Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Programació fonamental: problemes*

Sebastià Vila Marta

* L'elaboració d'aquest material ha comptat amb el suport d'un "ajut a la elaboració de material docent" de la UPC.

"Hay que cocinar con el mismo espíritu con que se habla con los amigos, se lee un buen libro o se hace algo que le gusta a uno mismo. De este modo, el acto de guisar produce una satisfacción que llega a quien consume ese plato como una energía positiva."

JUAN MARI ARZAK, Cocinar es fácil (1987)

Prefaci

"Conté la manera d'organitzar tota classe d'àpats senzills o d'etiqueta, de calcular berenars, sopars, escudelles, cuinats, rostits, salses, llegums, aviram, plats freds, canelons, arrossos, ous fregits, carns, peixos, amanides, escalivades, i diverses maneres de fer els púdings i pastes, i molts consells de pràctica i d'utilitat per comportar-se elegantment a taula, etc."

IGNASI DOMENÈCH, La Teca 15a ed. (1990)

Aquesta col·lecció de problemes està especialment dirigida a aquells alumnes dels cursos bàsics de programació de les escoles i facultats tècniques.

L'estructura del llibret s'ha dissenyat de manera que la introducció a aquesta matèria sigui la més progressiva possible. Al contrari que en moltes de les fonts de problemes de programació, en aquest llibret també s'ha incidit en els aspectes relacionats amb l'aprenentatge d'una notació algorítmica de caire imperatiu. No per això, però, s'han oblidat els aspectes més tradicionals del disseny d'algoritmes. Com a notació, s'ha escollit la definida a [7] per raons d'homogeneïtat.

Pel que fa a aquest aspecte, el llibret està pensat per a un curs fonamentat en l'ús d'esquemes de tractament seqüencial. No obstant això, altres enfocaments més formals també poden treure'n profit.

No fóra just atribuir-me la paternitat de tots els problemes. El llibret és un recull de problemes de molt diversos autors. A ells cal agrair l'elegància de molts dels problemes. Se'm fa difícil citar qui és l'autor de cada un. Alguns provenen d'antigues col·leccions de problemes editades per companys del Departament. Altres s'han obtingut revisant els arxius d'exàmens de cursos anteriors o bé de llibres sobre el tema. Molts s'han inventat *ad hoc* per al llibret. Totes les fonts publicades han estat citades a la bibliografia.

Tots ells han estat revisats i els seus enunciats reformats per unificar la notació i el vocabulari emprats. Els problemes s'han classificat en temes genèrics per facilitar-ne l'ús.

Finalment cal esmentar que aquest recull ho és de problemes per resoldre i no preten ser-ho

de problemes resolts. Tot i això, m'ha semblat molt adequat que l'alumne disposi d'algunes solucions com a model. Per això s'han resolt i afegit en un darrer capítol alguns problemes de dificultat mitjana.

Índex

1 Objectes i expressions

> *"Per elaborar un plat s'han de tenir preparats prèviament els ingredients necessaris, nets i tallats, perquè si s'ha de procedir alhora a la pesada i medició dels components i a la seva cocció, és molt possible que ens descuidem un detall o que alguna cosa ens surti malament."*
>
> MONTSERRAT SEGUÍ, Cuinar és senzill (1989)

1.1 Decidiu quines d'aquestes són declaracions d'objectes correctes i quines no. Justifiqueu-ho.

a)

var
  t, p, q : **enter**
fvar

b)

var
  l, **enter** : **real**
fvar

c)

var
  l : **boolea**
  m, q, l : **real**
fvar

d)

const
  pi : **enter** $= 3$
fconst

e)

const
  cert : **enter** = 1
fconst

f)

var
  mida : **real** = 3.2
fvar

g)

var
  grandaria, *factorQ* : **real**
  factorL : **enter**
fvar

h)

const
  *m*2 : **real** = 4
fconst

i)

const
  Revision : 1.24
fconst

1.2 Vegeu quines d'aquestes expressions són correctes sintàcticament i quines no. Interpreteu tot identificador com a variable o constant segons convingui.

a) $a + b + 4.0$

b) $3/2 + f(a) * l$

c) $3 - (-j)$

d) $4 * f(3 * (b \textbf{ i } c))$

e) $\textbf{no}(a \geq 1 \textbf{ div } (2 + q))$

f) $(3 - q(a, z) * 6$

g) $-\textbf{sqrt}(32.0 + l(\text{'a'}) + 27 \textbf{ mod } 2)$

h) $777.0012 * (\textbf{div} 46)$

i) $2 \textbf{ div } -f(\text{'m'})$

j) $0 - - - - - - -2 * - - - - - - - 0$

k) **no fals o cert**

l) **no fals** $= m($**cert**$) \geq$ **fals**

m) $q($**cert**, **cert**, $3 \leq 6, m) * 3$

1.3 Vegeu quines d'aquestes expressions són correctes sintàcticament i semàntica. Avalueu-ne les que ho siguin:

a)

const
  pi : **real** $= 3.1415$
  k : **enter** $= 3$
fconst
$pi * (k + 1.05)$

b)

const
  $epsilon$: **real** $= 4.0031$
  $fita$: **real** $= 3.56$
fconst
cert o $(epsilon * 2.0 - fita \geq 1.002)$

c)

var
  a, b : **enter**
  z : **real**
fvar
$-(a$ **mod** $b + 2)$ **div** $4 * z$

d)

var
  q, r, t : **boolea**
  $mida, lloc$: **enter**
fvar
q **i no**$(r$ **o** $r)$ **i** $(mida + 5 \geq 7)$

e)

const
  pi : **real** $= 3.141592$
  $lletraA$: **caracter** $=$ 'A'
fconst

var
x, y, z : **real**
fvar
$pi * 2.36 > pi * pi$ **o** $lletraA =$ 'A' **o** $x + 3.0 \leq \sin(y)$

1.4 Escriviu les expressions següents :

a) Donada una variable a : **caracter** trobeu una expressió que s'avalui a **cert** ssi a és una minúscula.

b) Donada una variable z : **real** trobeu una expressió que s'avalui a **cert** ssi succeeix que $z \in [1.0, 9.04]$.

c) Donada una variable v : **boolea** trobeu una expressió que s'avalui a **cert** ssi v conté el valor **cert**.

d) Trobeu una expressió que calculi el mateix valor que l'expressió matemàtica $\frac{a_1 + \log x}{b^\pi}$. Cal declarar aquells objectes utilitzats.

e) Si q és una variable de tipus **caracter**, quina expressió permetria classificar q com a caràcter alfabètic?

f) Si k és una constant de tipus **enter** i r és una variable de tipus **real**, quina expressió s'avaluaria a un **real** que és k vegades r?

1.5 Apliqueu el càlcul de proposicions per simplificar aquestes expressions booleanes:

a) **no**(**no**($a = 7$ **o** $4 * l = l * 4$))

b) ($a = b$) **o** **no**('A' < 'L')

c) **no**($a > 5$ **i** $a < 2$)

d) $b =$ **no**($b \neq b$)

e) **no**($3 < a$ **i** **no**($4 = 6$ **o** **fals**))

f) $b = q$ **i** (**no** r **o** $7 > 4$)

1.6 Complementeu (aplicant la negació) les expressions booleanes següents i simplifiqueu-les:

a) $a = 7$ **i** $b > n$

b) f $(x) \geq 2.9$ **o** a **mod** $q = 3$

c) $a = 8.9$ **i** **no**(s $(t) > 6.7$)

d) $b > 9.0$ **i** $l * 5.41 * q \geq t$ **i** $t = pi$

1.7 Simplifiqueu les proposicions següents després d'haver-les complementat:

a) $P \wedge (Q \Rightarrow P)$

b) $(P \vee Q) \wedge (\neg Q \vee R)$

c) $\neg(P \vee Q) \wedge Q$

d) $(Q \wedge R \wedge P) \vee \neg(R \vee S)$

e) $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$

f) $((P \Rightarrow Q) \Rightarrow R) \vee P$

g) $(P \vee Q) \wedge ((P \Rightarrow R) \vee (P \Rightarrow Q))$

1.8 Expresseu, usant un predicat i les funcions auxiliars necessàries, els enunciats que hi ha a continuació:

a) L'existència d'un enter parell que és múltiple de 13.

b) El fet que, donats dos reals qualssevol, sempre n'existeix un tercer que rau entre els dos primers.

c) El fet que la mitja de dos enters qualssevol sempre resta inclosa en l'interval tancat definit per aquests.

d) L'existència d'arrel quadrada per a tot real positiu.

e) La propietat enunciada com: "Si n és un enter parell, llavors la seva darrera xifra ha de ser parella".

1.9 Construïu un predicat per a cada enunciat que expressi:

a) Donat un cert enter k, el fet que k sigui parell.

b) Donats dos naturals i i j, el fet que siguin primers entre si.

c) Donat un natural p, la propietat de ser primer.

d) Donats tres naturals a, b, c, el fet que $mcd(a, b) = c$ en termes d'operacions elementals.

e) Donat un vector $\vec{v} = (a, b)$, el fet que $\vec{v}$ pertanyi al primer quadrant.

1.10 Suposeu l'existència d'un vector N-dimensional $\vec{x} = (x_0, \ldots, x_{n-1})$. Aleshores, la notació x_i denota la i-èsima component del vector $\vec{x}$. En aquest entorn, escriviu un predicat que expressi cada un dels enunciats següents:

a) r és la suma dels elements de $\vec{x}$.

b) m és el màxim valor de $\vec{x}$.

c) les components de $\vec{x}$ es troben ordenades de manera creixent.

d) Si $\vec{x}$ conté un 1, llavors també conté un 0.

e) Totes les components de $\vec{x}$ són diferents.

f) El màxim de $\vec{x}$ solament ocorre una vegada.

g) r és el producte dels elements positius de $\vec{x}$.

h) r és el màxim de les sumes dels segments de $\vec{x}$.

1.11 Aquestes expressions tenen un o més "forats" indicats pel símbol @. Completeu-les de manera que esdevinguin correctes usant els objectes declarats en cada cas:

a)

var
  a, b, c : **real**
fvar
$(-a * \log(c) > c/@)$

b)

var
  q, s : **boolea**
  x, y : **real**
fvar
no q **i** $(-x/3.0 > @)$ **i** @

c)

var
  c : **caracter**
  l : **enter**
fvar
$@ =$ 'a' **o** $l > 98002$

1.12 Una expressió és un algoritme? Raoneu la resposta.

1.13 Raoneu si l'afirmació següent és correcta: "Un algoritme pot ser modelat per un camí en un espai d'estats".

1.14 Per quina raó és necessari declarar els objectes que intervenen en un algoritme?

1.15 Quina és la diferència existent entre algoritme i programa?

2 Assignació. Descomposició d'accions

"Assaisonner: action de saler à point; action simple et cependant capitale pour la cuisine qui exige du sens, le goût, une grande subtilité, doublée de beaucoup d'attention et de discernement."

PAUL BOCUSE, La Cuisine du Marché (1980)

2.1 Vegeu si les assignacions següents són correctes i, en el seu cas, determineu-ne l'estat final:

a)

var
  a, b, c : **enter**
fvar
$a := 4 + (\mathbf{enter}(3.67))$

b)

const
  pi : **real** $= 3.1415$
  $total$: **enter** $= 7$
fconst
var
  $result$: **real**
fvar
$result := 3.90 * pi + \mathbf{real}(total)$

c)

var
  $acum1$: **real**
fvar
$acum1 := 7.01$
$acum1 := acum1 + 1.0$

d)

var
  $c1, c2, c3$: **caracter**
  result : **boolea**
fvar
{ $c1 = \text{'a'} \wedge c2 = \text{'k'} \wedge c3 = \text{'c'}$ }
$result := \mathbf{no}(c3 \geq c1 \ \mathbf{i} \ c3 \leq c2)$

2.2 Determineu quines assignacions satisfan els següents estats inicials i finals. Considereu a tal efecte la declaració d'objectes següent:

var
  a, i : **enter**
  b : **boolea**
  x, y, z : **real**
fvar

funcio arrodonit (**ent** r : **real**) **retorna enter**
{ Retorna l'enter més proper a r }

funcio parell (**ent** k : **enter**) **retorna boolea**
{ Retorna **cert** ssi k és parell }

a)

{ $z = Z \wedge z > 0 \wedge a = A$ }
...
{ $a = \lfloor Z \rfloor \wedge z = Z$ }

b)

{ $z = Z \wedge z > 0 \wedge a = A$ }
...
{ $a = \text{arrodonit } (Z) \wedge z = Z$ }

c)

{ $i = I \wedge i \geq 0$ }
...
{ $b = \text{parell } (i)$ }

2.3 Feu un algoritme que, donat un nombre enter que designa un període de temps expressat en segons, doni l'equivalent en dies, hores, minuts i segons.

2.4 Trobeu un algoritme que, donada una quantitat en pessetes, la descompongui en quantitats de moneda fraccionària. Això és, pessetes, duros, vint-i-cinc pessetes,... de manera que minimitzi el nombre de peces.

2.5 Usant la composició seqüencial d'accions, dissenyeu un algoritme que intercanviï el valor de a amb el de b. Considereu que a i b són objectes variables amb tipus a, b : **enter**.

2.6 Aquestes accions construïdes per composició alternativa tenen errors. Esbrineu quins són comprovant si compleixen totes les condicions establertes per a la construcció. Corregiu-los.

a) Donades dues variables de tipus **enter**, de nom a i b, calculeu-ne el màxim:

si
$\quad a > b \longrightarrow maxim := a$
$\quad [\!] \; a < b \longrightarrow maxim := b$
fsi

b) Calculeu el valor absolut de la variable z : **real** i l'assigneu a la variable *absolut* : **real**

$absolut := z$
si
$\quad z < 0 \longrightarrow absolut := -z$
fsi

2.7 Usant la composició seqüencial i alternativa d'accions, construïu una acció que assigni a la variable *result* : **enter** el valor menor dels continguts en les variables a, b, c : **enter**.

2.8 Escriviu un algoritme que permeti esbrinar quin tipus d'arrels té una equació de segon grau. Suposeu que les variables a, b, c : **real** contenen els coeficients de l'equació. El resultat cal emmagatzemar-lo a la variable r : **caracter**. Aquesta contindrà una 'd' si l'arrel és doble, una 'r' si hi ha dues arrels reals i una 'e' si hi ha arrels complexes.

2.9 L'estructura alternativa és més complicada quan és factible que més d'una condició sigui certa alhora. Per quina raó?

2.10 Dissenyeu un algoritme, usant estructures alternatives i seqüencials, tal que donats x, y, z : **real** variables amb un cert valor inicial diferent, obtingui una permutació dels mateixos valors sobre les variables $a1, a2, a3$: **real** que compleixi $a1 \geq a2 \geq a3$.

2.11 [*Algoritme d'Euclides*] Aquest algoritme iteratiu calcula el màxim comú divisor pel mètode d'Euclides. Simuleu-ne una execució. Feu-ne un gràfic a l'espai d'estats i identifiqueu:

a) L'estat inicial.

b) L'estat final.

c) La propietat invariant.

Com es determinaria l'estat final a partir de la propietat invariant i de la condició de la

iteració?

