PRACTICA de PRO2. Reproducción en el laboratorio

versión 26-mayo-2014

Generado por Doxygen 1.8.2

Viernes, 23 de Mayo de 2014 17:58:20

Índice general

Capítulo 1

PRACTICA de PRO2.

PRIMAVERA 2014.

Los científicos de un laboratorio de investigación biológica desean llevar a cabo una serie de experimentos para estudiar el ciclo de vida de una especie de organismos celulares sencillos.

Los científicos del laboratorio jugaran con organismos de células estructurados en forma de árbol. Cada célula del organismo contiene dos atributos: el identificador (un número natural mayor que zero) y la actividad, ya que una célula puede ser o bien activa (true) o pasiva (false).

Por cada experimento que realizen recibiran al inicio unos N organismos iniciales, con un M maximo organismos por el experimento, dónde M siempre será mas grande estrictamente que N.

Entonces a partir de estos N organismos iniciales dispondran de cinco opciones para poder trabajar con ellos:

Opción 1. Aplicar un estirón a un subconjunto de organismos.

Opción 2. Aplicar un recorte a un subconjunto de organismos.

Opción 3. Aplicar una ronda de reproducción en el experimento.

Opción 4. Obtener el ranking de reproducción de todos los organismos existentes.

Opción 5. Consultar el estado de un subconjunto de organismos.

Después de todo ello, el experimento finalizará cuando o bien lo finalizen manualmente, o todos los organismos hayan muerto o se haya alcanzado el máximo permitido.

Capítulo 2

Índice de clases

2.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 3

Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

Capítulo 4

Documentación de las clases

4.1. Referencia de la Clase Celula

Representa una celula con atributos identificador y actividad.

Métodos públicos

- Celula ()

  Creadora por defecto.
- Celula (int iden, bool activ)

  Creadora con valores concretos.
- void modificar_id (int iden)

  Modificadora del identificador.
- void modificar_activa (bool activ)

  Modificadora de la actividad.
- int consultar_id () const

  Consultora del identificador.
- bool consultar_activa () const

  Consultora de la actividad.
- void leer_celula ()

  Operación de lectura.
- void escribir_celula () const

  Operación de escritura.

Atributos privados

- int id

  Identificador de la celula.
- bool activa

  Indica si es activa (true) o pasiva (false)

4.1.1. Descripción detallada

Representa una celula con atributos identificador y actividad.

Definición en la línea 14 del archivo Celula.hpp.

4.1.2. Documentación del constructor y destructor

4.1.2.1. Celula::Celula ( )

Creadora por defecto.

Se ejecuta automáticamente al declarar una celula

Precondición

cierto

Postcondición

El resultado es una celula sin valores determinados para sus atributos

Definición en la línea 9 del archivo Celula.cpp.

```
{}
```

4.1.2.2. Celula::Celula (int *iden,* bool *activ*)

Creadora con valores concretos.

Precondición

iden $>$ 0

Postcondición

El resultado es una celula con identificador "iden" y actividad "activ"

Definición en la línea 11 del archivo Celula.cpp.

```
{
    id = iden;
    activa = activ;
}
```

4.1.3. Documentación de las funciones miembro

4.1.3.1. void Celula::modificar_id (int *iden*)

Modificadora del identificador.

Precondición

iden $>$ 0

Postcondición

El parámetro implícito pasa a tener identificador "iden"

Definición en la línea 19 del archivo Celula.cpp.

```
{
    id = iden;
}
```

4.1.3.2. void Celula::modificar_activa (bool *activ*)

Modificadora de la actividad.

Precondición

cierto

Postcondición

El parámetro implícito pasa a tener actividad "activ"

Definición en la línea 24 del archivo Celula.cpp.

```
{
    activa = activ;
}
```

4.1.3.3. int Celula::consultar_id ( ) const

Consultora del identificador.

Precondición

cierto

Postcondición

El resultado es el identificador del parámetro implícito

Definición en la línea 31 del archivo Celula.cpp.

```
{
    return id;
}
```

4.1.3.4. bool Celula::consultar_activa ( ) const

Consultora de la actividad.

Precondición

cierto

Postcondición

El resultado es la actividad del parámetro implícito

Definición en la línea 36 del archivo Celula.cpp.

```
{
    return activa;
}
```

4.1.3.5. void Celula::leer_celula ( )

Operación de lectura.

Precondición

cierto

Postcondición

Se han leido los atributos del parámetro implícito en el canal standard de entrada.

Definición en la línea 43 del archivo Celula.cpp.

```
{
    id = readint();
    if (id != 0) activa = readbool();
}
```

4.1.3.6. void Celula::escribir_celula ( ) const

Operación de escritura.

Precondición

cierto

Postcondición

Se han escrito los atributos del parámetro implícito en el canal standard de salida.

Definición en la línea 51 del archivo Celula.cpp.

```
{
    cout « id « " ";
    if (activa) cout « "1 ";
    else cout « "-1 ";
}
```

4.1.4. Documentación de los datos miembro

4.1.4.1. int Celula::id [private]

Identificador de la celula.

Definición en la línea 18 del archivo Celula.hpp.

4.1.4.2. bool Celula::activa [private]

Indica si es activa (true) o pasiva (false)

Definición en la línea 20 del archivo Celula.hpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- Celula.hpp
- Celula.cpp

4.2. Referencia de la Clase Experiment

Representa un experiment como un conjunto de organismes.

Métodos públicos

- Experiment ()

  Creadora por defecto.
- Experiment (int m)

  Creadora con tamaño predeterminado.
- int reproduccion (Ranking &R)

  Ronda de reproducción de organismes.
- void estiron (int x)

  Crecimiento de un organismo.
- void recorte (int x)

  Decrecimiento de un organismo.
- int tamano () const

  Consultora de tamaño.
- int tamano_maximo () const

  Consultora de tamaño máximo.
- int consultar_vius () const

  Consultora de organismos vivos.
- bool muerto () const

  Consultora de muerte de todos los organismos.
- void leer_experiment (int marca)

  Operación de lectura.
- void escribir_organisme (int x)

  Operación de escritura de un organisme del experiment.
- void escribir_ultims (int x)

  Operación de escritura de los ultimos hijos de la ronda.

