

Pràctica de PRAP

Doble Laberint

Siegfried-Angel Gevatter Pujals

Grup 12

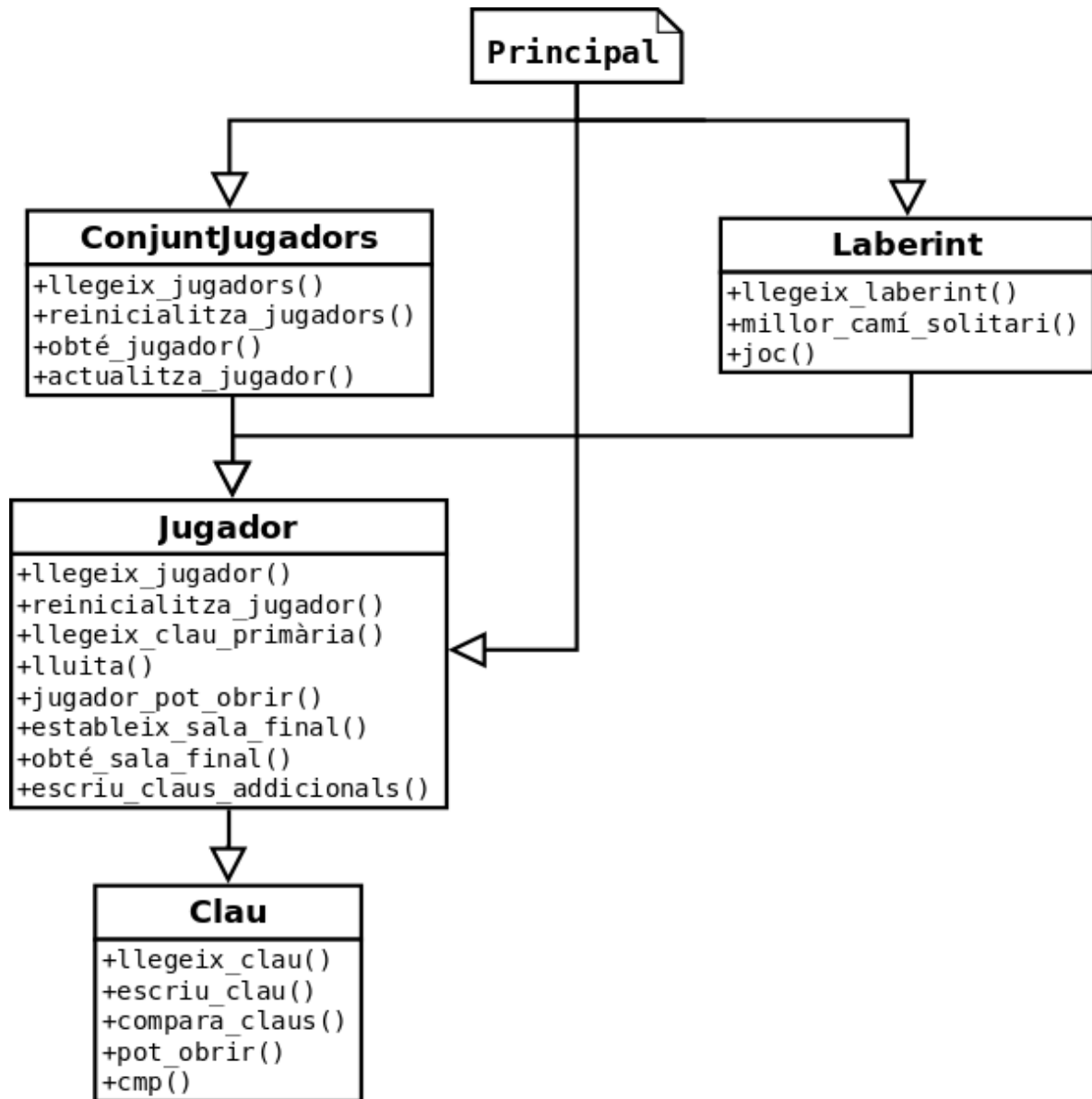
Juan Luis Esteban

19/05/2010

Índex de continguts

1. Diagrama modular.....	3
2. Especificació dels mòduls.....	4
2.1. Mòdul ConjuntJugadors.....	4
2.2. Mòdul Jugador.....	4
2.3. Mòdul Clau.....	6
2.4. Mòdul Laberint.....	6
3. Programa principal.....	8
4. Implementació dels mòduls.....	9
4.1. El mòdul ConjuntJugadors.....	9
Dades.....	9
Operacions.....	9
4.1.1. L'operació llegeix_jugadors.....	9
4.1.2. L'operació reinicialitza_jugadors.....	9
4.1.3. L'operació obté_jugador.....	9
4.2. El mòdul Jugador.....	10
Dades.....	10
Operacions.....	10
4.2.1. L'operació llegeix_jugador.....	10
4.2.2. L'operació reinicialitza_jugador.....	10
4.2.3. L'operació llegeix_clau_primària.....	11
4.2.4. L'operació lluita.....	11
Operacions privades.....	11
4.2.5. L'operació jugador_pot_obrir.....	11
4.2.6. L'operació estableix_sala_final.....	12
4.2.7. L'operació obté_sala_final.....	12
4.2.8. L'operació escriu_claus_addicionals.....	12
4.3. El mòdul Clau.....	13
Dades.....	13
Operacions.....	13
4.3.1. L'operació llegeix_clau.....	13
4.3.2. L'operació escriu_clau.....	13
4.3.3. L'operació compara_claus.....	13
4.3.4. L'operació pot_obrir.....	14
4.3.4. L'operació cmp.....	14
4.4. El mòdul Laberint.....	15
Estructures de dades (privades).....	15
Dades.....	15
Operacions.....	15
Operacions privades compartides entre varies operacions d'aquest mòdul.....	15
4.4.1. L'operació llegeix_laberint.....	16
Operacions privades.....	16
4.4.2. L'operació millor_camí_solitari.....	17
Operacions privades.....	17
4.4.2. L'operació joc.....	19
Operacions privades.....	20
4.5. Detalls addicionals.....	24
4.5.1. L'algorisme de cerca per bombolla.....	24

1. Diagrama modular



Possible disseny alternatiu:

- S'hauria pogut unir **ConjuntJugadors** i **Jugadors** en una sola classe per estalviar haver de recuperar (`obté_jugador`) i tornar a desar (`actualitza_jugador`) el jugador. D'altra banda, llavors caldrien dues variables (un **ConjuntJugadors** i un identificador natural) per a actuar sobre un jugador, en lloc de només una (**Jugador**).

2. Especificació dels mòduls

2.1. Mòdul *ConjuntJugadors*

Especificació

Sobre *Jugador*.

Tipus *ConjuntJugadors*