$\{\ A > 0 \wedge A > 0 \wedge a = A \wedge b = B\ \}$
mentre $a \neq b$ **fer**
  si
    $a > b \longrightarrow a := a - b$
    $[]\ b > a \longrightarrow b := b - a$
  fsi
fmentre
$\{\ a = b = \text{mcd}\ (A, B)\ \}$

2.12 Aquestes construccions iteratives són incorrectes. Esbrineu per quina raó. Suposeu, per a les deduccions, que tots els objectes usats són del tipus adequat i han estat correctament inicialitzats.

a)

mentre $valor < inf + 0.6$ **fer**
  $i := i + j$
fmentre

b)

mentre $suma \geq 0.0$ **fer**
  $l := f + suma + l$
fmentre

c)

mentre $residuQ \leq 6$ **o** $l = l$ **fer**
  $h := k + 1$
  $s := \log(residuQ)$
  $residuQ = residuQ + 1$
fmentre

d)

mentre $q \leq 6$ **fer**
  $q := q + 1$
  $m := r$ **i** $(s = t)$
fmentre
{Post: $m \in \{\textbf{cert}, \textbf{fals}\} \wedge q = 3$ }

e)

{Prec: $i = 1$ }
mentre i **mod** $2 = 1$ **fer**
  $i := i + 2$
fmentre

2.13 Dissenyeu un algoritme iteratiu que permeti dividir dos enters usant únicament sumes i productes. Això és, una acció que satisfaci l'especificació:

$$\{\ a = A \wedge b = B \wedge a > 0 \wedge b > 0\ \}$$
$$\ldots$$
$$\{\ r = A\ \mathbf{div}\ B\ \}$$

Verifiqueu el seu funcionament fent una simulació. Preneu nota de totes les dificultats amb què ens trobem en fer aquest exercici.

2.14 [*El problema del pot de cafè*] Aquest problema s'atribueix originalment a *Carel Scholten* i fou explicat per *E. W. Dijkstra* a *D. Gries* com a exemple de la potència que comporta raonar usant el concepte d'invariant.
Suposeu que es té un pot de cafè amb grans negres i grans blancs. Repetiu el procés següent mentre sigui possible:

"Seleccioneu aleatòriament dos grans del pot. Si són del mateix color, llenceu-los i afegiu un gra negre al pot. Si són de colors diferents, torneu el gra blanc al pot i llenceu el negre."

Trobeu una expressió que permeti saber el color de l'únic gra que resta al pot després d'haver repetit el procés. Aquesta expressió és funció del nombre de grans blancs i negres que inicialment hi havia al pot.

3 Subprogrames

"Si en una recepta no s'explica detingudament una salsa, arranjament de verdures, preparació d'una crema, bescuit, etc. s'ha de buscar a la secció corresponent."

MONTSERRAT SEGUÍ, Cuinar és senzill (1989)

3.1 Donades les següents especificacions de subprogrames, escriviu-ne les capçaleres. Analitzeu amb atenció els mecanismes de pas de paràmetre i els tipus d'aquests mecanismes. Noteu que per escriure les capçaleres d'aquests subprogrames no és necessari saber com s'implementen.

a) Un subprograma tal que, donats dos enters, diu si són primers entre si.

b) Un subprograma tal que, donades tres variables reals, obté el mínim interval tancat que les conté totes tres.

c) Un subprograma tal que, donats dos caràcters i un tercer, calcula si el darrer està dins l'interval que va del primer al segon.

d) Un subprograma tal que, donat un caràcter, ens calcula si el caràcter en qüestió és una lletra majúscula o no.

e) Un subprograma tal que, donat un enter, calcula el signe de l'enter i ens torna $-1, 0, 1$ segons si l'enter és negatiu, zero o positiu.

f) Un subprograma tal que, donat un real r, obté l'àrea de la cua de la funció normal centrada i reduïda definida per l'interval $(-\infty, r]$.

3.2 Tot seguit es troba un conjunt d'especificacions de subprogrames i un algoritme que les utilitza. Es demana que esbrineu l'estat final de l'algoritme. Noteu que no cal saber com estan implementats els subprogrames per resoldre l'exercici.

a)

funcio f (**ent** x : **real**) **retorna real**
{ $f(x) = \int_0^x e^{-\sin y} dy$ }
accio inc (**ent** x : **real**, **entsor** y : **real**)
{ incrementa en x el valor de y }
algoritme test1 **es**
  var
    m, n : **real**
  fvar
  $m := 3.984$
  $n := 1.0e-2$
  $inc(8.0, m)$
  $n := f(m)$
falgoritme

b)

{ Donat un valor x, retorna el seu factorial }
funcio fact (**ent** x : **enter**) **retorna enter**
{ $a = A \wedge b = B \wedge c = C$ }
accio max3 (**ent** a, b, c : **enter**, **sor** m : **enter**)
{ $m =$ maxim (A, B, C) }
algoritme test2 **es**
  var
    q, n, m : **enter**
    s, t : **real**
  fvar
  $s := 3.087$
  $q :=$ **enter**(s)
  $n := 7; m := 2$
  si
    $q = 3 \longrightarrow$
      max3 (q, n, m, m);
      $n :=$ fact (m)
    $\square$ $q \neq 3 \longrightarrow n := -3$
  fsi
falgoritme

c)

{ Donat el número de dia a partir d'1/1/90, calcula la temperatura màxima que s'enregistrarà a París }
funcio maxima (**ent** *ndia* : **enter**) **retorna real**

```
algoritme BonTemps es
    var
        bontemps : boolea
    fvar
    si
        maxima (4022) ≥ 18.0 ⟶ bontemps := cert
      □ maxima (4022) < 18.0 ⟶ bontemps := fals
    fsi
falgoritme
```

3.3 Seguidament hi ha uns algoritmes dels quals únicament interessen els objectes que hi participen. Per a cada un, escriviu-ne el diagrama de temps de vida. En una segona passada pel problema, esbrineu-ne l'estat final.

a)

```
algoritme cas1 es
    const
        v : enter = 1
    fconst
    var
        a, b, c : enter
    fvar
    c := 67
    b := −9
    A (a)
    B (c, b)
falgoritme

accio A (sor c : enter) es
    var
        m, p : real
    fvar
    C (m)
    c := 4
faccio

accio C (sor m : real) es
    m := real(v + 1)
faccio

accio B (ent l, k : enter) es
    var
        x : real
```

```
        z : caracter
    fvar
    x := 1.05
    z := caracter(l)
faccio
```

b)

```
accio A (ent a, b : enter, sor c : enter) es
    var
        m : enter
    fvar
    c := a
faccio

accio B (ent b, a : enter, entsor q : enter) es
    var
        m : enter
    fvar
    m := 34
    q := enter(3.98)
faccio

accio AB (ent a, b : enter) es
    var
        m, n : enter
    fvar
    n := b
    A (a, b, m)
    B (a, b, n)
faccio

funcio F (ent a : enter) retorna enter es
    retorna a + 2
ffuncio

accio AA (ent a, b : enter, sor c : enter) es
    var
        r : enter
    fvar
    A (a, b, r)
    A (a + b, b, r)
    c := r
faccio
```

```
algoritme cas2 es
    var
        a, b, c, d : enter
    fvar
    a := 3
    b := 45
    c := 2
    B (a, b, c)
    AA (a, F (b), d)
    AB (a, b)
falgoritme
```

3.4 Tot seguit es defineixen un conjunt d'objectes i subprogrames. En base a aquests, decidiu si les crides que es troben a cada subapartat són correctes o no i digueu per què.

```
const
    a : real = −0.098
    b : real = 0.8901
    k : enter = 8
    c : caracter = 'p'
fconst

var
    va, vb : real
    vk : enter
    vc : caracter
fvar

funcio fun2 (ent a, b : enter) retorna enter
```

a)

```
{ Acció que volem cridar }
accio acc1 (ent a, b : enter, sor s : real)
...
{ Crida a l'acció }
acc1 (k, enter(a), b)
```

b)

```
{ Acció que volem cridar }
accio acc2 (ent max : enter, sor l : real)
...
{ Crida a l'acció }
acc2 (va, vb)
```

c)

```
{ Acció que volem cridar }
accio acc3 (ent a, b : enter, sor w : real)
...
{ Crida a l'acció }
acc3 (3, enter(a), vb)
```

d)

```
{ Acció que volem cridar }
accio acc4 (ent a, b : real, entsor y : caracter)
...
{ Crida a l'acció }
acc4 (real(k), a, 'Q')
```

e)

```
{ Acció que volem cridar }
accio acc5 (ent a, b : enter, sor t : real)
...
{ Crida a l'acció }
acc5 (k + fun2 (2, 7), enter(va), va)
```

3.5 Sovint es defineix una funció com aquell subprograma que no modifica l'estat de l'algoritme. En aquest cas, què podríeu dir d'aquest subprograma?

```
{ a = A ∧ b = B }
accio suma (ent a, b : enter, sor r : enter)
{ a = A ∧ b = B ∧ r = A + B }
```

3.6 Dissenyeu una funció tal que, donats dos intervals $[a, b]$ i $[c, d]$ en els reals, retorni l'interval intersecció.

3.7 Dissenyeu una funció tal que, donats tres punts del pla, indiqui si aquests estan sobre una mateixa recta o no.

4 Cerca i recorregut de seqüències elementals

"Sauf quelques petites exceptions, toutes les formules indiquées aux aubergines sont applicables aux courgettes."

AUGUSTE ESCOFFIER, Ma Cuisine (1934)

4.1 Un operador introdueix pel teclat d'un computador una frase que acaba sempre en punt. Dissenyeu un algoritme que calculi quants caràcters té la frase. El resultat cal escriure'l pel canal de sortida estàndard.

4.2 Un operador introdueix pel teclat d'un computador una seqüència d'enters que acaba amb zero. Es vol un algoritme que calculi la suma dels enters que formen la seqüència. Escriviu el resultat pel canal estàndard de sortida.

4.3 Un operador introdueix una frase acabada en un punt. Dissenyeu un algoritme que reconegui la presència del caràcter 'Z'.

4.4 Aigües avall d'Ascó, a la riba de l'Ebre, existeix un aparell que mesura la intensitat de radiació emesa per les aigües del riu. Aquest aparell envia el valor mesurat cada δ minuts cap a un computador dedicat i finalitza la seqüència amb una mesura especial de valor -1.0. El computador imprimeix a cada interval de temps δ les dades següents:

- El valor de la mesura,
- El valor de la mitjana de totes les dades rebudes fins al moment.
- El carácter '*', com a indicatiu de perill, si la distància entre la mesura i la mitjana supera un llindar prefixat Ψ.

Es vol l'algoritme que defineix el comportament d'aquest computador. Per dissenyar-lo suposeu que existeix el subprograma següent:

```
{ Obté la mesura següent }
accio ObteMesura (sor m : real)
```

4.5 Dissenyeu un algoritme capaç de calcular el primer nombre enter múltiple de 13 i cap-i-cua.

4.6 Escriviu un algoritme per determinar, donat un N, el k més petit que compleix l'equació $k \geq 0 \wedge 2^k \geq N$.

4.7 En una planta química es disposa d'un dispositiu sensor que mesura la concentració d'anhídrid sulfurós d'un reactor. Aquest dispositiu envia informació cap a un computador periòdicament: cada δ dècimes de segón, envia la concetració en ppm. La seqüència de concentracions sempre acaba amb el valor -1.0. El computador disposa de l'acció LlegirConcentracio que permet rebre la informació provinent del sensor. Aquesta acció té com a especificació:

```
{ Obté la següent mesura del sensor }
accio LlegirConcentracio (sor c : real)
```

Construïu un algoritme que escrigui la mitjana de les tres darreres mesures cada vegada que una nova mesura arriba.

4.8 Un operador introdueix una frase acabada en punt en un computador. Dissenyeu un algoritme que esbrini si es repeteix la primera lletra de la frase.

4.9 Es desitja obtenir un algoritme que tabuli totes les potències de 2 inferiors a 2048 tals que la seva darrera xifra pertanyi a l'interval $[0, 5]$.

4.10 Donada una seqüència d'enters que arriba pel canal estàndard d'entrada i essent 0 el sentinella, es demana que dissenyeu un algoritme per calcular el màxim.

4.11 Dissenyeu un algoritme tal que, donada una seqüència d'enters acabada en zero, pugui esbrinar si són tots positius. Obteniu la seqüència del canal estàndard d'entrada.

4.12 Es vol saber si en el conjunt $[127, 4592]$ hi ha algun element que sigui múltiple de 13. Dissenyeu un algoritme que pugui prendre aquesta decisió.

4.13 Donat un polígon com una seqüència de vèrtex acabada amb el vèrtex inicial, construïu un algorisme que pugui esbrinar si és un polígon regular o no. Els vèrtexs vénen donats per una seqüència de parelles de reals que l'usuari introdueix pel teclat. El resultat cal escriure'l per la sortida estàndard. Considereu que com a mínim hi ha tres vèrtexs.

4.14 En una comunitat de veïns cal contractar el servei de neteja. La presidenta de l'escala ha obtingut les característiques de la planta del rebedor calculant les coordenades de la poligonal que la defineix. L'empresa de neteja necessita saber la superfície del rebedor per pressupostar el servei. Es demana que construïu un algorisme capaç de calcular-la. Considereu la mateixa tècnica per obtenir les dades que en el problema anterior.

4.15 Un natural x es diu que és quadrat perfecte si existeix algun y natural tal que $y^2 = x$. Construïu un algoritme que determini si, donat un interval $[a, b]$, aquest comprèn algun quadrat perfecte.

4.16 En un computador primitiu construït a principi dels 80 cal fer el càlcul de la funció $f(x) = e^x$ per alguns valors de x. A tal efecte cal usar el desenvolupament en sèrie

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

aturant el càcul quan dues aproximacions successives x_i i x_{i+1} difereixen en menys d'un cert paràmetre k de l'algoritme. Dissenyeu un algoritme que efectuï aquest càlcul.

4.17 Un operador introdueix una frase acabada en el caràcter '.'. Dissenyeu un algorisme capaç de saber si hi ha més caràcters 'a' o bé n'hi ha més de 'b'.

4.18 Construïu una funció tal com:

$\{\ n > 0, \text{EsPrimer}\ (n) \iff n \text{ és un primer}\ \}$
funcio EsPrimer (**ent** n : **enter**) **retorna boolea**

4.19 Un operador introdueix en un computador una seqüència de dades provinents d'un rellotge de control de presència. Les dades estan agrupades de tres en tres enters. Cada triplet d'enters indica hores, minuts i segons. L'operador introdueix triplets un darrere l'altre fins arribar al triplet $(0, 0, 0)$ que es considera la marca de final. La seqüència de triplets cal considerar-la com la història d'hores d'entrada (HE) i sortida de (HS) d'un cert operari d'una empresa:

$$HE_1, HS_1, HE_2, HS_2, HE_3, HS_3, \ldots, HE_i, HS_i, \ldots$$

Dissenyeu un algoritme que calculi el nombre d'hores que aquest operari ha treballat.

4.20 Construïu una funció que, donat un natural, esbrini quants 1 té en la seva representació decimal. Sabríeu modificar-lo perquè comptés el nombre de 1 que hi ha en la representació en una base qualsevol k?

4.21 Construïu una funció que calculi $\sqrt{a}$ per a qualsevol a real mitjançant el desenvolupament en sèrie

$$x_1 = a$$
$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$$

que aturi el càlcul quan dues aproximacions successives difereixin en menys d'ϵ.

4.22 Donats dos enters a i b no negatius, dissenyeu un algoritme que calculi el valor ab. Useu únicament la suma d'enters com a primitiva.

4.23 [*Algoritme del comerciant rus*] Construïu un algoritme per calcular la divisió entre dos enters no negatius usant únicament sumes, restes, productes, mòduls i divisions per dos.

4.24 Sigui a un enter positiu, dissenyeu una funció que calculi la part entera de $\sqrt{a}$ usant solament aritmètica entera.

4.25 [*Regla de Ruffini*] Construïu un algoritme que us permeti efectuar la divisió d'un polinomi $a_0 + a_1x + \ldots + a_nx^n$ pel polinomi mònic de primer grau $x - b$ usant la regla de *Ruffini*. Definiu un mecanisme per obtenir les dades pel canal estàndard d'entrada i per comunicar els resultats a través del canal standard de sortida.