Atributos privados

- int MAXORGANISMES

  Indica los maximos organismos posibles del experiment.
- int mida

  Indica los organismos vivos y muertos del experiment.
- int vius

  Indica los organismos vivos del experiment.
- vector< Organisme > EXP

  Sequencia de todos los organismes del experiment.
- vector< vector< bool > > emparellats

  Matriu que indica si dos organismos se han reproducido (true) o no (false)

4.2.1. Descripción detallada

Representa un experiment como un conjunto de organismes.

Definición en la línea 17 del archivo Experiment.hpp.

4.2.2. Documentación del constructor y destructor

4.2.2.1. Experiment::Experiment ( )

Creadora por defecto.

Se ejecuta automáticamente al declarar un experiment.

Precondición

cierto

Postcondición

El resultado es un experiment vacio

Definición en la línea 9 del archivo Experiment.cpp.

```
{
    MAXORGANISMES = 0;
    mida = 0;
    vius = 0;
}
```

4.2.2.2. Experiment::Experiment (int *m*)

Creadora con tamaño predeterminado.

Permite declarar un experiment nuevo con tamaño maximo especificado.

Precondición

m $>$ 2

Postcondición

El resultado es un experiment de tamano maximo m

Definición en la línea 16 del archivo Experiment.cpp.

```
{
    MAXORGANISMES = m;
    EXP = vector<Organisme>(m);
    emparellats = vector<vector<bool> >(m,vector<bool>(m,false));
}
```

4.2.3. Documentación de las funciones miembro

4.2.3.1. int Experiment::reproduccion (Ranking & *R*)

Ronda de reproducción de organismes.

Precondición

cierto

Postcondición

El resultado es el numero de hijos producidos en la ronda de reproduccion y además el parametro implicito se modifica despues de la ronda de reproduccion y R actualizado

Definición en la línea 25 del archivo Experiment.cpp.

```
{
    int fills = 0;
    int final = mida;
    int i = 0;
    int j = 1;
    vector<bool> aparicions(final,false);
    while (j < final and mida != MAXORGANISMES) {
        while (i < final - 1 and (aparicions[i] or not EXP[i].esta_vivo()))
      ++i;
        j = i + 1;
        while (j < final and (aparicions[j] or emparellats[i][j] or
      not EXP[j].esta_vivo())) ++j;
        if (j < final and i < final - 1) {
            Organisme h;
            h.reproduccion(EXP[i],EXP[j]);
            aparicions[i] = true;
            aparicions[j] = true;
            emparellats[i][j] = true;
            emparellats[j][i] = true;
            if (h.esta_vivo()) {
                EXP[mida] = h;
                ++mida;
                R.actualizar_ranking(i,j,mida);
                ++fills;
                ++vius;
            }
        }
        else if (i < final - 1) {
            ++i;
            j = i + 1;
        }
    }
    return fills;
}
```

4.2.3.2. void Experiment::estiron (int *x*)

Crecimiento de un organismo.

Precondición

1 <= x <= M y organisme x pot creixer

Postcondición

El elemento xessimo del p.i. ha sufrido un estiron

Definición en la línea 59 del archivo Experiment.cpp.

```
{
    if (EXP[x-1].consultar_potcreixer()) {
        EXP[x-1].estiron();
    }
}
```

4.2.3.3. void Experiment::recorte (int *x*)

Decrecimiento de un organismo.

Precondición

1 <= x <= M y organisme x vivo

Postcondición

El elemento xessimo del p.i. ha sufrido un recorte

Definición en la línea 66 del archivo Experiment.cpp.

```
{
    if (EXP[x-1].esta_vivo()) {
        EXP[x-1].recorte();
        if (not EXP[x-1].esta_vivo()) --vius;
    }
}
```

4.2.3.4. int Experiment::tamano ( ) const

Consultora de tamaño.

Precondición

cierto

Postcondición

El resultado es el numero de organismes del parametro implicito

Definición en la línea 76 del archivo Experiment.cpp.

```
{
    return mida;
}
```

4.2.3.5. int Experiment::tamano_maximo ( ) const

Consultora de tamaño máximo.

Precondición

cierto

Postcondición

El resultado es el maximo numero de organismes del parametro implicito

Definición en la línea 81 del archivo Experiment.cpp.

```
{
    return MAXORGANISMES;
}
```

4.2.3.6. int Experiment::consultar_vius ( ) const

Consultora de organismos vivos.

Precondición

cierto

Postcondición

El resultado es el numero de organismes vivos del parametro implicito

Definición en la línea 86 del archivo Experiment.cpp.

```
{
    return vius;
}
```

4.2.3.7. bool Experiment::muerto ( ) const

Consultora de muerte de todos los organismos.

Precondición

cierto

Postcondición

El resultado es si todos los organismos estan muertos

Definición en la línea 91 del archivo Experiment.cpp.

```
{
    return (vius == 0);
}
```

4.2.3.8. void Experiment::leer_experiment (int *marca*)

Operación de lectura.

Precondición

marca $<$ tamaño maximo

Postcondición

Se han leido los atributos del parámetro implícito en el canal standard de entrada.

Definición en la línea 98 del archivo Experiment.cpp.

```
{
    int num = 0;
    for (int i = 0; i < marca; ++i) {
        ++num;
        EXP[i].leer_organisme(num);
    }
    vius = marca;
    mida = marca;
}
```

4.2.3.9. void Experiment::escribir_organisme (int *x*)

Operación de escritura de un organisme del experiment.

Precondición

1 $<=$ x $<=$ M

Postcondición

Se han escrito los atributos del elemento xessimo del p.i. en el canal standard de salida.

Definición en la línea 111 del archivo Experiment.cpp.

```
{
    if (x <= mida) {
        cout « x « " : ";
        EXP[x-1].escribir_organisme();
        cout « endl;
    }
}
```

4.2.3.10. void Experiment::escribir_ultims (int *x*)

Operación de escritura de los ultimos hijos de la ronda.

