guesswho

Generado por Doxygen 1.8.11

Índice general

Capítulo 1

Índice de clases

1.1. Lista de clases

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

Capítulo 2

Indice de archivos

2.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

Capítulo 3

Documentación de las clases

3.1. Referencia de la plantilla de la Clase bintree< T >

`#include <bintree.h>`

Clases

- class const_inorder_iterator
- class const_level_iterator
- class const_postorder_iterator
- class const_preorder_iterator
- class inorder_iterator
- class level_iterator
- class node
- class postorder_iterator
- class preorder_iterator

Tipos públicos

- typedef unsigned int **size_type**

Métodos públicos

- bintree ()

  Constructor primitivo por defecto.
- bintree (const T &e)

  Constructor primitivo.
- bintree (const bintree< T > &a)

  Constructor de copia.
- void assign_subtree (const bintree< T > &a, node n)

  Reemplaza el receptor por una copia de subol.
- ~bintree ()

  Destructor.
- bintree< T > & operator= (const bintree< T > &a)

Operador de asignaciDoxyCompactList
node root () const
Obtener el nodo ra
void prune_left (node n, bintree< T > &dest)
Podar el subol a la izquierda de un nodo.
void prune_right (node n, bintree< T > &dest)
Podar el subol a la derecha de un nodo.
void insert_left (const bintree< T >::node &n, const T &e)
Insertar un nodo como hijo a la izquierda de un nodo.
void insert_left (node n, bintree< T > &rama)
Insertar un ol como subol a la izquierda de un nodo.
void insert_right (node n, const T &e)
Insertar un nodo como hijo a la derecha de un nodo.
void insert_right (node n, bintree< T > &rama)
Insertar un ol como subol a la derecha de un nodo.
void clear ()
Hace nulo un ol.
size_type size () const
Obtiene el nmero de nodos.
bool empty () const
Comprueba si un ol estces nulo).
bool operator== (const bintree< T > &a) const
Operador de comparaci igualdad.
bool operator!= (const bintree< T > &a) const
Operador de comparaci desigualdad.
void replace_subtree (node pos, const bintree< T > &a, node n)
Reemplaza el subol a partir de pos por una copia de subol.
preorder_iterator ***begin_preorder*** *()*
preorder_iterator ***end_preorder*** *()*
const_preorder_iterator ***begin_preorder*** *() const*
const_preorder_iterator ***end_preorder*** *() const*
inorder_iterator ***begin_inorder*** *()*
inorder_iterator ***end_inorder*** *()*
const_inorder_iterator ***begin_inorder*** *() const*
const_inorder_iterator ***end_inorder*** *() const*
postorder_iterator ***begin_postorder*** *()*
postorder_iterator ***end_postorder*** *()*
const_postorder_iterator ***begin_postorder*** *() const*
const_postorder_iterator ***end_postorder*** *() const*
level_iterator ***begin_level*** *()*
level_iterator ***end_level*** *()*
const_level_iterator ***begin_level*** *() const*
const_level_iterator ***end_level*** *() const*

3.1.1. Descripción detallada

template<typename T>
class bintree< T >

TDA bintree.

Representa un ol binario con nodos etiquetados con datos del tipo T.

T debe tener definidas las operaciones:

- T & operator=(const T & e);
- bool operator!=(const T & e);
- bool opertaor==(const T & e);

Son mutables. Residen en memoria dinca.

Un ejemplo de su uso:

Autor

{Miguel Garcia Silvente}
{Juan F. Huete Guadix}

3.1.2. Documentación del constructor y destructor

3.1.2.1. template<typename T > bintree< T >::bintree ( ) [inline]

Constructor primitivo por defecto.

Crea un ol nulo.

3.1.2.2. template<typename T> bintree< T >::bintree (const T & *e*) [inline]

Constructor primitivo.

Parámetros

e	Etiqueta para la ra

Crea un ol con un nico nodo etiquetado con e.

3.1.2.3. template<typename T> bintree< T >::bintree (const bintree< T > & *a*) [inline]

Constructor de copia.

Parámetros

a	ol que se copia.

Crea un ol duplicado exacto de a.

3.1.2.4. template<typename T > bintree< T >::~bintree ( ) [inline]

Destructor.

Destruye el receptor liberando los recursos que ocupaba.