4.26 Donat un enter positiu n, dissenyeu una acció que escrigui per la sortida estàndard els seus divisors.

4.27 Construïu una funció que calculi $n!$ donat n.

4.28 Construïu una funció que calculi, donats m i n, el valor $\binom{m}{n}$. No oblideu l'eficiència.

4.29 [*Factorització de Hörner*] Dissenyeu un algoritme per avaluar un polinomi en un punt. Utilitzeu a tal efecte el mètode de *Hörner*. Aquest mètode té com a principi la factorització següent:

$$\begin{aligned} P(x) = & \\ & = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \\ & = a_0 + (a_1 + (a_2 + (\cdots (a_n)x \cdots)x)x)x \end{aligned}$$

En dissenyar l'algoritme, suposeu que les dades arriben pel canal d'entrada i decidiu vosaltres mateixos el format d'entrada més adequat per al problema. Aquest mètode és més eficient que l'avaluació d'un polinomi usant l'expressió habitual ($a_0 + a_1 x + \cdots + a_n x^n$). Raoneu per què.

4.30 Dissenyeu un algoritme tal que, donat un enter positiu, n'escrigui una representació en base 3 pel canal estàndard de sortida.

4.31 Dissenyeu un algoritme tal que, donada una seqüència de caràcters acabada en punt que s'obté del canal d'entrada, escriu en el canal de sortida la mateixa seqüència però sense espais.

4.32 [*Nombres perfectes*] Es diu que un cert natural n és perfecte si i només si aquest és igual a la suma de tots els seus divisors llevat d'ell mateix. Per exemple,

$$28 = 1 + 2 + 4 + 7 + 14$$

Dissenyeu una funció tal que, donat un natural, esbrina si és perfecte.

4.33 [*Arrel digital*] Donat un nombre natural n, es defineix la seva arrel digital com el resultat de sumar els dígits que el componen tot iterant el procés amb el nou nombre fins arribar a un nombre d'un sol dígit. Aquest darrer és l'arrel digital del primer. Per exemple,

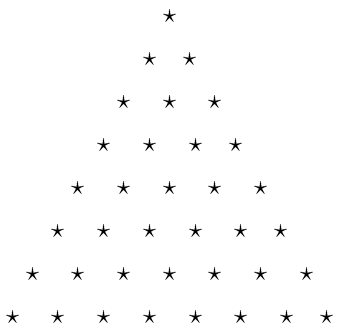
$$374 \Rightarrow 3 + 4 + 7 = 14 \Rightarrow 1 + 4 = 5 \Rightarrow \text{ArrelDig}(374) = 5$$

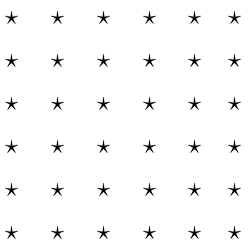
Dissenyeu una funció tal que, donat un natural, en calculi l'arrel digital.

4.34 A la botiga dels germans Pastafullada és tradició presentar les llaunes de conserva apilades triangularment. En el primer pis n'hi ha una, en el segon dues, en el tercer tres i així successivament. Per exemple, sis llaunes es posen així:

Els germans tenen grans problemes per fer les comandes de llaunes. Fixeu-vos que no tot nombre de llaunes pot apilar-se triangularment: per exemple, 8 llaunes no poden apilar-se. Dissenyeu un algoritme tal que, donat un natural, comprova si és un nombre adequat per muntar piles.

4.35 Els germans Pastafullada estan encantats del seu algoritme. Ara volen perfeccionar-lo i volen saber quins són els nombres adequats per fer piles triangulars i quadrades indistintament. El 36, per exemple, n'és un:





Dissenyeu un algoritme que indiqui si un cert natural és vàlid o no per fer piles triangulars i quadrades indistintament.

4.36 Dissenyeu un algoritme tal que, donada una expressió que un operador introdueix, indiqui si és una expressió ben parentitzada —els parèntesis estan ben posats. L'expressió acaba en un `'.'`. Per exemple, són expressions invàlides:

$$((x+2)(4\star 8)$$
$$(f(x)))(3-a\star 7)$$
$$)a($$

4.37 Donada un seqüència d'enters acabada en -1, determineu si és una seqüència creixent estrictament.

4.38 Construïu un algoritme que permeti determinar quin és el primer nombre natural n que compleix $25n+3 \leq n^2$.

5 Estructures de dades

"Los postres deben ser, pues, un plato complementario del resto del menú; esto es, escogido en función de los platos que hemos comido."

ANTONIO ROCHE, El libro de los postres (1990)

5.1 Dissenyeu un tipus de dades per representar cada una de les entitats següents:

a) Un interval a la recta real.

b) Un vector de $\mathbf{R}^3$.

c) Una matriu de 3×3 quadrada en els reals.

d) Un polinomi de grau tres a $\mathbf{Z}[x]$.

e) Una matriu de polinomis cúbics amb coeficients enters.

f) Un complex.

g) Un cub amb el centre de la base situat en un cert punt de l'espai.

h) Un segment de $\mathbf{R}^3$.

5.2 Donades les següents especificacions informals d'objectes coneguts per tothom, definiu un conjunt d'operacions per a cada un d'ells i escolliu-ne una representació adequada:

a) Un conjunt d'enters.

b) Una llista de noms de persones.

c) Una llista de noms de persones i els seus DNI de manera que, donat el DNI, poguem conèixer el nom de la persona.

5.3 Considereu els tipus de dades següents i obteniu-ne més d'una representació:

a) El tipus de dades que representa un complex.

b) El tipus de dades que representa un polinomi de coeficients reals.

c) El tipus de dades que representa una matriu d'enters.

Discutiu, cas a cas, quins són els avantatges i inconvenients de cada representació.

5.4 [*Organització territorial*] Donada l'estructura de dades següent

```
tipus
    pais = tupla
        r : regions
        president : nom
    ftupla
    regions = taula [1..40] de regio
    regio = tupla
        capital : ciutat
        n : nom
        c : comarques
    ftupla
    comarques = taula [1..50] de comarca
    comarca = tupla
        n : nom
        capital : ciutat
        hab : enter
    ftupla
    ciutat = tupla
        n : nom
        hab : enter
    ftupla
    nom = taula [1..100] de caracter
ftipus
```

Escriviu l'expressió d'accés o el petit algoritme indicat a continuació:

a) El nom de la tercera regió.

b) El nombre d'habitants de la capital de la setena comarca de la regió quarta.

c) La tercera lletra del nom de la primera comarca de la dissetena regió.

d) El nombre d'habitants que viuen en capitals de regió.

e) Per a una regió donada r, el nombre d'habitants que no viuen en capitals de comarca.

f) Quantes regions tenen una capital que comença amb 'A'.

5.5 [*Inversió de mots*] Suposeu una taula $t[0..N]$ de caràcters que conté un mot. Construïu un algoritme que, sobre la mateixa taula i sense taules auxiliars, obtingui el mot invertit.

5.6 [*Mots palíndroms*] Un mot m és palíndrom quan és igual al seu invers: $m = m^{-1}$. Dissenyeu una funció booleana tal que donat un mot m retorna **cert** si i només si m és palíndrom. Considereu que m s'emmagatzema en una taula de caràcters $t[1..N]$ i que la longitud de m és N.

5.7 [*Mots palindromables*] Un mot $m = (m_1, \ldots, m_N)$ es diu que és palindromable ssi existeix una permutació s tal que en aplicar-la al mot m s'obté un nou mot $m_s = (m_{s(1)}, \ldots, m_{s(N)})$ que resulta ser palíndrom. Dissenyeu una funció tal que donat un mot m amb $|m| = N$ retorni **cert** si i només si m és un mot palindromable.

5.8 [*Càlcul matricial*] Definiu un tipus matriu real de 4×4 i dissenyeu subprogrames per:

a) Calcular el determinant d'una matriu.

b) Donades dues matrius, obtenir-ne el producte sobre una tercera.

c) Donada una matriu, transposar-la sense usar espai auxiliar.

5.9 Definiu el tipus vector d'enters de 20 dimensions i dissenyeu subprogrames per:

a) Sumar dos vectors.

b) Fer el producte escalar.

5.10 t és una taula d'enters amb índexs en el rang [1..10].

a) Dissenyeu un algoritme que calculi el màxim valor contingut.

b) Dissenyeu un algoritme que calculi el segon valor més gran.

c) Dissenyeu un algoritme que indiqui si conté el valor 10.

5.11 [*Representació lineal de matrius*] t és una taula de 15 enters sobre la qual volem simular una matriu de 3×5. Indiqueu quina és l'expressió tal que, donats f i c índexs d'un element de la matriu, obté l'índex de l'element corresponent a t.

5.12 Considereu la taula d'enters amb valors:

$$-1, 4, 2, -2, 3, 7, 5$$

a) Ordeneu-la usant el mètode d'inserció directa i compteu quantes assignacions cal fer-hi.

b) Ordeneu-la usant el mètode d'intercanvi i compteu quantes assignacions són necessàries.

5.13 Considereu el tipus de dades que permet representar un ciutadà amb el seu nom i el seu DNI. Dissenyeu un algoritme tal que, donada una taula de ciutadans on el ciutadà de nom "XXX" actua de sentinella, obtingui una taula d'enters que apunten a la taula de ciutadans de forma que resultin ordenats per nom.

5.14 [*Càlcul de la mediana*] Donada una taula d'enters $t[1..100]$, construïu un algoritme que en calculi la mediana.

5.15 Construïu un algoritme d'ordenació rudimentari per a taules de 4 elements usant comparacions element-element.

5.16 Considereu el tipus de dades

```
tipus
    taupers = taula [1..100] de pers
    pers = tupla
        edat, dni : enter
    ftupla
ftipus
```

i suposeu que t és un objecte de tipus taupers amb totes les seves components inicialitzades. Dissenyeu un algoritme d'ordenació que obtingui una ordenació de menor a major pel camp *edat* i que, per edats iguals, ordeni de major a menor usant el camp *dni*.

5.17 [*Join de taules*] El problema del *join* de taules consisteix en el següent: suposeu definits els tipus de dades

```
tipus
    taupers = taula [1..100] de pers
    pers = tupla
        edat, dni : enter
    ftupla
    taunom = taula [1..100] de nompers
    nompers = tupla
        nom : cadena
        dni : enter
    ftupla
ftipus
```

i també el objectes *tp* de tipus taupers i *tm* de tipus taunom inicialitzats correctament. Sabent que tots els DNI de *tp* existeixen a *tm*, construïu un algoritme que obtingui una taula amb elements de tipus

```
tipus
    resul = tupla
        nom : cadena
        edat : enter
    ftupla
ftipus
```

creuant la informació de tp i tm.

5.18 [*Distància entre seqüències*] Siguin F i G taules d'enters tals que contenen una seqüència creixent acabada en -1. Dissenyeu una funció que en calculi la distància. A tal efecte, es defineix la distància com

$$d(F,G) = \min\{|f-g| \text{ t.q. } f \in F \wedge g \in G\}$$

5.19 Siguin F i G taules d'enters tals que contenen una seqüència creixent acabada en -1. Dissenyeu una funció que determini si existeixen $f \in F$ i $g \in G$ tals que $|f-g| < 9$.

5.20 [*Mínim element comú*] Siguin F, G i H taules d'enters tals que contenen una seqüència creixent acabada en -1. Dissenyeu una funció que calculi el mínim element comú a les tres considerant que sempre hi ha un o més elements comuns.

5.21 [*Elements redundants*] Donada una taula A d'enters indexada en $[1 \ldots N]$, dissenyeu un algoritme que escrigui en el canal de sortida els elements de A suprimint els redundants i sense usar taules auxiliars. Un element és redundant si és igual a un element anterior de la taula.
Feu aquest disseny suposant:

a) Que A no està ordenada.

b) Que A està ordenada de manera creixent.

5.22 [*Operacions amb conjunts*] Siguin a i b dues taules d'enters que contenen elements positius no repetits acabats amb el sentinella -1. Aquestes taules representen conjunts. Dissenyeu un algoritme que emmagatzemi sobre una tercera taula la intersecció dels elements de a i b. Feu dues hipòtesis:

a) Que a i b no estan ordenades.

b) Que a i b estan ordenades de manera creixent.

5.23 Siguin $A[1 \ldots n]$ i $B[1 \ldots m]$ dues taules ordenades de manera creixent els elements de les quals són enters. Dissenyeu un algoritme que computi el nombre de parelles (i,j) existents que satisfan la relació $A[i] + B[j] > 0$.

5.24 Escriviu una funció que sigui capaç de determinar, per a una matriu A d'enters amb n files i m columnes, el nombre de components no negatives. Suposeu que $A[i, j]$ és decreixent en funció de i però creixent en funció de j.

5.25 Donada una taula $A[1 \ldots n]$ d'enters ordenada creixent i un enter $k > 0$, dissenyeu una funció que calculi el nombre de parelles (i, j) que compleixen:

$$1 \leq i \leq j \leq n \wedge A[i] - A[j] \leq k$$

5.26 Dissenyeu una funció que decideixi si una certa taula d'enters $A[1 \ldots 2n]$ amb $n \geq 2$ té les components parells ordenades de manera creixent i les senars de manera decreixent.

5.27 Dissenyeu una funció tal que, donada una taula d'enters amb el primer element diferent de zero, compti el nombre dels seus canvis de signe. Considereu que un canvi de signe és l'aparició de dos enters amb signes diferents separats com a molt per zeros.

5.28 [*Quadrats màgics*] Dissenyeu una funció que comprovi que una matriu quadrada d'ordre n és un quadrat màgic. Una matriu és quadrat màgic si i només si compleix:

- La suma dels elements d'una fila és idèntica per a tota fila i val S.
- La suma dels elements de qualsevol columna és idèntica i val S.
- La suma dels elements de qualsevol de les diagonals principals és S.

5.29 [*L'erola*] Donada una taula $T[0 \ldots N-1]$ d'enters amb $b > 0$ i N constant, es defineix una erola com una subseqüència de la taula d'elements amb idèntic valor. Dissenyeu una funció tal que, donada una taula com l'anomenada, calculi la longitud de l'erola més llarga.

5.30 [*El fet diferencial extremeny*] Sigui la taula $B[1 \ldots n]$, on $n > 0$ és fixat i els elements de la qual són caràcters 'V' o 'B' o 'N' en qualsevol ordre. Escriviu un algoritme que permuti els elements de manera que passin a formar la bandera extremenya. És a dir, quelcom de la forma

$$VVVVVVBBBBBBNNNNNN$$

Solament pot operar-se sobre la taula permutant dos elements qualssevol i l'algoritme ha de fer com a molt n intercanvis.

6 Problemes de mitjana envergadura

"La grande cuisine, ça ne veut pas dire cuisine compliquée... la grande cuisine ce peut être une dinde bouillie, une langouste cuite au dernier moment, une salade cueillie dans le jardin et assaisonnée a la dernière minute."

PAUL BOCUSE, La Cuisine du Marché (1980)

6.1 Donada una seqüència de caràcters acabada amb el caràcter '.' que arriba pel canal d'entrada estàndard, escriviu pel canal de sortida una taula amb la freqüència d'aparició de cada una de les vocals.

6.2 Construïu un algoritme que calculi quantes vegades es repeteix la primera paraula d'una frase acabada en '.'. Considereu que la frase arriba pel canal d'entrada i el resultat cal escriure'l pel canal de sortida.

6.3 [*Divisors primers*] Donat un enter positiu n, dissenyeu una acció que escrigui per la sortida estàndard els seus divisors.

6.4 [*Segments nuls*] Considereu la seqüència numèrica $s_1, \ldots, s_n$. Direm que un segment $s_i, s_{i+1}, \ldots, s_j$ és nul si compleix $s_i + s_{i+1} + \cdots + s_j = 0$. Definim la longitud d'un segment $s_i, s_{i+1}, \ldots, s_j$ com $j - i + 1$. Considereu A una taula que conté N enters ordenats creixentment i amb totes les components diferents de zero. Dissenyeu una funció que calculi la longitud del segment nul més llarg.

6.5 Construïu un algoritme que calculi la longitud mitjana de les paraules que hi ha en una seqüència de caràcters acabada en '.'. Considereu que la seqüència pot obtenir-se del canal d'entrada estàndard.

