PROGRAMATZEKO ALGORITMOAK.

ariketa bilduma

```
ALGORITMOA liburua_irakurtzen
(*Hau ez da benetako algoritmoa, Jokoa baizik*)
ALDAGAIAK kapitulu1,kapitulu2 (karakterea)
          ariketa             (osoa)
HASIERA irakur(kapitulu1)
         irakur(kapitulu2)
         ariketa:= 0
         ERREPIKA
           ikas_ariketa
           ariketa:= ariketa + 1
         HARIK_ETA ariketa=12
         idatz('ba dakit algoritmoak idazten')
AMAIA

  EKINTZA ikas_ariketa
    ALDAGAIAK ez_ulertu  (boolearra)
              kontzeptuberriak, ariketaebatzia (karakterea)
              ariketabetesarria (karakterea)
    HASIERA  ez_ulertu:= egiazkoa
             BITARTEAN ez_ulertu EGIN
               irakur(kontzeptuberriak)
               irakur(ariketaebatzia)
             AMBITARTEAN
             ERREPIKA
               ebatz(ariketabetesarria)
             HARIK_ETA besterikez
    AMAIA(* EKINTZA ikas_ariketa *)
```

udako euskal unibertsitatea

IRUÑEA 1984

P R O G R A M A T Z E K O

A L G O R I T M O A K

Kepa Sarasola

EUSKO JAURLARITZAREN LAGUNTZAZ

Jabegoa: U.E.U.ko INFORMATIKA Saila
Lege-gordailua: BI-979-84
ISBN: 84-398-1971-4
Inprimategia: BOAN S.A.

1.- ARIKETA-BILDUMARAKO SARRERA

1.1. INFORMATIKA ETA PROGRAMAZIOA

Informatika Fakultatean ikasten denean lehenengo kurtsoan bi asignatura aurkitzen dira informatikari bereziki lotuta: Informatika bera eta Programazioa. Zein da bi gai hauen arteko desberdintasuna? Konputailuei buruzko ikasketak oso luzeak izanik, bi atal independiente eta konplementagarri horien artean banatu dira.

Informatikak konputailuen osagaiak deskribatzen ditu, eta beraien arteko erlazioak ere bai. Osagai horiek Prozesurako Unitate Zentrala, Memoria, Unitate Aritmetiko-logikoa eta sarrera/irteerako unitateak dira. Horrez gainera, konputailuaren egitura eta ezaugarri fisikoekin oso lotuta dauden makina-lengoaia eta mihistadura-lengoaia azaltzen dira informatikan.

Programazioak, berriz, abstraktuki hartzen du konputailua. Areago,Programazioan ez da konputailu hitza aipatzen, abstraktuagoa den prozesadore hitza baizik. Konputailua kutxa beltza bezala erabiltzen da Programazioaren ikuspegitik; erabil daitezkeen ekintzak eta objektuak definitu behar dira prozesadore bat definitzeko, baina ez dira azaltzen objektuek eta ekintzek konputailuan duten adierazpenak. Definizio horietatik abiatzen da Programazioaren lana, objektuen adierazpena eta eragiketen funtzionamendua Informatikaren eskuetan gelditzen direlarik.

1.2. PROZESADOREETARAKO DEFINIZIOA: DATU MOTAK ETA EKINTZAK

Prozesadore batek ondo definitua izateko ondoko bi deskribapenak eduki behar ditu:

a) Prozesadorerako erabilgarriak diren objektu-moten deskribapena, eta

b) Prozesadorearentzat ulergarri eta exekutagarriak diren ekin-

tza primitiboen deskribapena.

Aurrerago joan aurretik azter ditzagun bi kontzeptuok.

1.2.1. Objektu-motak. Objektuak. Konstanteak eta aldagaiak.

Objektu baten motak objektuaren berezitasunak adierazten ditu. Objektu-mota bat definitzeko hurrengo bi deskribapenok eman behar dira:

a) Motari dagozkion balio posible guztiak (heina deritzona)

b) Balio horiekin erabilgarri diren eragiketa guztiak.

Objektuak dira programatzeko elementuak, zeinen edukiak prozesadoreak alda eta beha baititzake.

Ondoko hiru atalok bereizten dute objektu bat:

Izena. Beste objektuetatik bereizteko ezaugarria

Mota. Objektuaren klasea eta berezitasunak adierazten dituena. (balio posibleak eta eragiketa erabilgarriak)

Balioa. Une konkretu batetan objektuak daukan edukia.

Objektuen artean, balioaren finkotasuna kontutan hartuz, aldagaiak eta konstanteak bereizten ditugu ondoko erara:

Objektu aldagaien balioa alda daiteke prozesadorearen ekintza bat exetutatuz.

Objektu konstanteen balioa ostera ez da aldatuko, prozesadoreak ez du balioa aldarazteko ahalmenik. Normalki, konstanteen izena balioa bera izaten da.

Ingurua prozesadoreak erabiltzen edo aldatzen duen objektu-multzoa da. Prozesadorearen erabilpen bakoitzak bere ingurua definitu behar du.

Inguruaren egoera: inguruaren objektuek une konkretu batetan dauz katen balioek definitzen dute.

1.2.2. Ekintzak.

Ekintza bat prozesadoreak burututako iraupen finituko gertaera da, ekintzaren ondorioz inguruaren ondo definituta dagoen azkeneko egoera batetara iristen da.

Ekintzak primitiboak eta ezprimitiboak izan daitezke.

Ekintza primitiboak beste xehetasunik eman gabe, berez, prozesadorearentzat ulergarriak eta exekutagarriak dira.

Ekintza ezprimitiboak, berriz, ez dira ulergarriak prozesadorearentzat; beraz, ekintza primitibozko sekuentzia bezala azalduta age ri beharko dute nonbait.

1.3. PROZESUAK, PROGRAMAK ETA ALGORITMOAK.

Prozesadore batez baliatuz, problema bat ebazten denean, problemaren enuntziatuan agertzen dira datu-objektuen balioak eta kalkula tu gura diren emaitzen definizioak.

Objektu guzti horiek inguruaren partaideak izango dira. Honelako etan, inguruaren hasierako egoeratik (datuen egoeratik) inguruaren azkeneko egoerara (emaitzak kalkulatuta dituenera) eragiten duen iga roketari prozesua deitzen zaio, eta prozesuaren deskribapenari programa deitzen zaio.

Programa bat ondoko bi atal honek osatzen dute:

a) Prozesuaren inguruan dauden objektuen erazagupena edo deklarazioa, non objektuen ezaugarriak eta objektuen motak azaltzen baitira.

b) Prozesuan burutu behar den ekintza-sekuentziaren deskribapena.

Programak programatzeko lengoaiez idazten dira. Programatzeko lengoaia asko dago, eta bakoitzak badauzka bere arauak, objektuak erazagutzeko eta ekintza-sekuentziak idazteko. Bi programa balioki de oso itxura desberdina izan dezakete, programatzeko lengoaia desberdinak erabiliz gero.

Horrexegatik, problema batentzako programa idatzi aurretik tarteko pausoa ematen da: algoritmoaren idazketa. Algoritmoa, ez da idazten programatzeko lengoaia jakin batez eta programatzeko lengoaiekin lotuta dauden arau hestuak ere, ez ditu betetzen.

Lan honetan problema bilduma bat aurkezten da, eta problema bakoitzerako algoritmoak eraikitzen dira. Algoritmoak idazteko, oso prozesadore berezia erabiliko dugu, eta baita, programatzeko, oso lengoaia berezia ere. Erabiliko diren objektuak eta ekintzak ez dira - ezein lengoaiarenak, baina edozein programatzeko lengoaia hautatuz gero, erreza izango da programaren idazketa, algoritmoa ikusi ondoren. Algoritmoen objektuek eta ekintzek programatzeko lengoaia gehienen generalizazioa izan gura dute.

2.- ALGORITMOAK DEFINITZEKO PROZESADORE OROKORRA (HASIERAKO DEFINIZIOA).

2.1. PAUSOKAKO DEFINIZIOA

Textu honetan aurkezten diren ariketak, errezenetik zailenera daude ordenatuta, hain zuzen ere, lehenengo ariketak errezenak dira objektu-mota edo ekintza gutxien erabiltzen dituztelarik, hurrengo ariketetan kontzeptu berriak sartuko dira pausoka.

Lehenengo ariketa aztertzeko, ez ditugu aurkeztuko prozesadorearen datu-mota eta ekintza guztiak,baizik eta, ariketa horretarako beharrezkoak direnak soilki.

Zailtasuna handitzerakoan, datu-mota eta ekintza gehiago erantsi ko dira.

2.2 OBJEKTU-MOTAK (LEHENENGO AURKEZPENA)

Bi objektu-mota definituko dira hasteko: osoa eta karakterea.

Lehen azaldu denez, objektu-mota bat definitzeko, balio posibleak eta eragiketa erabilgarriak definitu behar dira. Honelaxe, bada, definituko ditugu oraingo osoa eta karakterea datu-motak.

A) Osoa objektu-mota

Balio posibleak: edozein zenbaki oso. (Balio absolutua oso altua denean errealitatean eta adierazpenerako problemak direla eta, balio absolutuari muga bat jartzen zaio. Muga hau, desberdina izaten da prozesadore desberdinetan).

Eragiketak: +, -, div, mod.

eragilea	eragigaien motak	emaitzaren mota	eragiketaren izena
+ :	oso x oso	oso	batuketa arrunta
- :	oso x oso	oso	kenketa arrunta
* :	oso x oso	oso	biderkaketa arrunta

div:	oxo x oso	oso	zatiketa osoa
mod:	oso x oso	oso	zatiketa osoaren hondarra.

Batuketa, kenketa eta biderkaketa ondo ezagunak dira. div eta mod eragiketak honela definitzen dira:

div eragileak zatiketa osoa adierazten du eta ematen duen emaitza bi eragigaien arteko zatidura osoa da, hamarrenik kalkulatu gabe lortzen den zatidura alegia.

mod eragileak zatiketa osoaren hondarra lortzen du emaitzatzat.

Adibideak: 15 mod 6=3, 29 mod 10=9

Edozein a eta b-tarako beti beteko da ondoko ekuazioa

a = (a div b) * b + (a mod b)

B) Karakterea objektu-mota

Balio posibleak: komatxo artean idatzitako karaktereak edo karaktere-kateak. Prozesadore batetatik beste batetara aldakuntza txikiak izaten dira, baina gehienetan hauexek dira karaktere posibleak:

letrak: 'a','b','c','d','f','g','h','i','j', 'k','l','m','n','o','p','q','r','s','t','u', 'v','w','x','y','z'.

digituak: '0','1','2','3','4','5','6','7', '8','9'.

karaktere bereziak: '+','-','=','*','%','(', ')',' ',' ', etabar, eta komatxo artean idatzitako karaktereez osatutako sekuentziak ere bai. Adibidez: 'hitza', 'LX21', '34'

Eragiketak: oraindik ez dugu karakterea motarentzako eragiketarik. Mota honetako objektuak ez dira ageriko

adierazpenetan, beti ageriko dira bakar bakarrik.

2.3. EKINTZA PRIMITIBOAK

Hiru ekintza primitibo aurkezten dira oraingoz.

Irakurketa

Prozesu bat exekutatzeko algoritmoarekin batera balio-sekuentzia bat aurkezten da. Sekuentziaren balioak hartzeko, irakurketa ekintza erabiltzen da. Sekuentziaren elementu guztiak objektu-mota berberekoak izaten dira, bai oso motakoak, bai karaktere motakoak.

Irakurketa ondoko eredua jarraituz idazten da:

irakur (<aldagaia>)

Non aldagaia delakoak aldagai konkretu baten izena adierazten baitu.

Ekintza honek sekuentzian oraindik irakurri barik dagoen lehenengo balioa aldagaiari asignatzen dio, hau da, aldagaiaren balio berria sekuentziatik hartutakoa izango da. Beraz sekuentziako elementu bat irakurtzeko aurrean dauden guztiak aldez aurretik irakurri behar dira.

Sekuentziako elementuen objektu-motak eta aldagaiarenak, biok, berdinak izan behar dute.

Idazketa

Prozesu batek lortzen dituen emaitzak erakusteko dagoen bide bakarra idazketa da. Idazketaren bidez, prozesuaren inguruaren objektuen balioak idatz daitezke.

Ondoko eredua jarraituz idazten da ekintza hau:

idatz (<objektua>)

Non objektua delakoak aldagai konkretu baten izena edo konstante bat adierazten baitu.

Ekintza honek objektuaren balioa idazten du.

Asignazioa

Ondoko eredua jarraituz idazten da ekintza hau:

<aldagaia> : = <adierazpena>

non aldagaia delakoak aldagai konkretu baten izena adierazten bai tu eta adierazpena delakoak inguruaren objektuak (aldagai edo kons tanteak) eta azaldutako eragiketak erabiliz lortutako balio bat.

Adibideak:

```
a : = (a + b) * c
b : = b div (c + a)
a : = b
```

Parentesirik egon ezik adierazpena kalkulatzeko, lehenago kalkulatzen dira mod, div eta * eragiketak eta gero batuketa eta kenketa. Maila bereko eragiketen ebaluazioa ezkerretatik eskubitara egiten da beti.

Asignazio ekintzak lehenengoz adierazpenerako balioa kalkulatzen du, gero kalkulatutako balioa aldagaiari ezartzen dio balio berritzat. Aldagaiaren antzinako balioa galdu egiten da.

Adierazpenean agertzen diren aldagai izenak beren balioek ordezkatuak izango dira adierazpenaren balioa kalkulatzeko. Adierazpenean agertzen diren aldagaien balioak ez dira aldatuko, asignazio-ekintzako ezkerreko aldagaian bakarrik aldatuko da balioa.

2.4. ALGORITMOEN IDAZKERA

Algoritmoak idazterakoan ondoko eredua jarraituko da:

```
algoritmoa <identifikadorea>
aldagaiak  [<identifikadorea> ( <objektu-mota> )]*
hasiera
           [<ekintza> ]*
amaia
```

Non azpimarratuta dauden hitzak eta karaktereak dauden bezalaxe

idatzi behar baitira.

‹identifikadorea› delakoak letra batez hasten den hitz bat adierazten du. Algoritmoa eta aldagaiak bereizteko izenak identifikadoreak izango dira.

‹ekintza› delakoak ikusitako ekintzetariko bat adierazten du.

‹objektu-mota› delakoa karakterea edo osoa izan daiteke. Aurrerago bestelako objektu-motak ikusiko dira.

* ikurrak zera adierazten du: Mako artean ([ ]) dagoena behin edo gehiagotan ere ageri daitekeela.

Adibidez: [‹ekintza›]* adierazpidea ekintza bakar bat bezala idatz daiteke algoritmo batetan, edo elkarren segidan datozen ekintza bat baino gehiago bezala ere idatz daiteke.

3.- PROBLEMA BILDUMA

1. ARIKETA EBATZIA

ENUNTZIATUA

Irakur ezazu bi balio oso, A eta B aldagaiei asignatzeko eta gero aldagaien balioak elkarren artean trukatu.

EBAZPENA

Ariketa honen lehenengo ebazpen intuitiboa ondokoa da:

```
algoritmoa TRUKEA
aldagaiak  A, B (osoa)
hasiera    irakur (A)
           irakur (B)

           A: = B

           B: = A
           idatz (A)
           idatz (B)
amaia
```

Baina horrela A aldagaiaren balioa galtzen dugu eta amaitzean A eta B aldagaiak, biok dute balio bera: b.

Oztopo hau hobeto erakusteko, irazkin gisa aldagaien balioak idatziko ditugu algoritmoko asignazio-ekintzen aurretik eta atzetik.

```
algoritmoa TRUKEA
aldagaiak  A, B (osoa)
hasiera    irakur (A)
           irakur (B)
           { A = a, B = b }
             A: = B
           { A = b, B = b }
```

```
        B: = A
        { A = b, B = b, azken ekintzak ez du ezer aldatu }
        idatz (A)
        idatz (B)
```

amaia

A aldagaiaren lehenengo balioa gordetzeko, beste aldagai oso bat erabiliko dugu.

```
Algoritmoa TRUKE1
aldagaiak  A, B, C (osoa)
hasiera    irakur (A)
           irakur (B)
           {A = a, B = b, C = ?}
            C: = A
           {A = a, B = b, C = a}
            A: = B
           {A = b, B = b, C = a }
            B: = C
           {A = b, B = a, C = a }
           idatz (A)
           idatz (B)
```

amaia

Badago beste modu bat problema hau ebazteko. Bigarren modu honetan kalkulua erabiliko da batuketa eta kenketa hain zuzen ere.

```
Algoritmoa TRUKE2
aldagaiak  A, B (osoa)
hasiera    irakur (A)
           irakur (B)
           {A = a, B = b}
            A: = A + B
           {A = a + b, B = b }
```

```
B: = A - B
{A = a + b, B = (a + b) - b = a }
A: = A - B
{A = (a + b) - a = b, B = a }
idatz (A)
idatz (B)
```

amaia

Oharra: + eta - eragiketek emaitza ezegokiak ematen badituzte, azken algoritmo hau ez da bidezkoa izango. Kontutan har ezazu zenbaki osoak adierazteko muga badagoela, eta muga hori gaindituz gero emaitza ez litzateke zuzena izango.