{ Descripció: Conté tots els jugadors, cadascun associat amb un identificador natural entre 1 i N. }

Operacions

Funció *llegeix_jugadors* () retorna **c**: *ConjuntJugadors*

{ Pre: cert }

{ Post: **c** conté tots els jugadors, cadascun associat amb el seu identificador. }

Acció *reinicialitza_jugadors* (e/s **c**: *ConjuntJugador*)

{ Pre: cert }

{ Post: tots els jugadors de **c** han estat reinicialitzats al seu estat inicial. }

Funció *obté_jugador* (**c**: *ConjuntJugadors*, **i**: natural) retorna **j**: *Jugador*

{ Pre: **i** és un enter entre 1 i N. }

{ Post: **j** és el jugador identificat amb el natural **i**. }

Acció *actualitza_jugador* (e/s **c**: *ConjuntJugadors*, **i**: natural, **j**: *Jugador*)

{ Pre: **i** és un enter entre 1 i N. }

{ Post: substitueix, al conjunt de jugadors **c**, el jugador identificat pel natural **i** pel nou jugador **j**. }

2.2. Mòdul *Jugador*

Especificació

Sobre *Clau*.

Tipus *Jugador*

{ Descripció: Un jugador guarda un conjunt de claus, una de les quals és la primària. També guarda un natural que servirà per emmagatzemar la sala final a la qual arribi el jugador al simular una partida. }

Operacions

Funció llegeix_jugador () retorna **j**: Jugador

{ Pre: cert }

{ Post: **j** és un Jugador inicialitzat al seu estat inicial i amb una clau primària llegida de l'entrada estàndard. }

Acció reinicialitza_jugador (e/s **j**: Jugador)

{ Pre: cert }

{ Post: **j** ha estat re-inicialitzat al seu estat inicial. }

Acció llegeix_clau_primària (e/s **j**: Jugador)

{ Pre: cert }

{ Post: el jugador **j** té una nova clau primària llegida de l'entrada estàndard. }

Acció lluita (e/s **j**: Jugador, e/s **a**: Jugador, s **m**: natural, s **p**: natural, s **i**: natural)

{ Pre: cert }

{ Post: **m** té com a valor el nombre d'índex de la clau del jugador **j** superiors als de la clau del seu adversari **a**, **p** té com a valor el nombre d'índex que són pitjors i **i** la quantitat d'índexs iguals. A més, les claus del perdedor han estat transferides al seu contrincant. }

Funció jugador_pot_obrir (**j**: Jugador, **pany**: Clau) retorna **b**: booleà

{ Pre: cert }

{ Post: **b** és cert si el jugador pot obrir el **pany** donat. }

Acció estableix_sala_final (e/s **j**: Jugador, **sala**: natural)