Precondición

1 <= x <= M

Postcondición

Se han escrito los atributos del los ultimos x organismes del p.i. en la ultima ronda de reproduccion en el canal standard de salida.

Definición en la línea 120 del archivo Experiment.cpp.

```
{
    for (int i = mida - x; i < mida; ++i) {
        cout « i + 1 « " : ";
        EXP[i].escribir_organisme();
        cout « endl;
    }

}
```

4.2.4. Documentación de los datos miembro

4.2.4.1. int Experiment::MAXORGANISMES [private]

Indica los maximos organismos posibles del experiment.

Definición en la línea 21 del archivo Experiment.hpp.

4.2.4.2. int Experiment::mida [private]

Indica los organismos vivos y muertos del experiment.

Definición en la línea 23 del archivo Experiment.hpp.

4.2.4.3. int Experiment::vius [private]

Indica los organismos vivos del experiment.

Definición en la línea 25 del archivo Experiment.hpp.

4.2.4.4. vector<Organisme> Experiment::EXP [private]

Sequencia de todos los organismes del experiment.

Definición en la línea 27 del archivo Experiment.hpp.

4.2.4.5. vector<vector<bool> > Experiment::emparellats [private]

Matriu que indica si dos organismos se han reproducido (true) o no (false)

Definición en la línea 29 del archivo Experiment.hpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- Experiment.hpp
- Experiment.cpp

4.3. Referencia de la Clase Organisme

Representa un organisme como arbol de celulas, el máximo identificador y si puede crecer o no.

Métodos públicos

- Organisme ()
  Creadora por defecto.
- Organisme (const Organisme &o)
  Creadora copiadora.
- void estiron ()
  Crecimiento de un organisme.
- void recorte ()
  Decrecimiento de un organisme.
- void reproduccion (Organisme &m, Organisme &n)
  Reproducción de organismes.
- bool esta_vivo () const
  Consultora de vida de un organisme.
- int consultar_tamano () const
  Consultora de tamaño del organisme.
- int consultar_id () const
  Consultora de maximo identificador.
- int consultar_maxid () const
  Consultora de maximo identificador.
- bool consultar_potcreixer () const
  Consultora de posibilidad de crecer.
- void leer_organisme (int &num)
  Operación de lectura.
- void escribir_organisme ()
  Operación de escritura.

Métodos privados

- int interseccion (Arbre< Celula > &a, Arbre< Celula > &b)
  Tamaño de la interseccion entre dos organismes.
- bool compatibles (Organisme &a, Organisme &b)
  Compatibilidad entre dos organismes.
- void crece (Arbre< Celula > &a)
  Immersion de estiron.
- void decrece (Arbre< Celula > &a, bool &canviamax)

Immersion de recorte.

- void i_leer_organisme (Arbre< Celula > &a, int marca)

  Immersion de lectura del organisme.
- void i_escribir_organisme (Arbre< Celula > &a)

  Immersion de escritura del organisme.
- bool te_activa (Arbre< Celula > &a)

  Tamaño de la interseccion entre dos organismes.
- void i_reproduccion (Arbre< Celula > &m, Arbre< Celula > &n, Arbre< Celula > &h, int &maximid, int &tamanofill, bool &casraro)

  Immersion de la reproduccion.
- void buscar_maxid (Arbre< Celula > &a, int &max)

  Busqueda del id maximo.

Atributos privados

- Arbre< Celula > Organ

  Arbre de totes les celules del organisme.
- bool potcreixer

  Indica si puede crecer (true) o no (false)
- int id

  Identificador del organisme.
- int maxid

  Identificador máximo de un organisme.
- int tamano

  Indica el tamaño del organisme.

4.3.1. Descripción detallada

Representa un organisme como arbol de celulas, el máximo identificador y si puede crecer o no.

Definición en la línea 15 del archivo Organisme.hpp.

4.3.2. Documentación del constructor y destructor

4.3.2.1. Organisme::Organisme ( )

Creadora por defecto.

Se ejecuta automáticamente al declarar un organisme.

Precondición

cierto

Postcondición

El resultado es un organisme vacio

Definición en la línea 235 del archivo Organisme.cpp.

```
{
    tamano = 0;
    id = 0;
    maxid = 0;
    potcreixer = true;
}
```

4.3.2.2. Organisme::Organisme (const Organisme & *o*)

Creadora copiadora.

Permite declarar un organisme nuevo como copia de otro ya existente.

Precondición

cierto

Postcondición

El resultado es un organisme igual que o

Definición en la línea 243 del archivo Organisme.cpp.

```
{
    Organ = o.Organ;
    potcreixer = o.potcreixer;
    id = o.id;
    maxid = o.maxid;
}
```

4.3.3. Documentación de las funciones miembro

4.3.3.1. int Organisme::interseccion (Arbre< Celula > & *a,* Arbre< Celula > & *b*) [private]

Tamaño de la interseccion entre dos organismes.

Precondición

a = A, b = B

Postcondición

El resultado es el tamaño de la intersección entre A y B

Definición en la línea 9 del archivo Organisme.cpp.

```
{
    if (a.es_buit() or b.es_buit()) {
        return 0;
    }
    else {
        Arbre<Celula> a1,a2,b1,b2;
        Celula c1 = a.arrel();
        Celula c2 = b.arrel();
        a.fills(a1,a2);
        b.fills(b1,b2);
        int x = interseccion(a1,b1);
        int y = interseccion(a2,b2);
        a.plantar(c1,a1,a2);
        b.plantar(c2,b1,b2);
        return x + y + 1;
    }

}
```

4.3.3.2. bool Organisme::compatibles (Organisme & *a,* Organisme & *b*) [private]

Compatibilidad entre dos organismes.

Precondición

a = A, b = B

Postcondición

El resultado es si A y B son compatibles (true) o no (false)

Definición en la línea 30 del archivo Organisme.cpp.

```
{
    int mida = interseccion(a.Organ,b.Organ);
    return (mida >= int((a.consultar_tamano() + b.
      consultar_tamano())/4));
}
```

4.3.3.3. void Organisme::crece (Arbre< Celula > & *a*) [private]

Immersion de estiron.