3.1.3. Documentación de las funciones miembro

3.1.3.1. template<typename T> void bintree< T >::assign_subtree (const bintree< T > & *a*, node *n*)

Reemplaza el receptor por una copia de subol.

Parámetros

a	Arbol desde el que se copia.
n	nodo rael subol que se copia.

El receptor se hace nulo y despue le asigna una copia del subol de a cuya ras n.

3.1.3.2. template<typename T > void bintree< T >::clear ( )

Hace nulo un ol.

Destruye todos los nodos del ol receptor y lo hace un ol nulo.

3.1.3.3. template<typename T > bool bintree< T >::empty ( ) const [inline]

Comprueba si un ol estces nulo).

Devuelve

true, si el receptor estces nulo). false, en otro caso.

3.1.3.4. template<typename T> void bintree< T >::insert_left (const bintree< T >::node & *n,* const T & *e*)

Insertar un nodo como hijo a la izquierda de un nodo.

Parámetros

n	nodo del receptor. !n.null().
e	etiqueta del nuevo nodo.

Desconecta y destruye el subol a la izquierda de n, inserta un nuevo nodo con etiqueta e como hijo a la izquierda

3.1.3.5. template<typename T> void bintree< T >::insert_left (node *n,* bintree< T > & *rama*)

Insertar un ol como subol a la izquierda de un nodo.

Parámetros

n	nodo del receptor. n != nodo_nulo.
rama	subol que se inserta. Es MODIFICADO.

Desconecta y destruye el subol a la izquierda de n, le asigna el valor de rama como nuevo subol a la izquierda y rama se hace ol nulo.

3.1.3.6. template<typename T> void bintree< T >::insert_right (node *n,* const T & *e*)

Insertar un nodo como hijo a la derecha de un nodo.

Parámetros

n	nodo del receptor. !n.Nulo().
e	etiqueta del nuevo nodo.

Desconecta y destruye el subol a la derecha de n, inserta un nuevo nodo con etiqueta e como hijo a la derecha

3.1.3.7. template<typename T> void bintree< T >::insert_right (node *n,* bintree< T > & *rama*)

Insertar un ol como subol a la derecha de un nodo.

Parámetros

n	nodo del receptor. !n.Nulo().
rama	subol que se inserta. Es MODIFICADO.

Desconecta y destruye el subol a la izquierda de n, le asigna el valor de rama como nuevo subol a la derecha y rama se hace ol nulo.

3.1.3.8. template<typename T> bool bintree< T >::operator!= (const bintree< T > & *a*) const [inline]

Operador de comparaci desigualdad.

Parámetros

a	ol con que se compara el receptor.

Devuelve

true, si el receptor no es igual, en estructura o etiquetas a a. false, en otro caso.

3.1.3.9. template<typename T> bintree< T > & bintree< T >::operator= (const bintree< T > & *a*) [inline]

Operador de asignaci

Parámetros

a	ol que se asigna.

Destruye el contenido previo del receptor y le asigna un duplicado de a.

3.1.3.10. template<typename T> bool bintree< T >::operator== (const bintree< T > & *a*) const [inline]

Operador de comparaci igualdad.

Parámetros

a	ol con que se compara el receptor.

Devuelve

true, si el receptor es igual, en estructura y etiquetas a a. false, en otro caso.

3.1.3.11. template<typename T> void bintree< T >::prune_left (node *n,* bintree< T > & *dest*) [inline]

Podar el subol a la izquierda de un nodo.

Parámetros

n	nodo del receptor. !n.null().
dest	subol a la izquierda de n. Es MODIFICADO.

Desconecta el subol a la izquierda de n, que pasa a ser un ol nulo. El subol anterior se devuelve sobre dest.

3.1.3.12. template<typename T> void bintree< T >::prune_right (node *n,* bintree< T > & *dest*) [inline]

Podar el subol a la derecha de un nodo.

Parámetros

n	nodo del receptor. !n.null().
dest	subol a la derecha de n. Es MODIFICADO.

Desconecta el subol a la derecha de n, que pasa a ser un ol nulo. El subol anterior se devuelve sobre dest.

3.1.3.13. template<typename T> void bintree< T >::replace_subtree (node *pos*, const bintree< T > & *a*, node *n*)

Reemplaza el subol a partir de pos por una copia de subol.