{ Pre: cert }

{ Post: la **sala_final** del jugador **j** és **sala**. }

Funció obté_sala_final (**j**: Jugador) retorna **sala**: natural

{ Pre: s'ha cridat *estableix_sala_final* com a mínim una vegada des de l'última crida a reinicialitza(). }

{ Post: **sala** és l'identificador de la última sala que s'hagi indicat com a final. }

Acció escriu_claus_addicionals (**j**: Jugador)

{ Pre: cert }

{ Post: les claus addicionals han estat escrites a la sortida estàndard, ordenades lexicogràficament. }

2.3. Mòdul Clau

Especificació

Tipus Clau

{ Descripció: combinació d'una clau o d'un pany. }

Operacions

Funció llegeix_clau () retorna **c**: Clau

{ Pre: cert }

{ Post: **c** és una clau (o un pany) llegida de l'entrada estàndard. }

Acció escriu_clau (**c**: Clau)

{ Pre: cert }

{ Post: la clau **c** han estat escrits a l'entrada estàndard. }

Acció compara_claus (**c**: Clau, **c2**: Clau, **s millor**: natural, **s pitjor**: natural, **s igual** natural)

{ Pre: cert }

{ Post: **millor** té com a valor el nombre d'índex de la clau **c** superiors als de la clau **c1**, **pitjor** té com a valor el nombre d'índex que són pitjors i **igual** la quantitat d'índexs iguals.

Funció pot_obrir (**c**: Clau, **pany**: Clau) retorna **b**: booleà

{ Pre: cert }

{ Post: **b** és cert si la clau **c** pot obrir el pany **p**. }

Funció cmp (**c1**: Clau, **c2**: Clau) retorna **b**: booleà

{ Pre: cert }

{ Post: **b** és cert si la clau **c1** és lexicogràficament superior a la clau **c2**. }

2.4. Mòdul Laberint

Especificació

Sobre *ConjuntJugadors*, *Jugador*.

Tipus Laberint

{ Descripció: conté la descripció d'un laberint, amb les seves sales i portes. A més, un cop s'ha simulat una partida guarda l'identificador del jugador que hagi aconseguit sortir del laberint (si n'hi ha) i informació sobre totes les lluites que hagin tingut lloc. }

Operacions

Funció **llegeix_labirint** () retorna l: Laberint

{ Pre: cert }

{ Post: l és una laberint, amb les seves sales i panys, llegit de l'entrada estàndard. }

Acció **millor_camí_solitari** (**labirint**: Laberint, **j**: Jugador, **n**: natural)

{ Pre: cert }

{ Post: la informació demanada per l'enunciat en quant a la sort del jugador sol al laberint ha estat escrita a la sortida estàndard. }

Acció **joc** (e/s **labirint**: Laberint, e/s **jugadors**: ConjuntJugadors)

{ Pre: cert }

{ Post: llegeix de l'entrada estàndard l'ordre en que els jugador entren al laberint, calcula el resultat del joc i escriu el resultat a la sortida estàndard. }

3. Programa principal

Sobre *Laberint*, *ConjuntJugadors*, *Jugador*.

```
var laberint: Laberint
var jugadors: ConjuntJugadors
var operació: enter

laberint := llegeix_labirint();
jugadors := llegeix_jugadors();

operació := llegeix_enter();
mentre operació != -5:
    si operació = -1:
        joc(laberint, jugadors)
    [] operació = -2:
        var n: natural
        var j: Jugador
        n := llegeix_natural()
        j := obté_jugador(jugadors, n)
        millor_camí_solitari(laberint, j, n)
    [] operació = -3:
        laberint := llegeix_labirint()
    [] operació = -4:
        var n: natural
        var j: Jugador
        n := llegeix_natural()
        j := obté_jugador(jugadors, n)
        llegeix_clau_primària(j)
        actualitza_jugador(jugadors, n, j)
    fsi
operació := llegeix_enter();
fmentre
```

4. Implementació dels mòduls

4.1. El mòdul *ConjuntJugadors*

Sobre *Jugador*.

Dades

- **jugadors**: vector de *Jugador* amb **N** elements.
Nota: El vector consta dels elements 0..N-1, però al tractar amb l'exterior aquests elements s'exposaran amb els identificadors 1..N.