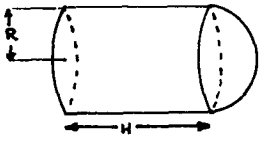
ARIKETA OSAGARRIAK

1.1 Irakur hiru balio oso A, B eta C aldagaiei emateko eta modu zirkularrez truka itzazu balioak ondoko erara:
B-ren hasierakoa balioa A-k edukiko du azkenean, hasierako C-ren balioa B-k edukiko du eta A-ren hasierako balioa C-k edukiko du

1.2 Emandako bi zenbaki osoren arteko zatidura eta hondarra kalkula itzazu. Zatiketa osoa eragiketa erabili.

1.3 Segundu-kopuru bat adierazten duen zenbaki bat irakurrita, kalkula itzazu zenbat egun, ordu, minutu eta segundu betetzen dituen.

1.4 Ondoko irudiaren bolumena kalkula ezazu altueraren eta erradioaren neurria irakurri ondoren



Irudian zilindro bat eta esferaerdi bat daude lotuta.

2. ARIKETA EBATZIA

OBJETU-MOTA ETA EKINTZA BERRIAK

Boolearra objektu-mota

Lehen azaldu denez objektu-mota bat definitzeko, balio posibleak eta eragiketa erabilgarriak definitu behar dira.

Defini dezagun boolearra objektu-mota:

Balio posibleak. Bi besterik ez dago: egiazkoa eta faltsua.

Eragiketak. Bi multzotan banatzen dira: eragiketa logikoak eta erlazio-eragiketak.

Eragiketa logikoek eragigai boolearrak dauzkate, eta honako hauek dira:

eragilea	eragigaien motak	emaitzaren mota	eragiketaren izena
edo :	boolearra x boolearra	boolearra	batuketa logikoa
eta :	boolearra x boolearra	boolearra	biderkaketa logikoa
ez :	boolearra	boolearra	ukapen logikoa

Eragiketa hauek Booleren Algebran definitzen dira.

Batuketa logikoaren eragilea bi eratara idatz daiteke: edo, $\vee$

Biderkaketa logikoaren eragilea ere, bi eratara: eta, $\wedge$

Erlazio-eragiketak honako hauek dira: $=, \neq, <, >, \leq, \geq$

Berdintasunak (=) eta desberdintasunak ($\neq$) edozein motatako bi eragigai onartzen dituzte. Txikiago ($<$), handiago ($>$), txikiago-edo-berdin ($\leq$) eta handiago-edo-berdin ($\geq$) eragiketetarako eragigaiek erlazio-ordena duten objektu-mota batetakoak izan behar dute. (Gure kasuan zenbaki osoak bakarrik, baina prozesadore askotan karaktereak ere izan daitezke).

Erlazio-eragiketen emaitzak boolearra motakoak dira.

Boolearra objektu-mota baldintzak adierazteko erabiltzen da.

Algoritmoetan sarritan erabiltzen dira baldintzak, balio batzuren arauera, tratamendu konkretu bat edo bestelako bat exekutatzeko.

Aukera ekintza primitiboa

Ondoko ereduari jarraituz idazten da ekintza hau:

aukera

B1 : E1

B2 : E2

.
.
.

BN : EN

amaukera

Non B1, B2,..., BN baldintzak baitira eta E1, E2,..., EN ekintzak.

Exekutatzeko modua ondokoa da. Dauden ordenan ebaluatzen dira baldintzak, lehenengo egiazkoa balioa lortzean jarraitzen zaion ekintza exekutatzen da eta besterik gabe aukera ekintza osoa exekutatutzat ematen da. Baldintza guztiak faltsua bezala ebaluatzen badira ezertxo ere exekutatu gabe amaitzen da aukera ekintzaren exekuzioa.

Ikusten denez, aukera ekintza erabiliz, problemaren inguruaren egoeraren arauera, hots, aldagaien balioen arauera, tratamendu desberdinak exekuta daitezke.

ENUNTZIATUA

Irakur itzazu bi zenbaki oso, bietariko handiena beste aldagai batetan gorde eta idatz gero.

EBAZPENA

Lehenengo soluzioan bi balioen arteko ordenarako dauden hiru aukerak bereizten ditugu aukeratzeko eskemaren bidez. (Datuen domeinuaren partiketa).

Algoritmoa HANDIA1

aldagaiak A, B, C (osoa)

hasiera irakur (A)

irakur (B)

{ A = a, B = b }

aukera

A > B : C := A

A = B : C := A

A < B : C := B

amaukera

{ A = a, B = b, C = max (a, b) }

idatz (C)

amaia

Bigarren soluzioan posible diren tratamendu guztiak bereizten ditugu eta bakoitza zein baldintzapetan eragin behar den zehaztu ere.

Baita tratamendu posibleak bi izango dira C := A edo C :=B.

Algoritmoa HANDIA2

aldagaiak A, B, C (osoa)

hasiera irakur (A)

irakur (B)

{ A = a, B = b }

aukera

A = B : C := A

A < B : C := B

amaukera

{A = a, B = b, C = max (a, b) }

idatz (C)

amaia

ARIKETA OSAGARRIAK

2.1. Kalkula ezazu irakurritako zenbaki oso baten balio absolutua.

2.2. Irakur itzazu hiru zenbaki oso, eta aztertu ea hirugarrena dagoen beste biek definitzen duten tartean. Eman itzazu soluzio bat baino gehiago.

2.3. 2 ariketarako ebazpena bilatu baina kalkulua erabiliz, hau da, erabil itzazu eragiketak baina aukera-ekintzarik ez. Lehengo ebazpena baina errezagoa al da? Argitsuagoa?

3. ARIKETA EBATZIA

OBJEKTU-MOTA ETA EKINTZA BERRIAK

Ekintza ez-primitiboen erabilpena

Algoritmoak idazterakoan ekintza primitibo baten ordez ekintza ez-primitibo bat idatz daiteke. Horrelakoetan jakin badakigu zer egingo duen ekintza ez-primitiboak (zelako tratamendu bete behar duen), baina geroago definitzeko uzten dugu; une horretan, ekintza ez-primitiboaren identifikadorea idazten dugu eta algoritmoarekin jarraitzen gara. Geroago, algoritmo orokorraren idazkera amaitzean, definitu gabe utzi ditugun ekintza ez-primitiboak definitu egingo ditugu banan-banan. Batzutan, algoritmoan bertan idatziko dugu ekintza ez-primitiboari dagokion ekintza primitiboen sekuentzia ekintza ez-primitiboaren izenaren ordez, baina beste batzutan algoritmoaren amai ondoren ekintza ez-primitiboaren definizioa erantsiko dugu. Ekintza ez-primitiboaren definizioa.

Ekintza hitzaz hasiko da eta ondoan ekintzen segida etorriko da hasiera eta amaia hitzen artean sartuta.

Adibidea

```
Ekintza HIRU
Hasiera
        idatz ('1')
        idatz ('2')
        idatz ('3')
amaia
```

Noiz jartzen da ekintza ez-primitiboaren izena bere ekintza primitiboen segidaren ordez eta noiz eransten zaio algoritmoari ekintza ez-primitiboaren definizioa?

Ekintza ez-primitiboaren definizioak ekintza asko ez badauka,

oso konplikatuak ez badira, eta gainera ez-primitiboa algoritmoan behin edo birritan besterik agertzen ez bada, orduan egokiagoa dirudi ordezkapenak.

Ostera, ekintza ez-primitiboaren definizioa konplikatua bada edo algoritmoan askotan aipatzen bada, orduan egokiagoa dirudi ekintza ez-primitiboaren definizioaren eransketak.

'Baldin' ekintza primitiboa

Ondoko bi ereduetako batek bezalakoa izan behar du:

baldin B orduan E1
bestela E2

baldin B orduan E

Non B baldintza, eta E, E1 eta E3 ekintzak baitira.

Lehenengo ereduaren kasuan B baldintza betetzen bada, E1 ekintza exekutatuko da, B baldintza betetzen ez bada ostera, E2 ekintza.

Bigarren ereduaren kasuan, B baldintza betetzen bada, E ekintza exekutatuko da, B baldintza betetzen ez bada 'baldin' ekintza osoa exekutatutzat ematen da ezer egin gabe.

E, E1 eta E2 ekintza bakarrak dira. Ekintza bat baino gehiago idatzi nahi bada hasiera eta amaia hitzen artean idatziko dira, eta ekintza bakar bat bezala hartuko dira.

Aukera ekintza erabiliz ere, baldin ekintzarako ikusi ditugun eredu bien ekintza baliokideak idatz daitezke.

aukera
B : E1
ez(B) : E2
amaukera

aukera
B : E
amaukera

Izatez, 'baldin' ekintza baldintza bakar batetarako aukera ekintzaren kasu partikularra besterik ez da.

ENUNTZIATUA

Irakur itzazu bi zenbaki oso A eta B aldagaietan utziz, bi balio etariko handiena B aldagaian eta txikiena A aldagaian egon daitezen eskatzen da.

A eta B-ren balioak elkarren artean trukatu beharko dira, A aldagaiarentzat irakurritako balioa handiena izanez gero.

EBAZPENA

Problemaren enuntziatua ikustean bi ekintza posible susmatzen dira: bata lehenengo ariketan bete zen elkarren arteko balio aldaketa eta bestea, ezer ez egitea problemaren eskaerak besterik egin gabe beteta daudelako. (A aldagaiarentzako balioa txikiena izanez gero).

Bi ekintza horiek noiz exekutatu behar diren adierazteko, bi partiketa edo kasu-azterketa jarrai ditzagun.

Ondoko bertsioetara garamatzate bi partiketa horiek:

Lehenengo bertsioa

Datuetarako kasu guztiak aztertzen ditugu, eta bakoitzari zein ekintza dagokion, hain zuzen, datuekiko partiketa.

Algoritmoa HANDIA1

aldagaiak A , B (osoa)

hasiera

irakur (A)

irakur (B)

{A = a, B = b}

aukera

A > B:

A= B:

A < B: ALDATU

amaukera

idatz (A)

idatz (B)

amaia

Non ALDATU izeneko ekintza ez baita primitiboa. Ekintza ez-primitiboa hau lehenengo ariketaren algoritmoaren antzekoa da.Baina oraingo ekintzan balioak ez dira irakurriko; ez eta idatzi ere HANDIA1 algoritmo orokorrean egiten baita. ALDATU ekintzak bi aldagaien balio-aldaketa besterik ez du egingo. Beraz ondokoa da:

Algoritmoa ALDATU

aldagaiak A, B, C (osoa)

hasiera

{A = a, B = b}

C := A

A := B

B := C

{A = a, B = b}

amaia

HANDIA1 algoritmoan sartu nahi badugu, C aldagaia definitu behar da.

Ondokoa da HANDIA1 algoritmoa ekintza primitiboz egina

Algoritmoa HANDIA1

aldagaiak A, B, C (osoa)

hasiera

irakur (A)

irakur (B)

```
{ A = a, B = b }
aukera
  A > B :
  A = B :
  A < B : hasiera
            C: = A
            A: = B
            B: = A
          amaia
  amaukera
      idatz (A)
      idatz (B)
amaia
```

Bigarren bertsioa

Tratamendu posible guztiak bilatzen ditugu (kasu honetan balio-aldaketa soilik), eta gero, zein baldintzapetan exekutatuko diren. Algoritmoa idazteko bide honi, tratamenduekiko partiketa deitzen zaio.

```
Algoritmoa  HANDIA2
aldagaiak   A, B (osoa)
hasiera
            irakur (A)
            irakur (B)
            {A = a, B = b }
            aukera
              A< B : ALDATU
            amaukera
              { A =(a eta b-ren arteko handiena), B =(a eta be
                  ren arteko txikiena) }
```

```
                idatz (A)
                idatz (B)
```

amaia

Baina "aukera" ekintzak baldintza bakar bat dakarrenean "baldin" ekintza bezala labur daiteke:

```
Algoritmoa  HANDIA2
aldagaiak   A, B (osoa)
hasiera
            irakur (A);
            irakur (B);
            {A = a, B = b}
            baldin A < B orduan ALDATU
            {A = a eta b-ren arteko handiena, B = b eta b-ren
            arteko txikiena}
            idatz (A)
            idatz (B)
amaia
```

HANDIA1 algoritmoarekin egin dugun bezalaxe ALDATU ekintza ez-primitiboa birfintzen dugu, eta ondokoa da lortzen den algoritmoa:

```
Algoritmoa  HANDIA2
aldagaiak   A, B, C (osoa)
hasiera
            irakur (A);
            irakur (B);
            {A = a, B = b}
            baldin A < B orduan hasiera
                                  C : = A
                                  A : = B
                                  B : = C
```

```
                    amaia
                {A = a eta b-ren arteko maximoa, B = a eta b-ren
                arteko minimoa}
                idatz (A)
                idatz (B)
    amaia
```

Oharra: Bertsio biak sortzeko abia-puntuak diferenteak izan dira. Batean datuekiko partiketatik abiatu gara, bestean tratamenduekiko partiketatik. Baina lortutako algoritmoak guztiz baliokideak dira.

Bi bide desberdinetatik lortutako baliokidetasunak algoritmoen egokitasuna baieztatzen digu.

ARIKETA OSAGARRIAK

3.1. Bigarren mailako ekuazio baten ebazpena.

(Erro karratua kalkulatzeko erabil ezazu ERROOSO eragiketa. Eragiketa honek zenbaki oso bat hartzen du eragigaitzat eta beste bat ematen du emaitzatzat. Honela izanez, emaitza ez da guztiz zehatza izango, baina oraingoan honela egingo dugu).

3.2. Azterketa baten kalifikazioa emanda (0-tik 10-erako zenbaki oso bat) kalkula ezazu dagokion maila (oso txarto (0-1-2), txarto(3-4), nahiko (5-6), ondo (7-8), bikain (9-10).

4. ARIKETA EBATZIA

ENUNTZIATUA

Irakur ezazu hiru zenbaki oso eta idatz beraietariko handiena. (Hiru zenbakiok diferenteak direla emango da).

EBAZPENA

Aurreko ariketetan bezala datuekiko partiketak eta tratamendue kiko partiketak bi bertsio diferente sortzen dute. Oraingoan, datu gehiago dagoenez, partiketa bakoitza bi partiketatan bana daiteke bi bertsio gehiago sortuz.

Lehenengo bertsioa

Datuekiko partiketa egiteko, hiru balioen arteko ordena-konbinazio guztiak aztertuko ditugu.

```
Algoritmoa  hirumax1
aldagaiak   A, B, C, MAX (osoa)
hasiera
            irakur (A);
            irakur (B);
            irakur (C);
            {A = a, B = b, C = c, c ≠ a, c ≠ b, a ≠ b}
            aukera
              (A > B) eta (B > C) : MAX : = A;
              (A > C) eta (C > B) : MAX : = A;
              (B > A) eta (A > C) : MAX : = B;
              (B > C) eta (C > A) : MAX : = B;
              (C > A) eta (A > B) : MAX : = C;
```

(C > B) *eta* (B > C) : MAX : = C;

amaukera

{MAX = a, b eta c-ren arteko handiena.}

idatz (MAX)

amaia

Bigarren bertsioa

Tratamenduekiko partiketaz ebatziko dugu.

Zeintzu dira tratamendu posibleak? Hiru: MAX : = A, MAX : = B eta MAX : = C.

Noiz erabiliko dugu zein?. MAX : = A tratamendua A aldagaiaren balioa besteenak baino handiagoa izatekotan, eta B eta C aldagaiekin berdintsu.

Hona hemen algoritmoa:

Algoritmoa HIRUMAX2

aldagaiak A, B, C, MAX (osoa)

hasiera

irakur (A)

irakur (B)

irakur (C)

{A = a, B = b, C = c, a ≠ b, a ≠ c, b ≠ c}

aukera

(A > B) *eta* (A > C) : MAX : = A;

(B > A) *eta* (B > C) : MAX : = B;

(C > A) *eta* (C > B) : MAX : = C;

amaukera

{A = a, B = b, C = c, MAX = handiena}

idatz (MAX)

amaia

Hirugarren bertsioa

Datuen arteko ordena aztertzeko, bi mailatan egiten dugu partiketa. Lehenengoa bi aldagairekin, eta gero horietako handiena eta hirugarren aldagaiarekin.