6.6 [*Telegrames*] Donat un text que acaba amb la paraula "stop", construïu un algoritme que calculi el cost d'enviar-lo. Per calcular el cost, considereu les regles següents:

- Totes les paraules tenen cost 1 excepte els monosíl·labs, que tenen cost 0.
- Un monosíl·lab és una paraula que conté una vocal com a màxim.

- Les paraules amb més de 8 lletres són penalitzades amb un cost de 2.

Considereu que la seqüència pot obtenir-se del canal d'entrada.

6.7 Construïu un algoritme que calculi quantes paraules d'una seqüència de caràcters acabada amb un punt són palíndroms —és a dir cap-i-cues—. Considereu que la seqüència pot obtenir-se del canal d'entrada.

6.8 Considereu el conjunt de nombres naturals M definit com:

- $1 \in M$
- Si $x \in M$ llavors $2x + 1$ i $3x + 1$ són de M
- Cap altre element no pot pertànyer a M

Suposeu que es disposa d'una taula del tipus

tipus
Tau = **taula** [1..100] **de enter**
ftipus

Dissenyeu un algoritme que generi sobre una taula m d'aquest tipus els 100 elements menors de M. Òbviament, m no pot contenir elements repetits.

6.9 Considereu el tipus Tau definit al problema anterior. Considereu que existeixen dues taules a, b : Tau cada una d'elles plena amb valors significatius i que compleixen el següent: el valor de $a[i]$ indica la cel·la de la taula b que ocuparia la posició i-èsima en cas que b fos una taula ordenada. Dissenyeu una funció tal que, donades les dues taules i un enter x, obtingui de la manera més eficient possible la posició de x dins la taula b.

6.10 [*Matrius quasi nul·les*] Quan hi ha un percentatge elevat d'elements d'una matriu amb valor zero, es diu que la matriu és quasi nul·la. En tasques on sovint cal treballar amb matrius quasi nul·les, aquestes es representen d'una manera especial: s'usa una taula de triplets on cada triplet conté un dels elements distints de zero juntament amb la fila i la columna que ocupava a la matriu original. Per exemple,

$$\begin{pmatrix} 0 & 0 & 3 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 0 & 6 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{esdevé} \quad \begin{array}{ccc} f & c & v \\ 1 & 3 & 3 \\ 2 & 1 & 4 \\ 3 & 2 & 6 \\ 3 & 3 & 1 \\ 4 & 4 & 3 \\ 5 & 5 & 1 \end{array}$$

Es demana que construïu un algoritme tal que a partir de dues matrius de mida il·limitada però amb un màxim de MAX elements no nuls obtingui la suma i el producte d'aquestes matrius. Els resultats s'han d'escriure en el canal de sortida usant la notació tradicional de matrius. Les dades s'obtenen del canal d'entrada usant un format que cal dissenyar. Com a restricció, s'imposa que en cap moment les matrius no poden existir a la memòria en una altra forma que la representació comprimida explicada abans.

6.11 [*Vocals fantasmes*] Siguin u i v paraules formades amb lletres minúscules. Direm que u i v estan associades per vocals fantasmes si i només si u pot obtenir-se a partir de v permutant els seus caràcters però sense modificar l'ordre relatiu de les consonants.
Per exemple: si $u =$ "perla" llavors "parle", "pearl" i "paerl" estan associades a u per vocals fantasmes. No podem dir el mateix de "lepra" ja que les consonants han canviat d'ordre.
Considereu una seqüència formada per lletres minúscules, blancs i un punt final. Suposeu que la paraula més gran de la seqüència té 10 caràcters. Dissenyeu un algoritme que calculi el nombre de paraules de la seqüència que estan associades per vocals fantasmes a la primera paraula. Podeu suposar que la seqüència sempre tindrà una paraula o més.

6.12 [*Muntanyes i valls*] Es diu que una successió de nombres enters $a_1, \ldots, a_n$ te forma de muntanya si $\exists h : 1 < h < n : a_1 \leq \cdots a_{h-1} \leq a_h \geq a_{h+1} \geq \cdots \geq a_n$. Es defineix la forma de vall de la mateixa manera que la muntanya però canviant el signe de les desigualtats. Per exemple: la successió $-7, -1, 6, 21, 15$ és una muntanya. La successió $12, 5, 0, 12$ té forma de vall.
Dissenyeu un algoritme que, donada una seqüència de nombres enters, calculi la longitud de la subseqüència més llarga que forma una vall o una muntanya. Podeu suposar les propietats següents sobre la seqüència:

- La seqüència no és monòtona. Per tant, sempre hi haurà alguna muntanya o vall.
- Mai dos elements a_i i a_{i+1} de la seqüència no seran iguals.

6.13 [*Anagrames*] Una paraula w és un anagrama de v, si w pot obtenir-se a partir de v permutant les lletres. Per exemple, "RAPE" és un anagrama de "PARE". Considerant que pel canal d'entrada arriba una seqüència de caràcters —minúscules, espais i un punt final—, dissenyeu un algoritme que calculi quantes paraules són un anagrama de la primera. A tal efecte, suposeu que cap paraula té més de 10 caràcters i que sempre hi haurà com a mínim una paraula.

6.14 [*El nombre misteriós*] Sigui n un nombre enter de quatre xifres no iguals. Es defineix:

- la funció gran (n) com el nombre més gran que podem formar amb les xifres de n.
- la funció petit (n) com el nombre més petit que podem formar amb les xifres de n.
- definim dif $(n) =$ gran $(n) -$ petit (n)

Per exemple, si $n = 1984$, llavors gran $(n) = 9841$, petit $(n) = 1489$ i dif $(n) = 8352$.
Escriviu un algoritme que llegeixi un enter n, comprovi que compleix les condicions demanades i calculi, tot escrivint els valors, els termes de la successió

$$\text{dif } (n), \text{dif } (\text{dif } (n)), \text{dif } (\text{dif } (\text{dif } (n))), \ldots$$

fins que trobi un nombre k tal que dif $(k) = k$. Es pot demostrar que aquest punt fix sempre existeix i que, a més, sempre és $k = 6174$.

6.15 [*Un de romans*] Definiu i dissenyeu dues funcions:

- Aquella tal que, donat un enter $n > 0$ expressat en notació decimal, en calcula la representació usant xifres romanes.
- Aquella tal que, donat un enter $n > 0$ expressat en xifres romanes, en calcula la representació decimal.

6.16 [*Triangle de Tartaglia*] Dissenyeu un algoritme que escrigui les n primeres files del *triangle de Tartaglia*. Recordeu que el triangle té l'aspecte següent:

$$
\begin{array}{ccccccc}
 & & & 1 & & & \\
 & & 1 & & 1 & & \\
 & 1 & & 2 & & 1 & \\
1 & & 3 & & 3 & & 1 \\
1 & & 4 & & 6 & & 4 \quad 1 \\
 & & & \ldots & & &
\end{array}
$$

6.17 Donades les coordenades de 10 punts del pla $\mathbf{R}^2$, dissenyeu un algoritme que:

- Calculi la matriu de distàncies. És a dir, ompli una estructura de dades amb les distàncies entre dos punts qualssevol i i j de les dades.
- Construeixi un polígon usant el procediment següent: la primera aresta a_1 es crea unint els dos vèrtexs més propers; les arestes successives es calculen unint un dels dos extrems de la poligonal $a_1, \ldots, a_k$ amb el vèrtex més proper; finalment, la darrera aresta uneix els dos extrems de la poligonal.
- Escrigui pel canal de sortida les coordenades dels vèrtexs del polígon ordenadament.

Podeu suposar que les dades inicials s'obtenen del canal d'entrada amb el format

$$x_1 y_1 x_2 y_2 \ldots x_{10} y_{10}$$

.

6.18 [*Substitució polialfabètica*] La substitució polialfabètica és una tècnica d'encriptat per a textos molt senzilla d'implementar. Es fonamenta en una paraula clau k de mida k_m a partir de la qual es genera una matriu d'encriptació m. Per exemple, si $k =$ "TRIPOLI"

llavors la matriu m fóra:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z
T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H
P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Y	Z	A	B	C	D	E	F	G	H

Observeu que m té dimensions $k_m \times 26$ en el cas que l'alfabet per encriptar consti de 26 caràcters. L'encriptat del text es duu a terme mot a mot. Cada mot s'encripta fent correspondre a cada caràcter el corresponent encriptat segons la matriu usant files succesives de manera cíclica. Per exemple, el text

"TANT DE BO FOS SEMPRE PRIMAVERA"

resulta ésser després d'encriptar

"NRVJ RP JI XXI HPUJJM FGTUTNMHO"

Es demana que dissenyeu un algoritme tal que, donada una seqüència formada per una paraula clau una estrelleta, i un text acabat amb una estrelleta obtenible pel canal d'entrada, escrigui pel canal de sortida el text encriptat. En dissenyar aquest algoritme, intenteu minimitzar l'ús d'espai de memòria. Com a complement, podeu dissenyar també l'algoritme que fa la desencriptació.

6.19 [*Pastachoux SA*] L'empresa Pastachoux SA es dedica a la venda d'un conjunt de productes. L'empresa ha definit una seqüència que representa les vendes que ha fet als seus clients durant un cert període de temps. Aquesta seqüència té l'estructura següent:

per cada client:
  codi de client [**enter**]
  per cada venda al client
    codi del producte [**enter**]
    quantitat venuda [**enter**]
  final vendes client [0]
final seqüència [0]

Per exemple, la seqüència

124 39 10 44 8 39 15 0 112 44 34 22 67 44 78 23 75 0 0

representaria les vendes a dos clients (el 124 i el 112), el primer amb tres vendes (prod. 39, quant. 10, ...) i el segon amb quatre vendes. Noteu que:

a) Totes les vendes fetes a un client estan agrupades en el mateix bloc.

b) Dins un client, les vendes no estan ordenades.

c) Dins un client, el mateix producte pot aparèixer diverses vegades.

d) El nombre de vendes no està limitat i pot ser, circumstancialment, zero.

L'import d'una venda s'obté multiplicant la quantitat venuda pel preu. Els preus dels productes estan disponibles en una taula t que conté els codis de producte i el seu preu. Si es dóna el cas excepcional de trobar un producte no existent a t, se li assignarà preu zero.
Se sap que hi ha MAX codis de producte diferents com a molt i que la taula constant t pot consultar-se mitjançant les operacions

{Prec: $p \in t$ }
funcio PreuProducte (p : **enter**, t : TauProd) **retorna enter**
{Post: retorna el preu de p }

funcio ProducteExistent (p : **enter**, t : TauProd) **retorna boolea**
{Post: **cert** $\iff p \in t$ }

Considerant les condicions explicades, es demana que dissenyeu un algoritme que, llegint una seqüència de dades del canal d'entrada, escrigui en el canal de sortida un llistat amb la informació següent:

a) El volum de les vendes realitzades a cada client.

b) La quantitat total venuda de cada article.

c) El codi de producte de l'article més venut.

6.20 [*Salt del cavall*] Dissenyeu un algoritme que sigui capaç de produir la seqüència de posicions per les quals passa un cavall d'escacs pel tauler de manera que, iniciant el recorregut per qualsevol casella, passi per totes les caselles una i solament una vegada. Com a pista, es proposa la solució donada per *J. C. Warndorff* l'any 1823. Aquesta consisteix a moure el cavall sempre a la posició tal que, des d'ella, el nombre possible de moviments del cavall és mínim.

6.21 [*Combinacions*] Dissenyeu un algoritme per generar totes les combinacions d'elements del conjunt $[1, \ldots, m]$ agafats d'n en n. Recordeu que les combinacions ens indiquen les extraccions possibles d'elements del conjunt i que, per tant, l'extracció $\{1, 2, 3\}$ és equivalent a l'extracció $\{2, 1, 3\}$.

6.22 Donada una taula a es desitja reordenar els seus components segons una permutació. La permutació es representa com una taula d'enters $p[1], \ldots, p[n]$. Per exemple, suposem $a = (A, B, C, D, E)$ i la permutació $p = (4, 1, 5, 2, 3)$, llavors el resultat de permutar a segons p fóra $a = (D, A, E, B, C)$.

Construïu una acció que dugui a terme la modificació de a segons p i sense usar estructures auxiliars. Se suggereix que penseu en la descomposició de p com a producte de cicles. Per exemple: $p = (4, 5, 1, 3, 2, 8, 6, 7) = (1, 4, 3)(2, 5)(6, 8, 7)$.

6.23 [*El poema*] Donat un poema, es demana que contruïu un algoritme que compprovi si està format per estrofes amb rima $A_k, A_k, B_k, C_k, C_k, B_k$.
En aquest poema són separadors els caràcters: blanc, coma, punt i apòstrof. Els versos es troben separats per barres i el poema acaba amb un vers virtual que no conté cap mot. És a dir, dues barres entre les quals solament hi ha zero o més separadors.
Es considera que dos versos rimen si les seves darreres paraules tenen lletres idèntiques a partir de la vocal tònica —aquesta inclosa. Per poder reconèixer-les, aquestes vocals s'han escrit en majúscules, al contrari que la resta de lletres, que s'han escrit en minúscules.
Es considera incorrecte aquell poema que conté alguna estrofa amb un nombre de versos diferent de sis.
Com a exemple d'un poema que rima correctament i té l'estructura explicada a l'enunciat, podeu considerar:

```
lA nOia somniOsa, recolzAda /
Al finestrAl, nO pEr sA rIca estAda /
A lA vOra dEl rIu m'hA vIst captIu /
sinO perquE trobAnt-se tOta sOla, /
havIa deixAt cAure Una fullOla /
dE sAlze sObre El rIu.  /
nI l'Aura dE llevAnt m'Es exquisIda /
perquE Em dUgui perfUms dE lA florIda /
dEls presseguErs dEl pUig orientAl /
sinO perquE lA tImida fullOla /
pEr Ella Al cAire dE mA bArca vOla /
En tImid caminAl //
```

Txan-Tiu-Lin (trad. per Josep Carner)

6.24 [*Els periòdics*] Construïu un algoritme que tabuli el valor de $\frac{1}{n}$ per $n = 2, \ldots, 100$. El resultat ha de ser una taula de valors de n i $\frac{1}{n}$ on, en aquest darrer, es deixa un espai en blanc just abans del període (o primer zero, en cas de decimal exacte), i aquest només s'escriu una vegada. Per exemple, podeu considerar:

```
2  0.5 0
3  0.  3
4  0.25 0
5  0.2 0
6  0.1 6
7  0.  142857
8  0.125 0
9  0.  1
```

```
10  0.1 0
11  0.  09
12  0.08 3
13  0.  076923
...
```

6.25 [*Reconeixement d'imatges*] Construiu un algoritme que permeti reconèixer una subimatge d'una imatge donada. Aquests algoritmes sovint s'anomenen algoritmes de *pattern matching* i tenen molta importància en el camp de la visió per computador.
Les imatges per tractar estan formades per píxels de color blanc o negre. És a dir, la imatge o fotografia es parteix amb una malla quadrada de punts, i s'analitza el color de cada punt assignant-li blanc o negre segons la seva intensitat. A l'algoritme, les imatges li arriben pel canal d'entrada codificades de la manera següent:

```
7 5
.....
.xxx.
..x..
..x..
..x..
.xxx.
.....
```

És a dir, nombre de files, nombre de columnes i imatge, on la imatge està formada per caràcters cada un dels quals representa un píxel. La codificació és '.' per indicar el color blanc i 'x' per indicar el color negre.
L'algoritme rep com a entrada dues imatges i cal que indiqui en quin lloc de la primera imatge fa *matching* la segona, en cas que així sigui. Se sobrentén que una imatge fa *matching* amb una altra si els píxels de la primera i de la segona coincideixen un a un. Per exemple, la imatge

```
3 3
xxx
.x.
.x.
```

fa *matching* amb l'anterior a la posició $(2, 2)$.