Operacions

4.1.1. L'operació *llegeix_jugadors*

Funció *llegeix_jugadors* () retorna **c**: *ConjuntJugadors*

{ Pre: cert }

```
var i: natural
per a i := 0 fins a N - 1 fes:
    var j: Jugador
    llegeix_jugador(j)
    c.jugadors[i] = j
fper
```

{ Post: **c** conté tots els jugadors, cadascun associat amb el seu identificador. }

4.1.2. L'operació *reinicialitza_jugadors*

Acció *reinicialitza_jugadors* (e/s **c**: *ConjuntJugador*)

{ Pre: cert }

```
var i: natural
per a i := 0 fins a N - 1 fes:
    reinicialitza_jugador(c.jugadors[i])
fper
```

{ Post: tots els jugadors de **c** han estat reinicialitzats al seu estat inicial. }

4.1.3. L'operació *obté_jugador*

Funció *obté_jugador* (**c**: *ConjuntJugadors*, **i**: natural) retorna **j**: *Jugador*

{ Pre: **i** és un enter entre 1 i **N**. }

```
j := c.jugadors[i-1]
```

{ Post: **j** és el jugador identificat pel natural **i**. }

Acció actualitza_jugador (e/s c: ConjuntJugadors, i: natural, j: Jugador)

{ Pre: **i** és un enter entre 1 i N. }

```
c.jugadors[i-1] = j
```

{ Post: substitueix, al conjunt de jugadors **c**, el jugador identificat pel natural **i** pel nou jugador **j**. }

4.2. El mòdul Jugador

Sobre **Clau**.

Dades

- **clau_primària**: Clau.
- **claus_addicionals**: vector de Clau amb N-1 elements.
Justificació: el nombre màxim d'elements és N-1 ja que cada jugador té una clau primària, i en el cas límit que tots els jugadors arribin a la segona part del laberint i competeixin el jugador guanyador se les quedaria totes. Restem 1 ja que la seva pròpia clau es guarda separada, com a clau primària.
- **sala_final**: natural.
- **núm_claus**: natural.
*Nota: Utilitzem aquest índex per portar el compte de quantes claus addicionals ha guanyat el jugador i per tant saber quants elements del vector **claus_addicionals** són vàlids.*

Operacions

4.2.1. L'operació llegeix_jugador

Funció llegeix_jugador () retorna **j**: Jugador

{ Pre: cert }

```
j.sala_final := 0  
j.num_claus := 0  
llegeix_clau_primària(j)
```

{ Post: **j** és un Jugador inicialitzat al seu estat inicial i una la clau primària llegida de l'entrada estàndard. }

4.2.2. L'operació reinicialitza_jugador

Acció reinicialitza_jugador (e/s **j**: Jugador)

{ Pre: cert }

```
j.sala_final := 0  
j.num_claus := 0
```

{ Post: **j** ha estat re-inicialitzat al seu estat inicial. }

4.2.3. L'operació `llegeix_clau_primària`

Acció `llegeix_clau_primària` (e/s `j`: Jugador)

{ Pre: cert }

```
llegeix_clau(j.clau_primària)
```

{ Post: el jugador `j` té una nova clau primària llegida de l'entrada estàndard. }

4.2.4. L'operació `lluita`

Acció `lluita` (e/s `j`: Jugador, e/s `a`: Jugador, s `m`: natural, s `p`: natural, s `i`: natural)

{ Pre: cert }

```
compara_claus(j.clau_primària, a.clau_primària, m, p, i)
si m >= p:
    transfereix_claus(j, a.clau_primària, a.claus_addicionals,
a.núm_claus)
sinó:
    transfereix_claus(a, j.clau_primària, j.claus_addicionals,
j.núm_claus)
```

{ Post: `m` té com a valor el nombre d'índex de la clau del jugador `j` superiors als de la clau del seu adversari `a`, `p` té com a valor el nombre d'índex que són pitjors i `i` la quantitat d'índexs iguals. A més, les claus del perdedor han estat transferides al seu contrincant. }

Operacions privades

Acció `transfereix_claus` (e/s `j`: Jugador, `primària`: Clau, `addicionals`: vector de Clau, `numc`: natural)

{ Pre: cert }

```
j.claus_addicionals[núm_claus] := primària
j.núm_claus := j.núm_claus + 1
var i: natural
per a i := 0 fins a numc - 1 fes:
    j.claus_addicionals[j.núm_claus]
    j.núm_claus := j.núm_claus + 1
fper
```

{ Post: la clau principal i les claus addicionals donades han estat afegides al jugador `j`. }

4.2.5. L'operació `jugador_pot_obrir`

Funció `jugador_pot_obrir` (`j`: Jugador, `pany`: Clau) retorna `b`: booleà

{ Pre: cert }

```
b := fals
// Prova amb la clau primària
```

```

si pot_obrir(j.clau_primària, pany):
    b := cert
// Prova amb les claus addicionals
var i: natural
per a i := 0 fins a j.núm_claus - 1:
    si pot_obrir(j.claus_addicionals[i], pany):
        b := cert