Algoritmoa HIRUMAX3

aldagaiak A, B, C, MAX (osoa)

hasiera

irakur (A)

irakur (B)

irakur (C)

{A = a, B = b, C = c, a ≠ b, a ≠ c, b ≠ c}

aukera

A > B: {emaitza a eta c-ren arteko handiena da}

aukera

A > C : MAX : = A

A < C : MAX : = C

amaukera

A < B: {emaitza b eta c-ren arteko handiena da}

aukera

B > C : MAX : = B

B < C : MAX : = C

amaukera

amaukera

{A = a, B = b, C = c, MAX = a, b eta c-ren arteko handiena}

idatz (C)

amaia

Laugarren bertsioa

Bertsio honetan ere bi mailatan egiten dugu partiketa.

Algoritmoa HIRUMAX4

aldagaiak A, B, C, MAX (oso)

hasiera

irakur (A)

irakur (B)

irakur (C)

{A = a, B = b, C = c, a ≠ b, a ≠ c, b ≠ c}

aukera

A > B : MAX : = A

A < B : MAX : = B

amaukera

{ MAX = a eta b-ren arteko handiena}

aukera

C > MAX : MAX : = C

C < MAX :

amaukera

{MAX = a, b eta c-ren arteko handiena}

idatz (MAX)

amaia

Oharra: Aukera moduko ekintzak, baldin modukoak bezala idatz litezke.

ARIKETA OSAGARRIAK

4.1. Lor ezazu MAX aldagaian A, B eta C aldagaien balioetariko handiena, eta idatz gero.

4.2. 4. ariketan emandako algoritmoak osa itzazu hiru balioak desberdinak direneko baldintza egiazta dezaten. Baldintza hau betetzen ez bada idatz ezazu "EZ DIRA DIFERENTEAK".

4.3. 4. ariketako algoritmoak berridatz itzazu baina orain ez ezazu eman balioak diferenteak direla binaka hartuta, izan ere bi edo hiru balio berdin izan bait daitezke.

Ondoko bi bideetatik egin dezakezu:

a) Idatzitako algoritmoei aukera posible berriak erantsiz.

b) Idatzitako algoritmoen gorputza ere aldatuz.

4.4. Hiru balioren ordenaketa. Irakur ezazu hiru balio eta idatz itzazu ordenaz.

4.5. Irakur ezazu hiru zenbaki eta bi txikienak idatz itzazu.

4.6. Defini ezazu bi zenbakiren biderkaduraren zeinua biderkaketa kalkulatu gabe.

4.7. Enpresa batetan ondoko erara egiten da oporraldi ordainduaren kalkulua:

Pertsona batek urte bat bete ez badu enpresan lan egiten, berak lan egindako hilebeteko bi egunerako eskubidea du, bestela 28 egun gutxienez.

Errespontsabilitate-postuan lan egiten badu, 35 urte baino gehiago baditu eta bere antzinakotasuna 3 urtekoa baino handiagoa baldin bada, orduan bi egun gehiago izango ditu. 45 urte baino gehiago baditu berriz, eta bere antzinakotasuna 5 urtekoa baino handiagoa bada, 4 egun gehiago izango ditu.

A antzinakotasuna (urtetan), B adina (urtetan) eta E, errespon-

sabilitate postua (baldintza logikoa) abiapuntutzat hartuz, J oporregun-kopurua kalkulatuko duen algoritmoa idatz ezazu.

5. ARIKETA EBATZIA

OBJEKTU-MOTA ETA EKINTZA BERRIAK

Errepika ekintza

Ondoko ereduari jarraituz, idatzi behar dira errepika moduko ekintza primitiboak:

errepika

al;

a2;

.
.
.

an

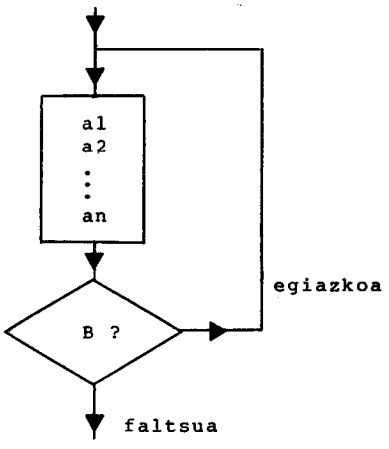
hariketa B

Non B baldintza baita eta al, a2, ..., an edintza primitiboak baitira.

Honela exekutatuko da "errepika" ekintza: Lehenengoz, al, a2, ..., an ekintzak exekutatzen dira, gero B baldintzaren ebaluaketak egiazkoa balioa ematen badu orduan "errepika" ekintza osoaren exekuzioa amaitutzat uzten da, bestela (B balditzaren ebaluaketak faltsua balioa emanez gero) al, a2, ..., an ekintza guztiak berriz exekutatzen dira eta berriz ebaluatzen da B baldintza. Honen balioaren arauera, lehen bezala, errepika ekintza amaitutzat uzten da edo al, a2, ..., an ekintza guztiak errepikatzen dira berriz.

Eta honela jarraituko da al, a2, ..., an ekintzak exekutatzean B baldintzak "egiazkoa" balioa eduki arte.

Ondoko irudiak, errepika ekintzaren exekuzioa erakusten du:



Errepika ekintzak bukaera aurki dezan, bere barruko ai ekintza batek bederen C baldintzaren ebaluaketa aldaraziko du.
Oharra: C baldintza aldagai boolear baten izen soila ere izan daiteke.

Errepika ekintza bat idazteko, ondoko puntuok hartu behar dira kontutan:

1) Zelako tratamendu egingo den errepikaketa bakoitzean hartuko den elementurako.
2) Zein den errepikaketa bukatzeko baldintza.
3) Errepika bakoitzean tratatzeko hurrengo elementua zelan hartuko den.
4) Zer egin behar den amaitu ondoren. Azken aztergaia ondo tratatzen denekoaz segurtatuz.
5) Zelan hasieratzen den errepikaketa. Lehenengo aztergaia ondo tratatzen denekoaz segurtatuz, bai eta lehenengo errepikaketa ondo exekutatzen denekoaz ere.

3. Sekuentziaren hurrengo elementua hartzeko, irakur (KAR) ekintza besterik ez da egin behar, honelaxe definituta baitago.

4. Errepikaketa amaitu ondoren ea azken elementua ondo tratatu dugun aztertu behar da, eta emaitzak idatzi egin behar ditugu.

Errepikatzeko ekintzak ondoko ordenan ezartzen baditugu, azken karakterea ere ("." karakterea) tratatu egiten da (ea 'a' karakterea den galdetzen baitugu)

```
errepika
  irakur (KAR)
  baldin KAR = 'a' orduan KONTA : = KONTA + 1
harik-eta KAR = '.'
```

Alderantziz ezartzen baditugu baldintza irakurketa aurrean karakterea ez da tratatzen, baina bost axola zaigu emaitza ez bait litzateke aldatuko ('.' karakterea ez da inoiz 'a' karakterea izango). Hots, ondoko bigarren ordena hau ere egokia da:

```
errepika
  baldin KAR = 'a' orduan KONTA : = KONTA + 1
  irakur (KAR)
harik-eta KAR = '.'
```

Aurreko bi kasuetan idatz (KONTA) erantsiko dugu lortutako emaitza idatz arazteko.

5. Hasieran KONTA aldagaiari 0 balioa asignatu behar diogu eta lehenengo karakterearen tratamendua ondo egiten delakoaz segurtatu behar gara.

Errepika ekintzarako aurreko puntuan eraiki dugun lehenengo aukeran ez dago hasieratzeko problemarik, lehenengo karakterea errepika ekintzan irakurtzen eta tratatzen delako, gainera sekuentziak puntua soilki eta ez beste karaktererik daukaneko kasuan ere emaitza ona (0 balioa) ematen duelako.

ENUNTZIATUA

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

Ebazpena

Sekuentziako karaktere guztiak aztertu beharko ditugu. Sekuentziako elementuak banan banan aztertzeko "errepika" ekintza erabiliko dugu. "Errepika" ekintza ondo definitzeko ondoko bost puntuoi erantzun behar diegu:

1- Zelako tratamendu egingo den errepikaketa bakoitzeko elementurako.

2- Zein den errepikaketa bukatzeko baldintza.

3- Zelan hartuko den tratatzeko hurrengo elementua errepikaketa bakoitzean.

4- Zer egin behar den amaitu ondoren.

5- Zelan hasieratzen den errepikaketa.

Erantzunak

1. Karaktere bakoitzerako zera egin behar dugu: aztertu ea 'a' letra den eta hala izanez gero, bat gehiago kontatu. Baina zelan kontatuko dugu? Errez: oso motako aldagai bat erabiliko dugu, hasieran zero balioa emango diogu aldagaiari eta 'a' letra bat agertzen denean, bat gehituko diogu balioari. Beraz, tratamendua ondokoa da:

baldin KAR = 'a' orduan KONTA : = KONTA + 1

non KAR aldagaiak aztertu behar den karakterea baitauka eta KONTA aldagaia kontatzekoa baita. KONTA aldagaiari 0 balioa asignatu beharko diogu "errepika" ekintza hasi baino lehenago.

2. Amaitzeko baldintza KAR = '.' izango da, '.' karaktereak sekuentziaren amaia adierazten du eta.

```
Algoritmoa  AKONTA1
aldagaiak   KAR (karakterea)
            KONTA (osoa)
hasiera
            KONTA : = 0;
            errepika { KONTA-k ageritako 'a' letran kopurua
              dauka }
              irakur (KAR)
              baldin  KAR = 'a' orduan KONTA : = KONTA + 1
            harik-eta KAR = '.';
            {KONTA-k sekuentzia osoan ageritako 'a' letren
            kopurua dauka}
            idatz (KONTA)
amaia
```

Errepika ekintzarako aurreko puntuan eraiki dugun bigarren aukeran osterantzean, lehenengo karaktereak aurretik irakurria izan behar du, eta sekuentziak puntua soilki eta ez beste karaktererik daukaneko kasurako tratamendu berezia sartu behar dugu, bestela, KONTA aldagaiak 1 balioa hartuko luke eta egongo ez den beste karaktere bat irakurri nahi izango da. Beraz ondokoa da algoritmoaren bigarren bertsioa:

```
Algoritmoa  AKONTA2
aldagaiak   KAR (karaktere)
            KONTA (osoa)
hasiera
            KONTA : = 0
            irakur (KAR)
```

```
baldin KAR ≠ '.'  orduan {sekuentzia ez da hutsa
                          Puntuaz gainera badu bes
                          te karaktererik.}
                  errepika
                    baldin KAR = 'a' orduan
                    KONTA ; = KONTA + 1
                    irakur (KAR)
                  harik-eta KAR = '.'
idatz (KONTA)
amaia
```

ARIKETA OSAGARRIAK

5.1. Puntu karaktere batez amaitzen den karaktere-sekuentzia bat emanik, zenbat karaktere dauzkan kontatu hurrengo kasuetan.

a) Puntua kontatuz.

b) Puntua kontatu barik.

5.2. Aurretik definitutako karaktere-sekuentzian 'A' letra ez diren karaktereak kontatu. Zuriguneak ere kontatuz.

a) Puntua kontatuz.

b) Puntua kontatu barik.

5.3. Karaktere-sekuentziaren bokaleak kontatu. Ondoren aldatu algoritmoa kontsonanteak konta ditzan.

5.4. Karaktere-sekuentzian konta ezazu zenbat 'a', zenbat 'e', zenbat 'i', zenbat 'o' eta zenbat 'u' dagoen.

5.5. Zero batez amaitzen den zenbaki osozko sekuentzia bat emanik, zenbat zenbaki oso negatibo dagoen konta.

5.6. Irakur ezazu zenbaki bat eta kalkula ezazu beraren faktoriala.

6. ARIKETA EBATZIA

ENUNTZIATUA

Puntu karaktereaz amaitzen den karaktere-sekuentzia batetan kalkula ezazu 'A' letren maiztasun erlatiboa hots, 'A' letraren agerpen-kopurua zati letra guztien arteko agerpenak. Eman portzentaia. (Puntuaz eta zuriguneez aparte ez dago beste karaktere berezirik).

EBAZPENA

Algoritmorako lehenengo hurbilketa ondokoa izan daiteke:

```
Algoritmoa  MAIZERL
aldagaiak   AKONTA, KONTA, PORTZENTAIA (osoa)
            KAR (karakterea)
hasiera
            akontatu;
            {AKONTA-K sekuentzian dauden 'a' letren kopurua
            dauka}
            kontatu;
            {KONTA-K sekuentzian dauden letren kopurua dauka}
            PORTZENTAIA : = (100 * AKONTA) DIV KONTA;
            idatz (PORTZENTAIA)
amaia
```

Non AKONTATU ekintza ez-primitiboa aurreko ariketaren AKONTA1 eta AKONTA2 algoritmoek definitzen baitute, eta KONTATU ekintza ez-primitiboa AKONTATU ekintzaren antzekoa baita. Kontatzeko baldintza besterik ez da aldatu behar, kar = 'a' baldintzaren ordez, (kar ≠ ' ') eta (kar ≠ '.') baldintza idatzi behar delarik.

Bigarren kontaketa egiteko, aldez aurretik sekuentzia hasteko moduan jarri beharko genuke. Eskakizun hori betetzeko ekintza primitibo berri bat erabiltzen da: Sekuentzia-hasiera. Ekintza hau exekutatuta, irakurriko den hurrengo aldagaiari sekuentziako lehenengo balioa asignatuko zaio.

Algoritmoaren azken hurbilketarako bi bertsio idatz daiteke. Batean AKONTATU eta AKONTA ekintza ez-primitiboen definizioa algoritmoaren azkenean erantsiz eta bestean ekintza ez-primitibo bezala definitu gabe, algoritmoaren lehenengo hurbilketan bertan ekintza horien ordez beren definiziak ezarriko ditugu.

Lehenengo bertsioa, ekintza ez-primitiboak definitzen dituena, honakoa da:

```
Algoritmoa  MAIZERL1
aldagaiak   AKONTA, KONTA, PORTZENTAIA (osoa)
            KAR (karakterea)
hasiera
            AKONTATU;
            { AKONTA-k sekuentzian dauden 'a' letren kopurua
            dauka }
            Sekuentzia-hasiera
            KONTATU
            {KONTA-k sekuentzian dauden letren kopurua dauka }
            PORTZENTAIA : = (100 * AKONTA) DIV KONTA
            idatz (PORTZENTAIA)
amaia
ekintza AKONTATU
hasiera AKONTA : = 0
        errepika { AKONTA-k ageritako 'a' letren kopurua dauka }
                 irakur (KAR)
                 baldin kar = 'a' orduan konta ; = konta + 1
        harik-eta kar = '.';
        {AKONTA-k sekuentzia osoan ageritako 'a' letren kopu-
        rua dauka }
amaia
ekintza AKONTATU
hasiera  KONTA : = 0
        errepika  {KONTA-k ageritako letren kopurua dauka}
                  irakur (kar)
                  baldin (kar ≠ ' ') eta (kar ≠ '.') orduan
                  KONTA : = KONTA + 1
```

```
        harik-eta kar = '.'
        {KONTA-k sekuentzia osoan ageritako letren kopurua
        dauka}
    amaia
```

Ohar zaitez sartu dugun SEKUENTZIA-HASIERA ekintzaz.

Bigarren bertsioa, ekintza ez-primitiboen ordez beren definizioak ezartzen dituena, ondoko algoritmoan aurkezten da:

```
    Algoritmoa  MAIZERL2
    aldagaiak   AKONTA, KONTA, PORTZENTAIA (osoa)
                KAR (karakterea)
    hasiera
                AKONTA : = 0
                errepika {AKONTA-k ageritako 'a' letren kopurua
                          dauka
                        irakur (kar)
                        baldin kar = 'a' orduan AKONTA : = AKONTA + 1
                harik-eta kar = '.';
                {KONTA-k sekuentzian osoan ageritako 'a' letren ko-
                purua dauka}
                sekuentzia-hasiera;
                KONTA : = 0
                errepika {KONTA-k ageritako letren kopurua dauka}
                        irakur (kar)
                        baldin kar= 'a' orduan KONTA : = KONTA + 1
                harik-eta kar = '.'
                {AKONTA: zenbat 'a' letra; KONTA: zenbat letra
                PORTZENTAIA : = (100* AKONTA) DIV KONTA
                idatz (PORTZENTAIA)
    amaia
```

Asmaturiko algoritmoak guztiz definituta daude. Baina ondoko sinplifikazio handiaz ohar gaitezke: kontaketa biak batera egin daitezke, kalkulatzen dituzten balioak independienteak dira eta, izan ere, bat kalkulatzeko ez dugu behar bestearen balioa aldez aurretik kalkulatuta edukitzea, kontaketa biak batera egiteko alda keta txikia egingo dugu aurreko ariketaren AKONTA1 algoritmoan.