6.26 Construïu una aplicació per obtenir el valor propi més gran d'una certa matriu. La matriu $A(n \times n)$ serà obtinguda pel canal d'entrada estàndard considerant el format següent:

$$< n >< fila_1 > \cdots < fila_n >$$

Per calcular el valor propi major s'usarà la seqüència següent de X_k i Z_k:

$$\begin{aligned} Z_k &= AX_k \\ X_{k+1} &= Z_k/|Z_k| \end{aligned}$$

tal que si es pren X_0 com qualsevol vector, llavors el valor $|Z_k|/|X_k|$ tendeix al valor

propi més gran i X_k al vector propi corresponent. Considereu que el mètode ha convergit quan dues aproximacions successives del valor propi siguin suficientment properes.

6.27 [*Justificació de textos*] Donat un text acabat en '.' que podeu llegir pel canal d'entrada, dissenyeu un algoritme que l'escrigui pel canal de sortida però justificant-lo a 70 columnes.

6.28 [*El teatre*] Per tal de facilitar la venda d'entrades, un teatre vol posar a disposició del públic un programa consistent en un menú de dues opcions. La primera escriu la llista de les representacions indicant el dia, el mes i l'hora en què es duran a terme i, si s'escau, el rètol "ENTRADES ESGOTADES". La segona opció, de venda d'entrades, ha de demanar el dia i el mes de la representació i també el nombre d'entrades que es desitja i la categoria.
El teatre, rectangular de 30 files i 20 seients per fila, es considera dividit en tres zones: les 7 primeres files de seients de primera categoria (2500 PTA per entrada), les files 8 a 20 de segona categoria (2000 PTA per entrada) i les files 21 a 30 de tercera categoria (1500 PTA per entrada). Dins de cada categoria de seients, aquests es consideren ordenats per files i dins de cada fila per columnes. Les entrades s'han de vendre segons aquesta ordenació.
El programa ha de donar missatges d'error en el cas en què, en el dia i el mes sol·licitats, no hi hagi cap actuació, o que les entrades estiguin esgotades o si no hi ha prou seients de la categoria demanada. En cas contrari, ha d'escriure el número de fila i seient de cada entrada i ha de dir quin és el total a pagar.

6.29 [*Calendaris*] La congruència de *Zeller* és un càlcul que permet obtenir el dia de la setmana per a una data qualsevol. Donat un dia determinat pel triplet (d, m, a), on d és el dia del mes, m és el mes de l'any i a és l'any,

a) se li resten dues unitats al mes m i si dóna zero o menys se li suma 12 al mes i se li resta una unitat a l'any. El nou mes obtingut l'anomenem m' i el nou any a'.

b) es calcula la centúria c (els dos primers dígits de l'any) a partir de l'any a'.

c) es calcula l'any dins la centúria y (els dos darrers dígits de l'any) a partir de l'any a'.

d) es calcula

$$f = \lfloor 2.6m' - 0.2 \rfloor + d + y + \left\lfloor \frac{y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c$$

e) f mòdul 7 representa el dia de la setmana

$$0 = \text{diumenge}$$
$$1 = \text{dilluns}$$
$$\ldots$$

Useu aquest mètode per construir una aplicació que llegeix del canal d'entrada un mes i un any tot escrivint pel canal de sortida el calendari corresponent. Per exemple, si

l'entrada és:

```
2 1990
```

la sortida hauria de ser

```
                1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28
```

6.30 [*Envolupant convexa*] Donada una seqüència S de punts del pla

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_y), (-1, -1)$$

es pot definir l'envolupant convexa d'aquests punts com aquell polígon convex que circumscriu tots els punts. Evidentment, els vèrtexs d'aquest polígon han de pertànyer a S.

Per saber si un cert punt $p_k \in S$ és un vèrtex de l'envolupant, cal veure si existeix un altre punt p_j tal que tots els punts d'S són dins un dels semiespais definits per la recta $\overline{p_k p_j}$.

Dissenyeu un algoritme per obtenir la llista dels vèrtexs de l'envolupant convexa de la seqüència de punts obtinguda del canal d'entrada. Estudieu les maneres possibles de millorar-ne l'eficiència.

6.31 [*Rotacions*] Direm que una successió numèrica $a_1, a_2, \ldots, a_n$ és una rotació de la també successió $b_1, b_2, \ldots, b_n$ si:

$$\begin{aligned} \exists j : 0 \leq j < n : \\ & a_{j+1} = b_1, \\ & a_{j+2} = b_2, \\ & \ldots, \\ & a_n = b_{n-j}, \\ & a_1 = b_{n-j+1}, \\ & \ldots, \\ & a_j = b_n \end{aligned}$$

Per exemple, les successions $3, 6, 5, 4, 8, 7, 2$ i $8, 7, 2, 3, 6, 5, 4$ són rotacions de l'original $5, 4, 8, 7, 2, 3, 6$.

Donada una taula T d'enters definida sobre $[1..N]$ amb tots els seus valors diferents, dissenyeu una acció tal que, per una seqüència d'enters, calculi el nombre de segments d'S que són una rotació de T. Aquesta acció no pot fer ús de taules auxiliars.

Per exemple, si suposeu que

$$S = 5, 3, 9, 6, 5, 3, 7, 7, 8, 4, 6, 5, 3, 7, 8, 2, 4, 7, 3, 7, 8$$
$$T = 6, 5, 3, 7, 8, 4$$

llavors el nombre de rotacions de T trobades a S són 3 i es corresponen amb els segments

$$s_1 = 7, 8, 4, 6, 5, 3$$
$$s_2 = 8, 4, 6, 5, 3, 7$$
$$s_3 = 4, 6, 5, 3, 7, 8$$

6.32 [*Erosió*] Podem representar un tall geogràfic usant una seqüència d'enters, entenent que cada enter representa l'alçada en metres respecte del nivell del mar i que les mesures s'han pres de metre en metre. Podem suposar que tots els meteors només erosionen les parts ascendents del tall —en el sentit de lectura de la seqüència—, i ho fan de la manera que mostra l'exemple següent:

$$\begin{aligned} &10, 9, 8, 10, 12, 14, 7, 8, 3 \rightarrow \\ &\quad 10, 9, 9, 11, 11, 13, 8, 7, 3 \equiv \\ &\quad 10, 9, 8+1, 10+1, 12-1, 14-1, 7+1, 8-1, 3 \end{aligned}$$

és a dir, incrementant l'alçada de la meitat inferior de la seqüència estrictament creixent, i minvant l'alçada de la meitat superior, ambdues en una unitat de mesura (noteu que si la longitud de la seqüència estrictament creixent és senar l'element central roman inalterat).
Dissenyeu un algoritme que llegeixi una seqüència pel canal d'entrada i escrigui la corresponent erosionada pel canal de sortida.

6.33 [*Cercle viciós*] Direm que un segment $x_1, \ldots, x_i$ (amb $i \geq 2$) és un segcercle si succeeix

$$x_1 = x_i$$
$$\forall j = 2 \ldots i-1 : x_1 \neq x_j$$

és a dir, comença i acaba amb el mateix número i aquest no apareix cap altra vegada en el segment. Definim el diàmetre del segcercle com la longitud menys un.
Una seqüència és segcercle divisible si la podem dividir en segcercles, de manera que tot element de la seqüència pertany a un i solament un segcercle. Per obtenir aquesta divisió s'ha de procedir de la manera següent: obtenir el primer element i cercar-ne una nova aparició. Quan el trobem haurem tancat el primer segcercle. Aquest procés el repetirem amb l'element següent (que trobem després del segcercle) fins haver tractat tots els elements. Noteu que l'últim element ha de tancar l'últim segcercle.
Construïu una acció tal que donada una seqüència segcercle divisible i sabent que no conté dos segcercles del mateix diàmetre, doni com a resultat una altra seqüència que contingui els segcercles originals ordenats per diàmetre creixent.

7 Problemes resolts

"Com que aquest llibre no vol ésser una lliçó de cuina, com que no va dirigit al o a la que no sap gens de cuinar, sinó a qui ja té nocions de cuina, les quantitats de condiments i elements no s'especifiquen sempre; per altra banda, com que cada casa és un món, cada casa és un gust."

FERRAN AGULLÓ, Llibre de la cuina catalana 2a ed. (1933)

7.1 Manufactures del Cardoner

7.1.1 Enunciat

L'empresa "Manufactures del Cardoner" ha instal·lat un dispositiu sensor en una de les cintes transportadores de la seva cadena de mecanitzat. Aquest dispositiu envia al canal estàndard d'entrada del computador una seqüència d'enters. Aquesta seqüència està formada pels vèrtexs dels objectes poligonals que passen per la cinta. Els vèrtexs es donen com a dues coordenades positives x y, i es presenten ordenats en el sentit de les agulles del rellotge. Per a cada objecte, es donen tots els seus vèrtexs i es repeteix el primer en darrer lloc. Els vèrtexs d'un objecte se separen dels vèrtexs del següent mitjançant el $(-1, -1)$ i el final de la seqüència és donat per un objecte sense vèrtex.

Les arestes dels polígons descrits sempre formen angles de noranta graus i, com a mínim, un polígon té dues arestes —excepció feta del polígon final de seqüència.

Donat un polígon com els descrits, es pot calcular la seva superfície de manera molt simple. Solament cal considerar totes les arestes horitzontals del polígon i,

- per cada aresta que va d'esquerra a dreta ($\rightarrow$) cal sumar el valor $l * y$, on l és la longitud d'aresta i y és la coordenada y de qualsevol dels vèrtexs de l'aresta.

- per cada aresta que va de dreta a esquerra ($\leftarrow$) cal restar el valor $l * y$, on l és la longitud d'aresta i y és la coordenada y de qualsevol dels vèrtexs de l'aresta.

Donada una seqüència d'enters, es defineix la *moda* com l'enter que més vegades surt a la seqüència.

ES DEMANA: Que dissenyeu un algoritme que escriu pel canal de sortida la moda de la superfície dels polígons definits per la seqüència que envia el sensor.

Considereu per exemple les peces de la figura 7.1:

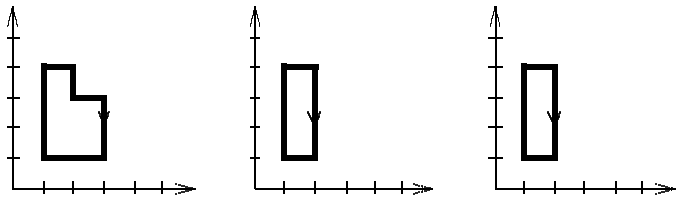


Fig. 7.1 Seqüència de peces

i la seva seqüència associada:

$$1\ 1\ 1\ 4\ 2\ 4\ 2\ 3\ 3\ 3\ 3\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1\ 1\ 4$$
$$2\ 4\ 2\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1\ 1\ 4\ 2\ 4\ 2\ 1\ 1\ 1\ -1\ -1\ -1\ -1$$

llavors el resultat de l'algoritme ha de ser:

$$3$$

7.1.2 Solució proposada

Considerem una seqüència de polígons per tractar acabada amb un sentinella. Llavors, per resoldre aquest problema solament cal fer un recorregut per calcular-ne la moda. No obstant això, per calcular la moda, cal comptabilitzar la freqüència d'aparició de cada superfície: suposem que existeix un objecte que s'encarrega d'aquesta feina. Llavors, un primer nivell de l'anàlisi fóra:

```
[0]   var
[1]       p : tPoligon
[2]       tf : tTauFreq
[3]   fvar

[4]   Inicialitzar (tf)
[5]   p := ObtenirPoligon ()
[6]   mentre no SentinellaP (p) fer
[7]       Acumular (Superficie (p), tf)
[8]       p := ObtenirPoligon ()
[9]   fmentre
[10]  EscriureEnter (Moda (tf))
```

Òbviament, el tipus "tPoligon" representa un polígon de la seqüència i el tipus "tTauFreq" representa a la taula on s'emmagatzemen les freqüències de les superfícies. Per altra banda, les operacions emprades responen a les especificacions següents:

```
[11]  accio Inicialitzar (entsor t : tTauFreq )
[12]  {Post:   Inicialitza a buit la taula t }

[13]  {Prec:   t és una taula inicialitzada }
[14]  accio Acumular (ent s : enter, entsor t : tTauFreq )
[15]  {Post:    Acumula a la taula de freqüències t l'aparició de la superfície s. }

[16]  {Prec:   t és una taula inicialitzada }
[17]  funcio Moda (ent t : tTauFreq ) retorna enter
[18]  {Post:    Retorna la moda de la taula de freqüències tf. Si hi ha superfícies
      diferents amb les mateixes aparicions retorna qualsevol d'elles. Si la taula de
      freqüències és buida, el resultat és indefinit. }

[19]  funcio ObtenirPoligon () retorna tPoligon
[20]  {Post:    Obté el polígon següent llegint-lo del canal estàndard d'entrada. }

[21]  funcio SentinellaP (ent p : tPoligon ) retorna boolea
[22]  {Post:    Retorna cert si i només si p és el polígon sentinella de la seqüència. }

[23]  funcio Superficie (ent p : tPoligon ) retorna enter
[24]  {Post:    Donat un polígon no sentinella, retorna la seva superfície. }
```

Discutim quina ha de ser la representació d'un polígon. A partir de les operacions especificades, podem deduir que un objecte polígon solament ha de contenir les informacions següents:

- La superfície del polígon representat

- La propietat de ser el polígon sentinella

A partir d'aquest fet, podem representar un polígon mitjançant una tupla definida com:

```
[25]  tPoligon = tupla
[26]      s : enter
[27]      f : boolea
[28]  ftupla
```

on el camp s dóna la superfície del polígon i el camp f s'avalua a **cert** si i només si el polígon és sentinella. Fet, aquest darrer, que permet implementar la funció "SentinellaP" de manera tan senzilla que es deixa com a exercici pel lector.