```

{ Post: **b** és cert si el jugador pot obrir el **pany** donat. }

4.2.6. L'operació estableix_sala_final

Acció estableix_sala_final (e/s j: Jugador, sala: natural)

{ Pre: cert }

```
j.sala_final = sala
```

{ Post: la **sala_final** del jugador **j** és **sala**. }

4.2.7. L'operació obté_sala_final

Funció obté_sala_final (j: Jugador) retorna sala: natural

{ Pre: cert }

```
sala := j.sala_final
```

{ Post: **sala** té com a valor la **sala_final** del jugador **j**. }

4.2.8. L'operació escriu_claus_addicionals

Acció escriu_claus_addicionals (j: Jugador)

{ Pre: cert }

```
ordena_claus(j.claus_addicionals)
var i: natural
per a i := 0 fins a núm_claus - 1 fes:
    escriu_clau(j.claus_addicionals[i])
fper

```

{ Post: les claus addicionals han estat escrites a la sortida estàndard, ordenades lexicogràficament. }

- L'acció *ordena_claus* al pseudo-codi anterior correspon a una dotzena de línies que implementen l'algorisme d'ordenació de la bombolla (per la seva simplicitat respecte a d'altres algorismes superiors com ara el de fusió). Es troba explicat al final d'aquest document, ja que s'utilitzarà més d'una vegada.

4.3. El mòdul Clau

Dades

- **c**: vector de naturals amb **M** elements.

Operacions

4.3.1. L'operació *llegeix_clau*

Funció *llegeix_clau* () retorna **c**: Clau

{ Pre: cert }

```
var i: natural
per a i := 0 fins a M - 1 fes:
    c.clau[i] = llegeix_natural()
fper
```

{ Post: **c** és una clau amb M índex llegits de l'entrada estàndard. }

4.3.2. L'operació *escriu_clau*

Acció *escriu_clau* (**c**: Clau)

{ Pre: cert }

```
var i: natural
per a i := 0 fins a M - 1 fes:
    escriu(c.clau[i])
fper
```

{ Post: els índex de la clau **c** han estat escrits a l'entrada estàndard. }

4.3.3. L'operació *compara_claus*

Acció *compara_claus* (**c**: Clau, **c2**: Clau, s **millor**: natural, s **pitjor**: natural, s **igual** natural)

{ Pre: cert }

```
millor := 0
pitjor := 0
igual := 0
var i: natural
per a i := 0 fins a M - 1 fes:
    si c.clau[i] > c2.clau[i]:
        millor := millor + 1
    sinó si c.clau[i] < c2.clau[i]:
        pitjor := pitjor + 1
```

```

sinó:
    igual := igual + 1
fsi
{{ Invariant: millor, pitjor i igual són la quantitat
    d'índexs de les claus donades que són superiors,
    inferiors i iguals (respectivament) considerant
    només els índex del 0 fins a l'i. }}
fper

```

{ Post: **millor** té com a valor el nombre d'índex de la clau **c** superiors als de la clau **c1**, **pitjor** té com a valor el nombre d'índex que són pitjors i **igual** la quantitat d'índexs iguals.

4.3.4. L'operació pot_obrir

Funció **pot_obrir** (**c**: Clau, **pany**: Clau) retorna **b**: booleà

{ Pre: cert }

```

b := 1
var i: natural
per a i := 0 fins a M - 1 fes:
    si c.clau[i] < pany.clau[i]:
        b := 0
    fsi
fper

```

{ Post: **b** és cert si la clau **c** pot obrir el pany **p**. }

4.3.4. L'operació cmp

Funció **cmp** (**c1**: Clau, **c2**: Clau) retorna **b**: booleà

{ Pre: cert }

```

b := 1
var i: natural
var fi: booleà
per a i := 0 fins a M - 1 fes:
    si no fi:
        si c1.clau[i] > c2.clau[i]:
            b := 1
            fi := 1
        sinó si c1.clau[i] < c2.clau[i]:
            b := 0
            fi := 1
        fsi
    fsi
fper

```

{ Post: **b** és cert si la clau **c1** és lexicogràficament superior a la clau **c2**. }

4.4. El mòdul Laberint

Sobre *ConjuntJugadors*, *Jugador*.

Estructures de dades (privades)

- Tupla Sala
 - id_sala**: natural
 - num_esq**: natural
 - num_dre**: natural
 - num_esq_ini**: natural
 - num_dre_ini**: natural
 - ocupada**: booleà
 - pany_esq**: Clau
 - pany_dre**: Clau
- Tupla Lluita
 - id_sala**: natural
 - j1**: natural
 - j2**: natural
 - millor**: natural
 - pitjor**: natural
 - igual**: natural
- Arbre ArbreSala: arbre de **Sala**.

