.
```

Adibidea: Hasieran Bukaeran



Hau da ebazpena:

```
package Motak is
   type Irudi_Puntu is record
     Horia, Urdina,  Gorria : Boolean;
     Intentsitatea : Integer ;
end record;
type Irudi is array (Integer range <>, Integer range <>)
                    of Irudi_puntu;
subtype Kolore is integer range 1..8;
end Motak;

with Motak;
procedure Ezkutatu_Laukia (Irudia: in out Irudi;
                           X1, Y1, X2, Y2 : Integer)  is
--Aurrebaldintza:(X1,Y1) eta (X2,Y2) irudiko bi puntuen
--    koordenatuak dira
--    X1<X2   eta   Y1<Y2
--Postbaldintza: irudia hasierakoa bezalakoa da baina
--    barruko lauki bat lausotuta azaltzen da.
--    A) (X1,Y1) eta (X2,Y2) puntuak lauki
--    lausotuaren diagonalaren muturrak dira
--    B) Lauki lausotuko puntu guztien kolorea
--    berdina da: hasierako irudiko laukian
--    maizago azaltzen zen kolorea
--    (oinarrizko koloreen konbinazio bat).
--    V) Lauki lausotuaren puntu guztietako
--    intentsitateak berdinak dira:
--    hasierako irudiko laukiko puntuen
--    intentsitateen batezbestekoa.

   Kolore_Kont: array (motak.Kolore) of Integer;
   Int_Kont, Int_Bb : Integer ;

begin
   -- Kontatu laukiko koloreak eta metatu intentsitatearen
   -- balioak
   for I_Kolore in motak.Kolore loop
         Kolore_Kont (I_Kolore ) := 0;
   end loop ;
   Int_Kont := 0;
   for I in X1..X2 loop
      for J in Y1 ..Y2 loop
         Kolorea := Konbinazioa (Irudi(I,J).Horia,
                                 Irudi(I,J).Urdina,
                                 Irudi(I,J).Gorria) ;
         Kolore_Kont (Kolorea) := Kolore_Kont (Kolorea) + 1 ;
         Int_Kol := Int_Kol + Irudi (I,J).Intentsitatea ;
      end loop ;
   end loop ;
```

```
   -- Lortu Kolore maizena
   Max := 0;
for I_Kolore in Motak.Kolore loop
      if Kolore_Kont (I_Kolore) > Max then
         Max := Kolore_Kont (I_Kolore) ;
         Kolore_Max:= I_Kolore ;
      end if ;
   end loop;

   -- Lortu batezbesteko intentsitatea
   Int_Bb := Int_Kont / ((X2-X1+1)*(Y2-Y1+1))

   -- Lausotu laukia
   Analizatu (Kolore_Max, H, U, G);
   for I in X1..X2 loop
      for J in Y1 ..Y2 loop
         Irudi (I,J).Horia  := H;
         Irudi (I,J).Urdina := U;
         Irudi (I,J).Gorria := G;
         Irudi (I,J).Intentsitatea := Int_Bb ;
      end loop ;
   end loop ;
end Ezkutatu_Laukia ;
```

XII.18. Karrera bukaerako proiektuen asignazioa I

Karrera-bukaerako 50 proiektu definitu dira fakultate batean ikasleen artean banatuak izateko. Ikasle bakoitzak proiektu bat eskatu du. Proiektu bakoitzerako eskatzaile bat egon daiteke, edo bat baino gehiago, edo bat ere ez. Proiektuak asignatzeko laguntza informatikoa inplementatzeko ondoko erazagupenak definitu dira:

```
Proiektu_Kopurua: constant Integer:= 50;
Ikasle_Kopuru_Maximoa: constant Integer:= 100;

type Eskatzaile is record
    Espediente_Zbkia: string(1..8);
    Batezbestekoa: Float ;
end record;
subtype Ikasle_Kopuru is Integer
    range 0 ..  Ikasle_Kopuru_Maximoa;
type Eskatzaile_Bektore is array (1..Ikasle_Kopuru_Maximoa)
    of Eskatzaile;
type Eskatzaile_Zerrenda is record
   Zerrenda: Eskatzaile_Bektore ;
   Eskatzaile_Kopurua: Ikasle_Kopuru := 0 ;
end record;
```

```
type Proiektu is record
   Kodea: Integer ;  — 1000tik 2000ra bitarteko zenbakia
   Titulua: string (1..100);
   Zuzendaria: string (1..30);
   Eskatzaileak: Eskatzaile_Zerrenda ;
end record;
type Proiektu_Bektore is array(1.. Proiektu_Kopurua) of
    Proiektu;

type Ikasle is record
   Espediente_Zbkia: string (1..8);
   Nota: Float;
   Proiektu_Eskatua: Integer; -- 1000tik 2000ra bitarteko
   -- zenbakia. Definituta dagoen proiektu baten kodea da
end record;
type Ikasle_Bektore is array (1..Ikasle_Kopuru_Maximoa) of
    Ikasle;
type Ikasle_Zerrenda is record
   Zerrenda: Ikasle_Bektore;
   Zenbat: Ikasle_Kopuru;
end record;
```

Hau da ebazpena:

```
with Motak;
procedure Hasieratu_Proiektuak
  (Proiektuak: in out Motak.Proiektu_Bektore;
   Ikasleak: in Motak.Ikasle_Zerrenda) is
-- Aurrebaldintza:
--    Proiektuak bektoreko eskatzaile-zerrenda hutsik dago;
--    proiektu guztietan eskatzaile-kopurua zeroa da.
-- Postbaldintza:Proiektuak parametroko proiektu bakoitzean
--    Eskatzaileak eremuan proiektu hori eskatu duten
--    ikasleen zerrenda dago
   Zer_Proiektu : Integer;
   procedure Gehitu (P : in out Motak.Proiektu_Bektore;
      I : in Motak.Ikasle) is
      Lag : Integer;
   begin
      I := 1;
      while P(I).Kodea /= I.Proiektu_Eskatua loop
         I:= I+1 ;
      end loop ;
      Lag := P(I).Eskatzaileak.Eskatzaile_Kopurua;
      P(I).Eskatzaileak.Eskatzaile_Kopurua :=  Lag + 1;
      P(I).Eskatzaileak.Zerrenda(Lag+1).Espediente_Zbkia :=
           I.Espediente_Zbkia;
      P(I).Eskatzaileak.Zerrenda(Lag+1).Batezbestekoa :=
           I.Nota;
   end Gehitu;
begin
   for I in 1 .. Ikasleak.Zenbat loop
      Gehitu (Proiektuak, Ikasleak.Zerrenda (I));
   end loop;
end Hasieratu_Proiektuak;
```

XII.19. Karrera bukaerako proiektuen asignazioa II

Aurreko prozedura erabiliz eta Eskatzaile_Hoberena funtzioa erabiliz (ez inplementatu!), Proiektuak_Asignatu azpiprograma egin:

```
function Eskatzaile_Hoberena (Proiektua: Proiektu)
                          return string is
-- Aurrebaldintza:  Gutxienez ikasle batek eskatu du Proiektu
--    Hau da, gutxienez eskatzaile bat dago
-- Postbaldintza:   Proiektua eskatu duen eskatzaile
--    hoberenaren espediente-zenbakia itzuli da
with Ada.Integer_Text_IO, Ada.Text_IO,
Motak,Eskatzaile_Hoberena;

procedure Proiektuak_Asignatu
  (Proiektuak: in out Motak.Proiektu_Bektore;
   Ikasleak: in Motak.Ikasle_Zerrenda) is
-- Aurrebaldintza:
-- Postbaldintza: Honako listatuak idatzi dira pantailan:
--   1) Proiektuen zerrenda, bakoitzerako bere zenbakia eta
--   nori eman zaion adierazten da (ikaslearen
--   espediente-zenbakia).
--   Proiekturen bat inori ere eman ez bazaio, EMAN GABE
--   hitzak azalduko dira espediente-zenbakiaren tokian.
--   2) Proiekturik gabe geratu diren ikasleen zerrenda:
```

Adibidez:

Lehenengo listatua:

Proiektu-Kodea	**Espediente-Zenbakia**
1003	15151151
1999	14141141
1004	*EMAN GABE*
1111	13333333
2000	*EMAN GABE*

Bigarren listatua:

Proiekturik gabeko ikasleak
15555555
14444444
12333333

Hau da ebazpena:

```
function Ba_Dago (I : in String;
                  A : in Motak.Asignazioen_Bektore)
                return Boolean is
         Aurkitua : Boolean := False;
         J : Integer := 1;
   begin
      while J <= Motak.Proiektu_Kopurua and not Aurkitua loop
         if A(J).Espediente_Zbkia = I
         then Aurkitua := True;
         else J := J + 1;
         end if;

end loop;
      return Aurkitua;
   end Ba_Dago;

   procedure Idatzi_Asignazioak
                  (A: in Motak.Asignazioen_Bektore) is
   begin
      Ada.Text_IO.Put_line
          ("Proiektu-Kodea Espediente-Zenbakia");
      for I in 1 .. Motak.Proiektu_Kopurua loop
         Ada.Integer_Text_IO.Put (A(I).Kodea);
         Ada.Text_IO.Put_LIne (A(I).Espediente_Zbkia);
      end loop;
   end Idatzi_Asignazioak;

   procedure Idatzi_Proiekturik_Gabe_Daudenak
               (A: in Motak.Asignazioen_Bektore) is
       Izena : String (1 .. 8);
   begin
      Ada.Text_IO.Put_line("Proiekturik gabeko ikasleak");
      for I in 1 .. Ikasleak.Zenbat loop
         Izena := Ikasleak.Zerrenda(I).Espediente_Zbkia;
         if not Ba_Dago (Izena, A)
         then
            Ada.Text_IO.Put_Line (Izena);
            Ada.Text_IO.New_Line;
         end if;
      end loop;
   end Idatzi_Proiekturik_Gabe_Daudenak;

   Izena : String (1 .. 8);
   Asigk : Motak.Asignazioen_Bektore;
begin
   for I in 1 .. Motak.Proiektu_Kopurua loop
      Asigk(I).Kodea := Proiektuak(I).Kodea;
      if Proiektuak(I).Eskatzaileak.Eskatzaile_Kopurua = 0
      then Izena := Eskatzaile_Hoberena (Proiektuak (I));
         Asigk(I).Espediente_Zbkia := Izena;
      else Asigk(I).Espediente_Zbkia := "EMANGABE";
      end if ;
   end loop;
   Idatzi_Asignazioak (Asigk);
   Idatzi_Proiekturik_Gabe_Daudenak (Asigk);
end Proiektuak_Asignatu;
```

Bibliografia

Castro, J., Cucker, F., Messeguer, X., Rubio, A., Solano, L., Vallés, B.
"*Curso de Programación*"
Mc Graw-Hill, 1992

Peyrin, J.P. eta Scholl, P.C.
"Algorithmique et Programmation"
Laboratoire IMAG, bp53 x, 38041 GRENOBLE-CEDEX, 1982

Skansholm, J.
"ADA 95 from the Beginning"
Addison-Wesley. 1996.

Shackelford, R.L.
"Introducing to Computing and Algorithms"
Addison-Wesley. 1998

Watt, D., Wichmann, B., Findlay, W
"ADA Lengoaia eta Metodologia"
EHUko Argitalpen Zerbitzua Leioa 1996.