Precondición

a = A

Postcondición

A ha sufrido un estiron

Definición en la línea 36 del archivo Organisme.cpp.

```
{
    if (not a.es_buit()) {
        Arbre<Celula> a1, a2;
        Celula c;
        c = a.arrel();
        a.fills(a1,a2);
        if (a1.es_buit() and a2.es_buit()) {
            Celula c1(c);
            Celula c2(c);
            ++maxid;
            c1.modificar_id(maxid);
            ++maxid;
            c2.modificar_id(maxid);
            tamano += 2;
            Arbre<Celula> aux1 = a1;
            Arbre<Celula> aux2 = a2;
            a1.plantar(c1,aux1,aux2);
            a2.plantar(c2,aux1,aux2);
            a.plantar(c,a1,a2);
        }
        else if (a1.es_buit()) {
            crece(a2);
            a.plantar(c,a1,a2);
        }
        else if (a2.es_buit()) {
            crece(a1);
            a.plantar(c,a1,a2);
        }
        else {
            crece(a1);
            crece(a2);
            a.plantar(c,a1,a2);
        }
    }
}
```

4.3.3.4. void Organisme::decrece (Arbre< Celula > & *a,* bool & *canviamax*) [private]

Immersion de recorte.

Precondición

a = A

Postcondición

A ha sufrido un recorte y canviamax indica si la maxid de A ha cambiado

Definición en la línea 73 del archivo Organisme.cpp.

```
{
    if (not a.es_buit()) {
        Arbre<Celula> a1, a2;
        Celula c = a.arrel();
        a.fills(a1,a2);
        if (a1.es_buit() and a2.es_buit()) {
            --tamano;
            if (c.consultar_id() == maxid) canviamax = true;
        }
        else {
            if (a1.es_buit()) {
            decrece(a2,canviamax);
            a.plantar(c,a1,a2);
            }
            else if (a2.es_buit()) {
            decrece(a1,canviamax);
            a.plantar(c,a1,a2);
            }
            else {
            decrece(a1,canviamax);
            decrece(a2,canviamax);
            a.plantar(c,a1,a2);
            }
        }
    }
}
```

4.3.3.5. void Organisme::i_leer_organisme (Arbre< Celula > & *a,* int *marca*) [private]

Immersion de lectura del organisme.

Precondición

cierto

Postcondición

Se han leido los atributos del parámetro implícito en el canal standard de entrada.

Definición en la línea 101 del archivo Organisme.cpp.

```
{
    Arbre<Celula> a1, a2;
    Celula c;
    c.leer_celula();
    if (c.consultar_id() != marca) {
        ++tamano;
        if (c.consultar_id() > maxid) maxid = c.
      consultar_id();
        i_leer_organisme(a1,marca);
        i_leer_organisme(a2,marca);
        a.plantar(c,a1,a2);
    }
}
```

4.3.3.6. void Organisme::i_escribir_organisme (Arbre< Celula > & *a*) [private]

Immersion de escritura del organisme.

Precondición

cierto

Postcondición

Se han escrito los atributos del parámetro implícito en el canal standard de salida

Definición en la línea 115 del archivo Organisme.cpp.

```
{
    if (a.es_buit()) {
        cout « "0 ";
    }
    else {
        Arbre<Celula> a1, a2;
        Celula c = a.arrel();
        a.fills(a1,a2);
        i_escribir_organisme(a1);
        c.escribir_celula();
        i_escribir_organisme(a2);
        a.plantar(c,a1,a2);
    }
}
```

4.3.3.7. bool Organisme::te_activa (Arbre< Celula > & *a*) [private]

Tamaño de la interseccion entre dos organismes.

Precondición

a = A

Postcondición

El resultado es si A tiene alguna celula activa (true) o no (false)

Definición en la línea 131 del archivo Organisme.cpp.

```
{
    if (a.es_buit()) return false;
    else {
        Celula c = a.arrel();
        if (c.consultar_activa()) return true;
        else {
            Arbre<Celula> a1, a2;
            a.fills(a1,a2);
            bool trobat = (te_activa(a1) or te_activa(a2));
            a.plantar(c,a1,a2);
            return trobat;
        }
    }

}
```

4.3.3.8. void Organisme::i_reproduccion (Arbre< Celula > & *m*, Arbre< Celula > & *n*, Arbre< Celula > & *h*, int & *maximid*, int & *tamanofill*, bool & *casraro*) [private]

Immersion de la reproduccion.

Precondición

maximid es el identificador maximo de m y h es el hijo de la reproduccion de entre m y n y m y n son compatibles

Postcondición

h contiene el organisme hijo resultante de la reproduccion entre m y n

Definición en la línea 148 del archivo Organisme.cpp.

```
{
    if (not m.es_buit() and not n.es_buit()) {
        Arbre<Celula> m1,m2,n1,n2,h1,h2;
        Celula c1 = m.arrel();
        Celula c2 = n.arrel();
        Celula aux(c1);
        if (not c1.consultar_activa() and not c2.
      consultar_activa()) {
            h.plantar(aux,m1,m2);
            ++tamanofill;
        }
        else {
            aux.modificar_activa(true);
            h.plantar(aux,m1,m2);
            ++tamanofill;
        }
        m.fills(m1,m2);
        n.fills(n1,n2);
        h.fills(h1,h2);
        i_reproduccion(m1,n1,h1,maximid,tamanofill,casoraro);
        i_reproduccion(m2,n2,h2,maximid,tamanofill,casoraro);
        m.plantar(c1,m1,m2);
        n.plantar(c2,n1,n2);
        h.plantar(aux,h1,h2);
    }
    else if (not m.es_buit() and n.es_buit()) {
        if (te_activa(m)) {
            Arbre<Celula> m1,m2,h1,h2;
            Celula c1 = m.arrel();
            h.plantar(c1,m1,m2);
            ++tamanofill;
            m.fills(m1,m2);
            h.fills(h1,h2);
            i_reproduccion(m1,n,h1,maximid,tamanofill,casoraro);
            i_reproduccion(m2,n,h2,maximid,tamanofill,casoraro);
            m.plantar(c1,m1,m2);
            h.plantar(c1,h1,h2);
        }
    }
    else if (m.es_buit() and not n.es_buit()) {
        if (te_activa(n)) {
            casoraro = true;
            Arbre<Celula> n1,n2,h1,h2;
            Celula c1 = n.arrel();
            Celula aux(c1);
            ++maximid;
            aux.modificar_id(maximid);
            h.plantar(aux,n1,n2);
            ++tamanofill;
            n.fills(n1,n2);
            h.fills(h1,h2);
            i_reproduccion(m,n1,h1,maximid,tamanofill,casoraro);
            i_reproduccion(m,n2,h2,maximid,tamanofill,casoraro);
            n.plantar(c1,n1,n2);
            h.plantar(aux,h1,h2);
        }
    }
}
```