Dades

- **jugador_guanyador**: natural.
- **laberint1**: ArbreSala.
- **laberint2**: ArbreSala.
- **lluites**: vector de Lluita amb N-1 elements.
- **num_lluites**: natural.

Operacions

Operacions privades compartides entre varies operacions d'aquest mòdul

Funció és_sala_final (**arbre**: ArbreSala) retorna **b**: booleà

{ Pre: cert }

```
si és_nul(hi(arbre)) o és_nul(hd(arbre)):  
    b := 1  
sinó:
```

```
b := 0
```

{ Post: **b** és cert si **sala** és una sala final. }

4.4.1. L'operació *llegeix_laberint*

Funció *llegeix_laberint* () retorna l: Laberint

{ Pre: cert }

```
l.num_lluites := 0  
llegeix_arbre_sala(l.laberint1, 0)  
llegeix_arbre_sala(l.laberint2, 1)
```

{ Post: **l** és una Laberint llegit de l'entrada estàndard. }

Operacions privades

Acció *llegeix_arbre_sala* (e/s **arbre**: ArbreSala, **segona_part**: booleà)

{ Pre: cert }

```
var id_sala: natural  
id_sala := llegeix_natural()  
si id_sala > 0:  
  var sala: Sala  
  llegeix_sala(sala, id_sala, segona_part)  
  var esq, dre: ArbreSala  
  llegeix_arbre_sala(esq, segona_part)  
  llegeix_arbre_sala(dre, segona_part)  
  plantar(arbre, sala, esq, dre)  
fsi
```

{ Post: **arbre** és un arbre de sales llegides de l'entrada estàndard. }

Acció *llegeix_sala* (e/s **sala**: Sala, **id_sala**: natural, **segona_part**: booleà)

{ Pre: cert }

```
sala.id = id_sala  
sala.ocupada = 0  
si no segona_part:  
  sala.num_esq_ini = llegeix_natural()  
  sala.num_dre_ini = llegeix_natural()  
fsi  
llegeix_clau(sala.pany_esq)  
si no segona_part:  
  llegeix_clau(sala.pany_dre)  
fsi
```

{ Post: **sala** és una Sala llegida de l'entrada estàndard. }

4.4.2. L'operació millor_camí_solitari

Acció millor_camí_solitari (laberint: Laberint, j: Jugador, n: natural)

{ Pre: cert }

```
j.reinicialitza()
var long, sala_final: natural
long := 1
sala_final := 0
millor_camí(laberint, j, laberint.laberint1, long, sala_final)
si sala_final == 0:
    long := 0
escriu_resultat(n, long, sala_final)
```

{ Post: la informació demanada per l'enunciat en quant a la sort del jugador sol al laberint ha estat escrita a la sortida estàndard. }

- Re-inicialitzem el jugador per evitar que pugui tenir claus addicionals d'un joc anterior. Això no té efecte sobre l'estat del programa, ja que no desem el jugador modificat al *ConjuntJugadors*.

Operacions privades

Acció millor_camí (laberint: Laberint, j: Jugador, arbre: ArbreSala, e/s long: natural, e/s sala_final: natural)

{ Pre: cert }

```
si no és_nul(arbre):
    var sala: Sala
    sala := arrel(arbre)
    si és_sala_final(arbre):
        si jugador_pot_obrir(j, sala.pany_esq)
        o jugador_pot_obrir(j, sala.pany_dre):
            var long_part2: enter
            long_part2 := millor_camí_part2(j,
                laberint.laberint2, sala.id_sala)
            si long_part2 > 0:
                sala_final = sala.id_sala
                l += long_part2
            fsi
        fsi
    sinó:
        var l1, l2, sala1, sala2: natural
        l1 := 1
        l2 := 1
        sala1 := 0
        sala2 := 0
```

```

    {{ H.I.: millor_camí retorna el camí més curt
        cap a una sala final que el jugador donat
        pot recórrer. }}

    si jugador_pot_obrir(j, sala.pany_esq):
        millor_camí(laberint, j, hi(arbre), l1, sala1)
    fsi

    si jugador_pot_obrir(j, sala.pany_dre):
        millor_camí(laberint, j, hd(arbre), l2, sala2)
    fsi

    si sala1 o (sala2 i l2 < l1):
        long := l + l1
        sala_final := sala1
    sinó si sala2:
        long := l + l2
        sala_final := sala2
    fsi
    fsi
fi

```

{ Post: **sala_final** és l'identificador de la sala final a la qual arriba el jugador, o zero si no arriba a cap; **long** és el nombre de sales per les quals ha de passar per arribar-hi. }

Funció millor_camí_part2 (j: Jugador, arbre: ArbreSala, id_sala: natural) retorna **long**: natural