Parámetros

pos	nodo a partir del que se colagar copia
a	Arbol desde el que se copia.
n	nodo rael subol que se copia.

El receptor se modifica colocando a partir de pos una copia del subol de a cuya ras n.

3.1.3.14. template<typename T > bintree< T >::node bintree< T >::root ( ) const [inline]

Obtener el nodo ra

Devuelve

nodo rael receptor.

3.1.3.15. template<typename T > bintree< T >::size_type bintree< T >::size ( ) const [inline]

Obtiene el nmero de nodos.

Devuelve

nmero de nodos del receptor.

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

- include/bintree.h
- include/bintree.hxx

3.2. Referencia de la Clase bintree< T >::const_inorder_iterator

Métodos públicos

- **const_inorder_iterator** (const const_inorder_iterator &i)
- bool **operator!=** (const const_inorder_iterator &i) const
- bool **operator==** (const const_inorder_iterator &i) const
- const_inorder_iterator & **operator=** (const const_inorder_iterator &i)
- const T & **operator*** () const
- const_inorder_iterator & **operator++** ()

Amigas

- class **bintree< T >**

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

- include/bintree.h
- include/bintree.hxx

3.3. Referencia de la Clase bintree< T >::const_level_iterator

Métodos públicos

- bool **operator!=** (const const_level_iterator &i) const
- bool **operator==** (const const_level_iterator &i) const
- const_level_iterator & **operator=** (const const_level_iterator &i)
- const T & **operator*** () const
- const_level_iterator & **operator++** ()

Amigas

- class **bintree< T >**

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

- include/bintree.h
- include/bintree.hxx

3.4. Referencia de la Clase bintree< T >::const_postorder_iterator

Métodos públicos

- bool **operator!=** (const const_postorder_iterator &i) const
- bool **operator==** (const const_postorder_iterator &i) const
- const T & **operator*** () const
- const_postorder_iterator & **operator=** (const const_postorder_iterator &i)
- const_postorder_iterator & **operator++** ()

Amigas

- class **bintree< T >**

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

- include/bintree.h
- include/bintree.hxx

3.5. Referencia de la Clase bintree< T >::const_preorder_iterator

Métodos públicos

- **const_preorder_iterator** (const const_preorder_iterator &i)
- **const_preorder_iterator** (const preorder_iterator &i)
- bool **operator!=** (const const_preorder_iterator &i) const
- bool **operator==** (const const_preorder_iterator &i) const
- const_preorder_iterator & **operator=** (const const_preorder_iterator &i)
- const T & **operator*** () const
- const_preorder_iterator & **operator++** ()

Amigas

- class **bintree< T >**

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

- include/bintree.h
- include/bintree.hxx

3.6. Referencia de la Clase bintree< T >::inorder_iterator

`#include <bintree.h>`

Métodos públicos

- **inorder_iterator** (const inorder_iterator &i)
- bool **operator!=** (const inorder_iterator &i) const
- bool **operator==** (const inorder_iterator &i) const
- inorder_iterator & **operator=** (const inorder_iterator &i)
- T & **operator*** ()
- inorder_iterator & **operator++** ()

Amigas

- class **bintree< T >**

3.6.1. Descripción detallada

template<typename T>
class bintree< T >::inorder_iterator

Clase iterator para recorrer el ol en Inorden

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

- include/bintree.h
- include/bintree.hxx

3.7. Referencia de la Clase bintree< T >::level_iterator

`#include <bintree.h>`

Métodos públicos

- **level_iterator** (const level_iterator &i)
- bool **operator!=** (const level_iterator &i) const
- bool **operator==** (const level_iterator &i) const
- level_iterator & **operator=** (const level_iterator &i)
- T & **operator*** ()
- level_iterator & **operator++** ()

Amigas

- class **bintree< T >**

3.7.1. Descripción detallada

template<typename T>
class bintree< T >::level_iterator

Clase iterator para recorrer el ol por niveles

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

- include/bintree.h
- include/bintree.hxx

3.8. Referencia de la Clase bintree< T >::node

Métodos públicos

- node ()
  Constructor primitivo.
- node (const T &e)
  Constructor primitivo.
- node (const node &n)
  Constructor de copia.
- bool null () const
  Determina si el nodo es nulo.
- node parent () const
  Devuelve el padre del nodo receptor.
- node left () const
  Devuelve el hizo izquierdo del nodo receptor.
- node right () const
  Devuelve el hizo izquierdo del nodo receptor.
- T & operator* ()
  Devuelve la etiqueta del nodo.
- const T & operator* () const
  Devuelve la etiqueta del nodo.
- void remove ()
  Elimina el nodo actual.
- node & operator= (const node &n)
  Operador de asignaci
  bool operator== (const node &n) const
  Operador de comparaci igualdad.
  bool operator!= (const node &n) const
  Operador de comparaci desigualdad.