A partir d'aquesta definició, per obtenir un polígon solament caldrà fer un recorregut per la seqüència de les seves arestes orientades tot aplicant —com a tractament— el mètode per al càlcul de la superfície exposat a l'enunciat. Això condueix a l'algoritme següent:

```
[29]  funcio ObtenirPoligon () retorna tPoligon es
[30]      var
[31]          p : tPoligon
[32]          a : tAresta
[33]          s : enter
[34]      fvar

[35]      a := PrimeraAresta ()
[36]      si
[37]          ArestaSent (a) −→
[38]              p.f := cert
[39]        [] no ArestaSent (a) −→
[40]              s := 0
[41]              mentre no ArestaSent (a) fer
[42]                  AcumulaSup (s, a)
[43]                  a := SeguentAresta (a)
[44]              fmentre
[45]              p.f := fals
[46]              p.s := s
[47]      fsi
[48]      retorna p
[49]  ffuncio
```

Observeu que s'ha considerat el cas en què la primera aresta obtinguda és l'aresta sentinella. En aquest cas, ens trobem davant del polígon buit i el càlcul és immediat. Altrament apliquem el càlcul de la superfície considerant les especificacions següents:

```
[50]  accio AcumulaSup (entsor s : enter, ent a : tAresta )
[51]  {Post:  Incrementa s amb el valor de superfície aportada per a segons l'algoritme
      expressat a l'enunciat. }
[52]  funcio PrimeraAresta () retorna tAresta
[53]  {Post:  retorna la primera aresta del canal d'entrada }
[54]  funcio SeguentAresta (ent a : tAresta ) retorna tAresta
[55]  {Post:  obté l'aresta següent del canal d'entrada. Si no n'hi ha més retorna un
      resultat indefinit. }
[56]  funcio ArestaSent (ent a : tAresta ) retorna boolea
[57]  {Post:  Retorna cert si a és l'aresta sentinella }
```

Si definim una aresta com els dos vèrtexs que la formen mantenint l'ordre horari $(x1, y1) \rightarrow (x2, y2)$ llavors el tipus corresponent pot definir-se de manera natural com:

```
[58]  tAresta = tupla
[59]      x1, y1, x2, y2 : enter
[60]  ftupla
```

Segons això, la implementació de l'acció "AcumulaSup" no és res més que una anàlisi per casos:

```
[61]  accio AcumulaSup (entsor s : enter, ent a : tAresta ) es
[62]      si
[63]          a.y1 = a.y2  ⟶
[64]              si
[65]                  a.x1 ≤ a.x2  ⟶  s := s + a.y1 * (a.x2 − a.x1)
[66]                □ a.x1 > a.x2  ⟶  s := s − a.y1 * (a.x1 − a.x2)
[67]              fsi
[68]        □ a.y1 ≠ a.y2  ⟶
[69]      fsi
[70]  faccio
```

de fet, aquest algoritme és fàcilment simplificable i pot reduir-se al que segueix mitjançant manipulacions algebraiques:

```
[71]  accio AcumulaSup (entsor s : enter, ent a : tAresta ) es
[72]      si
[73]          a.y1 = a.y2  ⟶  s := s + a.y1 * (a.x2 − a.x1)
[74]        [] a.y1 ≠ a.y2  ⟶
[75]      fsi
[76]  faccio
```

Per obtenir la primera aresta simplement cal llegir els primers quatre vèrtexs considerant que podem trobar l'aresta sentinella. Això és:

```
[77]  funcio PrimeraAresta () retorna tAresta es
[78]      var
[79]          a : tAresta
[80]      fvar
[81]      a.x1 := LlegirEnter ()
[82]      a.y1 := LlegirEnter ()
[83]      si
[84]          a.x1 ≠ −1 i a.y1 ≠ −1  ⟶
[85]              a.x2 := LlegirEnter ()
[86]              a.y2 := LlegirEnter ()
[87]        [] a.x1 = −1 o a.y1 = −1  ⟶
[88]              a.x2 := −1; a.y2 := −1
[89]      fsi
[90]      retorna a
[91]  ffuncio
```

Obtenir l'aresta següent i comprovar si és el sentinella es resol fàcilment com:

```
[92]  funcio SeguentAresta (ent a : tAresta ) retorna tAresta es
[93]      var
[94]          b : tAresta
[95]      fvar
[96]      b.x1 := a.x2; b.y1 := a.y2
[97]      b.x2 := LlegirEnter (); b.y2 := LlegirEnter ()
[98]      retorna b
[99]  ffuncio
[100] funcio ArestaSent (ent a : tAresta ) retorna boolea es
[101]     retorna (a.x1 = −1 i a.y1 = −1) o (a.x2 = −1 i a.y2 = −1)
[102] ffuncio
```

Resta implementar la taula de freqüències i les seves operacions. Els objectes de tipus "tTauFreq" han de ser capaços d'emmagatzemar quantes observacions s'han fet de cada superfície i les seves operacions han de permetre anotar una observació més i calcular-ne la moda.

A tal efecte considerem una taula de parelles observacions-superfície amb un apuntador al primer lloc buit. La seva implementació serà:

```
[103]  tipus
[104]      tTauFreq  = tupla
[105]          pbuit : enter
[106]          tf  : tTauSup
[107]      ftupla

[108]      tTauSup  = taula [1..MAXSUPS ] de sup

[109]      sup  = tupla
[110]          s : enter
[111]          n : enter
[112]      ftupla
[113]  ftipus
```

Observeu que és necessari limitar el màxim nombre de superfícies admissibles. En aquest cas, s'admeten fins a MAXSUPS superfícies.

Calcular la moda es redueix a cercar el màxim a la taula:

```
[114]  funcio Moda (ent t : tTauFreq ) retorna enter es
[115]      var
[116]          i : enter
[117]          m : enter
[118]      fvar

[119]      m := 1
[120]      per i en [2..t.pbuit − 1] fer
[121]          si
[122]              t.tf[i].n > t.tf[m].n  −→  m := i
[123]            ▯ t.tf[i].n ≤ t.tf[m].n  −→
[124]          fsi
[125]      fper
[126]      retorna t.tf[m].s
[127]  ffuncio
```

Observeu que la funció "Moda" no està definida per a taules de freqüència amb menys d'un element. Nogensmenys, aquesta és una restricció ben natural tot i que implica que la solució proposada requereix una seqüència no buida de polígons. També cal citar el cas en què hi ha diverses modes possibles. En aquest darrer no s'ha considerat important quina s'escollia.

Per acumular una observació cal comprovar si la superfície ja existia i incrementar les observacions en aquest cas o bé afegir una nova superfície. Val a dir que aquesta operació requereix una cerca sobre la taula que milloraria substancialment en cas que la taula fos ordenada per "superfície". No obstant això, donat que desconeixem el nombre de superfícies amb què ens trobarem, optem per una taula desordenada. Llavors, per acumular fem:

```
[128]  accio Acumular (ent s : enter, entsor t : tTauFreq ) es
[129]      var
[130]          i : enter
[131]      fvar

[132]      i := 1
[133]      mentre i < t.pbuit − 1 i t.tf[i].s ≠ s fer
[134]          i := i + 1
[135]      fmentre

[136]      si
[137]          t.tf[i].s = s  −→  t.tf[i].n := t.tf[i].n + 1
[138]        □ t.tf[i].s ≠ s  −→
[139]                t.tf[t.pbuit].n := 1
[140]                t.tf[t.pbuit].s := s
[141]                t.pbuit := t.pbuit + 1
[142]      fsi
[143]  faccio
```

Com pot observar-se, no s'ha pres cap precaució pel que fa al sobreeiximent de l'estructura de dades. Aquest problema, tot i que és important en una aplicació real, no ha rebut atenció en aquesta col·lecció de problemes.

Finalment per inicialitzar l'estructura solament cal implementar la funció "Inicialitzar". Aquesta senzillament consisteix en la inicialització de l'apuntador a primer buit:

```
[144]  accio Inicialitzar (entsor t : tTauFreq ) es
[145]      t.pbuit := 1
[146]  faccio
```

7.2 La lliga de futbol

7.2.1 Enunciat

Es disposa d'una seqüència de tuples que conté tots els resultats de tots els partits celebrats a la lliga de futbol del país X durant l'any 1990-91. El tipus dels elements de la seqüència és determinat per la definició següent:

```
tipus
    tPartit = tupla
        e1, e2 : tEquip
        g1, g2 : enter
    ftupla
ftipus
```

Noteu que:

- El final de la seqüència ve donat per una tupla sentinella on el camp $e1$ és nul.
- No es coneix a priori quins equips participen ni quants cops ho fan.
- Es disposa d'una funció per fer la lectura de les dades. Aquesta funció està definida tal com segueix:

  funcio LlegirElement () **retorna** tPartit
  { La primera vegada que és cridada retorna el primer partit de la seqüència. Les altres vegades retorna el següent element de la seqüència. Si es crida després de llegir el darrer element el resultat és indefinit. }

Considereu que un equip suma dos punts per cada partit que guanya, un punt per cada partit que empata i zero punts per aquells partits que perd. Dissenyeu un algoritme que generi un llistat de la classificació dels equips que han participat a la lliga. En aquest llistat, els equips cal que surtin ordenats segons la puntuació que han obtingut a la lliga.

Per exemple, si teniu les dades d'entrada:

```
Arganzuela 3 Pozoblanco 16
Pozonegro 4 Mansilla 2
...
Valdeconejos 8 Mirabete 2
```

llavors la sortida és:

```
Arganzuela 34
Mansilla 32
Valdeconejos 10
Mirabete 9
...
```

7.2.2 Solució proposada

Les nostres dades d'entrada són una seqüència de partits. El tractament per cada partit consisteix a acumular el seu efecte en la classificació. Usant un esquema de recorregut, obtenim:

```
[0]   var
[1]       p : tPartit
[2]       c : tClassificacio
[3]   fvar
[4]   InicialitzaClassificacio (c)
[5]   p := LlegirElement ()
[6]   mentre no NulEquip (p.e1) fer
[7]       si
[8]           p.g1 > p.g2  ⟶ SumarPunts (c, p.e1, 2)
[9]         ▯ p.g1 = p.g2  ⟶
[10]              SumarPunts (c, p.e1, 1)
[11]              SumarPunts (c, p.e2, 1)
[12]        ▯ p.g1 < p.g2  ⟶ SumarPunts (c, p.e2, 2)
[13]      fsi
[14]      p := LlegirElement ()
[15]  fmentre
[16]  OrdenarClassificacio (c)
[17]  EscriureClassificacio (c)
```

Les operacions emprades tenen per especificació:

```
[18]  accio InicialitzaClassificacio (sor c : tClassificacio )
[19]  {Post:   c és una taula de classificacions buida }
[20]  funcio NulEquip (ent e : tEquip ) retorna boolea
[21]  {Post:   Retorna cert si i només si e és un equip nul }
[22]  {Prec:   c està inicialitzada }
[23]  accio SumarPunts (entsor c : tClassificacio , ent e : tEquip , ent p : enter)
[24]  {Post:   Modifica la taula c sumant en el compte de l'equip e p punts }
```

```
[25]  {Prec:   c està inicialitzada }
[26]  accio OrdenarClassificacio (entsor c : tClassificacio )
[27]  {Post:   c és una taula de classificacions ordenada per punts }
[28]  {Prec:   c està inicialitzada }
[29]  accio EscriureClassificacio (ent c : tClassificacio )
[30]  {Post:   Escriu en el canal de sortida el contingut de c }
```

Definim el tipus "tClassificacio" com una taula on s'emmagatzemen un nombre indefinit de puntuacions de cada equip:

```
[31]  tipus
[32]      tElemClassificacio = tupla
[33]          e : tEquip ; p : enter
[34]      ftupla
[35]      tClassificacio = tupla
[36]          t : taula [0..MAX] de tElemClassificacio
[37]          n : enter
[38]      ftupla
[39]  ftipus
```

Amb això podem implementar trivialment:

```
[40]  accio InicialitzaClassificacio (sor c : tClassificacio ) es
[41]      c.n := 0
[42]  faccio
```

Considerant que l'acció "SumarPunts" actualitza els punts de l'equip e dins la classificació c sumant-li p punts i que, si l'equip no existia, el dóna d'alta, tenim:

```
[43]  accio SumarPunts (entsor c : tClassificacio , ent e : tEquip , ent p : enter) es
[44]      var
[45]          i : enter
[46]          trobat : boolea
[47]      fvar
[48]      trobat := fals; i := 0
[49]      mentre i < c.n i no trobat fer
[50]          trobat := c.t[i].e = e
[51]          i := i + 1
[52]      fmentre
```

```
[53]        si
[54]            trobat  ⟶  c.t[i − 1].p := c.t[i − 1].p + p
[55]          □ no trobat  ⟶
[56]                c.t[c.n].p := p
[57]                c.t[c.n].e := e
[58]                c.n := c.n + 1
[59]        fsi
[60]    faccio
```

De la mateixa manera, implementem l'acció "OrdenarClassificacio" i l'acció per escriure el resultat "EscriureClassificacio" com:

```
[61]    accio OrdenarClassificacio (entsor c : tClassificacio ) es
[62]        var
[63]            k, i, j : enter
[64]            x : tElemClassificacio
[65]        fvar
[66]        per i en [0 . . . c.n − 2] fer
[67]            x := c.t[i]; k := i
[68]            per j en [i + 1 . . . c.n − 1] fer
[69]                si
[70]                    c.t[j].e < x.e  ⟶
[71]                        k := j; x := c.t[k]
[72]                  □ c.t[j].e ≥ x.e  ⟶
[73]                fsi
[74]            fper
[75]            c.t[k] := c.t[i]; c.t[i] := x
[76]        fper
[77]    faccio
```

```
[78]    accio EscriureClassificacio (ent c : tClassificacio ) es
[79]        var
[80]            i : enter
[81]        fvar
[82]        per i en [0 . . . c.n − 1] fer
[83]            EscriureEquip (c.t[i].e);
[84]            EscriureEnter (c.t[i].p);
[85]            SaltaLinia ();
[86]        fper
[87]    faccio
```

Noteu que s'han fet algunes concessions:

- S'han suposat l'existència d'operacions pel tipus "tEquip" per
  - — comparar valors
  - — escriure valors pel canal de sortida
- S'ha suposat l'existència d'una operació per enviar un salt de línia al canal de sortida estàndard.

7.3 GroßenPaßten Bank

7.3.1 Enunciat

En el banc GroßenPaßten Bank de Zürich existeix un departament de comptes anònims. El banc posseeix la informació del nom del propietari de cada compte però en tots els documents aquest nom està encriptat i solament el cap de seguretat coneix la manera de desencriptar-lo.

A l'oficina de trameses arriben llistats de comptes amb el format següent:

```
#compte            (enter)
nom encriptat      (cadena de 20 car.  maxim)
saldo              (enter)
```

i el treballadors han d'obtenir, com a resultat de reordenar els anteriors, un llistat amb el format següent:

```
nom encriptat     #compte
```

però amb la dificultat afegida de que els llistats resultants han d'estar ordenats pel nom del propietari (ATENCIÓ: pel nom del propietari i no pel nom encriptat del propietari). Com que els treballadors no poden veure el nom original, es decideix fer un programa que resolgui la qüestió.

Al programador del GroßenPaßten Bank, que és de confiança, se li dóna la informació següent:

- Tots els llistats originals acaben amb les dades d'un compte especial que indica "fi de llistat". Aquest es caracteritza per tenir el número de compte igual a zero.

- Tots els noms, encriptats i desencriptats, estan escrits en lletres majúscules i no contenen espais ni caràcters no alfabètics.
- Per desencriptar un nom s'usa el procediment següent:

  — Cada caràcter té un enter associat segons s'indica a la taula següent:

  $$\begin{array}{lll} \text{A} & \ldots & 0 \\ \text{B} & \ldots & 1 \\ \text{C} & \ldots & 2 \\ & \ldots & \\ \text{Z} & \ldots & 25 \end{array}$$

  — Donat el nom encriptat, obtenim la seqüència d'enters associada:

  $$HQQVKA \rightarrow 8, 17, 17, 22, 11, 1$$

  — Agrupem de dos en dos els enters que hem obtingut

  $$8, 17, 17, 22, 11, 1 \rightarrow \begin{bmatrix} 8 \\ 17 \end{bmatrix}, \begin{bmatrix} 17 \\ 22 \end{bmatrix}, \begin{bmatrix} 11 \\ 1 \end{bmatrix}$$

  Si sobra algun element, aquest serà sempre el darrer de la seqüència obligatòriament i es tracta de forma especial. Noteu que això solament succeeix quan el nombre de caràcters del nom encriptat és senar.

  — Es multipliquen tots els vectors obtinguts per la matriu A considerant que el producte entre enters està definit com $a * b = ((a * b) \bmod 26)$, que la suma es defineix com $a + b = ((a + b) \bmod 26)$ i que A és

  $$A = \begin{bmatrix} 18 & 13 \\ 5 & 8 \end{bmatrix}$$

  Per exemple, considereu la primera parella del punt anterior; per aplicar la matriu A fem:

  $$\begin{aligned} \begin{bmatrix} 18 & 13 \\ 5 & 8 \end{bmatrix} \begin{bmatrix} 8 \\ 17 \end{bmatrix} &= \\ &= \begin{bmatrix} ((18 * 8 \ \mathbf{mod}\ 26) + (13 * 17 \ \mathbf{mod}\ 26)) \ \mathbf{mod}\ 26 \\ \ldots \end{bmatrix} = \\ &= \begin{bmatrix} 1 \\ 20 \end{bmatrix} \end{aligned}$$

  La forma especial de tractar l'element sobrant consisteix a deixar-lo tal com estava.

— La seqüència d'enters obtinguda així ens dóna directament el nom sense encriptar:

$$\begin{bmatrix} 1 \\ 20 \end{bmatrix} \begin{bmatrix} 20 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 11 \end{bmatrix} \rightarrow 1, 20, 20, 1, 3, 11 \rightarrow ATTACK$$

Dissenyeu un algoritme que escrigui el llistat indicat amb tots els clients ordenats segons el seu nom real (que no ha d'aparèixer al llistat).