{ Pre: cert }

```

si arrel(arbre).id_sala == id_sala:
    si jugador_pot_obrir(j, arrel(arbre).pany_esq):
        long := 1
    sinó:
        long := 0
sinó:
    long := 0

    {{ H.I.: millor_camí_part2 retorna el nombre de sales
        més una que ens cal recórrer fins arribar a la
        sala final cercada, o bé 0 si des de la sala d'origen
        donada no s'hi pot arribar. }}

    // Prova de trobar la sala final al costat esquerra
    var esq: ArbreSala
    esq := hi(arbre)
    si no és_nul(esq)

```

```

... i jugador_pot_obrir(j, arrel(esq).pany_esq):
    c := millor_camí_part2(j, esq, id_sala)
fsi

si c == 0:
    // No hi ha hagut sort, ara prova-ho a la dreta
    var dre: ArbreSala
    dre := hd(arbre)
    si no és_nul(dre)
        ... i jugador_pot_obrir(j, arrel(dre).pany_esq):
            c := millor_camí_part2(j, dre, id_sala)
        fsi
    fsi

// Hem pogut arribar a la sala final?
si c > 0:
    c := c + 1
fsi

fsi

```

{ Post: **long** és el nombre de sales de la segona part del laberint per les quals ha de passar el jugador per poder sortir-ne. }

4.4.2. L'operació joc

Acció joc (e/s **laberint**: Laberint, e/s **jugadors**: ConjuntJugadors)

{ Pre: cert }

```

var i: natural
var ordre: vector de naturals amb N elements

// Llegeix les dades
per a i := 0 fins a N - 1 fes:
    ordre[i] = llegeix_natural();
fper

// Re-inicialitza l'estat intern
laberint.num_lluites = 0;
reinicialitza_sales(laberint1);
jugadors.reinicialitza();

// Troba els jugadors que arriben a les sales finals
per a i := 0 fins a N - 1 fes:
    var j: Jugador
    j := obté_jugador(jugadors, ordre[i]);

```

```

    var sala: enter
    tria_sala(laberint1, j, sala);
    estableix_sala_final(j, sala);
    actualitza_jugador(jugadors, ordre[i], j);
fper

// Calcula la segona part del laberint
var guanyador: natural
guanyador := troba_guanyador(laberint2, jugadors);

// Escriu el resultat
escriu_guanyador(guanyador)
si guanyador > 0:
    var g: Jugador
    g := obté_jugador(jugadors, guanyador);
    escriu_claus_addicionals(g);
fsi
per a i := 1 fins a N:
    var j: Jugador
    var sala_final: natural
    j := obté_jugador(jugadors, i);
    sala_final := obté_sala_final(j);
    si sala_final:
        escriu_sala_final(i, sala_final)
    fsi
fper
ordena_lluites(laberint)
per a i := 0 fins a laberint.num_lluites - 1:
    var l: Lluita
    l := laberint.lluites[i];
    escriu_lluita(l)
fper

```

{ Post: la informació demanada per l'enunciat en quant al resultat del joc si els jugadors entren en l'ordre llegit de l'entrada estàndard ha estat escrita a la sortida estàndard. }

- L'acció *ordena_lluites* al pseudo-codi anterior correspon a una dotzena de línies que implementen l'algorisme d'ordenació de la bombolla. Es troba explicat al final d'aquest document, ja que s'utilitzarà més d'una vegada.

La funció de comparació és: $a.j1 > b.j2$ o $(a.j1 == b.j1 \text{ i } a.j2 > b.j2)$.

Operacions privades

Acció reinicialitza_sales (e/s **arbre**: ArbreSala)

{ Pre: cert }

```
si no és_nul(arbre):
  var sala: Sala
  var esq, dre: ArbreSala
  sala := arrel(arbre)
  esq := hi(arbre)
  dre := hd(arbre)
  sala.ocupada = 0;
  sala.num_esq = sala.num_esq_ini;
  sala.num_dre = sala.num_dre_ini;
  reinicialitza_sales(esq);
  reinicialitza_sales(dre);
  plantar(arbre, sala, esq, dre);
fsi
```

{ Post: l'arbre **arbre** ha estat inicialitzat per preparar-lo per simular el joc. }

Acció tria_sala (e/s **arbre**: ArbreSala, j: Jugador, e/s **sala_final**: natural)