Karaktere bakoitzerako tratamenduan, bi kontaketenak sartu behar ditugu, beraz:

baldin kar = 'a' orduan AKONTA : = AKONTA + 1

baldin (kar ≠ '.') eta (kar ≠ ' ') orduan KONTA : = KONTA + 1

AKONTA eta KONTA aldagaiei 0 balioa asignatuko diegu hasiera.

Errepikaketa amaitu ondoren maiztasun erlatiboaren portzentaia lortu behar da KONTA eta AKONTA aldagaien balioak erabiliz.

```
Algoritmoa  MAIZERL 3
aldagaiak   AKONTA, KONTA, PORTZENTAIA (osoa)
            KAR (karakterea)
hasiera
            AKONTA : = 0;
            KONTA : = 0;
            errepika
                    irakur (kar);
                    baldin (kar = 'a') orduan AKONTA : = AKONTA + 1
                    baldin (kar ≠ '.') eta (kar ≠ ' ') orduan
                    KONTA : = KONTA + 1
            harik-eta kar = '.'
            PORTZENTAIA : = (100 * AKONTA) DIV KONTA
            idatz (PORTZENTAIA)
amaia
```

ARIKETA OSAGARRIAK

6.1. Zero batez amaitzen den oso-sekuentzia baten zenbaki osoen batezbesteko aritmetikoa kalkula ezazu.

6.2. Kalkula ezazu oso-sekuentziaren zenbaki positiboen batezbesteko aritmetikoa.

6.3. Bi zenbaki osoren biderkadura eta zatidura osoa kalkula itzazu, baina batuketa eta kenketa soilik erabiliz. Zenbaki oso positiboak eta 0 baino handiagoak hartu lehenengoz

6.4. Karaktere-sekuentzia batetan azter ezazu ea zein letra agertzen den gehiagotan, 'A' letra edo 'E' letra.

6.5. Eraiki ezazu ea zenbaki oso bat lehena den aztertuko duen algoritmo bat.

6.6. Oso-sekuentzia bat emanik idatz itzazu zenbakien faktorialak.

6.7. Zero batez amaitzen den oso-sekuentzia bat emanik, konta ezazu zenbat zenbaki diren sekuentziaren batezbesteko aritmetikoa baino handiagoak.

7. ARIKETA EBATZIA

ENUNTZIATUA

Puntu karaktereaz amaitzen den karaktere-sekuentzia batetan konta ezazu zenbat aldiz agertzen den 'TA' karaktere-bikotea.

EBAZPENA

Sekuentzia bat tratatu behar dugunez "errepika" ekintza bat erabili beharko dugu. Tratamendu errepikakor hau ondo definitzeko 5. ariketan azaldutako galderei erantzun behar diegu.

1. Zelako tratamendua egingo dugu elementu bakoitzerako?

Karaktere bat irakurtzen dugunean ea 'TA' karaktere-bikotea ageri den jakiteko aurreko karaktereari buruzko informazioa behar dugu, hain zuzen ere, ea aurreko karakterea 'T' karakterea izan den.

AURREKOAT aldagai boolearra erabiliko dugu. 'Egiazkoa' balioa badauka aurreko karakterea 't' izan dela jakingo dugu; bestela, 'faltsua' balioa badauka, aurreko karakterea ez da 't' izan.

Beraz, karaktere bakoitzerako bi aldagai eguneratu behar dugu, bata 'TA' karakterea bikotearen agerpen-zenbakiarena (azken irakurritako eta aurreko karaktereen arauera) eta bestea AURREKOAT, zeinean azken irakurritako karakterea ea 't' karakterea izan den gordeko baitugu, hurrengo karakterearen tratamenduan erabilia izateko.

Tratamendu oso honetarako bi bertsio aurkezten dugu. Lehenengoan irakurritako karakterearen balio esanguratsuen multzoaren partiketan oinarrituz, eta bigarrenean 'TA' karaktere-bikotearen kontatzailearen balioa eta AURREKOAT aldagaiarena aldatzeko baldintzak adieraziz. Lehenengoa datuekiko partiketa da eta bigarrena tratamenduekiko partiketa.

2. Zein da amaitzeko baldintza?

Karaktere-sekuentzia puntuaz amaitzen denez, errepikaketa amaitzeko baldintza kar = '.' izango da aurreko ariketetan bezala.

3. Zelan hartuko dugu sekuentziaren hurrengo elementua?

Karaktere-sekuentzia denez, kar karaktere motako aldagaia irakurriz hurrengo karakterea hartuko dugu.

4. Zer egin behar da amaitu ondoren?

Azken karakterea '.' delarik, azkeneko bikotea inoiz ere ez da 'TA' bikotea izango. Beraz azken karakterearen tratamendua ez da beharrezkoa.
'Errepika' ekintza amaitzean emaitza idatzi beharko dugu. Kasu honetan 'TA' karaktere-bikotearen agerpen-kopurua.

5. Zelan hasieratzen dugu errepikaketa?

'TA' bikotearen agerpenen kontatzaileari 0 balioa eman behar diogu.

AURREKOAT aldagaiari faltsua balio boolearra emango diogu lehenengo karakterearekin inoiz ere ez baita bikotea lortuko.

1. Bertsioa

Lehenengo bertsioan irakurritako karaktere esanguratsuen multzoaren partiketan oinarritzen gara:

Algoritmoa TAKONTATU1

aldagaiak KAR (karakterea)
AURREKOAT (boolearra)
ZB (osoa)

```
hasiera     irakur (KAR)
            baldin KAR = '.'
                 orduan idatz ('textu hutsa')
                 bestela {textua ez da hutsa}
                  hasiera
                    ZB : = 0;
                    AURREKOAT : = faltsua;
                    errepika
                            {kar : aztertzeko karakterea
                            ZB : 'AT' bikotearen agerpen-ko
                            purua.
                            AURREKOAT : ea aurreko karakterea
                            't' izan den }
                    aukera {kar aldagaiaren balio esanguratsu
                           en gaineko partiketa }
                      KAR = 't' : AURREKOAT : = egiazkoa
                      KAR = 'a' : baldin AURREKOAT orduan
                                          hasiera
                                            ZB  := ZB + 1
                                            AURREKOAT : = faltsua
                                          amaia
                      (KAR ≠ 'a') ∧ (KAR ≠ 't'): AURREKOAT:=faltsua
                    amaukera
                      irakur (KAR)
                 harik-eta kar = '.';
                 idatz (ZB)
            amaia {textua hutsa ez deneko tratamendua}
amaia
```

2. Bertsioa

Bigarren bertsioan tratamendu desberdin posibleak azaltzen ditugu, eta berorien arteko zein exekuta behar den jakiteko baldintzak ezartzen dizkiegu datuei, hots, tratamenduen arauera egiten dugu partiketa.

```
Algoritmoa  TAKONTATU2
aldagaiak   KAR (karakterea)
            AURREKOAT (boolearra)
            ZB (osoa)
hasiera
            irakur (KAR)
            baldin KAR = '.'
              orduan idatz ('textu hutsa')
              bestela {textua ez da hutsa}
                hasiera
                  ZB : = 0
                  AURREKOAT : = faltsua
                  errepika { kar: aztertzeko karakterea
                             ZB: 'AT' bikotearen agerpen-kopurua
                             AURREKOAT: ea aurreko karakterea 't'
                             izan den. }
                    { noiz aldatuko da ZB-ren balioa?
                      eta noiz aldatuko da AURREKOAT-en balioa? }
                    baldin AURREKOAT eta (KAR='a') orduan ZB:=ZB+1
                    baldin KAR = 't' orduan AURREKOAT: = egiazkoa
                                    bestela AURREKOAT: = faltsua
                    irakur (KAR)
                harik-eta KAR = '.'
                idatz (ZB)
    amaia {textua hutsa ez deneko tratamendua}
amaia
```

3. Bertsioa

Hirugarren bertsio bat ere asmatuko dugu. Oraingoan tratamendu sekuentzialaz ohartzen gara. 'TA' karaktere-bikoteak kontatzea eta 'A' karaktereak kontatzea antzekoak dira.

'A' karaktereak kontatzeko, sekuentziako karaktere guztiak aztertzen ditugu eta 'A' denean kontatu egiten dugu. Bikoteak kontatzeko ez dago berdin egiterik? saia gaitezen.

'A' karaktereak kontatzeko ondoko algoritmoa erabiltzen genuen:

```
Algoritmoa  AKONTA2
aldagaiak   KAR (karakterea)
            KONTA (osoa)
hasiera
            KONTA: = 0
            irakur (KAR)
            baldin KAR ≠ '.' orduan {sekuentzia ez da hutsa}
                                errepika
                                  baldin KAR = 'a' orduan
                                  KONTA: = KONTA + 1
                                  irakur (KAR)
                                harik-eta KAR = '.'
            idatz (KONTA)
amaia
```

Saia gaitezen antzeko algoritmoa idazten baina karaktere-bikoteak kontatzeko. Karaktere-bikoteak era bereziaz irakurriko eta tratatuko dira, baina eman dezagun egiteko posibleak direla eta gerorako utziko ditugu beraien definizioa, oraindik ekintza ez-primitiboak erabiliko ditugu tratamendu horiek adierazteko eta eman dezagun ere bikote bakoitzaren adierazpena aldagai sinpleak erabiliz lortuko dugula baina hau ere gerorako utziko dugu.

Bikoteak sekuentzialki tratatzeko algoritmoa hauxe litzateke:

```
Algoritmoa  TAKONTA3
aldagaiak   KONTA (osoa)
            {oraindik ez dakigu zelan adieraziko ditugun bikoteak}
hasiera
            KONTA: = 0
            irakur-lehenengo-bikote
            baldin bikotea-azkena-ez-bada
                  orduan {sekuentzia ez da hutsa}
                    errepika
                      baldin bikotea-'TA'-bada orduan
                      KONTA : = KONTA + 1
                      irakur-bikote
                  harik-eta bikotea-azkena-da
            idatz (KONTA)
amaia
```

Algoritmoa definituta, bikoteetarako adierazpidea aurkitu behar da, eta gero algoritmoan dauden ekintza ez-primitiboak eta baldintza bereziak adierazpidearen arauera idatzi.

Bikoteetarako adierazpidea karaktere motako bi aldagaien bidez lor daiteke. KAR1 aldagaiak bikotearen lehenengo karakterea edukiko du eta KAR2 aldagaiak bigarren karakterea.

Azal ditzagun algoritmoan agertzen diren ekintza ez-primitiboak eta baldintza bereziak adierazpide honen araueraz, KAR1 eta KAR2 aldagaiak erabiliz.

a) irakur-lehenengo-bikote ekintza

Intuitiboki bi karakteretako irakurketak errezena ematen du.

```
irakur (KAR1)
irakur (KAR2)
```

baina sekuentzia hutsa deneko kasua kontutan hartzeko, puntu

bakar bat dagoenekoa, alegia, karaktere bat bakarra irakurriko da, KAR2 aldagaiari asignatuz, eta bikotearen lehenengo karakterea emaitza aldatuko ez duen bat jarriko da, esate baterako ' ' zurigunea.

Beraz honela gelditzen da ekintza:

ekintza irakur-lehenengo-bikote

hasiera

KAR1 : = ' '

irakur (KAR2)

amaia

b) bikotea-azkena-ez-bada baldintza

Azken bikotea puntura iristerakoan ageriko da eta beraz bikotearen bigarren karakterea puntua izango da. Baldintza ondokoa izango da:

$KAR_2 \neq$ '.'

c) bikotea-'TA'-bada baldintza

(KAR_1 = 'T') eta (KAR_2 = 'A')

d) irakur-bikote ekintza

Hurrengo bikotea irakurtzeko, ez dira hurrengo bi karaktereak irakurri beharko, horrela eginez gero tartean dagoen karaktere-bikotea aztertzeke geldituko baitlitzateke. Adibidez

ETA ETXEETARA JOAN ZIREN.

datu-sekuentzia balitz eta binakako irakurketa eginez 'ET', 'A ' 'ET', 'XE', 'ET', ..., bikoteak aztertuko genituzke baina ez: 'TA', ' E','TX', etabar.

Aurreko bikotearen bigarren karakterea, KAR2 aldagaiak daukana hain zuzen, bikote berriaren lehenengoa izango da, eta bikote berriaren bigarren karakterea sekuentziatik irakurtzekoa da.

ekintza irakur-bikote

```
hasiera
        KAR1 : = KAR2
        irakur (KAR2)
amaia
```

Kontuz ibili behar da aurreko asignazioa eta irakurketa idazterakoan, alderantziz idazten badira irakur (KAR2)) lehenengoz eta KAR1 : = KAR2 gero, aldagai biek sekuentziako karaktere bera edukiko dute eta.

Algoritmoan zeuden ekintza ez-primitiboak eta baldintzak idazteko ez dugu aurkitu gaindiezinezko oztoporik aukeratutako adierazpiderako. Hona hemen bikoteak sekuentzialki tratatzeko algoritmoa:

```
Algoritmoa  TAKONTA3
aldagaiak   KONTA (osoa)
            KAR1, KAR2 (karakterea) {bikoteak adierazteko}
hasiera
            KONTA: = 0;
            KAR1 : = ' ';
            irakur (KAR2)
            baldin KAR2 ≠ '.' orduan { sekuentzia ez da hutsa}
                                errepika
                                  baldin (KAR1 = 'T') eta
                                 (KAR2  = 'A')
                                    orduan KONTA: = KONTA + 1
                                  KAR1 : = KAR2
                                  irakur (KAR2)
                                harik-eta KAR2  = '.'
             idatz (KONTA)
amaia
```

7. ARIKETA OSAGARRIAK

7.1. Berrartu 'TA' karaktere-bikotearen agerpen-kopurua kontatzeko algoritmoaren bertsioak, baina ondoko enuntziatu aldaketak kontutan hartuz:

a) 'TA' eta 'TI' karaktere-bikoteak konta itzazu.

b) 'TA' eta 'MA' karaktere-bikoteak konta itzazu.

c) Konta itzazu sekuentzian dagoen lehenengo karaktere-bikotearen agerpenak.

7.2. Sekuentzia batek puntuz amaitutako esaldi bat dauka. Esaldia zurigune batez edo gehiagoz banatutako hitzez osatzen da. Hitza karaktere zuririk ez daukan karaktere-sekuentzia da. Azkeneko puntua zurigune baten edo gehiagoren atzetik ere etor daiteke. Sekuentziaren hitzak kontatzeko eskatzen da.

7.3. Berrartu ezazu aurreko ariketa eta eman ezazu hitzak letrez soilki osatzen direla, beste edozein karakterek hitzaren amaia adierazten duela alegia.

7.4. Letra bakar batez osatutako hitzak konta itzazu.

7.5. Zeroaz amaitzen den zenbaki osozko sekuentzia batetan, hurrengo zenbakiok kalkulatzeko algoritmoak sor itzazu.

- Zeinu-aldaketen
- Monotonia gorakorra

7.6. Froga ezazu TAKONTATU1 eta TAKONTATU2 algoritmoak baliokideak direla Horretarako:

-1.2 algoritmoko baldin ekintzak aukera ekintzak bezala adieraziko dira.

- eta lortutako bi aukera ekintzetatik aukera ekintza bakar bat lortuko da.

8. ARIKETA EBATZIA

OBJEKTU-MOTA ETA EKINTZA BERRIAK

BITARTEAN ekintza primitiboa

Ekintza primitibo berri honek, ondoko eredua jarraitzen du:

bitartean c egin

a1;

a2;

.
.
.

an

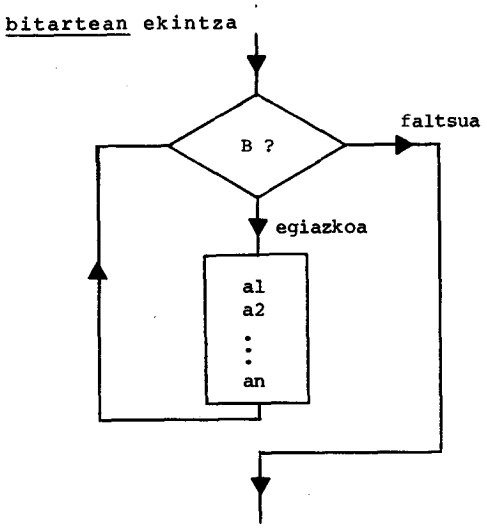
ambitartean

Non C letrak baldintza bat adierazten duen eta a1, a2, ..., an direlakoek ekintzak.