4.3.3.9. void Organisme::buscar_maxid (Arbre< Celula > & *a*, int & *max*) [private]

Busqueda del id maximo.

Precondición

a = A, max = 0

Postcondición

max indica el maxim id del arbre A

Definición en la línea 207 del archivo Organisme.cpp.

```
{
    if (not a.es_buit()) {
        Arbre<Celula> a1,a2;
        Celula c = a.arrel();
        a.fills(a1,a2);
        if (c.consultar_id() > max) max = c.consultar_id
      ();
        if (not a1.es_buit() and a2.es_buit()) {
            buscar_maxid(a1,max);
            a.plantar(c,a1,a2);
        }
        else if (a1.es_buit() and not a2.es_buit()) {
            buscar_maxid(a2,max);
            a.plantar(c,a1,a2);
        }
        else if (not a1.es_buit() and not a2.es_buit()) {
            int max2 = 0;
            buscar_maxid(a1,max);
            buscar_maxid(a2,max2);
            if (max2 > max) max = max2;
            a.plantar(c,a1,a2);
        }
        else a.plantar(c,a1,a2);
    }
}
```

4.3.3.10. void Organisme::estiron ( )

Crecimiento de un organisme.

Precondición

potcreixer = true

Postcondición

El parametro implícito sufre un estirón

Definición en la línea 253 del archivo Organisme.cpp.

```
{
    crece(Organ);
}
```

4.3.3.11. void Organisme::recorte ( )

Decrecimiento de un organisme.

Precondición

El parametro implícito esta vivo

Postcondición

El parametro implícito sufre un recorte

Definición en la línea 258 del archivo Organisme.cpp.

```
{
    bool canviamax = false;
    decrece(Organ,canviamax);
    potcreixer = false;
    if (canviamax) {
        int max = 0;
        buscar_maxid(Organ,max);
        maxid = max;
    }
}
```

4.3.3.12. void Organisme::reproduccion (Organisme & *m,* Organisme & *n*)

Reproducción de organismes.

Precondición

cierto

Postcondición

El p.i. (hijo) es el organsime resultante de la reproduccion de m y n si m y n son compatibles, altramente no hace nada.

Definición en la línea 270 del archivo Organisme.cpp.

```
{
    if ((m.esta_vivo() and n.esta_vivo()) and compatibles
      (m,n)) {
        bool casoraro = false;
        int maximid = m.consultar_maxid();
        i_reproduccion(m.Organ,n.Organ,Organ,
      maximid,tamano,casoraro);
        if (not casoraro) {
            maximid = 0;
            buscar_maxid(Organ,maximid);
        }
        maxid = maximid;
    }
}
```

4.3.3.13. bool Organisme::esta_vivo ( ) const

Consultora de vida de un organisme.

Precondición

cierto

Postcondición

El resultado es si el parametro implícito esta vivo o no

Definición en la línea 286 del archivo Organisme.cpp.

```
{
    if (Organ.es_buit()) return false;
    else return true;
}
```

4.3.3.14. int Organisme::consultar_tamano ( ) const

Consultora de tamaño del organisme.

Precondición

cierto

Postcondición

El resultado es el tamaño del organisme

Definición en la línea 292 del archivo Organisme.cpp.

```
{
    return tamano;
}
```

4.3.3.15. int Organisme::consultar_id ( ) const

Consultora de maximo identificador.

Precondición

cierto

Postcondición

El resultado es el identificador del p.i.

Definición en la línea 297 del archivo Organisme.cpp.

```
{
    return id;
}
```

4.3.3.16. int Organisme::consultar_maxid ( ) const

Consultora de maximo identificador.

Precondición

cierto

Postcondición

El resultado es el maximo identificador del p.i.

Definición en la línea 302 del archivo Organisme.cpp.

```
{
    return maxid;
}
```

4.3.3.17. bool Organisme::consultar_potcreixer ( ) const

Consultora de posibilidad de crecer.

Precondición

cierto

Postcondición

El resultado es si puede crecer el organismo o no

Definición en la línea 307 del archivo Organisme.cpp.

```
{
    return potcreixer;
}
```

4.3.3.18. void Organisme::leer_organisme (int & *num*)

Operación de lectura.

Precondición

num = identificador del organisme a llegir

Postcondición

Se han leido los atributos del parámetro implícito en el canal standard de entrada.

Definición en la línea 314 del archivo Organisme.cpp.

```
{
    id = num;
    tamano = 0;
    maxid = 0;
    potcreixer = true;
    i_leer_organisme(Organ,0);
}
```

4.3.3.19. void Organisme::escribir_organisme ( )

Operación de escritura.

Precondición

cierto

Postcondición

Se han escrito los atributos del parámetro implícito en el canal standard de salida.

Definición en la línea 325 del archivo Organisme.cpp.

```
{
    i_escribir_organisme(Organ);
}
```

4.3.4. Documentación de los datos miembro

4.3.4.1. Arbre<Celula> Organisme::Organ [private]

Arbre de totes les celules del organisme.

Definición en la línea 19 del archivo Organisme.hpp.

4.3.4.2. bool Organisme::potcreixer [private]

Indica si puede crecer (true) o no (false)

Definición en la línea 21 del archivo Organisme.hpp.

4.3.4.3. int Organisme::id [private]

Identificador del organisme.