{ Pre: cert }

```
var sala: Sala
var esq, dre: ArbreSala
sala := arrel(arbre)
esq := hi(arbre)
dre := hd(arbre)

sala_final := 0

si és_sala_final(arbre):
  si no sala.ocupada:
    sala.ocupada = 1
    plantar(arbre, sala, esq, dre)
    sala_final := sala.id_sala
  fsi
sinó
  si jugador_pot_obrir(j, sala.pany_esq) i sala.num_esq > 0
  ... i (no jugador_pot_obrir(j, sala.pany_dre)
  ... o sala.num_dre <= sala.num_esq):
    tria_sala(esq, j, sala_final);
    si sala_final > 0:
      sala.num_esq := sala.num_esq - 1
      plantar(arbre, sala, esq, dre);
    fsi
  sinó si jugador_pot_obrir(j, sala.pany_dre)
  ... i sala.num_dre > 0:
```

```

    tria_sala(dre, j, sala_final);
    si sala_final > 0:
        sala.num_dre := sala.num_dre - 1
        plantar(arbre, sala, esq, dre);
    fsi
    fsi // sinó: el jugador es desintegra
fsi

```

{ Post: **sala_final** és l'identificador de la sala final a la qual arriba el jugador, o bé 0; a més, si el jugador ha arribat a una sala final, **arbre** ha estat actualitzat per reflectir aquest fet (marcant la sala final com a ocupada i decremantant els comptadors de les sales que hi porten). }

Funció troba_guanyador (**laberint**: Laberint, **arbre**: ArbreSala, e/s **jugadors**: ConjuntJugadors) retorna **id_jugador**: natural

{ Pre: cert }

```

id_jugador := 0
si no és_nul(arbre):
    var sala: Sala
    var esq, dre: ArbreSala
    sala := arrel(arbre)
    esq := hi(arbre)
    dre := hd(arbre)
    si és_nul(esq) i és_nul(dre):
        var i: natural
        // Estem en una sala final. Ha arribat aquí
        // algun jugador?
        per a i := 1 fins a N:
            var j: Jugador
            j := obté_jugador(jugadors, i)
            si obté_sala_final(jugador) == sala.id_sala:
                si jugador_pot_obrir(jugador, sala.pany_esq):
                    id_jugador := i
                fsi
            fsi
        fper
    sinó
        var j1, j2: natural
        j1 := troba_guanyador(esq, jugadors)
        j2 := troba_guanyador(dre, jugadors)
        si j1 i j2:
            // Ordena'ls, l'índex menor primer
            si j1 > j2:
                var tmp: natural;
                tmp := j1
                j1 := j2

```

```

        j2 := tmp
    fsi
    // Lluita
    var jugador1, jugador2: Jugador
    var m, p, i: natural;
    jugador1 := obté_jugador(jugadors, j1)
    jugador2 := obté_jugador(jugadors, j2)
    lluita(jugador1, jugador2, m, p, i)
    laberint.lluites[núm_lluites] = {sala.id_sala,
    j1, j2, m, p, i};
    núm_lluites := núm_lluites + 1
    actualitza_jugador(jugadors, j1, jugador1)
    actualitza_jugador(jugadors, j2, jugador2)
    si m > p o (m == p i j1 < j2):
        id_jugador := j1
    sinó:
        id_jugador := j2
    sinó si j1:
        id_jugador := j1
    sinó si j2:
        id_jugador := j2
    fsi
    // Comprova que pugui obrir la porta
    var j: Jugador
    j := obté_jugador(jugadors, id_jugador)
    si no jugador_pot_obrir(j, sala.pany_esq):
        id_jugador = 0;
    fsi
}
}

```

{ Post: **id_jugador** és l'identificador del jugador que guanya la partida, o 0 si cap arriba al final. }

Comencem a la sala que duu a la sortida del laberint, però mitjançant recursivitat avancem fins a cadascuna de les sales connectades amb les sales finals. L'identificador dels jugadors que han arribat a una sala final i que tenen la clau necessària per accedir des d'allà a la segona part del laberint, són retornats a la invocació (de la funció) pare.

Aquesta recull els jugadors que li arriben, que poden ser zero, un o dos (pels camins de l'esquerra i de la dreta). En cas que siguin dos, té lloc una lluita que decideix en favor d'un dels dos, descartant l'altre jugador. Finalment, es comprova si el jugador pot obrir la porta per passar des de la sala de la qual prové a la sala actual; si és així, és retornat. El procediment descrit en aquest paràgraf es repeteix fins que totes les crides retornen. Si l'última retorna l'identificador d'un jugador, aquest és el guanyador.

4.5. Detalls addicionals

4.5.1. L'algorisme de cerca per bombolla

{ Pre: cert }

```
si num_elements > 0:
  var i, j: natural
  per a i := 0 fins a num_elements - 1 fes:
    j := num_elements - 1
    mentre j > i:
      si cmp(elements[j-1], elements[j]):
        intercanvia(elements[j-1], elements[j])
      fsi
      j := j - 1
    fmentre
  fper
fsi
```

{ Post: **elements** és un vector ordenat d'acord amb la funció de comparació *cmp*. }