Amigas

- class **bintree< T >**

3.8.1. Documentación del constructor y destructor

3.8.1.1. template<typename T> bintree< T >::node::node (const T & *e*) [inline]

Constructor primitivo.

Parámetros

e	Etiqueta del nodo

3.8.1.2. template<typename T> bintree< T >::node::node (const node & *n*)

Constructor de copia.

Parámetros

n	Nodo que se copia

3.8.2. Documentación de las funciones miembro

3.8.2.1. template<typename T > bintree< T >::node bintree< T >::node::left ( ) const [inline]

Devuelve el hizo izquierdo del nodo receptor.

Precondición

!null()

3.8.2.2. template<typename T > bool bintree< T >::node::operator!= (const node & *n*) const [inline]

Operador de comparaci desigualdad.

Parámetros

n	el nodo con el que se compara

3.8.2.3. template<typename T > T & bintree< T >::node::operator* ( ) [inline]

Devuelve la etiqueta del nodo.

Precondición

Si se usa como consultor, !null()

3.8.2.4. template<typename T > const T & bintree< T >::node::operator* ( ) const [inline]

Devuelve la etiqueta del nodo.

Precondición

!null()

3.8.2.5. template<typename T> bintree< T >::node & bintree< T >::node::operator= (const node & *n*) [inline]

Operador de asignaci

Parámetros

n	el nodo a asignar

3.8.2.6. template<typename T > bool bintree< T >::node::operator== (const node & *n*) const [inline]

Operador de comparaci igualdad.

Parámetros

n	el nodo con el que se compara

3.8.2.7. template<typename T > bintree< T >::node bintree< T >::node::parent ( ) const [inline]

Devuelve el padre del nodo receptor.

Precondición

!null()

3.8.2.8. template<typename T > void bintree< T >::node::remove ( )

Elimina el nodo actual.

Precondición

!null()

3.8.2.9. template<typename T > bintree< T >::node bintree< T >::node::right ( ) const [inline]

Devuelve el hizo izquierdo del nodo receptor.

Precondición

!null()

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

- include/bintree.h
- include/node.hxx

3.9. Referencia de la Clase bintree< T >::postorder_iterator

`#include <bintree.h>`

Métodos públicos

- **postorder_iterator** (const postorder_iterator &i)
- bool **operator!=** (const postorder_iterator &i) const
- bool **operator==** (const postorder_iterator &i) const
- postorder_iterator & **operator=** (const postorder_iterator &i)
- T & **operator*** ()
- postorder_iterator & **operator++** ()

Amigas

- class **bintree< T >**

3.9.1. Descripción detallada

template<typename T>
class bintree< T >::postorder_iterator

Clase iterator para recorrer el ol en PostOrden

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

- include/bintree.h
- include/bintree.hxx

3.10. Referencia de la Clase Pregunta

En cada estructura pregunta se almacena la cadena de la pregunta y el número de personajes que aún no han sido eliminados. Si el número de personajes es 1, entonces la cadena pregunta contiene el nombre del personaje.

`#include <pregunta.h>`

Métodos públicos

- Pregunta ()
  Constructor por defecto de la pregunta.
- Pregunta (const Pregunta &pregunta)
  Constructor de copias.
- Pregunta (const string atributo, const int num_personajes)
  Constructor de la pregunta.
- ~Pregunta ()
  Destructor.
- Pregunta & operator= (const Pregunta &pregunta)
  Operador de asignación.
- string obtener_pregunta () const
  Devuelve el atributo sobre el que se pregunta en el nodo.
- string obtener_personaje () const
  Devuelve el personaje del nodo.
- int obtener_num_personajes () const
  Devuelve el número de personajes sin eliminar al llegar a esta pregunta.
- bool es_personaje () const
  Devuelve true si el nodo es de personaje.
- bool es_pregunta () const
  Devuelve true si el nodo es de pregunta.