7.3.2 Solució proposada

L'algoritme que es demana requereix la lectura d'una seqüència de dades referents a clients, l'emmagatzemament de les dades en algun lloc on puguem ordenar-les i la posterior escriptura d'aquestes dades. Segons això, un primer esquema fóra:

```
[0]  algoritme encripta es
[1]      var
[2]          mag : tMagatzem
[3]          dades : tClient
[4]      fvar
[5]      InicialitzaMagatzem (mag)
[6]      LlegirDadesClient (dades)
[7]      mentre no DarreraDada (dades) fer
[8]          EmmagatzemaDades (dades, mag)
[9]          LlegirDadesClient (dades)
[10]     fmentre
[11]     OrdenaMagatzem (mag)
[12]     LlistaMagatzem (mag)
[13] falgoritme
```

Vegeu que s'han usat dos tipus de variables diferents, una on es guarden les dades d'un client en particular (*dades*) i una on es guarda la informació de tots els clients llegits (*mag*). Cal notar que els tipus usats els deixem per definir quan sigui necessari. Pel que fa al tractament vegeu que s'ha aplicat un esquema de recorregut atès que les dades d'entrada poden considerar-se una seqüència acabada amb un sentinella. L'especificació dels subprogrames usats és:

```
[14] accio InicialitzaMagatzem (sor m : tMagatzem )
[15] {Post:   El magatzem m està buit }
```

```
[16]  accio LlegirDadesClient (sor c : tClient )
[17]  {Post:   c és el client següent del canal d'entrada }

[18]  accio EmmagatzemaDades (ent c : tClient , entsor m : tMagatzem )
[19]  {Post:   Emmagatzema les dades del client c en el magatzem m }

[20]  accio OrdenaMagatzem (entsor m : tMagatzem )
[21]  {Post:   m és un magatzem amb els clients ordenats convenientment segons el
      criteri de l'enunciat }

[22]  accio LlistaMagatzem (ent m : tMagatzem )
[23]  {Post:   S'escriu pel canal de sortida el contingut del magatzem de la manera
      especificada a l'enunciat }

[24]  funcio DarreraDada (ent c : tClient ) retorna boolea
[25]  {  Retorna cert si i només si c és el client sentinella }
```

Continuem refinant les accions que manquen. Per exemple, podem continuar amb l'acció "LlegirDadesClient". Aquesta té un paràmetre de sortida en què ens retorna les dades del client següent o del primer si és la primera vegada que es crida. Per tant, per implementar-la es necessita saber la representació del tipus "tClient". En aquest tipus cal conservar tota la informació que ens cal del client. De cada client arriba el número de compte, el nom encriptat i el saldo. Tot i això, el saldo és irrellevant. En conseqüència, definim:

```
[26]  tipus
[27]      tClient  = tupla
[28]          compte : enter
[29]          nopmc : t_nom
[30]      ftupla
[31]  ftipus
```

on el tipus t_nom encabeix un nom encriptat de les mateixes característiques que les descrites a l'enunciat. Amb això, l'acció que llegeix la informació d'un client és:

```
[32]  accio LlegirDadesClient (sor client : tClient ) es
[33]      var
[34]          aux : enter
[35]      fvar

[36]      client.compte := LlegirEnter ()
[37]      LlegirNom (client.nom)
[38]      aux := LlegirEnter ();
[39]  faccio
```

Noteu que el saldo el llegim però l'obviem ja que no és necessari. Noteu també l'ús de l'acció "LlegirNom" que llegeix un nom encriptat. Ara es pot implementar la funció:

```
[40]  funcio DarreraDada (ent client : tClient ) retorna boolea es
[41]      retorna client.compte = 0
[42]  ffuncio
```

Resolt això, es poden refinar les accions que manipulen les dades dels clients. Abans cal establir la representació del tipus "tMagatzem". De manera habitual fem:

```
[43]  tipus
[44]      tMagatzem  = tupla
[45]          ndades : enter
[46]          tc : taula [0..MAX − 1] de tClient
[47]      ftupla
[48]  ftipus
```

on MAX representa el nombre màxim de clients que poden tractar-se. Amb aquesta definició ja pot tractar-se l'acció "EmmagatzemaDades". Considereu que aquesta té un paràmetre d'entrada sortida que és el magatzem i un paràmetre d'entrada que és el client per emmagatzemar. Si no es té en compte el possible sobreeiximent del magatzem, la implementació és:

```
[49]  accio EmmagatzemaDades (entsor m : tMagatzem , ent dades : tClient ) es
[50]      m.tc[m.dades] := dades
[51]      m.ndades := m.ndades + 1
[52]  faccio
```

Per inicialitzar el magatzem, simplement cal

```
[53]  accio InicialitzaMagatzem (sor m : tMagatzem ) es
[54]      m.ndades := 0
[55]  faccio
```

Pel que fa a llistar el contingut del magatzem, la implementació és prou senzilla:

```
[56]  accio LlistaMagatzem (ent m : tMagatzem ) es
[57]      var
[58]          i : enter
[59]      fvar
[60]      per i en [0..m.ndades − 1] fer
[61]          EscriuNom (m.tc[i].nomc)
[62]          EscriuEnter (m.tc[i].compte)
[63]      fper
[64]  faccio
```

Passem a considerar el gruix del problema amb l'acció que ordena les dades del magatzem. Com que aquestes dades han d'estar ordenades usant el nom no encriptat del client, serà necessari desencriptar el nom i ordenar-les segons aquest darrer. De fet, l'únic moment en el qual es fa necessari desencriptar és en el moment de comparar els noms de dos clients per saber quin d'ells és menor. Segons això, i usant una ordenació pel mètode d'inserció directa, fem:

```
[65]  accio OrdenaMagatzem (entsor m : tMagatzem ) es
[66]      var
[67]          aux : tClient
[68]          p, i, min : enter
[69]      fvar

[70]      per p en [0..m.ndades − 2] fer
[71]          min := p
[72]          per i en [p + 1..m.ndades − 1] fer
[73]              si
[74]                  Menor (m.tc[i].nomc, m.tc[min].nomc) −→
[75]                      min := i
[76]              □ no Menor (m.tc[i].nomc, m.tc[min].nomc) −→
[77]              fsi
[78]          fper
[79]          aux := m.tc[p]
[80]          m.tc[p] := m.tc[min]
[81]          m.tc[min] := aux
[82]      fper
[83]  faccio
```

Després d'això solament resta implementar la funció "Menor", que, donats dos noms encriptats, ens retorna **cert** si i només si el primer és menor que el segon. Expressat més formalment:

```
[84]  funcio Menor (ent n1, n2 : tNom ) retorna boolea
[85]  { Retorna cert si i només si n1 desencriptat és menor que n2 desencriptat }
```

Per implementar aquesta acció cal decidir la representació del tipus "tNom". Prenguem com a representació un vector de caràcters amb el conveni que la cadena de caràcters acaba amb un sentinella (caràcter NUL). Segons això, i considerant que l'enunciat diu que el nombre màxim de caràcters és 20, tenim:

```
[86]  tipus
[87]      tNom  = taula [0..20] de enter
[88]  ftipus
```

Per implementar la funció, apliquem literalment el procediment esmentat a l'enunciat:

```
[89]  funcio Menor (ent n1, n2 : tNom ) retorna boolea es
[90]      var
[91]          nd1, nd2 : tNom
[92]      fvar
[93]      Desencripta (n1, nd1)
[94]      Desencripta (n2, nd2)
[95]      retorna MenorCadena (nd1, nd2)
[96]  ffuncio
```

La funció "MenorCadena" retorna **cert** si i només si el primer paràmetre és menor que el segon i tots dos paràmetres són de tipus t_nom amb el contingut desencriptat. L'acció "Desencripta", desencripta el primer paràmetre i assigna el resultat al segon. Segons això, la implementació resulta:

```
[97]  accio Desencripta (ent n1 : tNom , sor n2 : tNom ) es
[98]      var
[99]          i, j : enter
[100]     fvar
[101]     i := Longitud (n1)
[102]     n2[i] := n1[i]
[103]     si
[104]         i mod 2 = 1  −→
[105]             n2[i − 1] := n1[i − 1]
[106]             i := i − 1
```

```
[107]            ▯ i mod 2 ≠ 1 −→
[108]        fsi
[109]        per j en [0..i − 1] pas 2 fer
[110]            Calcula (n1[j], n1[j + 1], n2[j], n2[j + 1])
[111]        fper
[112]    faccio
```

La funció "Longitud" retorna la longitud d'una cadena de caràcters. Pot implementar-se usant un senzill recorregut i per aquest motiu no la implementarem aquí. L'acció "Calcula" desencripta una parella de caràcters segons el mètode proposat per l'enunciat. Cal considerar que per passar del caràcter al codi enter adequat ho podem fer usant l'expressió Codi (c) − Codi ('A'). Amb això tenim,

```
[113]    { Desencripta la parella (c1, c2) i retorna la nova parella a (cr1, cr2) }
[114]    accio Calcula (ent c1, c2 : caracter, sor cr1, cr2 : caracter) es
[115]        var
[116]            cc1, cc2, tmp1, tmp2 : enter
[117]        fvar

[118]        cc1 := Codi (c1) + Codi ('A')
[119]        cc2 := Codi (c2) + Codi ('A')
[120]        tmp1 := (18 ∗ cc1) mod 26
[121]        tmp2 := (13 ∗ cc2) mod 26
[122]        cr1 := Caracter ((tmp1 + tmp2) mod 26)
[123]        tmp1 := (5 ∗ cc1) mod 26
[124]        tmp2 := (8 ∗ cc2) mod 26
[125]        cr2 := Caracter ((tmp1 + tmp2) mod 26)
[126]    faccio
```

On la funció "Caràcter" té un paràmetre i el que fa és retornar el caràcter associat a aquest codi segons la codificació de l'enunciat.

```
[127]    { Retorna el caràcter associat a v seguint la codificació donada a l'enunciat }
[128]    funcio Caracter (ent v : enter) retorna caracter es
[129]        retorna Car (v + Codi ('A'))
[130]    ffuncio
```

Observeu que en aquests darrers algoritmes s'han emprat les funcions "Codi" i "Car". Aquestes funcions s'encarreguen de convertir un caràcter en el seu codi ASCII i viceversa, respectivament.

7.4 Experiments PSI

7.4.1 Enunciat

Un departament de Parapsicologia necessita un programa que calculi la puntuació PSI dels experiments que duu a terme. En cada experiment, intervenen n subjectes. Cadascun dels n subjectes està aïllat, pensa una paraula qualsevol i prova de transmetre aquesta paraula per telepatia als altres. Passat un cert temps, cadascun dels n subjectes escriu en una targeta una de les paraules que pensa que ha rebut telepàticament.

El resultat de l'experiment es representa per una llista L de $n > 0$ paraules. La puntuació PSI de l'experiment, notada amb ψ, es calcula molt senzillament. Sigui β el nombre de vegades que apareix la paraula més freqüent de la llista L; si no hi ha paraules repetides a L llavors $\beta = 1$. La puntuació ψ de l'experiment representat per L és:

$$\psi = \frac{\beta}{n}$$

El programa rebrà pel CEE la llista de paraules separades per un o més blancs, salts de línia o tabuladors i acabada amb el caràcter '#'. Les paraules, tret de la paraula especial, són subseqüències de lletres minúscules de longitud $\leq$ MAXLONG, on MAXLONG $= 15$. El nombre de paraules no repetides d'un experiment mai no serà superior a MAXSUBJ, on MAXSUBJ $= 30$.

El programa haurà d'imprimir pel canal estàndard de sortida la puntuació ψ de l'experiment representat per l'entrada.

7.4.2 Solució proposada

L'estructura fonamental de l'algoritme es recolza en el realitzar un recorregut de la seqüència de mots que les persones escriuen. S'entén que un mot és una seqüència de lletres minúscules. Cada mot se separa del següent per un o més blancs, tabuladors o salts de línia. El final de seqüència és donat pel caràcter especial '#'. Aquest caràcter, però, no queda clar si sempre es troba separat de la darrera paraula amb algun caràcter separador o bé sempre és el darrer caràcter de la darrera paraula. A fi i efecte que seqüències amb un o altre conveni funcionin correctament, cal estudiar una solució comúna.

Sigui $\omega_0 \sqcup \omega_1 \sqcup \ldots \sqcup \omega_n$ la seqüència de mots d'entrada, on el caràcter '#' indica un separador. Fixant-se en l'acabament de la seqüència es distingeixen dos casos:

$$\begin{cases} \ldots \omega_n \sqcup \text{'\#'} \\ \ldots \omega_n \text{'\#'} \end{cases}$$

Cal trobar un criteri de finalització de la seqüència que englobi ambdós casos des d'un punt de vista uniforme. A tal efecte, convenim que el caràcter de marca '#' sempre ve a continuació de la darrera paraula sense separadors pel mig. Per altra banda, quan ens trobem amb el cas $\ldots\omega_n\llcorner$'#', aplicarem el mateix criteri: ara la darrera paraula ja no és ω_n sinó la paraula nul·la (aquella que té longitud zero).

Aquest nou conveni que s'acaba d'exposar pot afectar el tractament aplicat als elements recorreguts: si ω_j és una paraula nul·la, no s'ha de tractar. Tot i això, aquest fet no és un inconvenient greu com es veurà tot seguit.

Per altra banda, la seqüència per tractar usa com a sentinella la darrera paraula. Però aquesta paraula, en general, també ha de ser tractada. Això condueix a un esquema com el següent:

```
[0]   var
[1]       el : tElementLlista
[2]       n : enter
[3]       tf : tTaulaFreq
[4]   fvar
[5]   tf := InicialitzarTauFreq ()
[6]   el := ObtenirElement ()
[7]   n := 0
[8]   mentre no Sentinella (el) fer
[9]       n := n + 1
[10]      AugmentarFrequencia (tf, Paraula (el))
[11]      el := ObtenirElement ()
[12]  fmentre
[13]  {Post:    Tractats tots els mots llevat del sentinella }
[14]  { Tractem el sentinella segons si és un mot nul o no }
[15]  si
[16]      Nula (Paraula (el))  −→
[17]    □ no Nula (Paraula (el))  −→
[18]          n := n + 1
[19]          AugmentarFrequencia (tf, el)
[20]  fsi
[21]  {  Calculem el factor PSI i l'escrivim  }
[22]  EscriureReal (MajorFreq (tf)/real(n))
```

Observeu que, en escriure aquest algoritme, s'ha suposat l'existència d'alguns tipus de dades i també d'un conjunt d'operacions sobre ells. El tipus "tElementLlista" representa cada un dels elements que poden obtenir-se del canal d'entrada. És a dir, les paraules resultants de l'experiment. Les operacions aplicables es defineixen com segueix fent ús del tipus "tParaula"

per representar un mot:

```
[23]  {Prec:   El canal d'entrada conté algun element }
[24]  funcio ObtenirElement () retorna tElementLlista
[25]  {Post:   Retorna l'element següent de la llista del canal d'entrada }

[26]  {Prec:   el és un tElementLlista inicialitzat }
[27]  funcio Paraula (ent el : tElementLlista ) retorna tParaula
[28]  {Post:   Retorna la paraula associada a el }

[29]  {Prec:   el és un tElementLlista inicialitzat }
[30]  funcio Sentinella (ent el : tElementLlista ) retorna boolea
[31]  {Post:   Retorna cert si i només si el és el sentinella }
```

L'altre tipus utilitzat és "tTaulaFreq". Aquest representa una taula de freqüències indexada per elements de tipus "tParaula" i dotada de les operacions següents:

```
[32]  funcio InicialitzarTaulaFreq () retorna tTaulaFreq
[33]  {Post:   Retorna una taula on tots els mots tenen freqüència zero }

[34]  {Prec:   t és un "tTaulaFreq" inicialitzat }
[35]  accio AugmentarFrequencia (entsor t : tTaulaFreq , ent p : tParaula )
[36]  {Post:   La freqüència de p a t augmenta en 1 }

[37]  {Prec:   t és un "tTaulaFreq" inicialitzat que conté almenys un mot}
[38]  funcio MajorFreq (ent t : tTaulaFreq ) retorna real
[39]  {Post:   Retorna la freqüència corresponent al mot més freqüent}
```