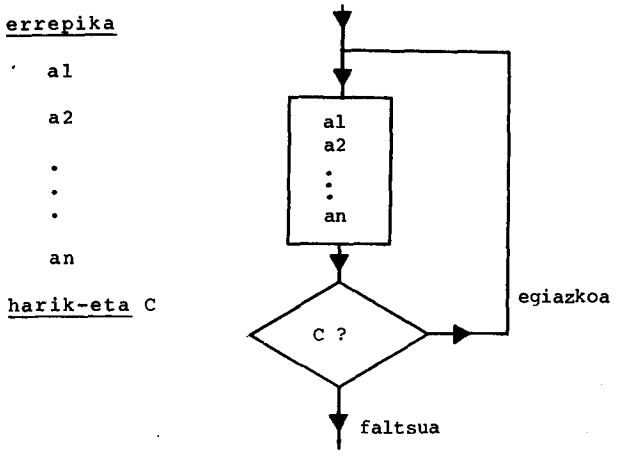
Honela exekutatuko da bitartean ekintza:

c baldintza egiazkoa bezala ebaluatzen den bitartean, a1, a2, ..., an ekintza-multzoa exekutatuko da. Hau da, lehenengoz c baldintza ebaluatzen da, faltsua bezala ebaluatzekotan behin ere ez da exekutatuko ekintza-multzoa, eta algoritmoaren hurrengo ekintza, "am bitartean" hitz mugatzailearen ostean datorrena alegia, exekutatuko da; bestela, baldintza egiazkoa bezala ebaluatzen bada, a1, a2, ..., an ekintza guztiak exekutatuko dira eta ondoren berriz ebaluatuko da c baldintza eta tratamendua hasierakoa bezelaxekoa izango da.

Ondoko irudiak exekuzioa adierazten du:



Konpara ezazu errepika ekintzaren eskemarekin



Errepika ekintzan a1, a2, ..., an errepikatzeko ekintzak gutxienez behin batean exekutatzen dira, bitartean ekintzan ordea, hasieran C baldintzak faltsua balioa badauka behin ere ez dira exekutatzen.

Errepika ekintzarekin gertatzen zen bezala, bitartean ekintzan ere c bukatzeko baldintzaren ebaluaketa aldarazteko ekintzaren bat

egon behar da errepikatzen direnen artean, bestela c baldintza hasi eran betetzen ez bada (normala izaten denez) ez da hurrengoan ere beteko eta bitartean ekintza ez da inoiz ere amaituko.

Bitartean moduko ekintza bat idazteko, bost puntu hartu beharko dugu kontutan, hain zuzen, errepika ekintza bat idazteko erabiltzen genituen puntu berberaiek:

1. Zelako tratamendu egingo den errepikaketa bakoitzeko elementurako.

2. Zein den errepika segitzeko baldintza. (Kasu honetan ez da bukatzekoa).

3. Zelan hartuko den tratatzeko hurrengo elementua errepikaketa bakoitzean.

4. Zer egin behar den amaitu ondoren.

5. Zelan hasieratzen den errepikaketa.

ENUNTZIATUA

Puntu karaktere batez amaitzen den karaktere-sekuentzia bat emanik, zenbat karaktere dauzkan konta ezazu.

a) Puntua kontatu barik.

b) Puntua kontatuz.

EBAZPENA

a) Puntu kontatu barik.

Karaktere-sekuentzia bat tratatu behar da, ez dakigu zenbat karaktere tratatu behar diren, amaitzeko baldintza soilki ezagutzen dugu. Beraz, ekintza errepikakor bat erabili behar dugu, bai errepika ekintza, bai bitartean ekintza. Ariketa honetan bitartean erabiliko dugu, baina bietako edozein erabil zitekeen.

Bitartean ekintza definitzeko, errepikatzeko xehetasunak azter-

tu behar ditugu. Azter ditzagun puntua ez kontatzeko kasurakoak.

. Zelako tratamendu egingo da elementu bakoitzerako?

Puntua ez bada, kontatu egingo dugu. Hasieran kontadore bat de klaratu eta hasieratu egin behar da.

. Zelan hartuko da errepikaketa bakoitzerako hurrengo aztergaia?

Karaktere-sekuentziatik karaktere bat irakurriko dugu.

. Zein den amaitzeko baldintza?

Puntu karakterea agertzea, sekuentziaren azkena adierazten du eta. Bitartean ekintzan kontrako baldintza hartzen da, kasu honetan errepikaketa jarraitzeko baldintza da eta.

Oraindaino ondoko eskema idatz daiteke:

```
bitartean   Kar ≠ '.'  egin
                Konta: = konta + 1
                irakur (kar)
ambitartean
```

.Zer egin behar da amaitu ondoren? Aztergaia ondo tratatzen denekoaz segurtatu.

Amaitu ondoren kontadorearen balioa idatziko da.

Azken karakterea, puntua hain zuzen ere, ez da tratatzen, horixe gura genuen.

. Zelan hasieratzen da errepikaketa?

Lehenengo karakterea aurretik irakurri behar da, baldintzan balio duntzat ematen baita.

Kontadoreak zero balioa eduki behar du hasi aurretik. Lehenengo errepikaketan 1 balioa hartuko du.

Sekuentzia hutsaren kasuan ere algoritmoa egokia da. Zero idazten da eta lehenengo karakterea soilki irakurtzen da ez besterik.

Hona hemen algoritmo osoa:

```
Algoritmoa PUNTUAEZ1  {karakterea kontatu puntua ezik}
aldagaiak  KONTA (osoa)
           KAR (karakterea)
hasiera
           KONTA:=0
           irakur(KAR)
           bitartean KAR ≠ '.' egin
                KONTA : = KONTA + 1;
                irakur (KAR)
           ambitartean
                idatz (KONTA)
amaia
```

b) Puntua kontatuz

Kasu honetan algoritmoaren eraikiketa lehengoarena bezalakoa da azken karakterearen tratamenduari dagokiona ezik. Puntu karakterearen kontaketa bitartean ekintzatik kanpoan egin behar da. Azken elementuaren tratamendu bezala. Ondokoa izango da algoritmoa:

```
Algoritmoa  PUNTUAERE 1 {Karaktereak kontatu, puntua ere bai}
aldagaiak   KONTA (osoa)
            KAR (karakterea)
hasiera
            KONTA : = 0
            irakur (KAR)
            bitartean  KAR ≠ '.' egin
                 KONTA : = KONTA + 1
                 irakur (KAR)
            ambitartean
              KONTA : = KONTA + 1 {Azken karakterearen tratamendua}
              idatz(KONTA)
amaia
```

Errepika ekintza erabiliz ere idatz daitezke problema hauetarako algoritmoak. Ondokoak lirateke errepika ekintza erabiliz lortutako algoritmoak:

```
Algoritmoa  PUNTUAEZ2  {karaktereak kontatu puntua ezik}
aldagaiak   KONTA (osoa)
            KAR (karakterea)
hasiera
            KONTA = 0
            irakur (KAR)
            baldin KAR = '.' orduan idatz ('sekuentzia hutsa')
                              bestela
                                    errepika
                                      KONTA : = KONTA + 1
                                      irakur (KAR)
                                    harik-eta KAR = '.'
            idatz (KONTA)
amaia

Algoritmoa  PUNTUAERE2  {karaktereak kontatu, puntua ere}
aldagaiak   KONTA (osoa)
            KAR (karakterea)
hasiera
            KONTA : = 0
            errepika
                 irakur (KAR)
                 KONTA : = KONTA + 1
            harik-eta  KAR = '.'
            idatz (KONTA)
amaia
```

Ariketa honetarako proposaturiko algoritmoak ikusita, zera esan daiteke:

Tratamendu errepikakorrak adierazteko, errepika ekintza edo bitartean ekintza erabil daitezke. Kasu batzutarako errepika ekintza era biltzen duen algoritmoak sinpleago ematen du, adibidez, PUNTUAERE1 algoritmoak PUNTUAERE2 baino luzeagoa da, lehenengo karakterea eta azkena era berezi batez tratatzen baitira.

Beste kasutan alderantziz gertatzen da eta bitartean ekintza era biltzen duen algoritmoa errepika ekintza baino sinpleago ematen du, puntu bakar batek osatutako sekuentzia hutsaren kasu bereiztu behar delako.

8. ARIKETA OSAGARRIAK

8.1. Puntuaz amaitzen den karaktere-sekuentzia bat emanda, idatz eza zu sekuentzian agertzen den lehenengo hitza, zurigunerik ez diren karaktereek osatzen duten lehenengo azpisekuentzia hain zuzen.

8.2. Ordena goranzkorrez ordenatutako oso-sekuentzia bat emanda, konta ezazu zenbat oso diren 27 baino txikiagoak.

9. ARIKETA EBATZIA

ENUNTZIATUA

Irakur ezazu zenbaki oso bat eta idatz itzazu beraren zatigai guztiak.

EBAZPENA

N zenbaki osoaren zatigai guztiak ezagutzeko, 1-etik N-raino dauden zenbakiak aztertu behar ditugu. Zenbaki hauek ez dira sekuentziatik irakurriak izango, sekuentzian N zenbakia bera besterik ez dago eta. Baina N zenbakia ezagututa 1-etik N-raino dauden zenbakiak sor daitezke. Horretarako, I aldagai oso bati hasieran 1 balioa ematen diogu eta ekintza errepikakor bat erabiliz, I aldagaiaren balioari 1 gehituz, bilatutako zenbakiak lortuko dira banan banan N zenbakia lortu arte.

Lortutako zenbaki bakoitzerako tratamendua ondokoa izango da: baldin I zenbakia N zenbakiaren zatigaia bada, orduan idatzi I zenbakia, bestela ez idatzi.

Zatigaitasuna aztertzeko baldintza zera da: N mod I=0.

Azter ditzagun ekintza errepikakorra idazteko puntuak.

(bitartean adibidez).

1. Zein da elementu bakoitzerako tratamendua?

Idazketa, N-ren zatigaia baldin bada, beraz:

baldin N mod I=0 orduan idatz (I)

2. Zelan hartzen da hurrengo errepikaketarako elementua?

I aldagaiari bat gehituz. (Ez da irakurtzen).

I: = I + 1

3. Zein da errepikaketak amaitzeko baldintza?

I aztergaia N zenbakia baino handiagoa izatea.

I > N

Bitartean ekintza erabiltzen badugu, errepikatzen jarraitzeko baldintza bilatu behar da.

I < = N

4. Zelan hasieratzen da errepikaketa?

I aldagaiari hasierako balioa emanez.

I : = 1

5. Zer egin behar da errepikaketak amaitzean?

Kasu honetan ezertxo ere ez. Zatigaiak idazteko eskatzen da eta idatzita daude guztiak, I aldagaiak (N+1) balioa dauka ekintza errepikakorra amaitzean eta berori ez da inoiz ere zatigaien artean egongo.

Beraz, algoritmoa honako hau da:

```
Algoritmoa  ZATIGAIAK
aldagaiak   N (osoa)  {zenbakia irakurtzeko}
            I (osoa)  {letik N-rainoko zenbakiak gordetzeko}
hasiera
            irakur (N)
            I : = 1
            bitartean I < = N egin
              {I baino txikiagoak diren N-ren zatigaiak idatzita
              daude.}
              {I zenbakia aztertu:}
              baldin N mod I = 0 orduan idatz (I)
              {Hurrengo zenbakia lortu:}
              I : = I + 1
              ambitartean
amaia
```

9. ARIKETA OSAGARRIAK

9.1. Irakur ezazu zenbaki oso bat. Demagun n izan dela. Fibonacci-ren n-garren zenbakia kalkula ezazu. Hona hemen Fibonacciren zenbakien definizioa:

$$FIB(N) = \begin{cases} 1 & \text{baldin } n = 0 \\ 1 & \text{baldin } n = 1 \\ FIB(N-1) + FIB(N-2) & \forall n > 1 \end{cases}$$

9.2. Idatz itzazu a(n) = (2 * (n-1)) div 3 gai generala duen segidaren gaiak, baina 1000 baino txikiagoak direnak besterik ez.

10. ARIKETA EBATZIA

OBJEKTU-MOTAK ETA EKINTZA BERRIAK

TAULA objektu-mota

Orain arte definitutako objektu-motak zerak dira: osoa, karaktere a eta boolearra. Hirurok bakunak dira, hau da, mota horietako aldagai batek balio bakar bat dauka inguruaren egoera bakoitzean.

Orain difinituko dugun TAULA objektu-motako aldagai batek, ordea balio bat baino gehiagok osatua dago, aldagai-multzo batek alegia. Bakunak ez diren objektu-motei egituratuak deritze.

TAULA objektu-mota egituratua da. TAULA motako den aldagai bat osatzen duten aldagai guztiak mota berberekoak dira, eta indize batek ordenatuta daude.

TAULA motako aldagai bat deklaratzeko ondoko hiru puntuok definitu behar dira:

- Taularen izena.
- Osagaien kopurua (indizeetarako tartea definituz)
- Osagaien mota.

Adibidez: KALIFIKAZIOAK (10 osagai osotako taula)
DEITURAK (4 osagai karakteretako taula)

Taularen osagai bat identifikatzeko, taularen izena eta osagaiari dagokion indizea idatzi behar dira.

Adibidez:

KALIFIKAZIOAK(6) delakoak KALIFIKAZIOAK taularen seigarren aldagaia adierazten du.

DEITURAK(2) delakoak DEITURAK taularen bigarren aldagaia adierazten du.

Taulako aldagai bat identifikatzen duen indizea konstante osoa, aldagai osoa edo balio osoa ematen duen adierazpena izan daiteke.

Adibidez:

KALIFIKAZIOAK(I) Taularen edozein osagai izan daiteke I aldagaiaren balioaren arauera.

I aldagaiak une batetan 6 balioa badu KALIFIKAZIOAK(I)-ik taularen 6. osagaia adierazten du.

KALIFIKAZIOAK(2*I-3) berdin; I aldagaiaren balioaren arauera aldagai bat edo beste bat adierazten du.

I aldagaiak une batetan 3 balioa badu KALIFIKAZIOAK(2*I-3) delakoak taularen 3. osagaia adierazten du.

Objektu-motak definitzeko, balio posibleen multzoa eta balioekin posible diren eragiketak definitu behar ziren. Defini dezagun taula objektu mota:

Balio posibleen multzoa osagaien objektu-motarena berbera da, baina taulak aldagai bat baino gehiago daukanez, horrenbeste balio edukiko du.

Eragiketa posibleak, osagaiak erabiliz egin daitezkeenak berberaiek dira. Ez dugu definitzen taula osoa eragigaitzat hartzen duen eragiketarik. Taula osatzen duten aldagaiak beren motako edozein aldagai bezalaxe erabiliko dira.

ENUNTZIATUA

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, atzetik aurrera idatz ezazu, baina puntua azken tokian utzirik. Puntu aurrean dagoen azken karakterea lehenengoa bezala idatziko da, sekuentziaren azkenaurrekoa bigarrena bezala idatziko da, eta honela jarraituz sekuentziaren lehenengoa idatzi arte. Sekuentziak

ezein kasutan ere ez dauka 100 karaktere baino gehiago.

EBAZPENA

Sekuentzia baten karaktere guztiak tratatu behar direnez, ekintza errepikakor bat erabili beharko da, errepika edo bitartean. Baina zein izango da karaktere bakoitzerako tratamendua? karakterea idatzi? Ez, horrela eginez gero ordena berean idatziko lirateke sekuentziaren karaktereak eta kasu honetan, alderantziz idatzi behar dira. Taula baten erabilpena oso egokia da problema honetarako, sekuentziaren karaktere guztiak taulako aldagaietan gorde baitaitezke, gero atzekoz aurrera karaktere guztiak idatziko direlarik.

Algoritmorako lehenengo hurbilpena honakoa da:

Algoritmoa ALDERANTZIZ
aldagaiak GORDELEKU (100 osagai karakteretako taula)
KONTA (osoa) {zenbat karaktere gorde diren jakiteko}
hasiera
TEXTUAGORDE
{GORDELEKU taulako aldagaiek textua daukate}
{KONTA aldagaiak gordetako karaktereen kopurua dauka}
IDATZ ALDERANTZIZ
amaia

Algoritmoaren lehenengo pausoan sekuentziaren karaktere guztiak gordetzen dira. Bigarren pausoan, alderantziz idazten dira.