Amigas

- ostream & operator<< (ostream &os, const Pregunta &pregunta)
  Operador de inserción de flujo.

3.10.1. Descripción detallada

En cada estructura pregunta se almacena la cadena de la pregunta y el número de personajes que aún no han sido eliminados. Si el número de personajes es 1, entonces la cadena pregunta contiene el nombre del personaje.

3.10.2. Documentación del constructor y destructor

3.10.2.1. Pregunta::Pregunta ( )

Constructor por defecto de la pregunta.
Reserva los recursos.

3.10.2.2. Pregunta::Pregunta (const Pregunta & *pregunta*)

Constructor de copias.

Parámetros

pregunta	Pregunta a copiar

Construye la pregunta duplicando el contenido de *pregunta* en la pregunta receptora.

3.10.2.3. Pregunta::Pregunta (const string *atributo*, const int *num_personajes*)

Constructor de la pregunta.

Parámetros

atributo	Atributo sobre el que se pregunta en este nodo. En el caso de que haya un único personaje restante, este campo almacena su nombre.
num_personajes	Número de personajes que quedan al llegar a esta pregunta.

3.10.2.4. Pregunta::∼Pregunta ( )

Destructor.
Libera los recursos ocupados por la pregunta receptora.

3.10.3. Documentación de las funciones miembro

3.10.3.1. string Pregunta::obtener_personaje ( ) const

Devuelve el personaje del nodo.

Precondición

El nodo debe ser un nodo de personaje (num_personaje==1).

3.10.3.2. string Pregunta::obtener_pregunta ( ) const

Devuelve el atributo sobre el que se pregunta en el nodo.

Precondición

El nodo debe ser un nodo de pregunta (num_personaje>1).

3.10.3.3. Pregunta& Pregunta::operator= (const Pregunta & *pregunta*)

Operador de asignación.

Parámetros

pregunta	Pregunta a copiar

Devuelve

Referencia a la pregunta receptora.

Asigna el valor de la pregunta duplicando el contenido de *pregunta* en la pregunta receptora.

3.10.4. Documentación de las funciones relacionadas y clases amigas

3.10.4.1. ostream& operator<< (ostream & *os,* const Pregunta & *pregunta*) [friend]

Operador de inserción de flujo.

Parámetros

os	Stream de salida.
pregunta	Pregunta a escribir.

Devuelve

Referencia al stream de salida.

Escribe en la salida la pregunta, escribiendo primero la cadena de la pregunta y después el número de personajes que quedan al llegar a esta pregunta.

La documentación para esta clase fue generada a partir del siguiente fichero:

- include/pregunta.h

3.11. Referencia de la Clase bintree< T >::preorder_iterator

`#include <bintree.h>`

Métodos públicos

- **preorder_iterator** (const preorder_iterator &i)
- bool **operator!=** (const preorder_iterator &i) const
- bool **operator==** (const preorder_iterator &i) const
- preorder_iterator & **operator=** (const preorder_iterator &i)
- T & **operator*** ()
- preorder_iterator & **operator++** ()

Amigas

- class **bintree**< **T** >

3.11.1. Descripción detallada

template<**typename T**>
class bintree< **T** >**::preorder_iterator**

Clase iterator para recorrer el ol en PreOrden

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

- include/bintree.h
- include/bintree.hxx

3.12. Referencia de la Clase QuienEsQuien

T.D.A. QuienEsQuien.

`#include <quienesquien.h>`

Métodos públicos

- QuienEsQuien ()
  Constructor bsico de la clase.
- QuienEsQuien (const QuienEsQuien &quienEsQuien)
  Constructor de copia.
- ~QuienEsQuien ()
  Destructor del pregunta.
- vector< string > getPersonajes () const
  Destructor del pregunta.
- vector< string > getAtributos () const
  Destructor del pregunta.
- bintree< Pregunta > getArbol () const
  Destructor del pregunta.
- vector< vector< bool > > getTablero () const
  Destructor del pregunta.
- bintree< Pregunta >::node getJugada_actual () const
  Destructor del pregunta.
- void limpiar ()
  Vacia todos los datos del QuienEsQuien receptor.
- void crear_nodo (vector< int > v, typename bintree< Pregunta >::node &nodo, int profundidad)
  Vacia todos los datos del QuienEsQuien receptor.
- QuienEsQuien & operator= (const QuienEsQuien &quienEsQuien)
  Sobrecarga del operador de asignacin.
- void mostrar_estructuras_leidas ()
  Escribe en la salida estandard las estructuras atributos, personajes y tablero.
- bintree< Pregunta > crear_arbol ()
  Este mtodo construye el rbol de preguntas para todos los personajes del tablero.
- void usar_arbol (bintree< Pregunta > arbol_nuevo)
  Sustituye el rbol actual por el rbol pasado por parmetro.
- void escribir_arbol_completo () const
  Escribe el arbol generado en la salida estandard.
- void eliminar_nodos_redundantes ()
  Mtodo que modifica el rbol de preguntas para que haya preguntas redundantes.
- void iniciar_juego ()
  Inicializa el juego.