Definición en la línea 23 del archivo Organisme.hpp.

4.3.4.4. int Organisme::maxid [private]

Identificador máximo de un organisme.

Definición en la línea 25 del archivo Organisme.hpp.

4.3.4.5. int Organisme::tamano [private]

Indica el tamaño del organisme.

Definición en la línea 27 del archivo Organisme.hpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- Organisme.hpp
- Organisme.cpp

4.4. Referencia de la Clase Ranking

Representa un ranking de todos los organismos del experiment.

Métodos públicos

- Ranking (int N, int M)

  Creadora con valor predeterminado.
- void actualizar_ranking (int a, int b, int c)

  Actualizadora del ranking.
- void escribir_ranking () const

  Operación de escritura.

Atributos privados

- vector< list< pair< int, int > > > rank

  Sequencia de listas de familias del ranking.
- vector< int > posicions

  Sequencia de las posiciones dels organismes.

4.4.1. Descripción detallada

Representa un ranking de todos los organismos del experiment.

Definición en la línea 18 del archivo Ranking.hpp.

4.4.2. Documentación del constructor y destructor

4.4.2.1. Ranking::Ranking (int *N*, int *M*)

Creadora con valor predeterminado.

Precondición

N > 1, N < M

Postcondición

El resultado es un ranking con tamaño inicial N y tamaño maximo M

Definición en la línea 9 del archivo Ranking.cpp.

```
{
    rank = vector<list<pair<int,int> > >(M);
    posicions = vector<int>(M,0);
    for (int i = 0; i < N; ++i) posicions[i] = i+1;
}
```

4.4.3. Documentación de las funciones miembro

4.4.3.1. void Ranking::actualizar_ranking (int *a*, int *b*, int *c*)

Actualizadora del ranking.

Precondición

a i b identificadores de los padres del identificador de su respectivo hijo c

Postcondición

El p.i. se ha actualizado

Definición en la línea 16 del archivo Ranking.cpp.

```
{
    posicions[c-1] = c;
    list<pair<int,int> >::iterator it = rank[a].end();
    pair<int,int> aux;
    aux.first = b+1;
    aux.second = c;
    rank[a].insert(it,aux);                          // Insertem en la
       posició del pare el identificador
```

```
                                                    // de la mare i del
       fill
    list<pair<int,int> >::iterator it2 = rank[b].end();
    pair<int,int> aux2;
    aux2.first = a+1;
    aux2.second = c;
    rank[b].insert(it2,aux2);                       // Insertem en la
       posició de la mare el identificador
                                                    // del pare i del fill

    for (int i = 0; i < 2; ++i) {
        int aux;
        if (i == 0) aux = a+1;                      // Cas en el que es
       tracte el pare (a)
        else if (i == 1) aux = b+1;                 // Cas en el que es
       tracte la mare (b)
        int j = 0;
        while (posicions[j] != aux) ++j;
        --j;
        while (j >= 0 and (rank[posicions[j]-1]).size() < (rank
      [aux-1]).size()) {
            int x = posicions[j];
            posicions[j] = aux;
            posicions[j+1] = x;
            --j;
        }
        while (j >= 0 and (rank[posicions[j]-1]).size() == (rank
      [aux-1]).size()) {
            if (posicions[j] > posicions[j+1]) {
                int x = posicions[j];
                posicions[j] = aux;
                posicions[j+1] = x;
                --j;
            }
            else {
                j = -1;
            }
        }
    }
}
```

4.4.3.2. void Ranking::escribir_ranking ( ) const

Operación de escritura.

Precondición

cierto

Postcondición

Se han escrito los atributos del parámetro implícito en el canal standard de salida.

Definición en la línea 61 del archivo Ranking.cpp.

```
{
    int i = 0;
    while (i < posicions.size() and posicions[i] != 0) {
        cout « posicions[i] « " :";
        list<pair<int,int> >::const_iterator it = rank[posicions[i]-1].
      begin();
        while (it != rank[posicions[i]-1].end()) {
            cout « " " « (*it).first « " " « (*it).second;
            ++it;
        }
        cout « endl;
        ++i;
    }
}
```

4.4.4. Documentación de los datos miembro

4.4.4.1. vector<list<pair<int,int> > > Ranking::rank [private]

Sequencia de listas de familias del ranking.

Definición en la línea 22 del archivo Ranking.hpp.

4.4.4.2. vector<int> Ranking::posicions [private]

Sequencia de las posiciones dels organismes.

Definición en la línea 24 del archivo Ranking.hpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- Ranking.hpp
- Ranking.cpp

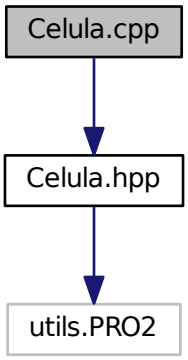
Capítulo 5

Documentación de archivos

5.1. Referencia del Archivo Celula.cpp

Código de la clase Celula.

Dependencia gráfica adjunta para Celula.cpp:



5.1.1. Descripción detallada

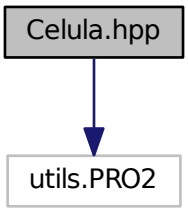
Código de la clase Celula.

Definición en el archivo Celula.cpp.

5.2. Referencia del Archivo Celula.hpp

Especificación de la clase Celula.

Dependencia gráfica adjunta para Celula.hpp:



Clases

- class Celula

  Representa una celula con atributos identificador y actividad.