Textua gordetzeko, banan banan irakurriko dira karaktereak eta bakoitza taulako aldagai batetan gordeko da. Bitartean ekintza batez egingo dugu.

KAR aldagaia karaktereak irakurtzeko erabiliko dugu.

I aldagaia, taulako aldagaiak bereizteko erabiliko dugu indize

bezala.

KONTA aldagaia, karaktereak kontatzekoa da, gero idazteko lehena zein den jakiteko balio du.

Ekintza errepikakorrean tratatu behar diren elementuak sekuentziaren karaktereak dira.

Karaktere bakoitzerako tratamendua taulan gordetzean datza, oraindik erabili ez den lehenengo aldagaian. Eta gainera karakterea kontatu.

Hurrengo karakterea hartzeko sekuentziatik irakurri behar dugu.

Errepikaketa amaitzeko baldintza, '.' karakterea irakutzea da.

Orain arte ondoko ekintza daukagu.

```
bitartean  KAR ≠ '.' egin
        I : = I + 1
        GORDELEKU(I)   : = KAR
        KONTA : = KONTA + 1
        irakur (KAR)
ambitartean
```

Lehenengo karakterea ondo tratatzeko, errepikaketa, hasieratzeko KAR, I eta KONTA aldagaien beren hasierako balioa hartu behar dute. KAR aldagaiak sekuentziaren lehenengo karakterearen irakurketaren bidez. I aldagaiak 0 balioa hartu behar du I : = I + 1 errepikaketan exekutatu ondoren GORDELEKU(I) taularen lehenengo aldagaia bezala ulertua izan dadin, KONTA aldagaiak ere 0 balioa hartuko du karaktere-kontaketaren hasierapenari dagokionez. Ondokoak dira hasieratzeko ekintzak.

```
irakur (KAR)
I : =0
KONTA : = 0
```

Sekuentziak '.' karakterea bakarrik daukaneko kasuan karaktere hori besterik ez da idatziko (errorea sortuko litzateke eta), eta KONTA aldagai kontatzailea 0 balioarekin gelditzen da nahi zen bezala.

Sekuentziaren azken karakterea, '.' hain zuzen, ez da kontatzen, ezta gordetzen ere, nahi zen bezala exekutatu egiten da. Errepikaketa bukatzean ez da beste ekintzarik exekutatu behar.

Hona hemen TEXTUAGORDE ekintza ez-primitiboa.

```
ekintza     TEXTUAGORDE
Aldagaiak   KAR (karakterea)  {karaktereak irakurtzeko}
            I (osoa)         {taularako indizea}
            KONTA (osoa)     {karaktereak kontatzeko}
hasiera
            irakur (KAR)
            I : = 0
            KONTA: = 0
            bitartean  KAR ≠ '.'
                       egin  {KAR aztertzeko karakterea}
                             {I azken erabilitako aldagaiaren indizea}
                             {KONTA gordetako karaktereen kopurua}
                         I: = I + 1
                         GORDELEKU(I) : = KAR;
                         KONTA: = KONTA + 1
                         irakur (KAR)
            ambitartean
amaia
```

Textua alderantziz idazteko, beste ekintza errepikakor bat erabiliko da, tratatzeko elementuak GORDELEKU taularen aldagaiak izango direlarik. Tratamendua balioa idaztekoa da. Hurrengo aldagaia eskura

tzeko, indizea aldatu behar da; textua alderantziz idatzi gura denez indizearen balioari bat kendu eta kitto.

Amaitzeko baldintza GORDELEKU1 aldagaiaren balioa idatzita edukitzea da, beraz, idatzi ondoren I-ren balioari bat kentzen zaionez I indize-aldagaiak 0 balioa ez badauka errepikatzen jarraitu behar da.

Hasieratzeko, I indize-aldagaiari idazteko lehenengo aldagaiaren indizea eman behar zaio. Kasu honetan KONTA aldagaiaren balioa, bertan kontatu ditugu karaktereak eta.

Errepikaketa amaitzeko ez dago ekintza berezirik, eta azken karakterea ondo idazten denekoaz ere segurtatzen gara.

```
ekintza     IDATZALDERANTZIZ
aldagaiak   I (osoa)
hasiera     I: = KONTA
            bitartean  I >= 1
              egin  {I: Idaztekoa den aldagaiaren indizea}
                    idatz (GORDELEKU(I))
                    I: = I - 1
            ambitartean
amaia
```

ALDERANTZIZ algoritmoan, algoritmo orokorrean, TEXTUAGORDE eta IDATZALDERANTZIZ ekintzen ordez beren deskribapenak jartzen baditugu ondoko algoritmoa lortuko dugu:

```
Algoritmoa  ALDERANTZIZ
aldagaiak   GORDELEKU (100 osagai karakteretako taula)
            KONTA (osoa)  {zenbat karaktere gorde diren jakiteko}
            KAR (karakterea) {sekuentziaren karaktereak irakurtzeko}
            I (osoa)      {taularako indizea}
hasiera
            irakur (KAR);
            I: = 0;
            KONTA: = 0;
            bitartean   KAR ≠ '.'
              egin { KAR: aztertzeko karakterea}
                   {I: azken erabilitako aldagaiaren indizea}
                   {KONTA: gordetako karaktereen kopurua }
              I: = I + 1;
              GORDELEKU(I) : = KAR;
              KONTA: = KONTA + 1
              irakur (KAR)
            ambitartean
              I: = KONTA
            bitartean   I >: 1
              egin  {I: Idaztekoa den aldagaiaren indizea }
                    idatz (GORDELEKU(I))
                    I: = I - 1
            ambitartean
amaia
```

Konta eta I aldagaien erabilpenak aztertuz gero, beren balioki-detasunaz kontura gaitezke. Bietako bat bakarra erabiliko dugu, beraz. KONTA aldagaiaren helburua gordetako karaktereen kopurua jasotzea zen, baina balio hori I aldagaiaz ere lor daiteke, balio berdinak dauzkate eta. Beraz, sinplifikazioa egin daiteke ALDERANTZIZ

algoritmoan, ondoko moduan gelditzen delarik:

```
Algoritmoa  ALDERANTZIZ
aldagaiak   GORDELEKU (100 osagai karakteretako taula)
            KAR (karakterea)  {sekuentziaren karaktereak irakurtzeko}
            I (osoa)          {taularako indizea eta karaktere-kontadorea}
hasiera
            irakur (KAR)
            I: = 0
            bitartean  KAR ≠ '.'
              egin     {Kar: aztertzeko karakterea}
                       {I; azken erabilitako aldagaiaren indizea}
                I: = I + 1
                GORDELEKU(I) : = KAR;
                KONTA: = KONTA + 1
                irakur (KAR)
            ambitartean
              {I aldagaiak irakurritako karaktereen kopurua du}
              bitartean  I>= 1
                egin     { I: Idaztekoa den aldagaiaren indizea}
                  idatz (GORDELEKUA(I))
                  I: = I - 1
              ambitartean
amaia
```

10. ARIKETA OSAGARRIAK

10.1. Taula datu-mota erabili gabe, asma ezazu 10. ariketa ebatzirako beste ebazpen bat.

10.2. Zenbaki osozko taula batetan balio txikiena bila ezazu.

10.3. Karaktere-sekuentzia bat digitoz osatuta dago ('0'tik, '9'rainoko karakterez) eta puntu karaktereaz amaitzen da. Digito-sekuentziak adierazten duen zenbaki osoa idatz ezazu, baina zenbaki oso bezala. Bi ebazpen asma:

a) Taula datu-motaz baliatuz.

b) Taularik erabili gabe.

10.4. 100 balio oso daukan DATOS izeneko taula ezagututa kalkula itzazu ondoko emaitzak:

1) Zenbat dira 27 baino txikiagoak.

2) 100 balioen batezbesteko aritmetikoa.

10.5. Puntu karaktereaz amaitzen den karaktere-sekuentzia bat ea kapikua den azter ezazu. (Kapikua baldin bada berdin irakurtzen da , aurreraka zein atzeraka).

11. ARIKETA EBATZIA

ENUNTZIATUA

Puntu karaktereaz amaitzen den karaktere-sekuentzia batetan letrez eta zuriguneez aparte beste karaktererik ez dago. Idatz ezazu sekuentzia hori, baina ondoan azaltzen den kodearen arauera,

'A' letrak 'D' bezala idatziko dira

'B' letrak 'E' bezala idatziko dira

'C' letrak 'F' bezala idatziko dira

. .
. .
. .

'Z' letrak 'C' bezala idatziko dira

Zuriguneak ez dira aldatuko.

EBAZPENA

Algoritmoaren eraikipenaren zailtasuna zatitzeko, demagun lehenengo hurbilketa batetan KODETU ekintza ez-primitiboa daukagula. Ekintza beronek KARKODE aldagaian jarriko du KAR aldagaiaren balioari dagokion kodea.

KODETU ekintza edukiz gero, karaktere-sekuentziaren kodeketa erreza da. Bitartean ekintza errepikakor baten bidez sekuentziaren karaktere guztiak tratatuko dira, non karaktere bakoitzerako tratamendua bere kodearen idazketa izango baita. Puntua ez da kodetzen baina idatzi egin behar da.

Ondokoa da algoritmoa:

```
Algoritmoa  KODESEKU
aldagaiak   KAR (karakterea)  {Sekuentziatik irakurtzekoa}
            KARKODE (karakterea) {Karaktere kodetua gordetzekoa}
hasiera
            irakur (KAR)
            bitartean  KAR ≠ '.'  egin
              KODETU  { KARKODE aldagaiak KAR karaktereari dago
                        kion karakterea hartu du }
              idatz (KARKODE)
              irakur (KAR)
              ambitartean
              idatz ('.')
amaia
```

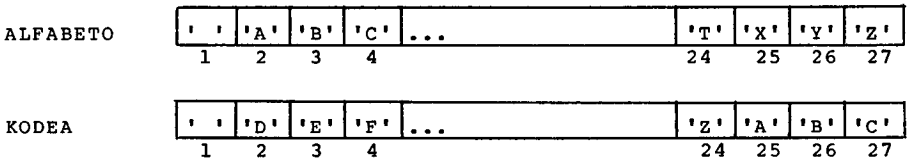
Orain KODETU ekintza ez-primitiboa deskribatu behar da beste algoritmo baten bidez, noski. Hiru bertsio aurkeztuko dugu. Lehenengoan bi taula erabiliko dira, bata alfabetoko karaktereak gordetzeko eta bestea karaktereei dagozkien kodeetarako. Bigarren eta hirugarren bertsioetan taula bakar bat erabiliko da kodetzeko arauak zerikusi handia baitu ordena alfabetikoarekin, hain zuzen, taulan hiru leku atzerago dagoen karakterea da karaktere bakoitzari dagokion kodea.

LEHENENGO BERTSIOA

Lehenengo bertsioko taula batetan zurigunea eta letra guztiak gordetzen dira lehenengo taulako karaktereei dagozkien karaktereak.

Hots, lehenengo taulako I aldagaian dagoen karaktereari dagokion kodea, bigarren taulako I aldagaian dagoen karakterea da.

Ondoko irudian ikusten dira bi taulok:



KODETU ekintza ezprimitiboak KAR aldagaiak duen karakterea ALFABETO taulan bilatuko du. Aurkitzean, KARKODE aldagaiari asignatuko dio KODEA taulan indize berbera duen aldagaiaren karakterea. Adibidez, KAR aldagaiak 'T' letra badauka, ALFABETO(24) aldagaian aurkituko du eta KARKODE-ri KODEA(24) aldagaiaren balioa emango dio, 'Z' karakterea hain zuzen.

ALFABETO taulan KAR aldagaiaren balioa bilatzeko ekintza errepikakor bat erabiliko dugu, non ALFABETO taulako aldagaiak banaka hartuko baitira bilatutako karakterea aurkitzen ez den bitartean.

ekintza KODETU

aldagaiak ALFABETO (27 karakteretako taula) { letrak eta zurigunea edukitzeko }
KODEA (27 karakteretako taula) { ALFABETO-ko karaktereetarako kodeak }
I (osoa) {ALFABETO eta KODEA tauletarako indizea}

hasiera

```
ALFABETO: = 'ABCDEFGHIJKLMNOPQRSTUVXYZ'

KODEA: =    'DEFGHIJKLMNOPQRSTUVXYZABC'

{ 1.- KAR karakterearen bilaketa ALFABETO taulan }
I :=1
bitartean  KAR <> ALFABETO(I) egin
I := I + 1
ambitartean
```

ambitartean

{ 2.- I-ren balioaren erabilpena KARKODE kalkulatzeko}

KARKODE: = KODEA (I)

amaia

ALFABETO eta KODEA taulei balioa emateko, adierazpen laburtua erabili dugu. Ondokoa da edintza-multzo baliokidea

```
ALFABETO: = 'ABCDEFGHIJKLMNOPQRSTUVXYZ'
  ALFABETO(1) : = ' '
  ALFABETO(2) : = 'A'
  ALFABETO(3) : = 'B'
     .              .
     .              .
     .              .
  ALFABETO(26) : = 'Y'
  ALFABETO(27) : = 'Z'
```

KODETU ekintza ezprimitiboa algoritmoan bere izenaren ordez idazten badugu, honako hau litzateke algoritmo osoa:

```
Algoritmoa  KODESEKU1
aldagaiak   KAR (karakterea)   { Sekuentziatik irakurtzekoa }
            KARKODE (karakterea) { Karaktere kodetua gordetzekoa}
            ALFABETO (27 karakteretako taula)
                     { letrak eta zurigunea edukitzeko }
            KODEA (27 karakteretako taula)
                     { ALFABETO-ko karaktereetarako kodeak}
            I (osoa) { ALFABETO eta KODEA  tauletarako indizea }
hasiera
            ALFABETO: = 'ABCDEFGJIJKLMNOPQRSTUVXYZ';
            KODEA: =    'DEFGHIJKLMNOPQRSTUVXYZABC';
```

```
irakur (KAR);
bitartean KAR ≠ '.' egin
  {KAR karakterea bilatu ALFABETO taulan}
  I: = 1
  bitartean KAR ≠ ALFABETO(I) egin
    I: = I + 1
  ambitartean
  {I-ren balioaren erabilpena KARKODE kalkulatzeko}
  KARKODE: = KODEA(I)
  idatz (KARKODE)
  irakur (KAR)
ambitartean
  idatz ('.')
```

amaia

ALFABETO taularen karaktereek ez dute ordena alfabetikoz egon behar. Beraz, bilaketarako denborak txikitu gura badira ALFABETO-ko lehenengo aldagaietan karaktere arruntenak, gehien agertzen direnak, gordeko dira. Baina, kontuz, kasu horretan, KODEA taula ere aldatu beharko da.

Bigarren bertsioa

Bigarren bertsioan taula bakar bat erabiltzen da, kodetzeko araua ordena alfabetikoarekin zerikusi handia duela aprobetxatuz. Taulako aldagaietan letrak ordena alfabetikoan ezartzen baditugu, letra baten kodea aurkitzeko, lehenengoz taulan bilatu behar dugu letra, eta dagokion kodea hiru aldagai atzerago dagoen letra izango da. Azkeneko hiru letretarako kodeak, taulako lehenengo hiru letren artean aurkituko dira. Zurigunerako kodea zurigunea bera da

eta beste modu batetara kalkulatu behar da, oraingoan zurigunea ez baita egongo taulan.