- set< string > informacion_jugada (bintree< Pregunta >::node jugada_actual)
  Dado un estado del tablero devuelve los nombres de los personajes que an no han sido tumbados en el tablero.
- float profundidad_promedio_hojas ()
  Este mtodo permite calcular la media de la profundidad de las hojas del rbol. Este valor representa el nmero (promedio) de preguntas necesarias para adivinar cada personaje. A menor profundidad promedio, mejor solucin. Esta mtrica es un indicador para evaluar la calidad de vuestra solucin.
- float **calcular_profundidad_promedio** (const bintree< Pregunta > &a, typename bintree< Pregunta >::node n)
- void tablero_aleatorio (int numero_de_personajes)
  Rellena los datos del QuienEsQuien con un tablero calculado aleatoriamente.
- void preguntas_formuladas (bintree< Pregunta >::node jugada)
  Muestra una descripción de las preguntas formuladas anteriormente y las respuestas dadas por el usuario hasta ahora. Dado un estado del tablero devuelve los nombres de los personajes que aún no han sido tumbados en el tablero.
- void aniade_personaje (string nombre, vector< bool > caracteristicas)
  Inserta el personaje nuevo en el árbol ya construido a partir de su nombre y descripción asociada (vector<bool>).
- void elimina_personaje (string nombre)
  Elimina un personaje del arbol de QuienEsQuien.

Amigas

- istream & operator>> (istream &is, QuienEsQuien &quienEsQuien)
  Operador de extraccin de flujo.
- ostream & operator<< (ostream &os, const QuienEsQuien &quienEsQuien)
  Operador de insercin de flujo.

3.12.1. Descripción detallada

T.D.A. QuienEsQuien.

Conjunto de personajes con sus atributos.

3.12.2. Documentación del constructor y destructor

3.12.2.1. QuienEsQuien::QuienEsQuien (const QuienEsQuien & *quienEsQuien*)

Constructor de copia.

Parámetros

quienEsQuien	QuienEsQuien a copiar.

3.12.3. Documentación de las funciones miembro

3.12.3.1. void QuienEsQuien::aniade_personaje (string *nombre,* vector< bool > *caracteristicas*)

Inserta el personaje nuevo en el árbol ya construido a partir de su nombre y descripción asociada (vector<bool>).

Parámetros

nombre	Nombre del jugador a insertar en el árbol.
caracteristicas	Descripcion asociada al jugador a insertar en el árbol.

Devuelve

QuienEsQuien con un nuevo personaje de nombre pasado por parámetro.

Precondición

El arbol del QuienEsQuien receptor debe haber sido construido previamente.

3.12.3.2. void QuienEsQuien::elimina_personaje (string *nombre*)

Elimina un personaje del arbol de QuienEsQuien.

Parámetros

nombre	Nombre del jugador a eliminar.

Devuelve

QuienEsQuien con un personaje eliminado de nombre pasado por parámetro.

Precondición

El arbol del QuienEsQuien receptor debe haber sido construido previamente.

3.12.3.3. void QuienEsQuien::eliminar_nodos_redundantes ( )

Mtodo que modifica el rbol de preguntas para que haya preguntas redundantes.

Postcondición

El rbol de preguntas se modifica.

3.12.3.4. set<string> QuienEsQuien::informacion_jugada (bintree< Pregunta >::node *jugada_actual*)

Dado un estado del tablero devuelve los nombres de los personajes que an no han sido tumbados en el tablero.

Parámetros

jugada_actual	Nodo del estado del tablero.