5.2.1. Descripción detallada

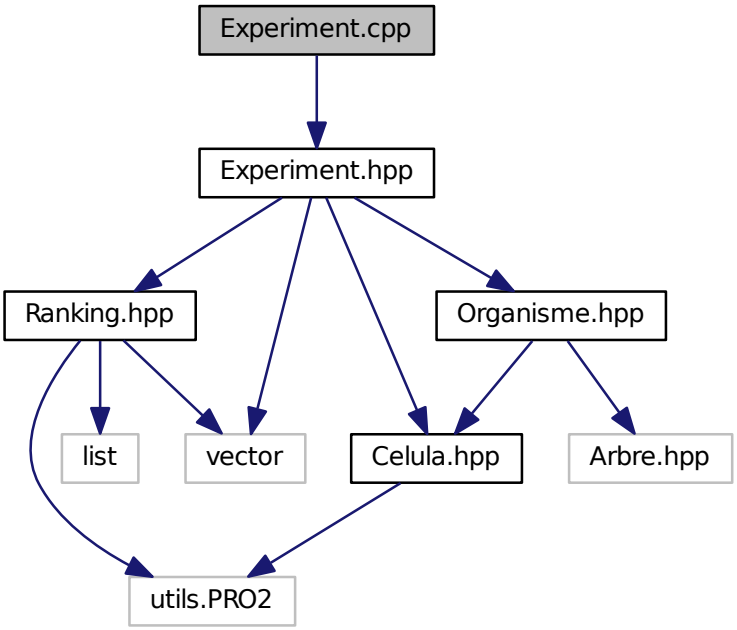
Especificación de la clase Celula.

Definición en el archivo Celula.hpp.

5.3. Referencia del Archivo Experiment.cpp

Código de la clase Experiment.

Dependencia gráfica adjunta para Experiment.cpp:



5.3.1. Descripción detallada

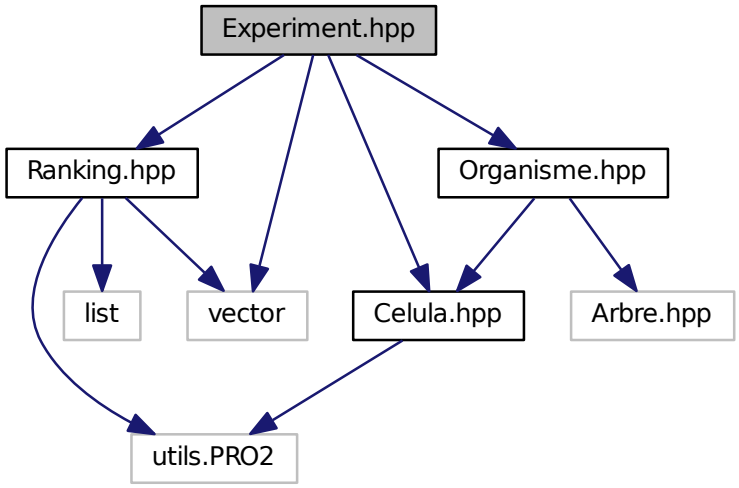
Código de la clase Experiment.

Definición en el archivo Experiment.cpp.

5.4. Referencia del Archivo Experiment.hpp

Especificación de la clase Experiment.

Dependencia gráfica adjunta para Experiment.hpp:



Clases

- class Experiment

  Representa un experiment como un conjunto de organismes.

5.4.1. Descripción detallada

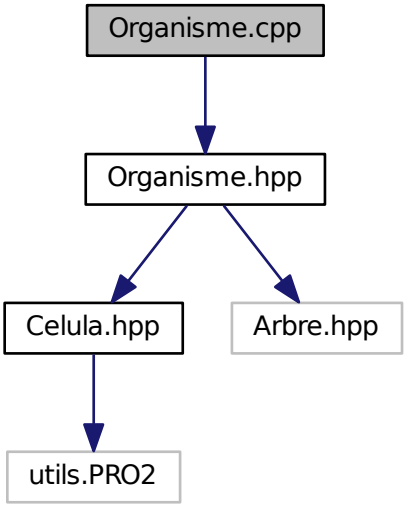
Especificación de la clase Experiment.

Definición en el archivo Experiment.hpp.

5.5. Referencia del Archivo Organisme.cpp

Código de la clase Organisme.

Dependencia gráfica adjunta para Organisme.cpp:



5.5.1. Descripción detallada

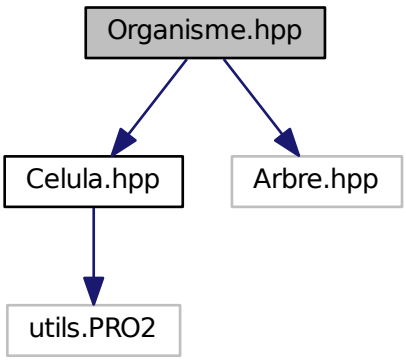
Código de la clase Organisme.

Definición en el archivo Organisme.cpp.

5.6. Referencia del Archivo Organisme.hpp

Especificación de la clase Organisme.

Dependencia gráfica adjunta para Organisme.hpp:



Clases

- class Organisme

  Representa un organisme como arbol de celulas, el máximo identificador y si puede crecer o no.

5.6.1. Descripción detallada

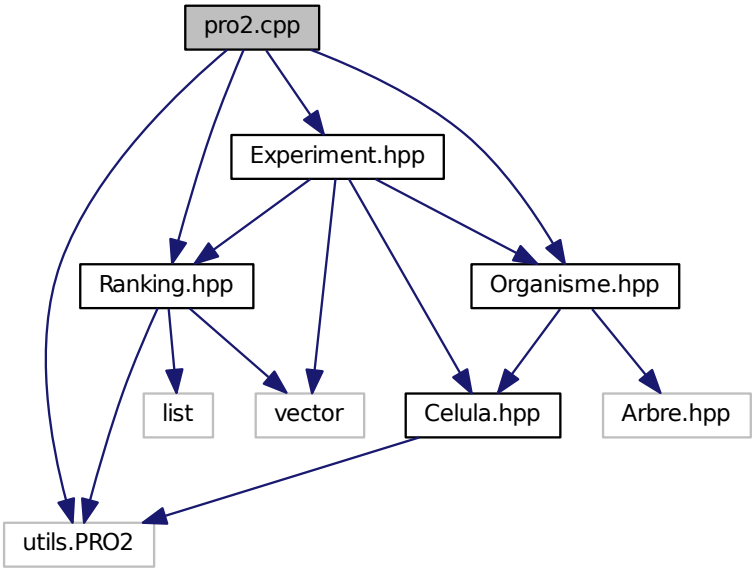
Especificación de la clase Organisme.