Hona hemen KODETU ekintzarako bigarren bertsioa:

```
ekintza      KODETU
aldagaiak    ALFABETO (26 karakteretako taula)
             P (osoa) {taulan bilatzeko indizea}
             Q (osoa) {bilatu nahi den kodearen taulako indizea}
hasiera
             ALFABETO: = 'ABCDEFGHIJKLMNOPQRSTUVXYZ'

             baldin KAR = ' ' orduan  KARKODE: = ' '
                   bestela
                     hasiera
                       {KAR karakterearen bilaketa}
                       P:= 1
                       bitartean KAR ≠ ALFABETO(P) egin
                         P: = P + 1
                       ambitartean
                        { KAR = ALFABETO(P)   }
                        { kodearen kalkulaketa, P erabiliz}
                         baldin P  23 orduan  Q: = P - 23
                                      bestela Q: = P + 3
                         KARKODE: = ALFABETO(Q)
                     amaia
amaia  { KODETU}
```

KODETU ekintza ezprimitiborako bigarren bertsio hau KODESEKU algoritmoan idazten badugu, honako hau litzateke algoritmo osoa:

```
Algoritmoa  KODESEKU2
aldagaiak   KAR(karakterea)  {sekuentziatik irakurtzekoa}
            KARKODE (karakterea) {Karaktere kodetua gordetzekoa}
            ALFABETO (26 karakteretako taula)
                  {letrak edukitzekoa}
            P (osoa)  {taulan bilatzeko indizea}
            Q (osoa)  {bilatu nahi den kodearen taulako indizea}
hasiera
            ALFABETO: = 'ABCDEFGHIJKLMNOPQRSTUVXYZ'

            irakur (KAR)
            bitartean  KAR ≠ '.'egin
              baldin  KAR = ' ' orduan  KARKODE = ' '
                              bestela
                                hasiera
                                  {KAR karakterearen bilaketa}
                                   P: = 1
                                  bitartean KAR ≠ ALFABETO(P)egin
                                    P: = P + 1
                                  ambitartean
                                  {KAR = ALFABETO(P) }
                                  {Kodearen kalkulaketa, P erabiliz }
                                  baldin P  23 orduan Q:=P - 23
                                               bestela Q:= P + 3
                                  KARKODE: = ALFABETO(Q)
                                amaia
            idatz (KARKODE)
            irakur (KAR)
```

```
        ambitartean
        idatz ('.')
amaia
```

Hirugarren bertsioa

Bigarren bertsioko taulari A, B eta C karaktereak eransten badizkiogu, azkeneko hiru karaktereen kodeak bilatzeko, beste karaktere guztienak kalkulatzen diren bezala kalkula daitezke. Kasu honetan azken hiru karaktereen kodea taulan hiru leku atzerago aurkituko da.

Ondokoa da algoritmoa:

```
ekintza     KODETU
aldagaiak   ALFABETO (29 karakteretako taula)
            P (osoa)  {KAR karakterea taulan bilatzeko indizea}
            Q (osoa)  {bilatu nahi den karaktererako taulako
                       indizea}
hasiera
            ALFABETO: ='ABCDEFGHIJKLMNOPQRSTUVXYZ'

            baldin KAR = ' ' orduan KARKODE: = ' '
                           bestela
                              hasiera
                                {KAR karakterearen bilaketa}
                                P: = 1
                                bitartean KAR ≠ ALFABETO(P) egin
                                  P:= P+1
                                ambitartean
                                  {KAR = ALFABETO(P)     }
```

```
                                {Kodearen kalkulaketa,
                                P erabiliz }
                                Q: = P + 3
                                KARKODE: = ALFABETO(Q)
                            amaia
    amaia
```

KODETU ekintza ezprimitiborako hirugarren bertsioa KODESEKU algoritmoan idatziz honako algoritmo hau lortzen dugu:

```
    Algoritmoa  KODESEKU3
    aldagaiak   KAR (karakterea) {sekuentziatik irakurtzekoa}
                KARKODE (karakterea) {karaktere kodetua gordetzekoa}
                ALFABETO (29 karakteretako taula)
                        {letrak eduketzekoa }
                P (osoa) {taulan bilatzeko indizea}
                Q (osoa) {bilatu nahi den kodearen taulako indizea}
    hasiera
                ALFABETO: ='ABCDEFGHIJKLMNOPQRSTUVXYZ'

                irakur (KAR)
                bitartean  KAR <> '.' egin
                  baldin KAR = ' ' orduan  KARKODE := ' '
                                   bestela
                                     hasiera
                                       {KAR karakterearen bilaketa}
                                       P: = 1
                                       bitartean KAR <> ALFABETO(P) egin
                                         P := P + 1
```

```
                                    ambitartean
                                    { KAR = ALFABETO(P) }
                                    { Kodearen kalkulaketa, P
                                    P erabiliz          }
                                    Q := P + 3
                                    KARKODE := ALFABETO(Q)
                                amaia
                        idatz (KARKODE)
                        irakur (KAR)
                        ambitartean
                    idatz ('.')
            amaia
```

11.- ARIKETA OSAGARRIAK

11.1. Berrartu 11. ariketa, baina edozein kodetarako balio dezan, hots, kodea adierazteko ez da arau errazik egongo.

Esaterako: ABCDEFGHIJKLMNOPQRSTUVWXYZ'

JCEWZALBCQNMVUDYTSOLHGFIKP

eta zurigunea ez da aldatzen.

11.2. Berrartu 11. ariketa, baina ondoko aldaketokin:

a) Zifrarik ere egon daiteke karaktere-sekuentzian, baina zifrak ez dira aldatuko.

b) Sekuentzian edozein karaktere ager daiteke, baina letrak soilki kodetzen dira, beste karaktere guztiak berdin idazten dira.

11.3. Karaktere-sekuentzia batek letrak, zuriguneak eta azken puntua dauzka. Konta ezazu zenbat aldiz agertzen den letra bakoitza.

11.4. Puntuaz amaitzen den karaktere-sekuentzia batetan konta ezazu zenbat aldiz agertzen diren sekuentzian dauden karaktereak.

Sekuentzian ez dauden karaktereak ez dira kontatuko.

11.5. Zeroaz amaitzen den zenbaki osozko sekuentzia bat emanik, konta ezazu zenbat amaitzen diren zifra bakoitzarekin, hau da zenbat amaitzen diren zeroaz, zenbat amaitzen diren bataz... zenbat amaitzen diren bederatziaz.

12. ARIKETA EBATZIA

ENUNTZIATUA

Puntu karaktereaz amaitzen den karaktere-sekuentzia bat emanda, konta ezazu zenbat aldiz agertzen den lehenengo hitza. Sekuentzian letrak eta zuriguneak besterik ez dago.

Oharrak:

- Lehenengo hitzaren lehenengo letra, baldin badago, sekuentziaren lehenengo karakterea izan daiteke edo zurigune batzuren atzean ere, etor daiteke.
- Azken hitzaren azken karaktereak, baldin badago, zuriguneak eduki ditzake atzean edo puntua bakarrik.
- Hitzak banatzeko zurigune bat edo gehiago egon daiteke'

EBAZPENA

Enuntziatu honetarako algoritmoa asmatzeko, beheranzko eraikuntza izeneko metodologia erabiliko dugu.

Problema batetan erabili behar diren datuak asko edo anizkunak (egituratuak) badira, edo deskribatu behar diren tratamenduak zailak badira, egokia izaten da beheranzko eraikuntza deritzon metodologiaz baliatzea.

Datuekiko edo tratamenduekiko zailtasun handia dagoenean arterga asko eduki behar da kontutan, eta gainera guztiak batera, hutsak egiteko arriskua handia delarik. Beheranzko eraikuntzak pausoka edo zatika lortu gura du algoritmoa, problematik azpiproblema batzu atereaz.

Problemaren zati zailak, ondo definitzen dira azpiproblemak bezala, eta problema nagusirako algoritmo osoa eraikita dagoenean, orduan ebatziko dira azpiproblemak. Problema gehiago izango dira,

baina berauen ebazpenak errazagoak izango dira, eta gainera, azpiproblemak ondo definituta badaude azpiproblema bakoitzaren ebazpena beste azpiproblemen ebazpena kontutan hartu gabe eraiki daiteke. Laburtuz, problema bakar bat ebazteko problema asko sortzen dugu, baina problema berriak errazagoak dira problema iturria baino.

Algoritmoaren definizioa ondokoa da: Inguruaren hasierako egoeratik (datuen egoeratik), inguruaren azkeneko egoerara (emaitzak kalkulatuta daudeneko egoera) igarotzearen deskribapena.

Hau da, problema bat ebatziko duen algoritmo bat definitzeko ondo definitu behar dira hasierako eta azkeneko egoerak. Bi egoera horien artean dauden bitarteko egoerak definitzean datza beheranzko eraikuntza. Azpiproblema bakoitza bitarteko egoera batetik hurrengora daraman igaroketa bezala definituko da.

Igaroketa honen deskribapena ekintza ezprimitibo bezala definituko dugu.

Hasierako problemaren zailtasun handiaren ordez, bere azpiproblemen zailtasun txikiagoak izango ditugu. Hori dela eta, metodologia honek beheranzko eraikuntza izena hartu zuen.

Oraingo enuntziatuko sekuentziaren lehenengo hitza zenbat aldiz agertzen den kontatzeko, azaldutako problema bi azpiproblematan bana daiteke.

1- Lehenengo hitza gorde (baldin badago)

2- Lehenengo hitzari jarraitzen zaion sekuentzia-zatia tratatu.

Algoritmoa LEHENHITZA

aldagaiak KAR(karakterea) {Sekuentziako karaktereak irakurtzeko}

LEHENKONTA (osoa) {Lehenengo hitzaren agerpenak kontatzeko}

LEHENA (20 karakteretako taula) {Lehenengo hitza gordetzeko}

LUZERA (osoa)

hasiera

irakur (KAR)

{letra bat, zurigunea edo puntua dauka KAR aldagaiak}

ZURIGUNEAK-JAUZI

{KAR aldagaiak letra bat edo puntua dauka}

baldin KAR = '.' *orduan* idatz ('ez dago hitz bat ere')

bestela

hasiera { KAR aldagaiak lehenengo hitzaren lehenengo letra dauka}

LEHEN-HITZA-GORDE

{ KAR aldagaiak lehenengo hitzari jarraitzen dion zurigunea edo puntua dauka}

{LEHENA eta LUZERA-k lehenengo hitza adierazten dute}

LEHENA-KONTATU

{LEHENKONTA, lehen hitzaren agerpen-kopurua dauka lehenengo hitza kontatu barik}

{KAR aldagaiak azkeneko puntua dauka}

idatz (LEHENKONTA + 1)

amaia {bestelakoa}

amaia {LEHENHITZA}

Non ondo definitu behar baititugu erabili diren ekintza ezprimitiboak (azpiproblemak). Hona hemen definizioak:

ZURIGUNEAK-JAUZI ekintza

Karaktereak irakurtzen dira KAR aldagaiak zurigunea ez daukan arte.

Hasierako egoera: KAR aldagaiak zurigunea, letra bat edo puntua dauka.

Azkeneko egoera: KAR aldagaiak letra bat edo puntua dauka.

LEHEN-HITZA-GORDE ekintza

Hartu letrak sekuentziatik zurigunea edo puntua ageri arte eta gorde karaktereak LEHENA taulan. Asignatu LUZERA aldagaiari gordetako karaktereen kopurua.

Hasierako egoera: KAR aldagaiak lehen hitzaren lehen letra dauka edo puntua.

Azkeneko egoera: KAR aldagaiak lehen hitzari amaia ematen dion puntua edo zurigunea dauka.
LEHENA taulak lehen hitza dauka.
LUZERA zenbat karaktere gorde diren LEHEN taulan.

LEHENA-KONTATU ekintza

Ekintza honek kontatzen du lehen hitza zenbat aldiz agertzen den, sekuentziatik lehenengo hitza kenduz gelditzen den sekuen-

tzia-zatian.

Hasierako egoera: KAR aldagaiak lehen hitzari amaia ematen dion puntua edo zurigunea dauka.
LEHEN eta LUZERA aldagaiek lehen hitza adierazten dute.

Azkeneko egoera: LEHENKONTA lehen hitzaren agerpen kopurua dauka lehenengo hiza bera kontatu barik

Orain banan banan hartu eta ebatziko ditugu hiru azpi-problemak.

ZURIGUNEAK-JAUZI ekintza

Ekintza errepikakor bat erabiliko da, bitartean adibidez, eta zuriguneak irakurriko dira, bestelako bat aurkitu arte.

Hona hemen ekintzaren definizioa.

ekintza ZURIGUNEAK-JAUZI

hasierak {KAR aldagaiak zurigunea, letra bat edo puntua dauka}

bitartean KAR = ' ' *egin*
irakur (KAR)

ambitartean

amaia {KAR aldagaiak letra bat edo puntua dauka}

LEHEN-HITZA-GORDE ekintza

KAR aldagàiak lehen hitzaren lehen karakterea daukalarik, sekuentziako karaktereak banaka gorde eta irakurriko dira, harik eta hitzari amaia emango dion zurigunea edo puntua irakurri arte. Taula erabiltzeko, I indizearen balioaz lortuko dugu gordetako karaktereen kopurua.

Hona hemen ekintzaren deskripzioa:

```
ekintza     LEHEN-HITZA-GORDE
hasiera     {KAR-ek lehenengo karakterea dauka}
            I: = 1
            bitartean (KAR ≠ ' ')eta  (KAR ≠ '.') egin
              LEHENA (I): = KAR
              I: = I + 1
              irakur (KAR)
            ambitartean {I aldagaiak gordetako karaktereen ko-
              puruagehi bat dauka}
              LUZERA: = I - 1
              {KAR aldagaiak lehen hitzari amaia ematen dion zu
              rigunea edo puntua dauka}
              {LEHENA eta LUZERAk lehen hitza adierazten dute}
amaia
```

LEHENA-KONTATU ekintza

Sekuentziatik lehen hitza kenduta, berori zenbat aldiz agertzen den kontatu behar du ekintza honek. Ekintza honek zaila ematen duenez, beheranzko eraikuntza izeneko metodologia erabiliko dugu. Sekuentzia hitzez hitz hartu eta tratatu gura dugu.

Beraz, ekintza batek hitz oso bat hartuko du (karaktereak taula batetan gordez eta gordetako karaktereen kopurua aldagai bati asignatuz). Beste ekintza batek lehen hitza eta azken irakurritako hitza konparatuko ditu, eta berdinak izanez gero kontatu egin behar ko du.

Hona hemen algoritmoa:

```
ekintza     LEHENA-KONTATU
aldagaiak   HITZA(20 karakteretako taula)  {hitzak gordetzeko
                                            taula}
```

```
HITZLUZERA(osoa)  {hitzen karaktere-kopurua}
BERDINAK (boolearra)  {konparaketaren emaitza}
```

hasiera

```
{KAR aldagaiak lehen hitzari amaia eman dion zu-
rigunea edo puntua dauka}
ZURIGUNEAK-JAUZI
{KAR aldagaiak letra bat edo puntua dauka}
LEHENKONTA: = LEHENKONTA + 1
```

bitartean KAR ≠ '.' *egin* {LEHENKONTA aldagaiak lehen hitzetik oraingo karaktererainoko lehen hitzaren agerpenak dauzka}
{KAR-ek hurrengo hitzaren lehen karakterea dauka}

```
  HITZA-GORDE
    {HITZA eta HITZLUZERA aldagaek irakurritako hi
    tza adierazten dute}
    {KAR aldagaiak hitzari amaia eman dion zurigu-
    nea edo puntua dauka}
  KONPARATU
    {BERDINAK = egiazkoa, baldin bi hitzak berdinak
    badira}
    {BERDINAK = faltsua, bestelakoetan}
```

baldin BERDINAK *orduan* LEHENKONTA: = LEHENKONTA + 1

```
        {KAR = ' '  edo KAR = '.'}
  ZURIGUNEAK-JAUZI
        {KAR = '.' edo hurrengo hitzaren lehen ka-
        rakterea}
```

ambitartean

amaia

LEHENA KONTATU azpiproblema (ekintza ezprimitiboa) definitzeko, beste azpiproblemak erabili dira. Azpiproblema berri beroriek bigarren mailakoak dira. Azkeneko hauek ere ondo definitzeko, aldagai-ingururako bitarteko egoerak definitu behar dira eta azpiproblema bakoitzaren betebeharra ere bai. Hona hemen definizioak:

ZURIGUNEAK-JAUZI ekintza

Lehen definitu den ekintza bera da.

HITZA-GORDE ekintza

Irakurri letrak sekuentziatik zurigunea edo puntua ageri arte eta gorde karaktereak HITZA taulan. Asignatu LUZERA aldagaiari gordetako karaktereen kopurua.

Hasierako egoera: KAR aldagaiak lehen hitzaren lehen letra dauka edo puntua.

Azkeneko egoera: KAR aldagaiak hartutako hitzari amaia ematen dion puntua edo zurigunea dauka.
HITZA taulak irakurritako hitza dauka.
HITZLUZERA aldagai osoak dauka zenbat karaktere gorde diren HITZA taulan.

Izan ere, LEHEN-HITZA-GORDE ekintza bezalakoa baita, baina LEHENA taula erabili beharrean, HITZA taula, eta LUZERA aldagaiaren ordez HITZLUZERA aldagaia.

KONPARATU ekintza

LEHENA taulak eta LUZERA aldagai osoak adierazten duten hitza konpara ezazu HITZA taulak eta HITZLUZERA aldagai osoak adierazten dutenarekin. BERDINAK aldagai boolearrari asigna iezaiozu egiazkoa balioa hitz biak berdinak badira eta faltsua bestela.