Per últim també s'utilitza una operació sobre "tParaula" que definim com:

```
[40]  {Prec:   p és una paraula inicialitzada correctament}
[41]  funcio Nula (ent p : tParaula ) retorna boolea
[42]  {Post:   Retorna cert ⟺ p te longitud nul·la}
```

Definides aquestes operacions, cal continuar decidint quina és la representació pel tipus "tElementLlista". Si ens fixem en les operacions que l'utilitzen, observem que cal poder respondre a dues interrogacions fonamentals: el mot associat a l'element i el fet que l'element sigui el sentinella. En base a això s'escull la representació següent:

```
[43]  const
[44]      MAXLONG : enter = 15
[45]      MAXSUBJ : enter = 30
[46]  fconst

[47]  tipus
[48]      { el '.' actua de sentinella }
[49]      tParaula = taula [0..MAXLONG] de caracter
[50]      tElementLlista = tupla
[51]          p : paraula
[52]          s : boolea    { cert si i només si és sentinella }
[53]      ftupla
[54]  ftipus
```

A partir d'aquí es poden implementar les operacions d'aquest tipus. Considereu "ObtenirElement". En aquest cas cal procedir seqüencialment fent saltar els separadors i obtenir el mot que els segueix. Cal tenir en compte que el mot pot ser nul i/o ser el sentinella. Amb aquestes consideracions escrivim:

```
[55]  funcio ObtenirElement () retorna tElementLlista es
[56]      var
[57]          c : caracter
[58]          i : enter
[59]          el : tElementLlista
[60]      fvar

[61]      { Fem un recorregut saltant els separadors }
[62]      c := LlegirCaracter ()
[63]      mentre Separador (c) fer
[64]          c := LlegirCaracter ()
[65]      fmentre
[66]      { 'c' conté el primer caracter no separador }

[67]      { Recorrem la seq. considerant que '#' pot actuar de separador }
[68]      i := 0
[69]      mentre c ≠ '#' i no Separador (c) fer
[70]          el.p[i] := c
[71]          i := i + 1
[72]          c := LlegirCaracter ()
[73]      fmentre
[74]      { c és marca final o separador }

[75]      el.p[i] := '.'    { Marquem final del mot }
[76]      el.s := c = '#'    { Comprovem si és el mot sentinella }
```

```
[77]      retorna el
[78]  ffuncio
```

La funció "Separador" és trivialment implementable com:

```
[79]  funcio Separador (ent c : caracter) retorna boolea es
[80]      retorna c = ' ' o c = 'SaltLinia' o c = 'Tabulador'
[81]  ffuncio
```

La resta de funcions que treballen sobre "tElementLlista" i "tParaula" són simples d'implementar:

```
[82]  funcio Sentinella (ent e : tElementLlista ) retorna boolea es
[83]      retorna el.s
[84]  ffuncio

[85]  funcio Paraula (ent l : tElementLlista ) retorna tParaula es
[86]      retorna l.p
[87]  ffuncio

[88]  funcio Nula (ent p : tParaula ) retorna boolea es
[89]      retorna p[0] = '.'
[90]  ffuncio
```

Solament manca implementar allò que té relació amb el tipus "tTaulaFreq". A tal efecte, cal decidir la seva representació. Essencialment es tracta d'una taula que associa mots a enters (la seva freqüència). Usant una taula d'aquests elements, que anomenarem elements del tipus "tElementTaula", i un apuntador al primer forat buit, tenim:

```
[91]  tipus
[92]      tElementTaula  = tupla
[93]          p : tParaula
[94]          n : enter
[95]      ftupla
[96]      taux  = taula [0..MAXSUBJ − 1] de tElementTaula
[97]      tTaulaFreq  = tupla
[98]          t : taux
[99]          pb : enter    { Apunta al primer lloc buit }
[100]     ftupla
[101] ftipus
```

Tot i que la taula és molt petita, ni que solament sigui per qüestió de complicar les coses, considerarem que es manté ordenada usant com a clau el mot. D'aquesta manera, aconseguirem fer més eficient l'accés mitjançant una cerca binària.

Amb aquestes consideracions, podem escriure de manera molt simple la funció següent:

```
[102]  funcio InicialitzarTaulaFreq () retorna tTaulaFreq es
[103]      var
[104]          tf : tTaulaFreq
[105]      fvar

[106]      tf.pb := 0
[107]      retorna tf
[108]  ffuncio
```

Per calcular quina de les freqüències és la màxima, utilitzem el recorregut tradicional:

```
[109]  { tf conté almenys un mot }
[110]  funcio MajorFreq (ent tf : tTaulaFreq ) retorna real es
[111]      var
[112]          i, max : enter
[113]      fvar

[114]      max := tf.t[0].n
[115]      per i en [1..tf.pb − 1] fer
[116]          si
[117]              tf.t[i].n > max  −→  max := tf.t[i].n
[118]            ▯ tf.t[i].n ≤ max  −→
[119]          fsi
[120]      fper
[121]      retorna real(max)
[122]  ffuncio
```

Per dur a terme l'acció que augmenta la freqüència d'aparició solament cal tenir en compte que, si el mot en qüestió no existeix, llavors cal inserir-lo en un lloc pertinent per mantenir l'ordenació. A tal efecte, definirem la funció "Localitzar" que ajudi en aquest menester. Després d'aquestes consideracions tenim:

```
[123]  { Si p és paraula nova, tf té prou espai }
[124]  accio AugmentarFrequencia (entsor tf : tTaulaFreq , ent p : tParaula ) es
[125]      var
[126]          pos : enter
[127]      fvar
[128]      pos := Localitzar (tf, p)
[129]      si
[130]          pos = tf.pb  −→
[131]              { p és nou element que cal afegir al final }
[132]              tf.t[pos].p := p
[133]              tf.t[pos].n := 1
[134]              tf.pb := tf.pb + 1
[135]        □ pos ≠ tf.pb  −→
[136]              si
[137]                  tf.t[pos].p = p  −→
[138]                      { p és una paraula existent }
[139]                      tf.t[pos].n := tf.t[pos].n + 1
[140]                □ tf.t[pos].p ≠ p  −→
[141]                      { p és una paraula inexistent que cal inserir a pos }
[142]                      BuidarPos (tf, pos)
[143]                      tf.t[pos].p := p
[144]                      tf.t[pos].n := 1
[145]                      tf.pb := tf.pb + 1
[146]              fsi
[147]      fsi
[148]  faccio
```

La funció que localitza una paraula a la taula té una doble funció: trobar la posició d'una paraula existent o bé indicar la posició on s'ha d'inserir una nova paraula. La seva especificació és donada per:

```
[149]  {Prec:   tf està inicialitzada }
[150]  funcio Localitzar (ent tf : tTaulaFreq , ent p : tParaula ) retorna enter
[151]  {Post:   Retorna la posició i on cal inserir p en cas que p ∉ tf o bé la posició i
       que conté p en cas que p ∈ tf }
```

Quant a l'acció "BuidarPos", la seva especificació és:

```
[152]  {Prec:   p és en el rang de definició de tf }
[153]  accio BuidarPos (entsor tf : tTaulaFreq , ent p : enter)
[154]  {Post:   Buida la posició p fent un desplaçament del contingut}
```

El primer subprograma l'implementem usant un esquema clàssic de cerca binària. És discutible si des del punt de vista de l'eficiència aquesta és una decisió correcta. Tal volta, donat el nombre màxim d'elements d'una taula de freqüències, una cerca lineal hauria estat més eficient. Tot i això, l'hem implementat mitjançant una cerca binària il·lustrant així l'ús d'un esquema tant important. A tal efecte suposarem que els elements de tipus "tParaula" estan dotats d'operacions de comparació booleanes. En cas contrari, aquestes haurien d'implementar-se. A tal efecte tenim:

```
[155]  funcio Localitzar (ent tf : tTaulaFreq , ent p : tParaula ) retorna enter es
[156]      var
[157]          m, e, d : enter
[158]      fvar

[159]      e := 0; d := tf.pb
[160]      {Inv:    (∀k : 0 ≤ k < e : tf.t[k] < p) ∧ (∀k : d ≤ k < tf.pb : tf.t[k] ≥ p) }
[161]      {Fita:   d − e }
[162]      mentre e < d fer
[163]          m := (e + d) div 2
[164]          si
[165]              tf.t[m].p < p  ⟶  l := m + 1
[166]            ▯ tf.t[m].p ≥ p  ⟶  d := m
[167]          fsi
[168]      fmentre

[169]      retorna d
[170]  ffuncio
```

La darrera acció es resol simplement fent un recorregut per la taula de freqüències i desplaçant totes aquelles posicions superiors a la de la cel·la on cal fer lloc. Com que el recorregut es fa sobre una taula i el rang d'aquest és ben establert, usem una construcció **per**:

```
[171]  accio BuidarPos (entsor tf : tTaulaFreq , ent p : enter) es
[172]      var
[173]          i : enter
[174]      fvar

[175]      per i en [tf.pb − 1..p] fer
[176]          tf.t[i + 1] := tf.t[i]
[177]      fper
[178]  faccio
```

7.5 Facturació d'una empresa telefònica

7.5.1 Enunciat

Una empresa telefònica vol calcular el percentatge de trucades internacionals d'un abonat respecte del seu nombre total de trucades.

Cada trucada és representa mitjançant una seqüència de caràcters de longitud variable (la seqüència de dígits del telèfon al qual es truca). Hi ha com a màxim 15 dígits per trucada. Les trucades d'un abonat se separen entre si pel caràcter '@'. Les trucades internacionals es distingeixen de la resta perquè comencen amb el prefix `07`. La darrera trucada de la seqüència té longitud zero. Com a mínim hi haurà una trucada (sense comptar la sentinella), però el nombre màxim de trucades és desconegut.

Dissenyeu un algoritme que llegeix pel canal d'entrada estàndard la seqüència de trucades d'un abonat i n'escriu el nombre total de trucades i el percentatge de trucades internacionals. Per exemple, si l'entrada és:

```
2106547@900343434@07491882105 6@922230967@091@0788976542 1@@
```

el resultat és: `6 33.33`, ja que l'entrada té 6 trucades en total i 2 trucades són internacionals.

7.5.2 Solució proposada

Si considerem la seqüència formada per trucades, llavors el problema es redueix a comptar quants elements té la seqüència i quants d'aquests elements són trucades internacionals. Si suposem el tipus "tTrucada" com a existent, llavors un esquema de recorregut senzill resol el problema:

```
[0]   var
[1]       t : tTrucada
[2]       nt, nti : real
[3]   fvar
[4]   nt := 0.0; nti := 0.0
[5]   t := ObtenirTrucada ()
[6]   mentre no TrucadaSentinella (t) fer
[7]       nt := nt + 1.0
[8]       si
[9]           Internacional (t) ⟶ nti := nti + 1.0
[10]        □ no Internacional (t) ⟶
```

```
[11]        fsi
[12]        t := ObtenirTrucada (t)
[13]    fmentre
[14]    EscriureReal (100.0 * nti/nt)
```

L'especificació corresponent a cada un dels subprogrames emprats és la següent:

```
[15]    {Prec:   S = t_0@β }
[16]    funcio ObtenirTrucada () retorna tTrucada
[17]    {Post:   S = β ∧ retorna t_0 }

[18]    funcio TrucadaSentinella (ent t : tTrucada ) retorna boolea
[19]    {Post:   retorna cert ⟺ |t| = 0 }

[20]    funcio Internacional (ent t : tTrucada ) retorna boolea
[21]    {Post:   retorna cert ⟺ t = x_0 x_1 δ ∧ x_0 = '0' ∧ x_1 = '7' }
```

El tipus "tTrucada" emmagatzema aquella informació necessària per implementar les funcions anteriors. Per aquest motiu, únicament cal que emmagatzemi informació dels dos primers dígits del número telefònic. En cas que el número tingui menys de dos dígits, aquests seran els emmagatzemats. Una implementació possible és:

```
[22]    tipus
[23]        tTrucada = taula [0..2] de caracter
[24]    ftipus
```

Per delimitar el prefix telefònic dins la taula, convenim que el caràcter '@' fa de sentinella. És important notar que aquest sentinella i la marca final de trucada en la seqüència d'entrada són dues entitats distintes tot i que coincideixin en el caràcter usat.

Amb aquestes definicions, pot implementar-se la funció "ObtenirTrucada" senzillament fent un recorregut de la seqüència d'entrada i obtenint-ne el dos primers caràcters de la primera trucada o, en cas que n'hi hagi menys, tots els caràcters que la conformen.

```
[25]    funcio ObtenirTrucada () retorna tTrucada es
[26]        var
[27]            c : caracter
[28]            i : enter
[29]            t : tTrucada
[30]        fvar
```

```
[31]        { Obtenim el primer dígit del mot }
[32]        c := LlegirCaracter ()
[33]        i := 0
[34]        mentre c ≠ '@' i i < 2 fer
[35]            t[i] := c
[36]            i := i + 1
[37]            c := LlegirCaracter ()
[38]        fmentre
[39]        { c = '@' ∨ i = 2 }
[40]        { Restaurem estat seqüència d'entrada }
[41]        mentre c ≠ '@' fer c := LlegirCaracter () fmentre
[42]        { c = '@' }
[43]        { Afegim sentinella a la trucada }
[44]        t[i] := c
[45]        retorna t
[46]    ffuncio
```

Les dues funcions que manquen són molt planeres i, fonamentalment, l'únic que fan és veure si la representació d'una trucada compleix certes propietats.

```
[47]    funcio Internacional (ent t : tTrucada ) retorna boolea es
[48]        retorna t[0] = '0' i t[1] = '7'
[49]    ffuncio

[50]    funcio TrucadaSentinella (ent t : tTrucada ) retorna boolea es
[51]        retorna t[0] = '@'
[52]    ffuncio
```

Per acabar, solament cal incidir en el fet que, tot i no semblar-ho, la implementació de la funció "Internacional" té en compte el cas en què la trucada té menys de dos dígits. En efecte, si és així, el sentinella '@' ha d'ocupar la posició 0 o 1 de t i per tant la funció retorna **fals**.

Bibliografia

[1] Castro, Jorge. **Ejercicios de Programación**. CPDA-ETSEIB núm. 698. Barcelona 1991.

[2] Castro, J.; Cucker, F.; Messeguer, X.; Rubio, A.; Solano, L.; Valles, B.; **Curs de Programació**. McGraw-Hill. Barcelona 1992.

[3] Col·lectiu de professors de l'assignatura d'IB1-ETSEIB. **Apunts d'Informàtica Bàsica I**. CPDA-ETSEIB. Barcelona 1988–92.

[4] Col·lectiu de professors de l'assignatura d'IB1-ETSEIB. **Problemes d'Informàtica Bàsica I**. CPDA-ETSEIB. Barcelona 1992.

[5] Col·lectiu de professors de l'assignatura d'INIPRO-FIB. **Problemes d'Iniciació a la Programació**. Documentació interna. Barcelona 1992.

[6] Gries, David. **The science of programming**. Springer-Verlag. New York 1981.

[7] Soto, Antoni. **Llenguatge Algorísmic**. CPDA-ETSEIB. Barcelona 1993.

"Demostrat que hi ha una cuina catalana pròpia, especial, completa, per a tots els gustos i tots els règims d'alimentació i dictades algunes regles de composició i cocció, queda satisfet l'autor i més si ha aconseguit escriure amb claredat i senzillesa, tal com pertoca a la naturalesa de la matèria."

FERRAN AGULLÓ, Llibre de la cuina catalana 2a ed. (1933)