Definición en el archivo Organisme.hpp.

5.7. Referencia del Archivo pro2.cpp

Programa principal para la PRACTICA de PRO2.

Dependencia gráfica adjunta para pro2.cpp:



Funciones

- int main ()

  Programa principal para la PRACTICA de PRO2.

5.7.1. Descripción detallada

Programa principal para la PRACTICA de PRO2.

Definición en el archivo pro2.cpp.

5.7.2. Documentación de las funciones

5.7.2.1. int main ( )

Programa principal para la PRACTICA de PRO2.

Definición en la línea 49 del archivo pro2.cpp.

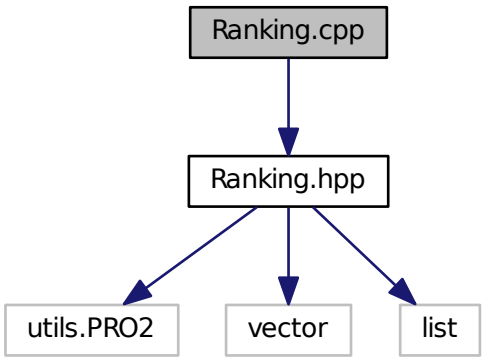
```
{
  // leer los valores enteros N (población inicial del experimento, mayor que
       1)
  int N = readint();

  // leer M (máximo permitido de población histórica, mayor que N).
  int M = readint();

  // leer los N organismos iniciales. Cada organismo ha de tener al
  // menos una célula (es decir, ha de estar vivo).
  Experiment exp(M);
  Ranking R(N,M);
  int fills_ronda;
  exp.leer_experiment(N);

  int op = readint();
  while (op != -6 and exp.tamano() < exp.tamano_maximo() and not exp.muerto())
      {
    if (op == -1) {
      // Aplicar un estirón a un subconjunto de organismos. Se proporciona una
      // secuencia de enteros no repetidos entre 1 y M.  Los organismos que
       estén
      // vivos (y no hayan empezado a sufrir recortes) en el momento de
       realizar la
      // tarea, cuyos identificadores estén en la secuencia, sufrirán el
      // correspondiente estirón.
      int x = readint();
      for (int i = 0; i < x; ++i) {
        int y = readint();
        exp.estiron(y);
      }
    }

    else if (op == -2) {
      // Aplicar un recorte a un subconjunto de organismos. Se proporciona una
      // secuencia de enteros no repetidos entre 1 y M.  Los organismos que
       estén
      // vivos en el momento de realizar la tarea, cuyos identificadores estén
       en la
      // secuencia, sufrirán el correspondiente recorte.
      int x = readint();
      for (int i = 0; i < x; ++i) {
        int y = readint();
        exp.recorte(y);
      }
    }

    else if (op == -3) {
      // Aplicar una ronda de reproducción a todos los organismos. Actualizar
       el
      // ranking consecuentemente. Escribir el número de hijos nacidos en la
       ronda.
      cout « "RONDA DE EMPAREJAMIENTOS" « endl;
      fills_ronda = exp.reproduccion(R);
      cout « "Nuevos organismos : " « fills_ronda « endl;
      cout « endl;
    }

    else if (op == -4) {
      // Obtener el ranking de reproducción de los organismos. Para cada
       organismo,
      // vivo o muerto, que haya existido hasta el momento de la consulta se ha
       de
      // obtener un listado de sus emparejamientos que hayan producido hijos.
       Cada
      // emparejamiento estará representado por el identificador de su
       compañero y el
      // del hijo. El listado vendrá ordenado por el identificador de los
       hijos.
      cout « "RANKING" « endl;
      R.escribir_ranking();
      cout « endl;
    }

    else if (op == -5) {
      // Consultar el estado de un subconjunto de organismos. Se proporciona
```

```
        una
      // secuencia de enteros no repetidos entre 1 y M.  Se escribirá la
        estructura
      // celular de los organismos existentes (vivos o muertos) en el momento
        de
      // realizar la tarea cuyos identificadores estén en la secuencia.
      int x = readint();
      cout « "ORGANISMOS" « endl;
      for (int i = 0; i < x; ++i) {
        int y = readint();
        exp.escribir_organisme(y);
      }
      cout « endl;
    }

    if (exp.tamano() < exp.tamano_maximo() or not exp.muerto()) op = readint();
  }

  cout « "FIN" « endl;
  cout « endl;

  // Tras el fin del proceso, se ha de indicar la causa.
  // Si ha sido porque la población ha llegado a M organismos (máximo
  // permitido), se han de realizar dos consultas adicionales:
  // Obtener el ranking
  // Consultar el estado de los organismos nacidos en la última ronda de
  // reproducción (atención: ésta ha podido ser incompleta)

  cout « "Organismos en total : " « exp.tamano() « endl;
  cout « "Organismos vivos : " « exp.consultar_vius() « endl;
  cout « endl;

  if (exp.tamano() == exp.tamano_maximo()) {
    cout « "ORGANISMOS" « endl;
    exp.escribir_ultims(fills_ronda);
    cout « endl;
    cout « "RANKING" « endl;
    R.escribir_ranking();
  }
}
```

5.8. Referencia del Archivo Ranking.cpp

Código de la clase Ranking.

Dependencia gráfica adjunta para Ranking.cpp:



5.8.1. Descripción detallada

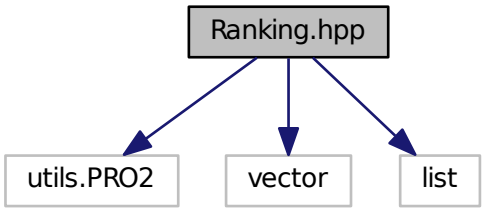
Código de la clase Ranking.

Definición en el archivo Ranking.cpp.

5.9. Referencia del Archivo Ranking.hpp

Especificación de la clase Ranking.

Dependencia gráfica adjunta para Ranking.hpp:



Clases

- class Ranking

  Representa un ranking de todos los organismos del experiment.

5.9.1. Descripción detallada

Especificación de la clase Ranking.

Definición en el archivo Ranking.hpp.

Índice alfabético