Hasierako egoera: LEHENA-k eta LUZERA-k lehen hitza adierazten dute

HITZA-k eta HITZLUZERA-k irakurritako hitza adierazten dute.

BERDINAK-ek edozein balio du (*egiazkoa*) edo *faltsua* .

Azkeneko egoera: BERDINAK = *egiazkoa*, bi hitzak berdinak badira.

BERDINAK = *faltsua*, bestela.

Azken bi azpiproblemen eraikipena ondokoa da:

HITZA GORDE ekintza

Aurretik esan bezala, LEHEN-HITZA-GORDE ekintza bezalakoa da, baina LEHENA taula HITZA taulaz eta LUZERA aldagaia HITZLUZERA aldagaiaz tratatu direlarik.

```
ekintza   HITZA-GORDE
hasiera   {KAR-ek lehenengo karakterea dauka}
          I: = 1
          bitartean  (KAR ≠ '.') eta (KAR ≠ ' ') egin
            HITZA(I) : = KAR
            I: = I + 1
            irakur (KAR)
          ambitartean {I aldagaiak gordetako karaktereen kopurua gehi bat dauka}
            LUZERA: = I - 1
            {KAR aldagaiak hitzari amaia ematen dion zurigunea edo puntua dauka}
            {HITZA eta HITZLUZERA-k gordetako hitza adierazten dute}
amaia
```

KONPARATU ekintza

Konparaketa ahalik lasterren amaitzeko, karaktereak banaka konparatzen hasi aurretik, ea bi hitzen karaktere-kopurua berbera den aztertuko da. Hitz bat bestea baino luzeago baldin bada, inola ere ez dira berdinak izango.

Karaktere-kopuruak berdinak badira, banaka konparatu behar dira karaktereak. Horretarako, 'bitartean' ekintza errepikakorra erabiliko da. Taularen indizea gehituko da banaka, 1-etik hasita, bi letra diferente aurkitu arte edo azken karaktereetaraino iritsi arte.

Azken bi letren konparaketa ez da egiten 'bitartean' ekintzaren baldintzaren bidez, baldin ekintzaren bidez baizik . Honela eginez piskat arintsuagoa da algoritmoa eta gainera taularen erabilpen egokiaz segurtatzen gara. Hain zuzen, azken bi letren konparaketa bitartean ekintzaren baldintza bezala egiten bada, I taularen indizeak hurrengo balioa hartzen du, eta beharbada orduan taulek duten indizerako goiko muga gainezkatzen da, honelakoetan bitartean ekintzaren baldintza berriro ebaluatzerakoan existitzen ez diren bi aldagai aipatzen dira, errorea gertatzen delarik.

```
ekintza  KONPARATU
hasiera
        baldin LUZERA ≠ HITZLUZERA orduan BERDINAK: = faltsua
          bestela
            hasiera
              I: = 1
              bitartean  (I < LUZERA) eta (LEHENA (I)=HITZA (I))
                egin
                I: = I + 1
              ambitartean
                baldin LEHENA (I) = HITZA (I) orduan
                BERDINAK: = egiazkoa
```

bestela BERDINAK := *faltsua*

amaia bestela

amaia KONPATATU

Oharra: Ikus ondoko bi ekintzen baliokidetasuna

1.- baldin LEHENA(I) = HITZA(I) orduan BERDINAK := egiazkoa
bestela BERDINAK := faltsua

2.-BERDINAK := LEHENA(I) = HITZA(I)

Pauso asko eman ondoren lehenengo hitza zenbat aldiz dagoen kontatzeko problemarako algoritmo bat asmatu dugu. Bitarteko egoerak eta azpiproblemak sortu ditugu, eta beraien erabilpenez baliatu gara.

Orain bi aukera daukagu azken algoritmoa aurkezteko.

1.- Algoritmoaren eta ekintzen definizioak (elkarren) segidan idatzi.

2.- Algoritmoa eta ekintzak bildu, ekintza ezprimitiboetarako deien ordez dagozkien ekintza-segidak ipiniz. Aldagaien ezagupenak (edo deklarazioak) ere bil daitezke.

Ondoren, bi aukerak jarraituz lortzen diren algoritmoak aurkezten dira. Algoritmoaren eta ekintzen definizioak elkarren segidan idatziz lortzen den algoritmo osoa hauxe da:

Algoritmoa LEHENHITZA1

aldagaiak

KAR (karakterea) sekuentziako karaktereak irakurtzeko

LEHENKONTA (osoa) lehenengo hitzaren agerpenak kontatzeko

LEHENA (20 karaktereetako taula) lehenengo hitza gordetzeko

LUZERA (osoa)

hasiera

irakur (KAR)

```
    {letra bat, zurigunea edo puntua dauka KAR aldagaiak}
    ZURIGUNEAK-JAUZI
    {KAR aldagaiak letra bat edo puntua dauka}
    baldin  KAR = '.' orduan idatz ('ez dago hitz bat ere')
       bestela
          hasiera  {KAR aldagaiak lehenengo hitzaren lehenen-
                   go letra dauka}
            LEHEN-HITZA-GORDE
                   {KAR aldagaiak lehenengo hitzari jarraitzen
                   dion zurigunea edo puntua dauka}
                   {LEHENA eta LUZERA: lehenengo hitza adieraz
                   ten dute}
            LEHENA-KONTATU
                   {LEHENKONTA: lehen hitzaren agerpen-kopurua
                   dauka lehenengo hitza kontatu barik}
                   {KAR aldagaiak azkeneko puntua dauka}
            idatz (LEHENKONTA + 1)
          amaia {bestela}
amaia  {LEHENHITZA1}

ekintza   ZURIGUNEAK-JAUZI
hasiera   {KAR aldagaiak letra bat, zurigunea edo puntua dauka}
          bitartean KAR = ' ' egin
            irakur (KAR)
          ambitartean
            {KAR aldagaiak letra bat edo puntua dauka}
amaia     {ZURIGUNEAK-JAUZI}
```

```
ekintza   LEHEN-HITZA-GORDE
hasiera   {KAR: lehenengo karakterea dauka}
          I: = 1
          bitartean (KAR ≠ ' ') eta (KAR ≠ '.') egin
            LEHENA(I) : = KAR
            I: = I + 1
            irakur (KAR)
          ambitartean {I aldagaiak gordetako karaktereen kopu
            rua gehi bat dauka }
          LUZERA: = I - 1
          {KAR aldagaiak: lehen hitzari amaia ematen dion
                          zurigunea edo puntua.}
          {LEHENA eta LUZERA: lehen hitza adierazten dute}
amaia     { LEHEN-HITZA-GORDE}

ekintza   LEHENA-KONTATU
aldagaiak
          HITZA (20 karakteretako taula) { hitzak gordetzeko taula}
          HITZLUZERA (osoa)              {hitzen karaktere-kopurua}
          BERDINAK (boolearra)           {konparaketaren emaitza}
hasiera   {KAR aldagaiak lehen hitzari amaia eman dion zurigunea
          edo puntua dauka}
          ZURIGUNEAK-JAUZI
          {KAR aldagaiak letra bat edo puntua dauka}
          LEHENKONTA: = LEHENKONTA + 1
          bitartean KAR ≠ '.' egin {LEHENKONTA aldagaiak lehen
                                    hitzetik oraingo karaktere
                                    rainoko lehen hitzaren ager
                                    penak dauzka. }
                                   {KAR: hurrengo hitzaren lehen
                                    karakterea}
```

```
                    HITZA-GORDE
                      {HITZA eta HITZLUZERA aldagaiek irakurri
                      tako hitza adierazten dute}
                      {KAR aldagaiak hitzari amaia eman dion
                      zurigunea edo puntua dauka}

                      KONPARATU
                        {BERDINAK = egiazkoa, baldin bi hitzak
                                    berdinak badira}
                        {BERDINAK = faltsua, berdinak ez badira}
                      baldin BERDINAK orduan
                      LEHENKONTA: = LEHENKONTA + 1
                        {KAR = ' ' edo KAR = '.'}
                      ZURIGUNEAK-JAUZI
          ambitartean
amaia

ekintza   HITZA-GORDE
hasiera   {KAR-ek lehenengo karakterea dauka}
          I: = 1
          bitartean  (KAR ≠ '.') eta (KAR ≠ ' ') egin
            HITZA(I) : = KAR
            I: = I + 1
            irakur (KAR)
          ambitartean  {I aldagaiak gordetako karaktereen ko-
                       purua gehi bat dauka}
            LUZERA: = I - 1
            {KAR aldagaiak hitzari amaia ematen dion zurigunea
            edo puntua dauka}
```

```
        {HITZA eta HITZLUZERA: gordetako hitza adierazten
        dute}
amaia

ekintza   KONPARATU
hasiera
        baldin LUZERA ≠ HITZLUZERA orduan BERDINAK: =faltsua
          bestela
            hasiera
              I: = 1
              bitartean  (I < LUZERA) eta (LEHENA(I)  = (HITZA(I))
                         egin
                I: = I + 1
              ambitartean
                baldin LEHENA (I) = HITZA orduan
                                    BERDINAK: = egiazkoak
                                          bestela
                                    BERDINAK: = faltsua
            amaia {bestela}
amaia
```

Algoritmo nagusiaren eta ekintzen definizioak biltzen badira, ekintza ezprimitiboetarako deien ordez dagozkien ekintza-segidak ipiniz, ondoko algoritmo orokorra lortzen da:

Algoritmoa eta ekintzen definizioa bilduta, ekintza ezprimitiboetarako deien ordez dagozkien ekintza-segidak ipiniz hain zuzen ere. Algoritmo hauxe lortzen da:

```
Algoritmoa  LEHENHITZA2
aldagaiak   KAR (karakterea) {sekuentziako karaktereak irakurtzeko}
            LEHENKONTA (osoa) {lehen hitzaren agerpenak kontatzeko}
            LEHENA (20 karakteretako taula) {lehen hitza gordetzeko}
            LUZERA (osoa) {lehen hitzaren karaktere-kopurua}
            HITZA (20 karakteretako taula) {beste hitzak gordetzeko}
            HITZLUZERA (osoa) {HITZA taulan gordetako karaktere-kopurua}
            I (osoa)  {taulak erabiltzeko aldagai indizea}
hasiera
            irakur (KAR)
              {KAR aldagaiak letra bat zurigunea edo puntua dauka}
            bitartean KAR ≠ ' ' egin irakur (KAR) ambitartean
              {KAR aldagaiak letra bat edo puntua dauka}
            baldin KAR = '.' orduan idatz ('ez dago bat ere')
              bestela
                hasiera {KAR aldagaiak lehenengo hitzaren lehenengo letra dauka}
                  I: = 1
                  bitartean (KAR ≠ ' ') eta (KAR ≠ '.') egin
                    LEHENA(I)  : = KAR
```

```
  I: = I + 1
  irakur (KAR)
ambitartean
  LUZERA: = I - 1
    {LEHENA eta LUZERA aldagaiek lehen hitza
     adierazten dute}
bitartean (KAR ≠ ' ')egin irakur (KAR) ambitartean
LEHENKONTA: = 0
bitartean KAR ≠ '.' egin
  I: = 1
  bitartean (KAR ≠ ' ') eta (KAR ≠ '.') egin
    HITZA(I) : = KAR
    I: = I + 1
    irakur (KAR)
  ambitartean
    HITZLUZERA: = I - 1
    baldin LUZERA = HITZLUZERA orduan
    BERDINAK: = faltsua
      bestela
        I: = 1
        bitartean (I< LUZERA) eta
                  (LEHENA(I)  = HITZA(I)egin
          I: = I + 1
        ambitartean
          BERDINAK: =(LEHENA(I) = HITZA(I))
    baldin BERDINAK orduan
                    LEHENKONTA : = LEHENKONTA + 1
    bitartean KAR = ' ' egin irakur (KAR)
              ambitartean
  ambitartean
```

idatz(LEHENKONTA + 1)

amaia {bestela}

amaia {LEHENHITZA}

12. ARIKETA OSAGARRIAK

12.1. Zeroaz amaitzen den zenbaki-sekuentzia bat emanda, agertzen den monotonia goranzkor luzeena idatz ezazu.

12.2. Zeroaz amaitzen den zenbaki-sekuentzia bat emanda. Zenbaki guztiak idatz itzazu hamar zifra-karaktere idatziz. Sekuentziako zenbaki batek ere ez du 10 zifra baino gehiago.

Adibidea: 427 zenbakia tratatzerakoan

'0''0''0''0''0''0''0''4''2''7' karaktereak idatziko dira.

12.3. Karaktere-sekuentzia batek telegrama-sekuentzia bat adierazten du.

Telegrama bakoitza 'AMTEL' hitzaz amaitzen da, eta hitzez osatuta dago. Hitzak banatzeko zurigune bat edo gehiago erabiltzen da. Bi eratako hitzak daude:

- kobratzeko hitzak
- Hitz laguntzaileak: 'STOP' eta 'AMTEL' (ez dira kobratzen)

Telegrama-sekuentzia telegrama hutsaz amaitzen da. Telegrama hutsa, kobratzeko hitzik ez daukana da.

Idatz ezazu zera egiten duen algoritmoa telegrama bakoitzerako: idatz telegramaren textua (hitzak zurigune bakar batez banatuta), gero telegramak dauzkan kobratzeko hitzen kopurua Hitz hauek moztuta idatziko dira.

Adibidea:

DATORREN ASTEAN STOP HORRA STOP JOANGO NAIZ STOP AMTEL AITA GAIXORIK DAGO STOP ZATOZ STOP AMTEL STOP AMTEL

1. ERANSKINA: EKINTZAK

- Asignazioa

<aldagaia> : = <adierazpena>

-Irakurketa

irakur (<aldagaia>)

- Idazketa

idatz (<objektu>) (konstantea edo aldagaia)

-Aukeraketa

aukera

<baldintza> : <ekintza>

<baldintza> : <ekintza>

. .
. .
. .

<baldintza> : <ekintza>

amaukera

- Baldintzazkoa

baldin <baldintza> orduan <ekintza>
edo
baldin <baldintza> orduan <ekintza>
bestela <ekintza>

- Errepikatzekoak

errepika	bitartean <baldintza> egin
<ekintza>	<ekintza>
<ekintza>	<ekintza>
.	.
.	.
.	.
<ekintza>	<ekintza>
harik-eta <baldintza>	ambitartean

- Datu-sekuentzia hasieraketa

Sekuentzi-hasiera

2. ERANSKINA : DATU-MOTAK

osoa: balio posibleak : zenbaki osoak

eragiketak : +, -, *, DIV, MOD

karakterea: balio posibleak: letrak: 'a', 'b', 'c', ...

zifrak: '0', ..., '9'

karaktere bereziak: '?', '(', '=', ...

eragiketak:

boolearra: balio posibleak: egiazkoa, faltsua

eragiketak : eta, edo, ez

=, ≠, < , > , <=, > = eragiketek emaitza boolearra ematen dute.

3. ERANSKINA : ALGORITMOEN IDAZKERA

algoritmoa <deitura>
aldagaiak [<deitura> (<mota>)]*
hasiera
    [<ekintza>]*
amaia

[**ekintza** <deitura>
    hasiera
        [<ekintza>]*
    amaia
]*

BIBLIOGRAFIA

Liburu honen iturri nagusia ondokoa da:

PEYRIN, Jean-Pierre eta SCHOLL, Pierre-Claude
ALGORITHMIQUE ET PROGRAMMATION
Laboratoire IMAG, bp53 x, 38041 GRENOBLE-CEDEX; Fevrier 1982

Algoritmoei buruzko beste liburu interesgarri bat:

BIONDI,J. et CLAVEL,G.
INTRODUCTION A LA PROGRAMMATION
Ed. Nasson, Paris 1981

AURKIBIDEA

1.- ARIKETA-BILDUMARAKO SARRERA 2

1.1. Informatika eta programazioa 3

1.2. Prozesadoreen definizioa:

Datu-motak eta ekintzak 3

1.3. Prozesuak, programak eta algoritmoak 5

2.- ALGORITMOAK DEFINITZEKO PROZESADORE OROKORRA

(HASIERAKO DEFINIZIOA) 7

2.1. Pausokako definizioa 8

2.2. Objektu-motak 9

2.3. Ekintza primitiboak 10

2.4. Algoritmoen idazkera 11

3.- PROBLEMA-BILDUMA 13

1. Ariketa 14

2. Ariketa 17

3. Ariketa 21

4. Ariketa 29

5. Ariketa 35

6. Ariketa 42

7. Ariketa 48

8. Ariketa 57

9. Ariketa 65

10. Ariketa 68

11. Ariketa 77

12. Ariketa 88

Eranskinak 107

Bibliografia 111