Devuelve

Conjunto de personajes que no han sido tumbados en el tablero.

Precondición

El arbol del QuienEsQuien receptor debe haber sido construido previamente.
El nodo indicado debe ser un nodo del arbol del QuienEsQuien receptor.

3.12.3.5. void QuienEsQuien::iniciar_juego ( )

Inicializa el juego.

Postcondición

Si la partida anterior no haba terminado se pierde el progreso.

3.12.3.6. QuienEsQuien& QuienEsQuien::operator= (const QuienEsQuien & *quienEsQuien*)

Sobrecarga del operador de asignacin.

Parámetros

quienEsQuien	objeto a copiar.

Devuelve

Referencia al objeto copiado.

3.12.3.7. void QuienEsQuien::preguntas_formuladas (bintree< Pregunta >::node *jugada*)

Muestra una descripción de las preguntas formuladas anteriormente y las respuestas dadas por el usuario hasta ahora. Dado un estado del tablero devuelve los nombres de los personajes que aún no han sido tumbados en el tablero.

Parámetros

jugada	Momento del juego a consultar.

Devuelve

Conjunto de personajes que no han sido tumbados en el tablero.

Precondición

El arbol del QuienEsQuien receptor debe haber sido construido previamente.
El nodo indicado debe ser un nodo del arbol del QuienEsQuien receptor.

3.12.3.8. float QuienEsQuien::profundidad_promedio_hojas ( )

Este mtodo permite calcular la media de la profundidad de las hojas del rbol. Este valor representa el nmero (promedio) de preguntas necesarias para adivinar cada personaje. A menor profundidad promedio, mejor solucin. Esta mtrica es un indicador para evaluar la calidad de vuestra solucin.

Devuelve

Profundidad media del arbol de preguntas.

Precondición

El arbol de preguntas debe haber sido construido previamente.

3.12.3.9. void QuienEsQuien::tablero_aleatorio (int *numero_de_personajes*)

Rellena los datos del QuienEsQuien con un tablero calculado aleatoriamente.

Parámetros

numero_de_personajes	Nmero de personajes que tiene el tablero a crear.

3.12.3.10. void QuienEsQuien::usar_arbol (bintree< Pregunta > *arbol_nuevo*)

Sustituye el rbol actual por el rbol pasado por parmetro.

Parámetros

arbol_nuevo	Arbol de preguntas que sustituye al actual.

3.12.4. Documentación de las funciones relacionadas y clases amigas

3.12.4.1. ostream& operator<< (ostream & *os,* const QuienEsQuien & *quienEsQuien*) [friend]

Operador de insercin de flujo.

Parámetros

os	Stream de salida
quienEsQuien	Quien es quien a escribir.

Devuelve

Referencia al stream de salida.

Escribe en *is* un quien es quien. El formato usado para la escritura es un TSV (tab-separated values) en el que las columnas tienen cabecera y son las preguntas. La ltima columna corresponde al nombre del personaje. Tras la cabecera se especifica en cada lnea un personaje, teniendo el valor 1 o 0 si tiene/no tiene el atributo de la columna. En la ltima columna se da el nombre del personaje.

3.12.4.2. istream& operator>> (istream & *is,* QuienEsQuien & *quienEsQuien*) [friend]

Operador de extraccin de flujo.

Parámetros

is	Stream de entrada
quienEsQuien	Quien es quien a leer.

Devuelve

Referencia al stream de entrada.

Lee de *is* un quien es quien y lo almacena en *quienEsQuien*. El formato aceptado para la lectura es un TSV (tab-separated values) en el que las columnas tienen cabecera y son las preguntas. La ltima columna corresponde al nombre del personaje. Tras la cabecera se especifica en cada lnea un personaje, teniendo el valor 1 o 0 si tiene/no tiene el atributo de la columna. En la ltima columna se da el nombre del personaje.

La documentación para esta clase fue generada a partir del siguiente fichero:

- include/quienesquien.h

Capítulo 4

Documentación de archivos

4.1. Referencia del Archivo include/quienesquien.h

Fichero cabecera del QuienEsQuien.

```
#include <string>
#include <iostream>
#include <map>
#include <vector>
#include <cassert>
#include <set>
#include <sstream>
#include "pregunta.h"
#include "bintree.h"
```

Dependencia gráfica adjunta para quienesquien.h: